Active Structural Acoustic Control of Clamped and Ribbed Plates

William Richard Johnson
Brigham Young University - Provo

Active Structural Acoustic Control of Clamped and Ribbed Plates

William R. Johnson

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Mechanical Engineering

Brigham Young University

December 2013

ABSTRACT

Active Structural Acoustic Control of Clamped and Ribbed Plates

William R. Johnson
Department of Mechanical Engineering, BYU
Master of Science

A control metric, the weighted sum of spatial gradients (WSSG), has been developed for use in active structural acoustic control (ASAC). Previous development of WSSG [1] showed that it was an effective control metric on simply-supported plates, while being simpler to measure than other control metrics, such as volume velocity. The purpose of the current work is to demonstrate that the previous research can be generalized to plates with a wider variety of boundary conditions and on less ideal plates. Two classes of plates have been considered: clamped flat plates, and ribbed plates.

On clamped flat plates an analytical model has been developed for use in WSSG that assumes the mode shapes are the product of clamped-clamped beam mode shapes. The boundary condition specific weights for use in WSSG have been derived from this formulation and provide a relatively uniform measurement field, as in the case of the simply-supported plate. Using this control metric, control of radiated sound power has been simulated. The results show that WSSG provides comparable control to volume velocity on the clamped plate. Results also show, through random placement of the sensors on the plate, that similar control can be achieved regardless of sensor location. This demonstrates that WSSG is an effective control metric on a variety of boundary conditions.

Ribbed plates were considered because of their wide use in aircraft and ships. In this case, a finite-element model of the plate has been used to obtain the displacement field on the plate under a variety of boundary conditions. Due to the discretized model involved, a numerical, as opposed to analytical, formulation for WSSG has been developed. Simulations using this model show that ASAC can be performed effectively on ribbed plates. In particular WSSG was found to perform comparable to or better than volume velocity on all boundary conditions examined. The sensor insensitivity property was found to hold within each section (divided by the ribs) of the plate, a slightly modified form of the flat plate insensitivity property where the plates have been shown to be relatively insensitive to sensor location over the entire surface of the plate. Improved control at natural frequencies can be achieved by applying a second control force. This confirms that ASAC is a viable option for the control of radiated sound power on non-ideal physical systems similar to ribbed plates.

Keywords: active structural acoustic control, ASAC, clamped plate vibration, ribbed plate vibration, active noise control, ANC, wave equation, Bernoulli-Euler beam theory, composite velocity, weighted sum of spatial gradients, WSSG

ACKNOWLEDGMENTS

I would like to thank, first of all, the other students who have worked with me on this project, including Dan Hendricks, Pegah Aslani, and Monty Anderson. In particular I would like to thank Dan who has collaborated with me over the past two years. I would also like to thank my advisor Dr. Jonathan Blotter, and my committee members Dr. Scott Sommerfeldt and Dr. Kent Gee, all of whom I feel I have been privileged to work with and learn from over the course of my masters degree. It has been a pleasure to take classes from them and meet with them on a weekly basis.

I would be remiss if I didn’t also thank my parents, Rick and Virginia Johnson, who have always believed in me and supported me in everything I do.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

V_{comp}^2	Squared composite velocity
$w(x,y)$	Plate displacement
$WSSG$	The weighted sum of spatial gradients objective function
α	WSSG weight for the displacement term
β	WSSG weight for the $\partial/\partial x$ term
β_m	WSSG β weight for the *m*th mode
γ	WSSG weight for the $\partial/\partial y$ term
γ_n	WSSG γ weight for the *n*th mode
δ	WSSG weight for the $\partial^2/\partial x\partial y$ term
δ_{mn}	WSSG δ weight for the *mn*th mode
L_x	Plate length in the *x* direction
L_y	Plate length in the *y* direction
ω	Excitation frequency (rad/s)
ω_{mn}	Plate natural frequencies (rad/s)
q	Index representing the *q*th applied force
m	Index representing the *m*th mode in the *x* direction
n	Index representing the *n*th mode in the *y* direction
D	Plate bending stiffness
ρ	Plate density
h	Plate thickness
$\bar{m}$	Mass per unit area of the plate
U	Volume velocity
$F(x,y)$	Applied forces on the plate
$\delta(\cdot)$	The Dirac delta function
d	Subscript indicating the disturbance force
c	Subscript indicating the control force
s	Subscript indicating the sensor(s)
$\phi(\cdot)$	Assumed mode shape functions for the clamped plate
λ_p	Eigenvalues of the system
k	The wavenumber
$\mathbf{R}$	Radiation resistance matrix
$\mathbf{v_e}$	Elemental radiator average velocity
Δx	Finite difference derivative spacing in x
Δy	Finite difference derivative spacing in y
H_{d_i}	Transfer function from the disturbance force to the *i*th sensor location (displacement over force)
H_{c_i}	Transfer function from the control force to the *i*th sensor location (displacement over force)
H_w	Transfer function from a given input to the displacement term of WSSG at the sensor location
$H_{\frac{\partial w}{\partial x}}$	Transfer function from a given input to the $\partial/\partial x$ term of $WSSG$ at the sensor location
$H_{\frac{\partial w}{\partial y}}$	Transfer function from a given input to the $\partial/\partial y$ term of $WSSG$ at the sensor location
$H_{\frac{\partial^2 w}{\partial x\partial y}}$	Transfer function from a given input to the $\partial^2/\partial x\partial y$ term of WSSG at the sensor location
H_1	Transfer function from a given input to the displacement at the first sensor measurement point
H_2	Transfer function from a given input to the displacement at the second sensor measurement point

H_3	Transfer function from a given input to the displacement at the third sensor measurement point
H_4	Transfer function from a given input to the displacement at the fourth sensor measurement point

CHAPTER 1. INTRODUCTION

Active structural acoustic control (ASAC) is a subfield of active noise control (ANC) concerned with reducing sound radiated by distributed structures such as plates and shells. For many noise sources ANC is accomplished by placing microphones and speakers in a sound field [2]. However, for distributed structures, a prohibitively large number of microphones and speakers are required in order to achieve global control [3] [4] [5] [6], and it is not always convenient to have control hardware in the sound field, as in the case of the interior of an airplane fuselage. Research showed that, instead of using speakers in a sound field, the control actuator could be moved to the surface of the distributed structure, which resulted in a significant reduction in the number of control actuators required to achieve control [7] [8]. Not long after this development, it was also suggested that a structural quantity could be measured on the plate, to achieve control [9]. However, this quantity would need to be related to the radiated sound power. By moving the sensor to the plate, a reduction in the number of sensors required for control, could be obtained. These two developments, the movement of control actuators and sensors to the boundary of the sound field, and the resulting simplification of the system required to achieve global noise control, are the main benefits of ASAC.

Although ASAC is potentially simpler than traditional ANC, it has inherited many of its problems. Two problems of particular significance for both techniques are the choice of sensor location [10], and the choice of the measurement quantity, or control metric, to be minimized. Without sufficient analysis either of these could result in sound amplification.

An important theoretical development that has guided research in resolving these problems is the radiation resistance matrix and its radiation mode shapes [9]. The radiation resistance matrix was developed as an alternative to the Rayleigh integral for use in calculating the radiated sound power from distributed structures. Instead of integrating over a surface in the far field, as the Rayleigh integral does, the radiation resistance matrix method discretizes the surface of the structure into elementary radiators and then estimates the radiated power through the calculation of the self and mutual impedances between each elementary radiator on the surface [11] [12]. The radiation resistance matrix contains the mutual radiation impedances, and its eigenvectors are the radiation mode shapes. The radiation mode shapes reveal the underlying sound radiation mechanisms of the structure. By targeting the radiation mode shapes of the structure, an error sensor on the structure can measure a quantity related to the radiated sound power of the structure and therefore provide control.

Several control metrics have been developed that target the radiation mode shapes of a distributed structure. At low frequencies the first radiation mode shape is the only efficient radiator, and has a drum-like appearance. Volume velocity, which is calculated as the cross sectional area of a space multiplied by the speed of the fluid flowing through it, is a measurable quantity on the surface of the plate and has a similar appearance to the first radiation mode [9]. This has become the predominant control metric in the literature because of its success in providing control [13] [14] [15]. Other research has developed a method to target specific radiation or structural modes using optimally shaped piezoelectric sensors [16] [17].

Some control metrics which do not target the radiation mode shapes have been developed. For example, the most basic control metric used in the literature is the measurement of squared velocity at a point on the plate [18]. Energy based control metrics have also been developed [19]. ASAC control metrics which have not targeted radiation modes have met with limited success at best.

Despite the variety of sensing techniques available, each of the methods described has one of two common problems. Either they require *a priori* knowledge about the radiation or structural vibration modes of the system to be controlled, or they require a large number of sensors. For example, several methods have been developed to measure volume velocity. One method uses an array of accelerometers to measure the velocity across the plate. For a plate of size 0.278 m x 0.247 m, to accurately measure volume velocity, 16 to 25 sensors was required [14], a prohibitively large number of sensors in practice. The increase in the number of sensors can be avoided by using a distributed piezoelectric sensor to measure volume velocity [15]; however, this sensor would need to be designed for the specific geometry. Therefore, while the increased complexity from the sensor array is alleviated, the geometry dependence of the sensor requires *a priori* knowledge about the system. If, instead of volume velocity, the radiation mode shapes were targeted directly through the use of shaped piezoelectric sensors [16] *a priori* knowledge would once again be required in order to shape the sensor properly for the given geometry. Finally, squared velocity, which requires only a single accelerometer to measure, is highly dependent on sensor location, and care must be taken not to place the sensor on a nodal line, once again requiring *a priori* knowledge of the system. As a result of the difficulty in implementing control control metrics in practice, ASAC has seen very little practical application.

A summary of the preceding discussion regarding ANC, ASAC, and control metrics can be seen in Fig. 1.1. From this brief survey of control metrics and measuring techniques it can be concluded that in order for ASAC to live up to its potential as a simpler and more effective technique than traditional active noise control, an control metric which requires only a few sensors, is insensitive to sensor location, and does not depend on plate geometry, is necessary. A recent control metric, termed composite velocity (also referred to as WSSG or the weighted sum of spatial

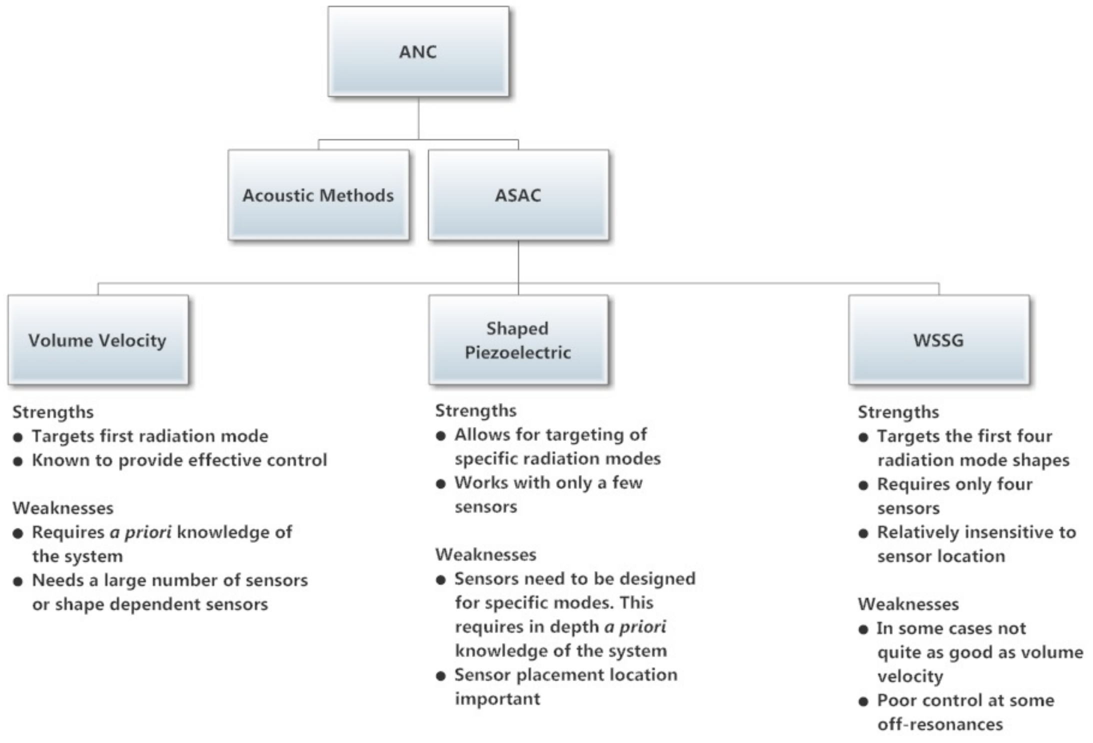

ANC
Acoustic Methods
ASAC
Volume Velocity
Shaped Piezoelectric
WSSG
Strengths
● Targets first radiation mode
● Known to provide effective control
Weaknesses
● Requires *a priori* knowledge of the system
● Needs a large number of sensors or shape dependent sensors
Strengths
● Allows for targeting of specific radiation modes
● Works with only a few sensors
Weaknesses
● Sensors need to be designed for specific modes. This requires in depth *a priori* knowledge of the system
● Sensor placement location important
Strengths
● Targets the first four radiation mode shapes
● Requires only four sensors
● Relatively insensitive to sensor location
Weaknesses
● In some cases not quite as good as volume velocity
● Poor control at some off-resonances

Figure 1.1: General overview of how the current research fits into active noise control

gradients in this thesis), has shown promise in resolving these issues. Composite velocity was developed as a weighted sum of spatial velocity gradients requiring only four sensors to measure, and was found to be relatively insensitive to sensor location on a simply-supported plate, thus achieving the simplicity possible with ASAC and avoiding the extra work *a priori* knowledge would require. Previous research developed composite velocity [1] [20] and showed that it had initial promise as a control metric. The purpose of this research is twofold. The first objective is to develop the composite velocity control metric theoretically for clamped plates and demonstrate its effectiveness. The second objective is to develop a numerical version of composite velocity to test its effectiveness on the less ideal but more practical ribbed plate. Results demonstrate that often the control metric performs better than previous ones while also being easier to implement.

1.1 Overview of WSSG and Volume Velocity

Previously it was mentioned that volume velocity, the most researched control metric to date, was chosen because of its similarity to the first radiation mode shape. In a similar manner terms of composite velocity were chosen for their likeness of the first four radiation mode shapes, thus attempting to control them. The original general expression for composite velocity, or V^2_{comp} [1], was given as:

$$V^2_{comp}(x,y) = \alpha\left(\frac{\partial w}{\partial t}\right)^2 + \beta\left(\frac{\partial^2 w}{\partial x \partial t}\right)^2 + \gamma\left(\frac{\partial^2 w}{\partial y \partial t}\right)^2 + \delta\left(\frac{\partial^3 w}{\partial x \partial y \partial t}\right)^2 \tag{1.1}$$

where w is the plate displacement, and α, β, γ, and δ are the weights for each accompanying term. These weights can be found using analytical or numerical methods. They are found analytically, mode by mode, for the simply-supported rectangular plate, by taking the derivatives of the analytical solution to the wave equation. The analytical solution assuming time harmonic motion is given in Eq. (1.2) as

$$w(x,y) = \sum_{q=1}^{F} \frac{f_q}{\rho h} \sum_{m}^{\infty}\sum_{n}^{\infty} \frac{W_{mn}(x,y)W_{mn}(x_q,y_q)(\omega^2_{mn} - \omega^2 - i\eta\,\omega^2_{mn})}{(\omega^2_{mn} - \omega^2)^2 + \eta^2\omega^4_{mn}} e^{j\omega t}, \tag{1.2}$$

where

$$W_{mn}(x,y) = \frac{2}{\sqrt{L_x L_y}} sin\left(\frac{m\pi x}{L_x}\right) sin\left(\frac{n\pi y}{L_y}\right) \tag{1.3}$$

and L_x and L_y are the lengths in the x and y directions of the plate, respectively. Also, q indexes the applied forces, m and n are the mode numbers, ω is the excitation frequency, ω_{mn} are natural frequencies of the plate, ρ is the plate density, h is the plate thickness, and η is the damping ratio. For convenience, the $e^{j\omega t}$ term, which represents the time harmonic motion, is hereafter left off. When the spatial derivatives $\partial/\partial x$, $\partial/\partial y$, and $\partial^2/\partial x \partial y$ of Eq. (1.3) are taken it can easily be seen that each of these terms will be scaled by $\frac{m\pi}{L_x}$, $\frac{n\pi}{L_y}$, and $\frac{mn\pi^2}{L_x L_y}$ respectively. This gave an obvious method to determine weights for each of these terms. By choosing $\alpha = 1$, $\beta = \left(\frac{L_x}{m\pi}\right)^2$, $\gamma = \left(\frac{L_y}{n\pi}\right)^2$, and $\delta = \left(\frac{L_x L_y}{mn\pi^2}\right)^2$ the magnitudes of each of the spatial gradient terms are scaled to be the same as the magnitude of the $\partial w/\partial t$ term. Using these weights yields an extremely uniform field over the surface of the plate at a specific mode.

Composite velocity was originally derived by taking spatial derivatives of the velocity under the assumption of time harmonic vibration, as shown above. With this assumption each of the velocity terms is scaled by the same constant value of $j\omega$ when the time derivative is taken. This constant scaling has no effect on the performance of the control algorithm, therefore the time derivative portion can simply be dropped, making the velocity portion of "composite velocity" a misnomer. In future derivations in this thesis, V_{comp}^2 will be referred to as the "weighted sum of spatial gradients" or WSSG for short, and will be expressed as:

$$\text{WSSG}(x,y) = \alpha\,(w)^2 + \beta\left(\frac{\partial w}{\partial x}\right)^2 + \gamma\left(\frac{\partial w}{\partial y}\right)^2 + \delta\left(\frac{\partial^2 w}{\partial x \partial y}\right)^2 \tag{1.4}$$

The previous research also predicted, through simulations, that WSSG provided a comparable level of control to other standard control metrics, such as volume velocity and squared velocity on a simply supported plate, while being insensitive to sensor location. To validate the current work, volume velocity, U, has been used as a standard metric for comparison. The volume velocity at the surface of a vibrating plate is given in Eq. (1.5) as

$$U = \int_0^{L_x}\int_0^{L_y} \frac{\partial w}{\partial t} dxdy. \tag{1.5}$$

This integral does not need to be evaluated exactly, it can be estimated with acceptable accuracy by discretizing the plate and summing over the discrete elements, as represented in Eq. (1.6)

$$U \approx \sum_i \sum_j \frac{\partial w(x_i, y_j)}{\partial t} \Delta x \Delta y \tag{1.6}$$

where i and j are the ith and jth discrete elements in the x and y directions respectively, and Δx and Δy are the element lengths. The discrete method shown here is generally used in this thesis.

1.1.1 Research Scope and Objectives

The primary goal of this research has been to extend the formulation of WSSG for simply-supported plates to clamped flat plates, and ribbed plates with different boundary conditions. The research has also focused on demonstrating the effectiveness of WSSG when compared to other commonly used control metrics. The research objectives are as follows:

1. Develop WSSG theoretically for a clamped flat rectangular plate and simulate control.
2. Develop WSSG theoretically for a ribbed rectangular plate for a variety of boundary conditions and simulate control.

The structure of this thesis will follow an outline similar to the research objectives stated above. Chapter 2 will cover the clamped flat rectangular plate theoretical development and control simulation development. Chapter 3 will cover the ribbed plate theoretical development and control simulations. Chapter 3 represents the case of greatest practical importance as many structures are built out of ribbed plates and very little has been published on the applicability of ASAC on these plates. This chapter represents the culmination of the research shown here. All work on WSSG prior to the ribbed plate was done on idealized cases. However chapter 3 focuses on a case which occurs frequently in industry and demonstrates the practicality of using ASAC with WSSG as an control metric. Finally Chapter 4 will give conclusions and suggestions for future work.

CHAPTER 2. WSSG FOR CLAMPED FLAT PLATES

Note: This chapter is currently being prepared as a journal article for publication in the Journal of the Acoustical Society of America under the title “Active structural acoustic control of clamped flat plates using a weighted sum of spatial gradients” with coauthors Daniel R. Hendricks, Pegah Aslani, Jonathan D. Blotter, Scott D. Sommerfeldt, and Kent L. Gee.

Abstract

In this paper, the weighted sum of spatial gradients, or WSSG (formerly known as composite velocity) is developed for use in active structural acoustic control (ASAC) on a clamped flat rectangular plate. In previous work, WSSG was developed on a simply-supported flat rectangular plate [1], and showed promise as an control metric. The displacement on the clamped plate has been modeled using an approximate analytical solution assuming shape functions corresponding to clamped-clamped beams. From the analytical formulation, weights, which were found to be the reciprocal of the wave number squared, have been derived to produce a uniform WSSG field across the plate. In active control simulations this quantity has been found to provide comparable control to volume velocity. Sensitivity analysis has also shown that comparable control, regardless of the sensor location, can be achieved using WSSG as a control metric. Experimental results have demonstrated that this control metric works effectively in practice, with results similar to the simulations. The results show that WSSG can be used as an effective control metric on clamped rectangular plates.

2.1 Introduction

Active structural acoustic control (ASAC) is an important subfield of active noise control (ANC). In ASAC, sound fields radiated by structures are controlled by using a control actuator applied to the vibrating structure as opposed to being placed in the sound field [11]. As research in ANC increased in the 1980's it was noted that for sound control of vibrating plates the number of actuators (speakers) in a sound field required to achieve global control was frequency dependent [3]. It was also noted alternatively that only one or two control forces applied to the structure [7], as opposed to multiple actuators placed in the sound field, could provide global sound reduction. This led to the suggestion of using an active control algorithm to control actuators on the structure [21] for improved control of global sound radiation and simplification of the control hardware.

Initial attempts at ASAC used error microphones placed in the radiated sound field. By moving the actuator to the surface of the plate significant global reduction, by as much as 40 dB in some cases, was achieved while avoiding the problem of an increasing number of actuators in the sound field. It was shown that only one to two actuators were needed in order to provide control regardless of the frequency of excitation. This was due to the fact that only one to two structural modes seemed to contribute to radiated sound at a given frequency [6].

Another important step for ASAC was the development of the radiation resistance matrix method which provided a simple way to calculate the radiated sound power of a distributed system. One of the key analytical tools in the development of ASAC was derived from this, namely the acoustical or radiation modes of a structure, which are the eigenvectors of the radiation resistance matrix. The radiation modes are a set of orthogonal modes which radiate sound power independently [11]. Each radiation mode is composed of several contributing structural modes and can be used to show which structural modes are significant in the radiation of sound power [9] [12].

For rectangular plates it was shown that for values of $kl < 0.5$, where k is the wave number and l is the length of the longer side of the plate, the first radiation mode is the only radiation mode which radiates substantially. The shape of this first mode is comparable to volume velocity and suggested that using an error sensor to measure volume velocity on the plate would be an effective way to control radiated sound in the sound field [9]. This was significant in that it removed the error sensor as well as the control actuator from the sound field and placed them both on the plate, thus allowing for control of a sound field while keeping it clear of measurement or control devices.

With this idea in mind, sensing techniques for volume velocity were investigated. Two of the methods pursued were accelerometer arrays [14] and distributed piezoelectric films [15]. An accelerometer array can give an accurate estimate of the volume velocity of the plate. However, it requires a large number of sensors to be implemented. For example, on a plate with dimensions of 0.278 m x 0.247 m it was predicted that 16 to 25 accelerometers would be needed to effectively measure the volume velocity [14]. With larger plates this number increases. Making a measurement in this manner does not reduce the complexity of the hardware required for control, thus eliminating one of the potential benefits of ASAC.

A distributed piezoelectric film avoids the necessity of a large number of error sensors to sense volume velocity, requiring only a single piezoelectric patch. In this case the difficulty is not in the number of sensors required but in the fact that the sensor needs to be designed specifically for the geometry in question. For a plate of size 0.314 m x 0.414 m a PVDF film was used with dimensions of 0.306 m x 0.406 m [15]. For a different size plate an entirely different sensor would need to be designed and built. Thus while effective, this method is highly geometry dependent and difficult to implement.

Other methods were developed for sensing radiation mode shapes, such as optimally shaped piezoelectric films designed to target the specific structural vibration modes contributing significantly to the important acoustic radiation modes [16]. This allowed for the first radiation mode to be targeted specifically as well as several other higher but significantly radiating modes to be targeted. Structural energy methods have also been investigated [19], but with limited success.

Optimally shaped piezoelectric films have the benefit of targeting the radiation modes of a structure more directly. Because only a few radiation modes contribute to sound radiation at low frequencies, and only a few structural modes contribute significantly to each radiation mode this method requires a smaller number of sensors than volume velocity. If the first radiation mode were the only mode of interest (also the target of volume velocity), three to four sensors would be required, depending on the frequency, to sense the contributing structural modes. By using 6 - 8 sensors, this method could also target multiple radiation modes [16].

On the other hand this method, like the distributed piezoelectric film, is highly dependent on the geometry. In order to sense specific structural modes the piezoelectric sensors need to be shaped appropriately. However, significant analysis of the structural and radiation modes is required to determine which modes are important and the required sensor shapes.

As previously described, many of the suggested methods of sensing for ASAC have had drawbacks due to the large number of sensors or sensor dependency on geometry. A new structural quantity, composite velocity, or V_{comp}^2, has recently been developed which, when sensed, provides similar control while avoiding these problems [1].

Composite velocity has been shown to be a near uniform structural quantity on a vibrating, simply supported, rectangular plate. This uniformity allows for insensitivity to sensor location, reducing the analysis required in choosing an appropriate sensing location, a difficulty encountered when making other point measurements of vibration. The quantity itself, which is a sum of weighted spatial gradients, can be measured using an array of only four accelerometers, and has been shown to provide similar control to other standard ASAC control metrics such as volume velocity. The purpose of this paper is to derive composite velocity (referred to as the weighted sum of spatial gradients, or WSSG in this paper) for a rectangular plate with clamped boundary conditions and to demonstrate its feasibility as a simpler, more robust, yet effective control metric for use in ASAC.

2.2 Development

The governing equation for a vibrating plate is given by

$$D\left(\frac{\partial^4}{\partial x^4}+2\frac{\partial^4}{\partial x^2\partial y^2}+\frac{\partial^4}{\partial y^4}\right)w(x,y)-\bar{m}\omega^2 w(x,y)=F(x,y), \tag{2.1}$$

where D is the bending stiffness, w is the displacement, ω is the excitation frequency, and $\bar{m}$ is the mass per unit area [22]. The plate excitation is assumed to be time harmonic, the displacement and force expressions can be assumed to be multiplied by $e^{j\omega t}$ although not stated explicitly. This is an assumption made throughout the remainder of this paper. Also, for the current formulation the plate will be assumed to be excited by a series of point forces, so that $F(x,y)=\sum_q F_q\delta(x-x_q)\delta(y-y_q)$, with F_q indicating the qth applied force as a complex number, $\delta(\cdot)$ the Dirac delta function, and x_q and y_q the location of the qth force. This expression models the shakers used to provide the experimental disturbance and control.

Although an exact analytical solution to Eq. (2.1) for clamped rectangular plates is not available, a method assuming the product of beam mode shapes as the eigenfunctions of the plate can be used to provide an approximate analytical solution [18] [23], written as

$$w(x,y) = \sum_q \sum_m^\infty \sum_n^\infty \frac{F_q \phi_m(x_q)\phi_n(y_q)\phi_m(x)\phi_n(y)}{D(I_1 I_2 + 2I_3 I_4 + I_5 I_6) - \bar{m}\omega^2 I_2 I_6}, \tag{2.2}$$

where

$$\phi_p(\xi) = C_1\left(\frac{\lambda_p \xi}{L_\xi}\right) - \frac{C_1(\lambda_p)}{S_1(\lambda_p)} S_1\left(\frac{\lambda_p \xi}{L_\xi}\right) \tag{2.3}$$

$$C_1(\xi) = cosh(\xi) - cos(\xi)$$

$$S_1(\xi) = sinh(\xi) - sin(\xi)$$

and

$$\begin{aligned} I_1 &= \int_0^{L_x} \phi_m^{(4)}(x)\phi_m(x)dx \qquad & I_2 &= \int_0^{L_y} \phi_n^2(y)dy \\ I_3 &= \int_0^{L_x} \phi_m''(x)\phi_m(x)dx \qquad & I_4 &= \int_0^{L_y} \phi_n''(y)\phi_n(y)dy \\ I_5 &= \int_0^{L_y} \phi_n^{(4)}(y)\phi_n(y)dy \qquad & I_6 &= \int_0^{L_x} \phi_m^2(x)dx. \end{aligned}$$

The subscripts *m* and *n* indicate structural mode numbers, and the values for λ_p, the eigenvalues of the system, are given by the characteristic equation

$$cosh(\lambda_p)cos(\lambda_p) = 1. \tag{2.4}$$

$\phi(\cdot)$ is the assumed shape function. The typical geometry and application of forces can be seen in Fig. 2.1. This method has been verified and shown to predict accurately the natural frequencies, when compared to the Rayleigh-Ritz method, and experimentally, for a clamped rectangular plate [18].

2.2.1 Derivation of the Control Metric for Clamped Plates

Previously, composite velocity was defined [1] as

$$V_{comp}^2 = \alpha\left(\frac{\partial w}{\partial t}\right)^2 + \beta\left(\frac{\partial^2 w}{\partial x \partial t}\right)^2 + \gamma\left(\frac{\partial^2 w}{\partial y \partial t}\right)^2 + \delta\left(\frac{\partial^3 w}{\partial x \partial y \partial t}\right)^2, \tag{2.5}$$

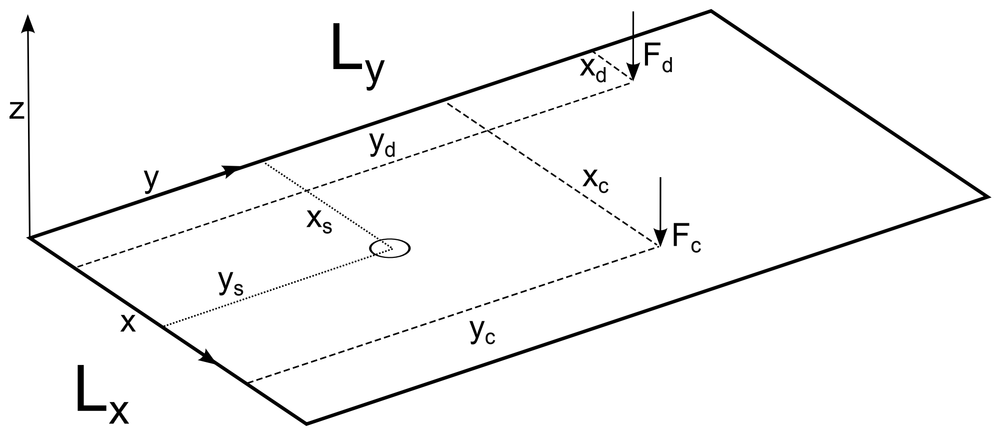


Figure 2.1: Coordinate system and force application on flat plate

where the choice of weights $\alpha, \beta, \gamma,$ and δ were important in deriving a uniform quantity for the simply-supported rectangular plate. However, it may be noted that due to the assumption of time harmonic vibration, the time derivative causes each of the terms to be scaled by a constant, $(j\omega)^2$, which has no effect on the algorithm's ability to provide control using this metric. In this formulation the time derivative will be removed, making 'composite velocity' a misnomer. The new formulation is given as

$$\mathrm{WSSG} = \alpha\,(w)^2 + \beta\left(\frac{\partial w}{\partial x}\right)^2 + \gamma\left(\frac{\partial w}{\partial y}\right)^2 + \delta\left(\frac{\partial^2 w}{\partial x \partial y}\right)^2, \tag{2.6}$$

where WSSG stands for the weighted sum of spatial gradients.

While Eqs (2.5) and (2.6) are fundamentally the same with regards to control for both the time harmonic simply-supported and clamped plates the choice of weights changes with the boundary conditions and is dependent on the eigenvalues of the solution to Eq. (2.1). To determine the weights, the derivatives of Eq. (2.2) with respect to *x* and *y*, as shown by Eqs. (2.7) – (2.9), are used:

$$\frac{\partial w}{\partial x} = \sum_q \sum_m^\infty \sum_n^\infty \frac{F_q \phi_m(x_q)\phi_n(y_q)\phi_m'(x)\phi_n(y)}{D(I_1 I_2 + 2I_3 I_4 + I_5 I_6) - \bar{m}\omega^2 I_2 I_6} \tag{2.7}$$

$$\frac{\partial w}{\partial y} = \sum_q \sum_m^\infty \sum_n^\infty \frac{F_q \phi_m(x_q)\phi_n(y_q)\phi_m(x)\phi_n'(y)}{D(I_1 I_2 + 2I_3 I_4 + I_5 I_6) - \bar{m}\omega^2 I_2 I_6} \tag{2.8}$$

and

$$\frac{\partial^2 w}{\partial x \partial y} = \sum_q \sum_m^\infty \sum_n^\infty \frac{F_q \phi_m(x_q) \phi_n(y_q) \phi_m'(x) \phi_n'(y)}{D(I_1 I_2 + 2 I_3 I_4 + I_5 I_6) - \bar{m}\omega^2 I_2 I_6} \tag{2.9}$$

where

$$\phi_m'(x) = \frac{\lambda_m}{L_x}\left[S_2\left(\frac{\lambda_m x}{L_x}\right) - \frac{C_1(\lambda_m)}{S_1(\lambda_m)} C_1\left(\frac{\lambda_m x}{L_x}\right)\right] \tag{2.10}$$

$$\phi_n'(y) = \frac{\lambda_n}{L_y}\left[S_2\left(\frac{\lambda_n y}{L_y}\right) - \frac{C_1(\lambda_n)}{S_1(\lambda_n)} C_1\left(\frac{\lambda_n y}{L_y}\right)\right] \tag{2.11}$$

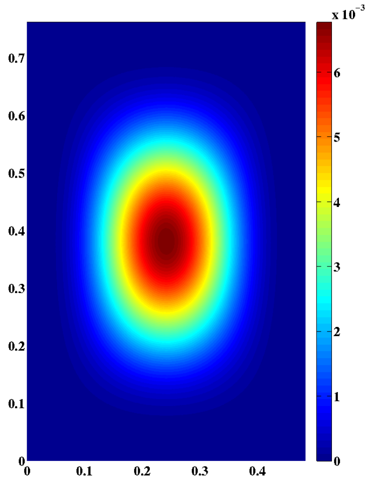
and

$$S_2(q) = sinh(q) + sin(q).$$

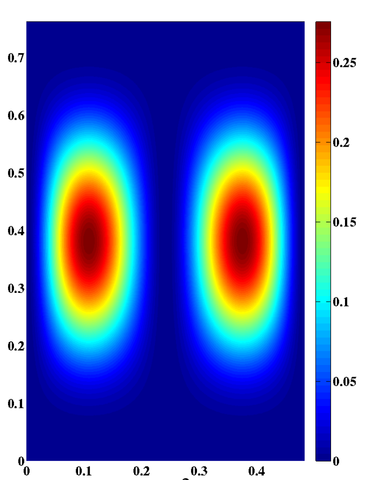
Each of the WSSG terms for the first mode can be seen plotted in Fig. 2.2. Note how the first term has maximum values in the middle, the x and y terms have maximum values on the outside edges, and the cross derivative term has maximum values on the corners. By summing these terms, with the weights chosen to normalize the maximum values, a uniform field can be created.

Terms in Eq. (2.7) are scaled by a factor of λ_m/L_x when the derivative is taken, as shown by Eq. (2.10). Likewise, Eqs. (2.8) and (2.9) are scaled by factors of λ_n/L_y and $\lambda_m\lambda_n/L_xL_y$ as shown by Eqs. (2.10) and (2.11). These scaling factors are responsible for the difference in magnitudes of the terms. To create a uniform WSSG field, the inverse of these terms are chosen and squared to form the weights. These inverse expressions, as the clamped plate weights, are given on the first line of Table 2.1. Note that each of the weights are scaled by α. There is no specific value defined for α, which can be used to scale all the terms by an equal amount. Normally the value of α is set to unity, however if the value were set so that $\alpha = (j\omega)^2$, this formulation for WSSG becomes equivalent to Eq. (2.5), composite velocity.

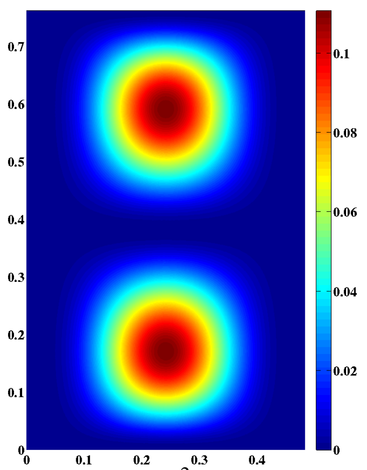
An alternative way to define the weights is in terms of the wave number. The wave numbers are inherent in the shape functions described by Eq. (2.3). For the clamped plate the wave number is $k = \lambda/L$, which can be substituted into the weights equations in the first line of the Table 2.1 to yield the general weights expressions shown on the second line of Table 2.1. For the case of the simply-supported plate $k_m = m\pi/L_x$ which yields the correct weight expression $\beta_m = \left(\frac{L_x}{m\pi}\right)^2$ [1]. This can be demonstrated in a similar manner for γ_n and δ_{mn}, verifying that the general weight expressions are independent of boundary conditions and reduce down to the correct weights when the wave numbers, corresponding to the chosen boundary condition, are used. This general



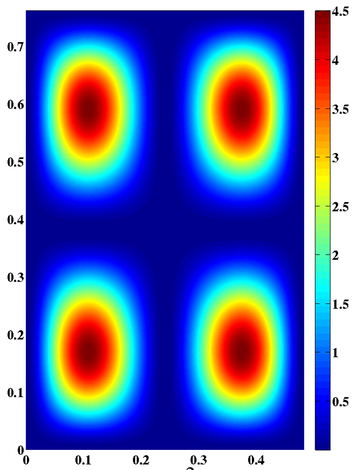
(a) w^2 term



(b) $\left(\frac{\partial w}{\partial x}\right)^2$ term



(c) $\left(\frac{\partial w}{\partial y}\right)^2$ term



(d) $\left(\frac{\partial^2 w}{\partial x \partial y}\right)^2$ term

Figure 2.2: WSSG terms for the clamped flat plate first mode

expression is significant because it shows that each term is weighted by the corresponding wave number components, and allows for the selection of weights on plates with non-ideal shapes if the wave number components in x and y can be measured.

At this point the derived weights are only valid for individual mode numbers m and n, while the total displacement given in Eq. (2.2) is a sum over an infinite number of modes. While individual modes can be weighted analytically, this cannot be done experimentally, as the contributions to displacement from each of the modes cannot be separated in real time. Several different weighting schemes have been investigated for the non-mode specific weights with very similar results in

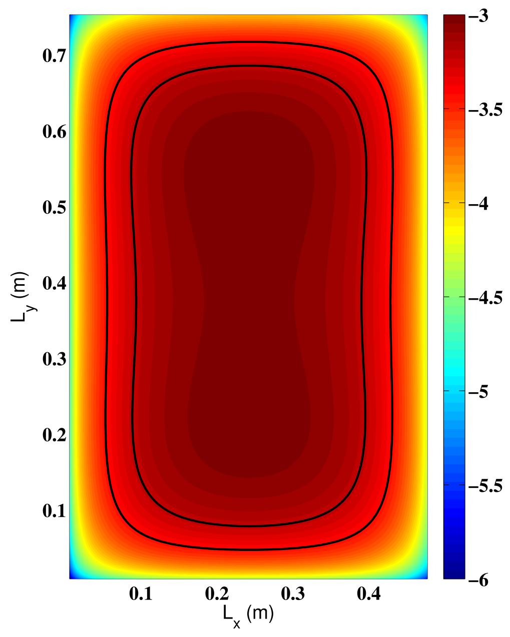
Table 2.1: Expressions for the weights in WSSG. The average weights shown were calculated using the geometry described in Table 2.2.

Quantity	$\partial w/\partial x$	$\partial w/\partial y$	$\partial^2 w/\partial x\partial y$
Clamped Weights	$\beta_m = \alpha\left(\frac{L_x}{\lambda_m}\right)^2$	$\gamma_n = \alpha\left(\frac{L_y}{\lambda_n}\right)^2$	$\delta_{mn} = \alpha\left(\frac{L_x L_y}{\lambda_m \lambda_n}\right)^2$
General Weights	$\beta_m = \alpha\left(\frac{1}{k_m}\right)^2$	$\gamma_n = \alpha\left(\frac{1}{k_n}\right)^2$	$\delta_{mn} = \alpha\left(\frac{1}{k_m k_n}\right)^2$
Avg. Weights	$\beta_{ave} = 0.00595$	$\gamma_{ave} = 0.00897$	$\delta_{ave} = 4.92x10^{-5}$

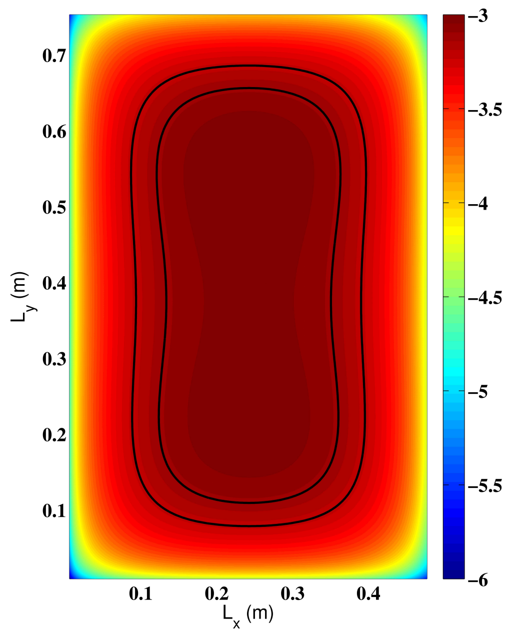
simulated control. It can be concluded that the control metric is not overly sensitive to the general weighting scheme as long as they are of the appropriate order of magnitude. The simplest method used was averaging the β_m, γ_n, and δ_{mn} values over the first 15 modes of the geometry examined.

2.2.2 Uniformity of WSSG Field

With the weights chosen as described above, the uniformity of the WSSG field on the clamped plate can be demonstrated. Using the equations shown in Table 2.1, and the plate geometry and material properties defined in Table 2.2, the WSSG field for the 1,2 mode is found by exciting the plate at the 1,2 mode natural frequency, and calculating the weights for and summing over only the 1,2 mode in Eqs. (2.2), (2.7), (2.8), and (2.9). This field is shown in Fig. 2.3(a), in decibels relative to the maximum value on the plate, to demonstrate uniformity. The inner black contour represents the boundary of the 3 dB down region, and contains approximately 50% of the plate. The outer black contour represents the boundary of the 6 dB down region, and contains approximately 66% of the plate. The standard deviation of the WSSG field is 6.7 dB. The simply-supported plate, used in previous work [1] was found to be uniform to within $\pm 0.01\%$ at the first mode. The clamped boundary conditions, and therefore the introduction of hyperbolic sines and cosines in the shape functions, are the cause of this difference in uniformity.



(a) WSSG field for the clamped plate 1,2 mode.



(b) WSSG field summed over the first 15 modes.

Figure 2.3: Examples of the WSSG field on the plate, in dB, relative to the maximum value on the plate, excited at the 1, 2 mode natural frequency. The inner black line represents the 3 dB down contour and the outer the 6 dB down.

A more practical case occurs when WSSG is summed over multiple modes (in Eqs. (2.2), (2.7), (2.8), and (2.9)), more closely modeling the displacement of an actual plate when it is excited. In this case the average weight values shown in Table 2.1 are used. The WSSG field summed over the first 15 modes using the average weights is shown in Fig. 2.3(b). In this case the 3 dB down region contains approximately 31% of the plate, and the 6 dB down region contains approximately 50% of the plate, so with more modes included the WSSG field becomes slightly less uniform. The standard deviation of the WSSG field in this case is 7.2 dB, a little higher than for a single mode. For the simply-supported plate in this situation, using average weights, excited at the 1,2 mode natural frequency, the 3 dB down region contained about 40% of the plate, and the 6 dB down region contained about 80% of the plate, once again more uniform due to the difference in boundary conditions and the resulting shape functions.

When WSSG is summed over multiple modes the uniformity decreases, due to the contributions from other modes to the displacement of the plate. Also, the field on the clamped plate is less uniform than the WSSG field on the simply-supported plate [1] due to the difference in boundary conditions. The clamped boundary prevents the plate from rotating at the edges, therefore driving the derivative terms to zero at the boundaries of the plate, which can be seen in Figs. 2.3(a) and 2.3(b). At higher modes and frequencies this transition area decreases. These results suggest that sensors ought to be placed away from the edges of the plate. Another important point to note is that while Figs. 2.3(a) and 2.3(b) show the WSSG field at the 1,2 natural frequency, there are no nodal lines in the field. In general nodal lines in the WSSG field do not appear regardless of the mode or frequency of excitation. This property significantly simplifies the choice of sensor location, reducing the importance of the exact location.

2.2.3 Comparison to Radiation Modes

One of the guiding factors in the original development of WSSG [1] was the relationship between the radiation mode shapes and the shape of the spatial derivatives used. The radiation modes of a plate are derived from the radiation resistance matrix. The radiation resistance matrix discretizes the plate geometry and provides a simplified and computationally more efficient method for calculating the radiated sound power at low frequencies. It is derived by dividing the plate into elemental radiators and calculating the mutual radiation impedances between each elemental radiator and all others. This matrix is given, for a rectangular plate [11], by

$$\mathbf{R} = \frac{\omega^2 \rho_0 A_e^2}{4\pi c} \begin{bmatrix} 1 & \frac{sin(kR_{12})}{kR_{12}} & \cdots & \frac{sin(kR_{1N})}{kR_{1N}} \\ \frac{sin(kR_{21})}{kR_{21}} & 1 & & \vdots \\ \vdots & & \ddots & \\ \frac{sin(kR_{N1})}{kR_{N1}} & \cdots & & 1 \end{bmatrix} \tag{2.12}$$

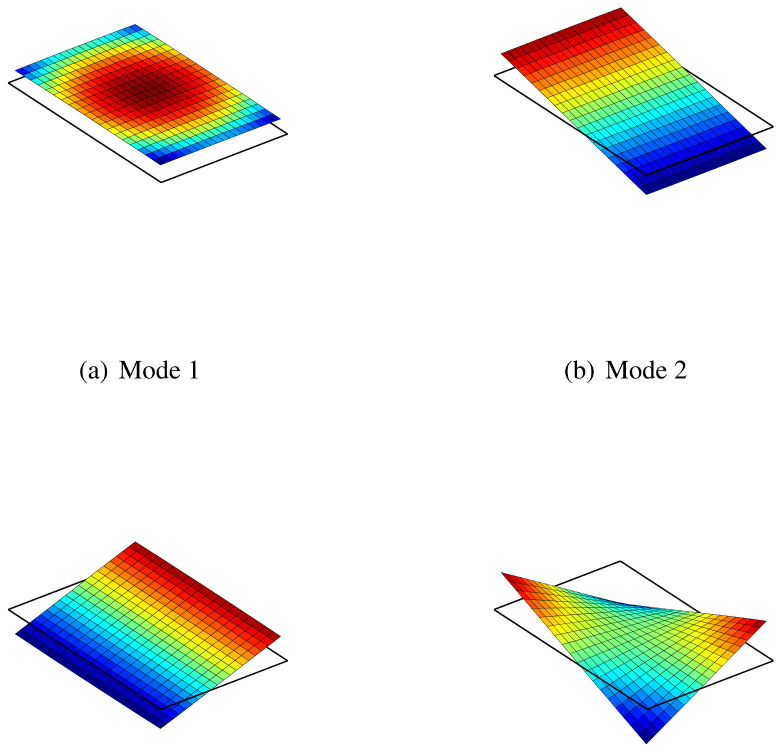
where ρ_0 is the density of the surrounding fluid, A_e is the elemental area, c is the speed of sound in the fluid, R_{ij} is the distance from the ith to the jth element, and N is the total number of elements. Using the radiation resistance matrix, the sound power is calculated [9] as

$$P = \mathbf{v_e}^H \mathbf{R} \mathbf{v_e} \tag{2.13}$$

where $\mathbf{v_e}$ is a vector containing the velocity of each element, and H is the Hermitian transpose.

The radiation mode shapes are found as the eigenvectors of the radiation resistance matrix, and break the matrix up into a set of mutually orthogonal vectors which reveal the underlying radiation mechanisms of the structure. The radiation mode shapes, which are a function of geometry and frequency, but not boundary conditions, generally only have a few of the structural modes contributing significantly, thus by targeting the radiation modes the relevant radiating structural modes can be targeted specifically. At low frequencies, it has been found that only a few of the radiation modes have any significant contribution to sound radiation.

The first four radiation modes of the rectangular plate are shown in Fig. 2.4 and the four terms of WSSG are shown in Fig. 2.2. Note the similarity between these four terms. The first radi-



(a) Mode 1

(b) Mode 2

(c) Mode 3

(d) Mode 4

Figure 2.4: The first four radiation mode shapes for a rectangular plate.

ation mode is similar in shape to the displacement, *w*, which is comparable to a monopole source; the second and third radiation modes are similar to the derivatives with respect to *x* and *y*, and can be compared to dipole like radiation; the fourth radiation mode is similar to the cross derivative, and is similar to quadrupole like radiation. Although these comparisons are made loosely, and are not exact, on the first mode of the clamped plate, at more complicated structural modes the derivative terms locally very closely resemble the radiation mode shapes. This similarity allows for a local measurement on the plate to control the radiation mode shapes globally, as opposed to volume velocity, which required a global measurement of the plate to control the first radiation mode. These similarities, which were also found for the simply supported plate, suggest that WSSG will be effective at targeting the first four radiation mode shapes for a clamped plate, and will therefore be a good candidate for use in ASAC.

Because of the way in which the weights are calculated, a uniform WSSG field is created. The uniformity of the field is considered an advantage as far as sensitivity to sensor placement is concerned, however it can be viewed in light of weighting the radiation modes. By weighting each of the terms the same, each of the radiation modes is given the same weight in the control algorithm, thus targeting each of the first four radiation modes equally.

2.3 Numerical Simulations

(2.13). To demonstrate the effectiveness of WSSG, control was simulated on a clamped plate. The plate dimensions and material properties for the simulation are given in Table 2.2.

Table 2.2: Plate properties and dimensions for control simulations.

Property	Value
Length in x (L_x)	0.483 m
Length in y (L_y)	0.762 m
Thickness	0.0031 m
Young's Modulus	69.8 GPa
Poisson's Ratio	0.33
Density	2700 kg/m^3
Damping ratio	2%

Equations (2.2), (2.6), (2.7), (2.8), and (2.9) were used as the analytical model for the plate. As stated previously point forces were used to excite and control the plate. The disturbance force was applied at the location $x_d = 0.083$ m, $y_d = 0.629$ m and the control force was applied at the location $x_c = 0.083$ m, $y_c = 0.127$ m with WSSG measured at $x_s = 0.286$ m, $y_s = 0.432$ m. The control was simulated by finding the optimal control force for each frequency to minimize WSSG at the sensor location. The radiated sound power was then calculated by using elementary radiators as described in Eq.

In order to demonstrate the effectiveness of WSSG it was compared to volume velocity, the most commonly used control metric for ASAC. For the analytical simulation volume velocity was found by discretizing the plate into 60 elements and estimating the velocity across each individual element as the velocity at the center of the element. This number of elements was calculated as sufficient to give an accurate measure of volume velocity based on the methods described by Sors and Elliott [14].

For the simulated control, shown in Fig. 2.5, the overall sound attenuation over the frequency range from $0-630$ Hz is 28.8 dB with WSSG as the control metric, and 9.0 dB with

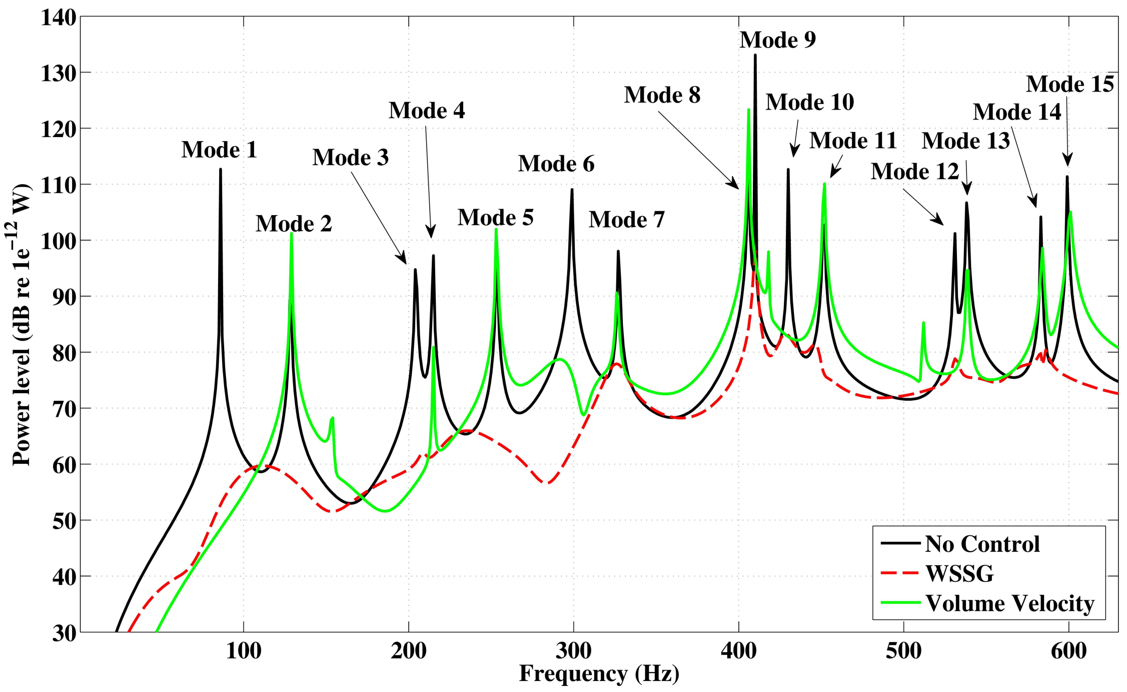


Figure 2.5: Simulated control of the clamped plate using WSSG and volume velocity as control metrics.

volume velocity as the control metric. After the fourth structural mode WSSG consistently provides better control than volume velocity over the frequency spectrum for the case examined.

The simulated control at each of the natural frequencies is shown in Fig. 2.6. It can be easily seen from this that at most of the natural frequencies WSSG outperforms volume velocity. The success of WSSG, especially at higher modes, is due to its ability to target the first four, as opposed to just the first, radiation modes. The radiated sound power from each of the first four radiation mode shapes is shown in Figs. 2.7, 2.8, 2.9, and 2.10. The overall sound attenuation for each of these modes is given in Table 2.3. Although volume velocity provides comparable control to WSSG at the first mode, WSSG significantly outperforms volume velocity on the other three modes, as expected from the theoretical development of the control metric.

There are several things to note in Figs. 2.7 – 2.10. Each of the radiation modes has a different set of structural modes contributing to their sound radiation. For example, in Fig. 2.7 the first, third and ninth structural modes radiate strongly, but the second, fourth and fifth do not. In the case of the fifth structural mode there is no apparent contribution to the sound radiation of

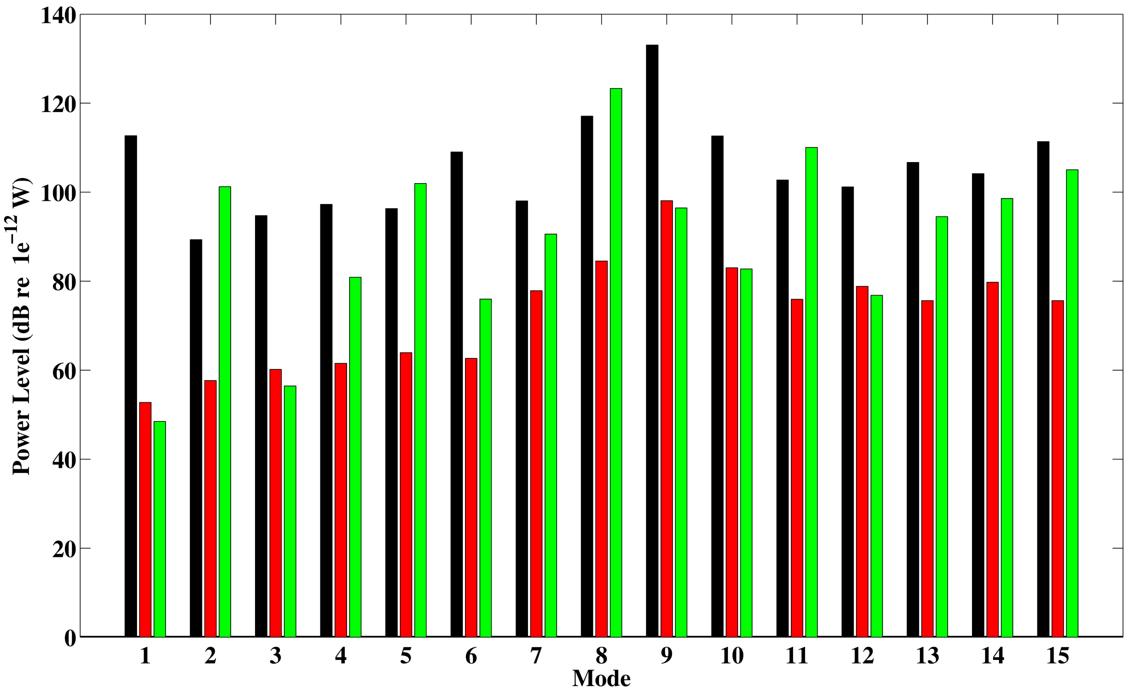


Figure 2.6: Simulated control sound power levels at the structural mode natural frequencies. Black is the radiated sound level without control, red represents the radiated sound level with control from WSSG, and green represents the radiated sound level with control from volume velocity.

Table 2.3: Overall sound attenuation in each of the four radiation modes targeted by WSSG, as shown in Figs. 2.5 – 2.10.

Mode	WSSG	Vol. Vel.
1	32.1	27
2	19.3	-0.2
3	16.0	9.0
4	26.6	16
Overall	28.8	9.0

the first radiation mode. Modes that don't appear to radiate sound power were still marked on the figures.

For the first mode volume velocity provides better or comparable control to WSSG over the frequency range, except for the 14th and 15th modes. The poor control of volume velocity and the good control of WSSG at these two modes caused WSSG to provide a better overall sound attenuation for the first radiation mode. For the other three radiation modes examined, WSSG outperformed volume velocity because it provides comparable or improved control at the signif-

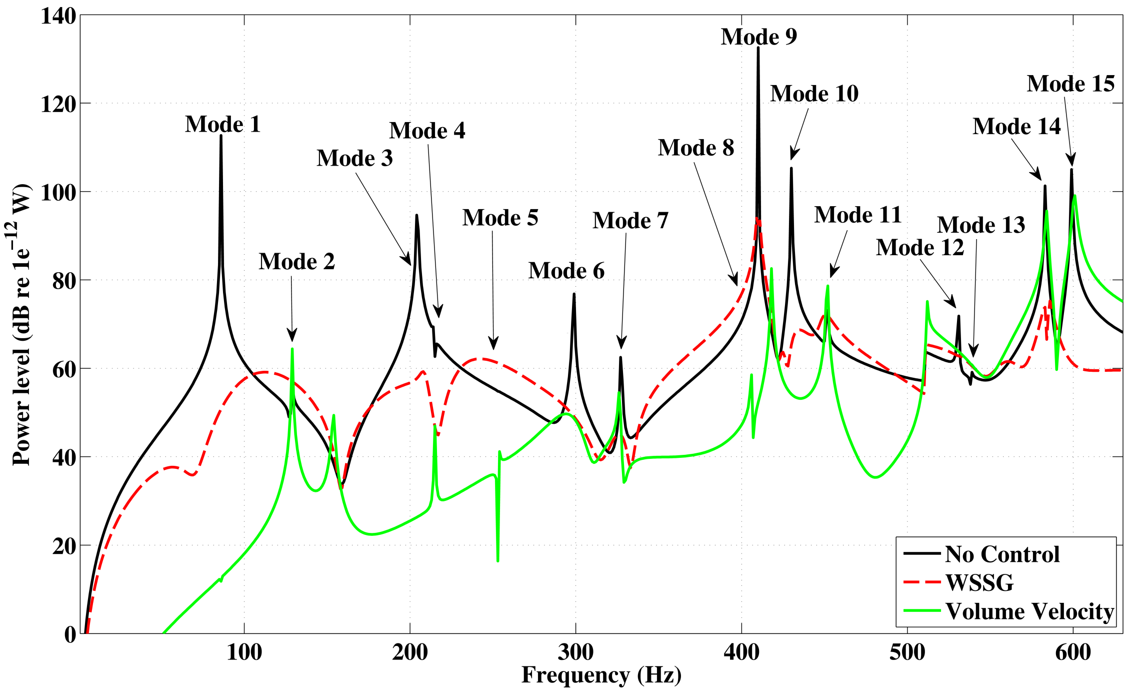


Figure 2.7: Sound radiated by the first radiation mode on the clamped rectangular plate.

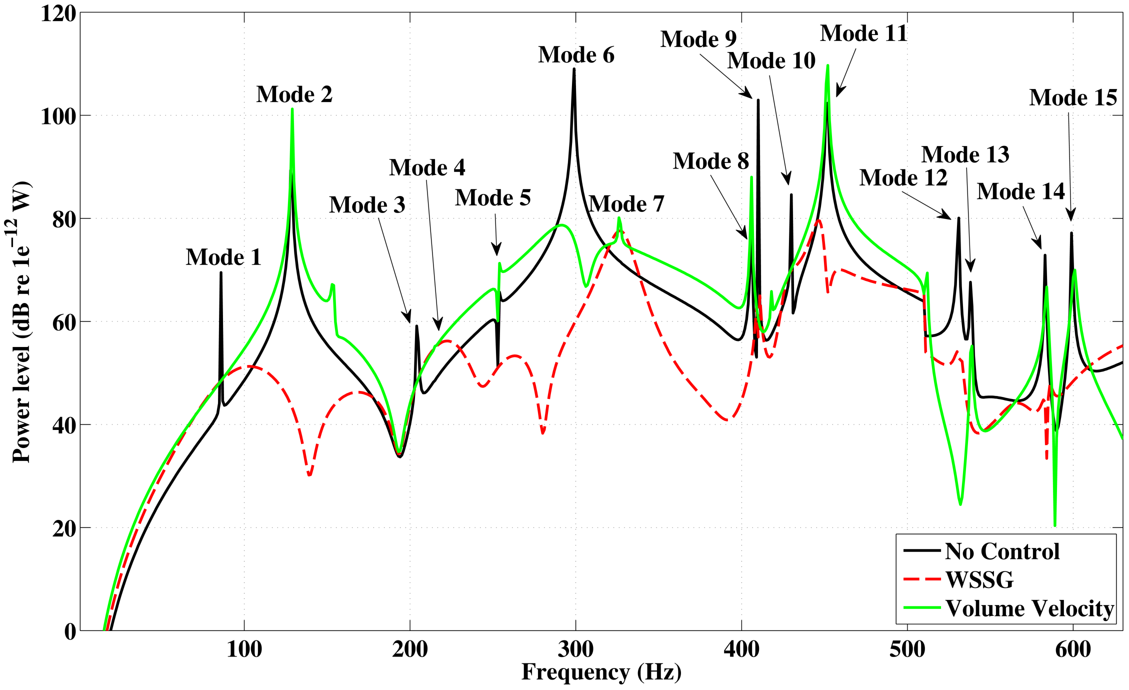


Figure 2.8: Sound radiated by the second radiation mode on the clamped rectangular plate.

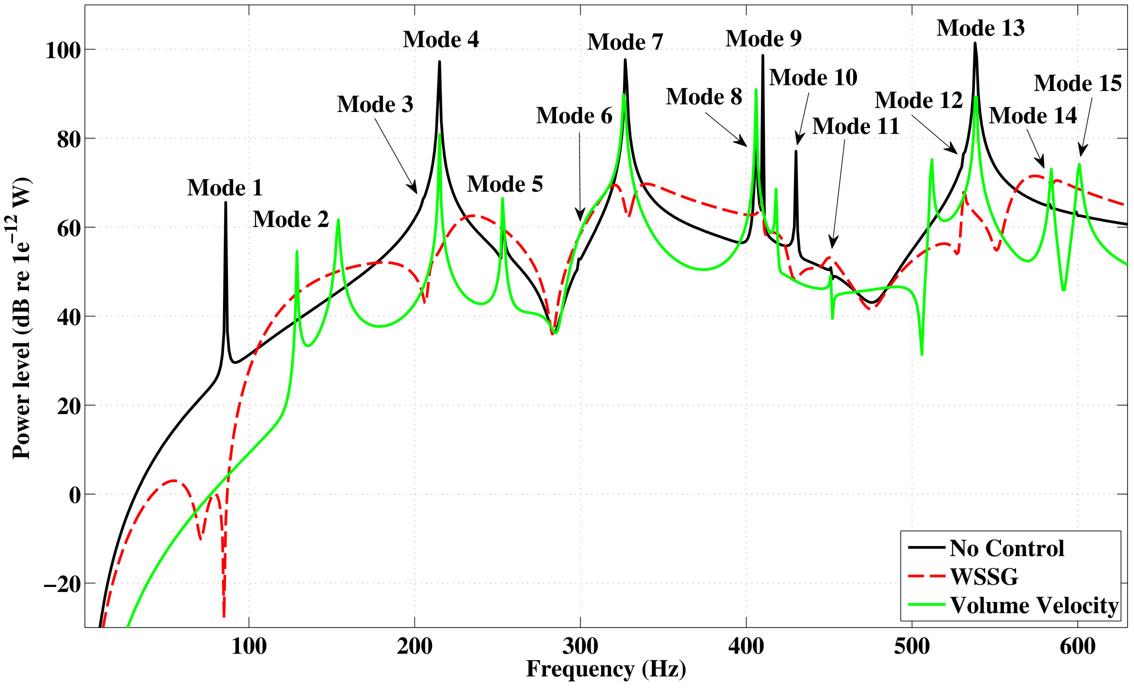


Figure 2.9: Sound radiated by the third radiation mode on the clamped rectangular plate.

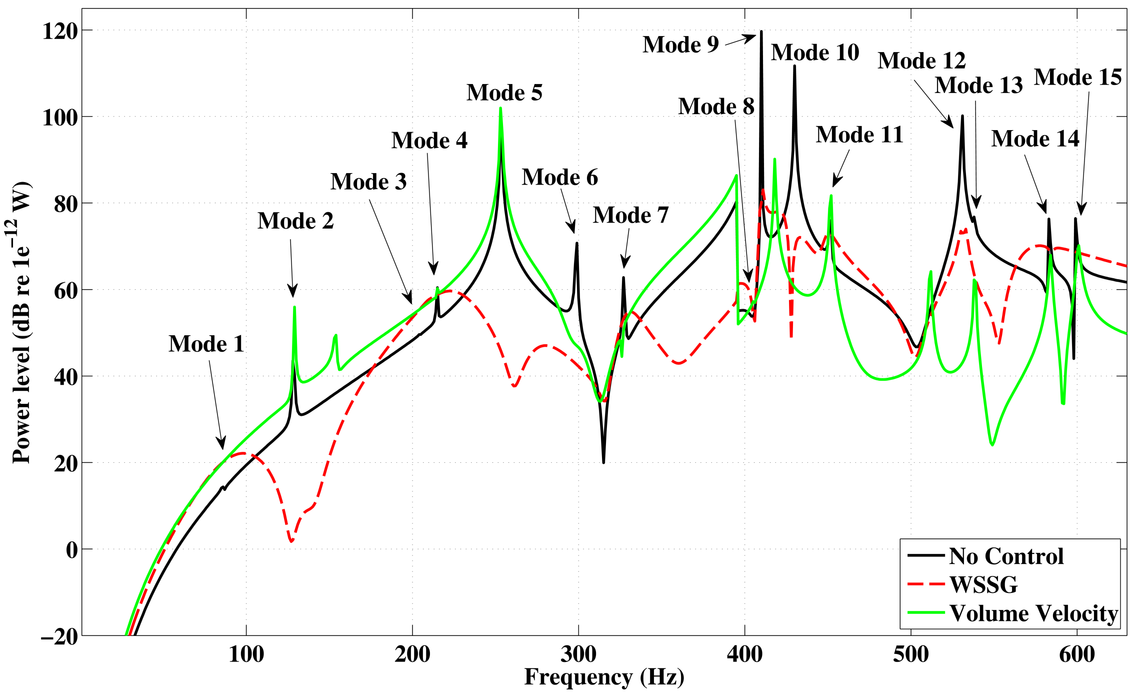


Figure 2.10: Sound radiated by the fourth radiation mode on the clamped rectangular plate.

icantly radiating natural frequencies. Specifically, in Fig. 2.5 volume velocity does not provide control at modes 2, 5, 7, 8, 11, 13, 14, or 15. With the exception of modes 14 and 15, which noticeably contribute to the radiated sound of multiple radiation modes, these structural modes contribute radiated sound through either the second, third, or fourth radiation mode, modes which volume velocity was not designed to target. It is therefore unable to provide consistent control at these structural mode natural frequencies. These results show that WSSG does successfully target the first four radiation modes, allowing WSSG to provide a better overall sound attenuation than volume velocity.

Control of the plate with the sensor placed at random locations was also simulated to determine control sensitivity to sensor location. Five locations, given in Table 2.4 were chosen at random on the plate, and then control was simulated at those locations, along with the original location. The overall attenuation for each sensor location is also shown in Table 2.4. A plot showing the control at these various locations is given in Fig. 2.11. Each of the structural modes is successfully controlled for radiated sound power, although the ability to control mode 9 is reduced when compared to the other modes. This is due to the choice of control force location for these simulations.

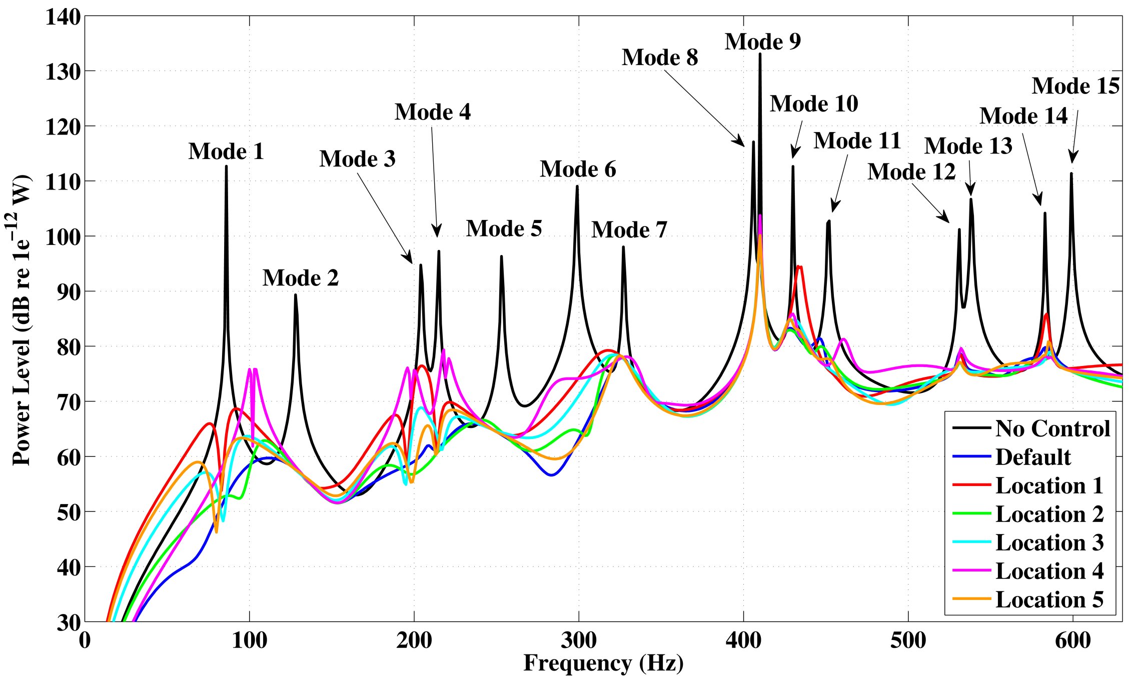


Figure 2.11: Simulated control of the clamped plate at multiple sensor locations.

Note the similarity in control regardless of the sensor location. The mean of the overall sound attenuation at these sensor locations is 27.5 dB with a standard deviation of 1.4 dB, showing a high degree of consistency, and thus verifying that the uniformity of the field, as discussed in section 2.2.2, does result in control insensitivity to sensor location.

Table 2.4: The sensor locations used to test the sensitivity of WSSG to sensor placement on the plate of dimensions 0.483 m x 0.762 m.

Sensor	x (m)	y (m)	Overall Sound Atten. (dB)
Default	0.286	0.432	28.8
1	0.088	0.566	25.4
2	0.193	0.299	28.6
3	0.126	0.499	27.9
4	0.386	0.130	26.1
5	0.208	0.538	28.0

2.4 Conclusions

A weighted sum of spatial gradients control metric (WSSG) has been developed for a clamped flat rectangular plate following the method used previously to develop composite velocity on a simply-supported flat rectangular plate. For a rectangular plate clamped on all four sides an exact analytical solution is not available, therefore an approximate solution assuming that the shape functions for the plate were the product of clamped-clamped beam shape functions, has been used. This solution was used to derive analytical expressions for the weights in WSSG, which caused the field to become relatively uniform. The development of WSSG with the choice of weights was intended to create a measurement quantity which was insensitive to sensor location, would target and control the first four radiation mode shapes, and could be measured in a much simpler manner experimentally than previous control metrics.

Simulated results verified the hypothesized effects. While only measuring the displacement at four locations on the plate, improved control over volume velocity (which required 16 - 25 accelerometers, or geometry dependent sensors) was achieved. In particular the control metric achieved improved control when compared to volume velocity at natural frequencies and modes higher than the fourth mode. This was due to the fact that WSSG could control the second, third, and fourth radiation modes whereas volume velocity could not. The second, third, and fourth radiation modes have a greater contribution from structural modes with higher natural frequencies when compared to the first radiation mode, accounting for this improved control.

Also, the WSSG measurement sensor location has been found to be relatively unimportant in simulations. This has been shown by placing the sensor in random locations and comparing the control using those locations. Control was most consistent above the fourth mode natural frequency, where more of the targeted radiation mode shapes become relevant.

From the observed effects it can be concluded that WSSG, which provides comparable control to volume velocity, has several advantages in implementation. It requires fewer sensors, its physical measurement location is geometry independent, it works on multiple boundary conditions (demonstrated on simply-supported [1] and clamped), and provides improved control over volume velocity.

CHAPTER 3. WSSG FOR RIBBED PLATES

Note: This chapter is currently being prepared as a journal article for publication in the Journal of the Acoustical Society of America under the title "Active structural acoustic control of ribbed plates using a weighted sum of spatial gradients" with coauthors Daniel R. Hendricks, Monty Anderson, Jonathan D. Blotter, Scott D. Sommerfeldt, and Kent L. Gee.

Abstract

A weighted sum of spatial gradients (WSSG) control objective for use in active structural acoustic control of finite ribbed plates has been developed and compared to volume velocity. A finite element model was used to obtain the displacement field on the plate, and a finite difference derivative approach was used to obtain the terms in WSSG. Previous results of ASAC on finite ribbed plates are minimal in the literature; however, volume velocity has been heavily used as a control metric on flat plates. WSSG was compared to volume velocity as a benchmark to measure success of control on a variety of boundary conditions including clamped and simply-supported plates. Simulation results show that WSSG provides comparable or improved control of radiated sound power on finite ribbed plates for four different boundary conditions. Results also show that WSSG is relatively insensitive to sensor location, and that multiple control forces can be used to successfully improve control. These results demonstrate that ASAC can be successfully performed on finite ribbed plates with the WSSG control metric, and provides an improvement over the volume velocity control metric.

3.1 Introduction

Active structural acoustic control (ASAC) was first proposed as a method for controlling the sound radiation of distributed structures such as plates and shells [21]. ASAC provided a distinct advantage over active noise control applications to distributed structures by requiring fewer control actuators [7] and removing control actuators from the sound field [11]. Although this method was initially developed with the end goal of minimizing sound radiation from the inside of an airplane fuselage the majority of the literature on ASAC has focused on simple ideal cases such as simply-supported flat plates, while more practical cases, such as ribbed plates (for ship hulls) and cylinders (for modeling aircraft fuselages) have been largely ignored.

Research on flat plates has successfully demonstrated the effectiveness of ASAC, and a variety of methods have been developed for improving ASAC control. Initially ASAC was accomplished by moving the control actuator to the surface of the vibrating structure [6] but it was quickly determined that the error sensor could also be removed from the sound field and placed on the structure as well. This could be done successfully by measuring a quantity on the structure correlated with the radiated sound in the sound field. Control objectives found to be related to this have included minimizing squared velocity at a point [18] and minimizing volume velocity [13]. One of the early developments and important tools in choosing effective control objectives is the acoustic or radiation mode shapes of a structure [9] [12], which reveal the underlying radiation mechanisms of a structure. The result of the radiation mode shape analysis was the predominance of volume velocity minimization in the literature as a control objective due to its similarity to the first radiation mode.

Volume velocity was shown to provide effective control at many of the natural frequencies of a vibrating plate [9] [13] [14]; however, it has proved difficult to measure experimentally. A variety of measurement methods have been developed such as accelerometer arrays [14] and distributed piezoelectric films [15]. Shaped PVDF sensors have also been used to target the structural modes associated with specific radiation modes [16] [17] allowing for an alternative to volume velocity in the estimation of the first radiation mode as well as a method to target higher modes. These methods have been effective in measuring volume velocity but their practicality has been hampered by their difficulty of implementation. Accelerometer arrays require too many sensors, distributed PVDF sensors are geometry dependent, and shaped PVDF sensors are mode shape dependent. Thus these methods require *a priori* knowledge of the system, manufacturing processes tailored to the specific geometry, or a relatively complicated experimental setup.

A more recently developed structural metric, termed the weighted sum of spatial gradients (WSSG) or composite velocity, has been shown, through simulations, to avoid many of these problems [1]. This structural metric was designed to target the first four radiation modes of a plate, was shown to be uniform on a simply supported plate, thus making it insensitive to sensor location, and can be measured with an array of four accelerometers without a priori knowledge of the structural geometry or modes. This makes it simpler to implement in practice and still provides similar control to volume velocity at structural modes contributing to the first radiation mode, while outperforming it at higher frequencies and structural modes which provide a more significant contribution to the higher radiation modes.

One of the purposes of the current research is to extend the application of WSSG and the applicability of ASAC on ribbed plates. Until recently there has been little theoretical development on ribbed plate vibration. Current research is motivated mainly with reducing ship vibration. Recently, models have been developed to represent, analytically, the displacement of a vibrating plate [24] [25] [26]; however, these require a large number of assumptions, such as the assumption of treating the rib as a line force and moment, to work properly. To mitigate vibration of these plates, analysis has shown that ribbed plate vibration can be reduced by irregularly spacing the stiffeners [25] [27]. ASAC has been investigated for an infinite ribbed plate using both line forces [28] and point forces [29] minimizing the far-field acoustic intensity and total sound pressure, respectively. Both of these studies found that significant control could be achieved, however the literature is lacking in exploration of ASAC on finite ribbed plates.

In the literature, the concept of ASAC has been demonstrated for flat rectangular plates both numerically and experimentally and shown to work effectively. ASAC has also been simulated for the simplest and most ideal cases of a ribbed plate. The main purpose of this paper is to extend the applicability of ASAC on ribbed plates by demonstrating that flat plate control metrics, specifically WSSG, can be effectively used and applied to achieve control on finite ribbed plates.

3.2 Formulation

The ribbed plate, as treated in this paper, can be separated into two distinct parts, the plate, by itself, and the ribs, separated from the plate. In previous analytical formulations examining ribbed plates a similar assumption was made [24] [25], allowing the ribs to be thought of as beams applying a line force and moment to the plate. In this paper the plate was modeled using finite element analysis, and not analytically, due to the lack of solutions for the boundary conditions used; however some of the assumptions used in the analytical papers still proved to be useful. In particular, in this paper it has been assumed that only the plate (and not the ribs) radiate sound. Thus in the finite element model, although the interaction between the ribs and the plate was certainly taken into account, the only data needed for the control simulations and calculation of radiated sound power was the displacement field on the plate itself.

Four different sets of boundary conditions on the ribbed plate have been analyzed. In the FEA models a contact constraint requiring the rib to move with the plate along the contact edge was required in each case. The first boundary condition examined had the plate simply-supported along all of its edges, with the rib simply-supported at its ends, as if it were a beam, hereafter referred to as the simply-supported plate, simply-supported ribs boundary condition. The second boundary condition examined also had the plate simply-supported, but this time with the ends of the ribs left free. This boundary condition is referred to as the simply-supported plate, free ribs boundary condition. The third boundary condition examined was with the plate clamped along all four edges, and the ribs clamped at the ends, referred to in the remainder of this paper as the clamped plate, clamped ribs boundary condition. The final boundary condition once again had the plate clamped along all four edges, but with the ribs free at the ends. This is the clamped plate, free ribs boundary condition. The four boundary conditions were chosen to demonstrate the generality of WSSG for use in ASAC, as well as for their potential to appear in practical situations.

Analytically WSSG is defined as a sum of weighted spatial gradients, given as

$$\mathrm{WSSG} = \alpha\,(w)^2 + \beta\left(\frac{\partial w}{\partial x}\right)^2 + \gamma\left(\frac{\partial w}{\partial y}\right)^2 + \delta\left(\frac{\partial^2 w}{\partial x \partial y}\right)^2, \tag{3.1}$$

where w is the displacement of the plate and α, β, γ, and δ are the weights for each of the spatial gradients. This formulation is identical to the formulation of composite velocity [1], Eq. (1.1), with the time dependence, due to time harmonicity, removed.

Because of the dependence on finite-element data, the derivatives cannot be taken analytically. Estimates of the terms using finite difference derivatives are defined as

$$w \approx \frac{w_1 + w_2 + w_3 + w_4}{4}, \tag{3.2a}$$

$$\frac{\partial w}{\partial x} \approx \frac{w_2 - w_1 + w_4 - w_3}{2\Delta x}, \tag{3.2b}$$

$$\frac{\partial w}{\partial y} \approx \frac{w_1 - w_3 + w_2 - w_4}{2\Delta y}, \tag{3.2c}$$

$$\frac{\partial^2 w}{\partial x \partial y} \approx \frac{w_2 - w_1 + w_3 - w_4}{\Delta x \Delta y} \tag{3.2d}$$

where the spacing for the finite difference derivatives is shown in Fig. 3.1. Each of the measurement points, w_i in Fig. 3.1, correspond to the displacement at a nodal location in the finite element data.

When composite velocity was originally formulated [1] each of the terms was chosen because of its similarity in shape to one of the four most efficiently radiating radiation mode shapes at low frequencies. Although important to the structure and stiffness of the plate, the ribs can be thought of as a line force applied to a flat plate. Thus, the radiation mode shapes for the rectangular plate in previous developments still apply to the ribbed plate, suggesting that WSSG has the potential to provide effective control on the ribbed plate.

Also, in previous work great care was taken to choose weights which would make the WSSG quantity relatively uniform over the surface of the plate. With the analytical model for the

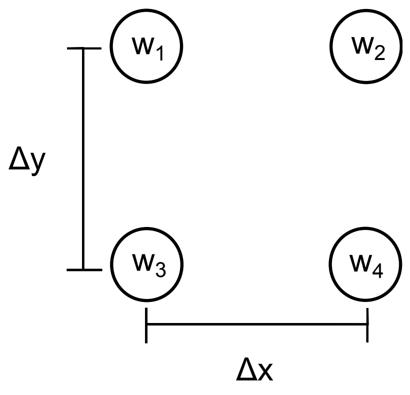


Figure 3.1: Geometric configuration of sensors for finite difference derivatives.

simply-supported flat plate, expressions for these weights were derived which scaled each of the terms so that each mode had the same maximum amplitudes. This resulted in a relatively uniform field which reduced the sensitivity of the WSSG sensor to location. In the case of the ribbed plate, without analytical equations to describe the vibration of the plate, analytical weight expressions cannot be derived. However, the general concept still applies. The weights can be found by calculating each of the WSSG terms numerically over the entire plate, for a given frequency, and then finding values for α, β, γ, and δ so that when each term is multiplied by its corresponding weight all of the maximum amplitudes match. This frequency dependent weighting was used throughout this paper.

3.2.1 The Finite Element Model as a Transfer Function

By assuming that the ribbed plate acts as a linear system, the total displacement due to multiple point forces on the plate at a given frequency is the sum of the displacements from the point forces applied individually. Transfer functions from the force location to the sensor location as a function of frequency were created from these data using the standard definition of a transfer function as output over input. For this application, the output was displacement at the sensor locations and the input was the applied force.

The displacement at the individual sensor locations (as shown in Fig. 3.1) in terms of the transfer functions are given as

$$w_1 = H_{d_1}F_d + \sum_i H_{ci_1}F_{ci} \tag{3.3a}$$

$$w_2 = H_{d_2}F_d + \sum_i H_{ci_2}F_{ci} \tag{3.3b}$$

$$w_3 = H_{d_3}F_d + \sum_i H_{ci_3}F_{ci} \tag{3.3c}$$

$$w_4 = H_{d_4}F_d + \sum_i H_{ci_4}F_{ci} \tag{3.3d}$$

where H_d are the transfer functions from the applied disturbance to the first, second, third, or fourth sensor in the array, and H_{ci} are the transfer functions from the ith applied control force to the first, second, third, or fourth sensor in the array. F_d and F_{ci} are the corresponding disturbance and control forces. Substituting these expressions into Eqs. (3.2) allows for the calculation of WSSG. After this substitution, and assuming F_d has a value of unity, the individual transfer functions can be rearranged to create more general transfer functions in terms of the derivatives with the new expressions given as

$$w \approx \frac{H_{wd} + \sum_i H_{wc_i} F_{c_i}}{4} \tag{3.4a}$$

$$\frac{\partial w}{\partial x} \approx \frac{H_{\frac{\partial w}{\partial x}d} + \sum_i H_{\frac{\partial w}{\partial x}c_i} F_{c_i}}{2\Delta x} \tag{3.4b}$$

$$\frac{\partial w}{\partial y} \approx \frac{H_{\frac{\partial w}{\partial y}d} + \sum_i H_{\frac{\partial w}{\partial y}c_i} F_{c_i}}{2\Delta y} \tag{3.4c}$$

$$\frac{\partial^2 w}{\partial x \partial y} \approx \frac{H_{\frac{\partial^2 w}{\partial x \partial y}d} + \sum_i H_{\frac{\partial^2 w}{\partial x \partial y}c_i} F_{c_i}}{\Delta x \Delta y} \tag{3.4d}$$

where H_w, $H_{\frac{\partial w}{\partial x}}$, $H_{\frac{\partial w}{\partial y}}$, and $H_{\frac{\partial^2 w}{\partial x \partial y}}$ are the transfer functions for the derivative terms of the disturbance or control forces (depending on the subscript), and can be expressed as

$$H_w = H_1 + H_2 + H_3 + H_4, \tag{3.5a}$$

$$H_{\frac{\partial w}{\partial x}} = H_2 - H_1 + H_4 - H_3, \tag{3.5b}$$

$$H_{\frac{\partial w}{\partial y}} = H_1 - H_3 + H_2 - H_4, \tag{3.5c}$$

$$H_{\frac{\partial^2 w}{\partial x \partial y}} = H_2 - H_1 + H_3 - H_4. \tag{3.5d}$$

The H_1, H_2, H_3, and H_4 transfer functions are the displacement over the control or disturbance force at the first, second, third, and fourth sensors, respectively.

Using WSSG in this form allows for the calculation of displacements using the finite element data and the optimization of the control force external to the finite element code. The radiated sound power can then be found by calculating the velocity of each element due to the disturbance and optimized control forces and then using the radiation resistance matrix method [9].

This method was validated by simulating control of WSSG on a simply-supported flat (no ribs) plate using both analytical and FEA formulations using the same dimensions and material properties as previous work [1] on WSSG for a simply-supported plate. This simulation can be seen in Fig. 3.2. There is a difference of 0.35 dB overall sound attenuation between the analytical and the finite element predictions. The slight discrepancies between the two curves can be attributed to the discretization and numerical error of the finite element model, as well as the difference in taking finite difference derivatives as opposed to the exact derivatives of the analytical model. The overall match between the two models demonstrates that the method described above using the finite element data is an accurate approximation and can be used to effectively predict the radiated sound power and find optimal control forces to minimize it.

3.3 Simulations

To demonstrate the effectiveness of WSSG as a control objective, simulations on a ribbed plate are presented in this section. Several specific cases were of interest because of the manner in which WSSG was developed. First, control of WSSG on a variety of boundary conditions is shown to demonstrate its applicability to a wide variety of situations. Second, simulations to demonstrate the insensitivity of the control, provided by WSSG, to the measurement point, are

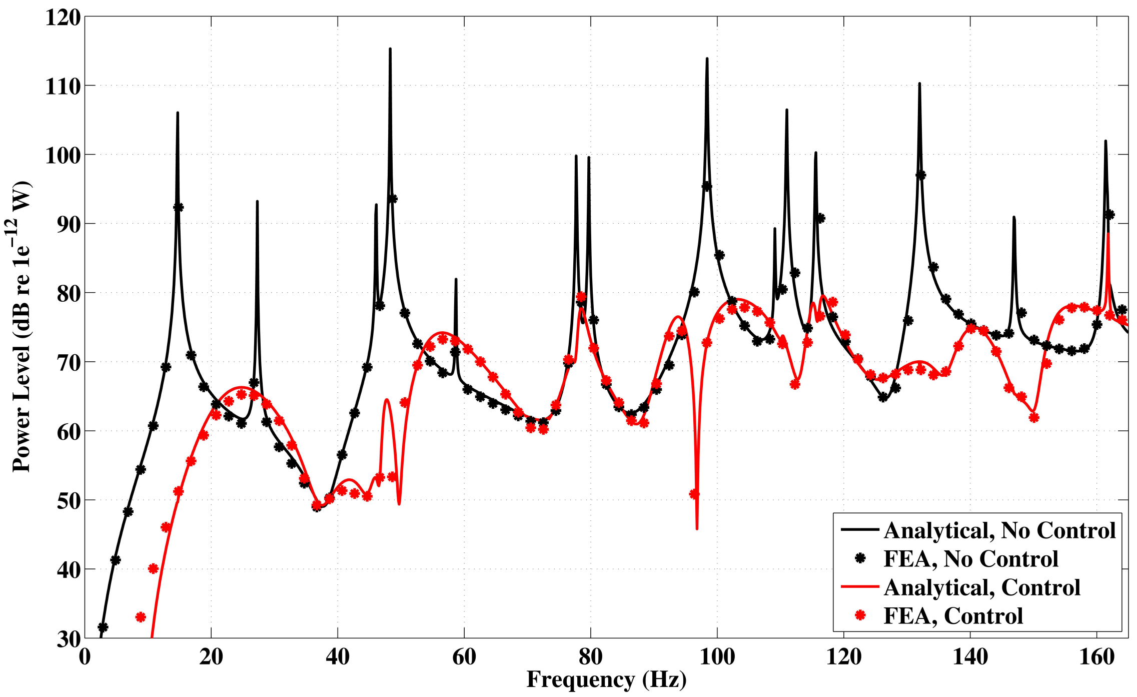


Figure 3.2: Simulated control for a flat simply-supported plate using both analytical and finite element models.

shown to demonstrate the effectiveness of the uniformity of the control field. Last of all, control of the ribbed plate with multiple applied control forces is demonstrated to show how control can be achieved over a larger range of natural frequencies. The measure used to compare the effectiveness of the different control objectives was the overall sound attenuation for a given frequency range. This was calculated by integrating the predicted sound power curve over the frequency range of interest, converting to sound power level (in dB), and then taking the difference between this and the same integral, converted to dB, of the uncontrolled curve. This measure tells how effective the control objective was over the entire frequency range of interest.

The typical geometry and distances for the simulation are shown in Fig. 3.3, where the subscript d stands for disturbance, s stands for sensor, c stands for control, and L_x and L_y are the lengths in x and y respectively. A finite element model defined by this geometry was created. Geometric and material properties for the model are shown in Table 3.1. The model used has two ribs placed at $x = 1/3L_x, 2/3L_x$ dividing the plate into left, middle, and right sections.

Using finite element software the displacement of the plate due to an individual unit point force at chosen disturbance and control locations was found. The point forces were applied harmonically over the frequency range from 0 Hz up to the fifteenth structural mode natural frequency (ranging from 220 - 280 Hz depending on the boundary conditions). The weights used were as described in Sec. 3.2.

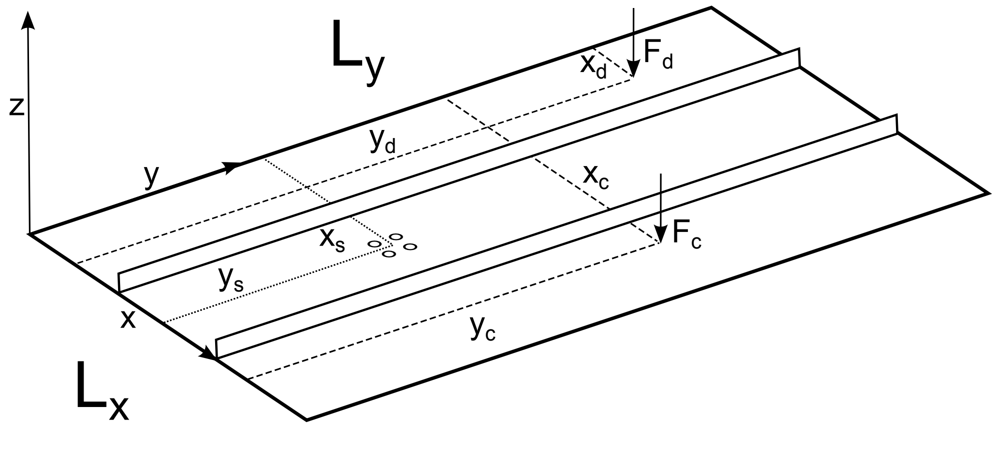


Figure 3.3: Typical geometry of the ribbed plate.

Table 3.1: Geometric and material properties used in the finite element model

Quantity	Value
L_x	0.483 m
L_y	0.762 m
Plate thickness	0.001 m
Rib height	0.025 m
Rib thickness	0.0009 m
Young's modulus	207 GPa
Poisson's ratio	0.29
Density	7800 kg/m^3

3.3.1 Effectiveness of WSSG on Different Boundary Conditions

Using the formulation described previously, ASAC on the ribbed plate, shown in Fig. 3.3 with the properties given in Table 3.1, was simulated under the four boundary conditions: simply-supported plate, simply supported ribs; simply-supported plate, free ribs; clamped plate, clamped ribs; and clamped plate, free ribs, with the disturbance, control and sensor locations given in Table 3.2. Note that for this case both the disturbance and control reside in the left-most section of the ribbed plate, and the sensor lies in the center section. In each of the simulations, WSSG was compared with volume velocity, the most commonly used ASAC control metric, for the ribbed plate.

Figure 3.4 shows control of the simply-supported plate simply-supported ribs boundary condition. For this boundary condition, over the frequency range shown in the figure, WSSG has an overall sound attenuation of 8.0 dB compared to an overall sound attenuation of -0.83 dB for

Table 3.2: Disturbance, control, and sensor locations used in the simulations

Quantity	Value
x_d	0.0805 m
y_d	0.635 m
x_c	0.0805 m
y_c	0.127 m
x_s	0.286 m
y_s	0.432 m

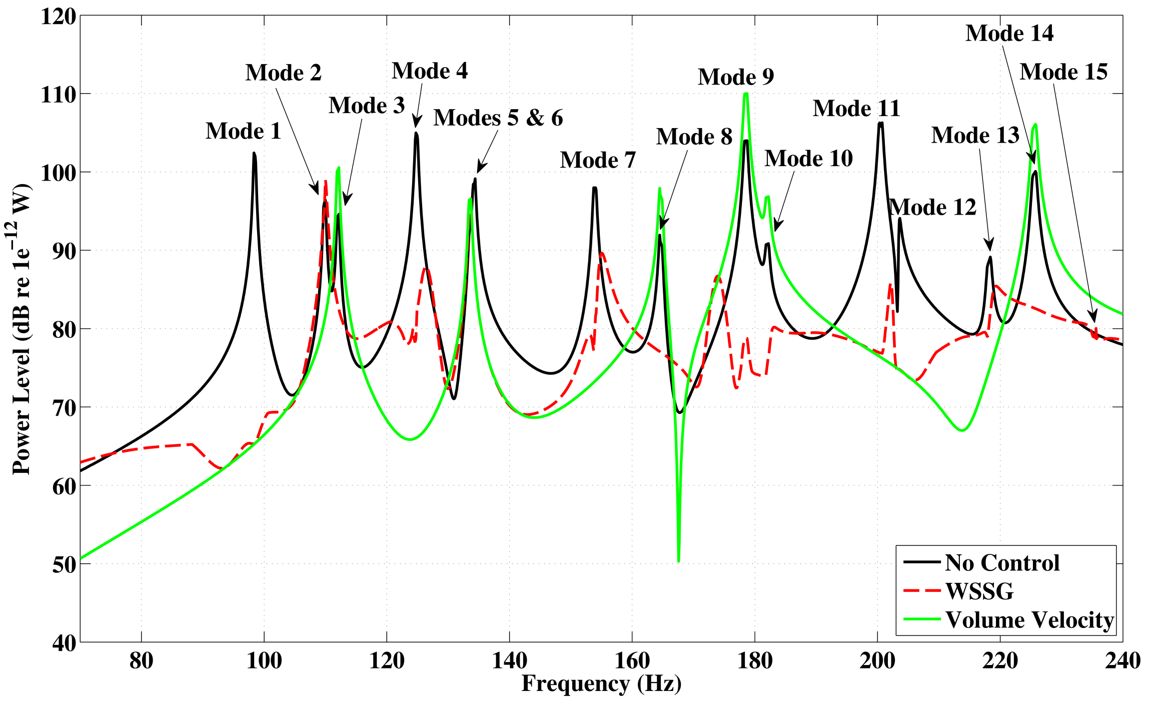


Figure 3.4: Simulated control of the ribbed plate with the simply-supported plate simply-supported ribs boundary condition.

volume velocity. Although volume velocity outperforms WSSG at modes 2, 4, 7, and 13, WSSG achieves a higher overall attenuation because of its ability to provide control at modes 9 and 14, two modes which volume velocity amplifies. WSSG also controls modes 3, 8, and 10 while volume velocity does not. Neither can control modes 5 and 6, which are close enough together to appear as a degenerate mode. Also, mode 15 ($f \approx 235$ Hz) has no noticeable sound radiation. This is due to the ribs constraining the plate to an extremely small displacement at this mode.

Figure 3.5 shows control of the simply-supported plate free ribs boundary condition. The overall sound attenuation for both WSSG and volume velocity, respectively, was 9.9 and -2.2 dB, clearly showing that WSSG outperforms volume velocity in this case as well. In particular, note that WSSG achieves control at modes 4, 8, 9, 11, and 15 while volume velocity outperforms WSSG only at modes 7 and 14. Volume velocity performs particularly badly at modes 9 and 15, causing significant amplification, therefore resulting in negative overall sound attenuation.

For the clamped plate clamped ribs boundary condition, simulated control can be seen in Fig. 3.6. For the clamped plate the overall sound attenuation for WSSG over the frequency range shown is 6.29 dB while for volume velocity the overall sound attenuation is -1.68 dB. The

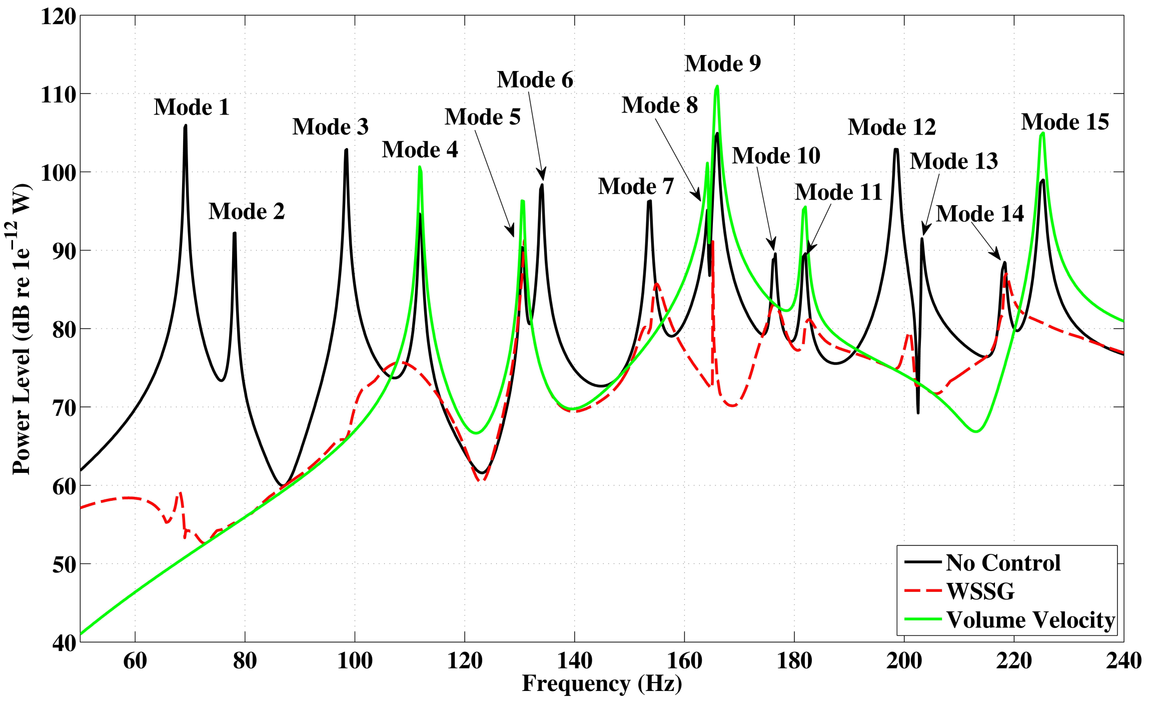


Figure 3.5: Simulated control of the ribbed plate with the simply-supported plate free ribs boundary condition.

difference in control in this case is due to similar reasons as in previous boundary conditions. While WSSG does amplify the radiated sound in some frequency ranges it either controls, or does not amplify, the radiated sound at all of the natural frequencies. Volume velocity, which provides improved control over WSSG in the lower range of the spectrum examined, significantly amplifies modes 9 and 14, while WSSG controls both of these modes. Beyond mode 5 volume velocity performs poorly, with WSSG providing better control at almost every mode. One mode of particular note is the mode at approximately 235 Hz, which is a triply degenerate mode, causing difficulty for both objectives. Because of multiple vibrating structural modes, degenerate modes have multiple out of phase components, each generally requiring an extra degree of freedom in the control force. Thus a single control force is usually ineffective at controlling these modes.

The clamped plate free ribs boundary condition simulated control is shown in Fig. 3.7. Under this boundary condition WSSG provides an overall sound attenuation of 7.38 dB over the frequency range shown, while volume velocity provides an overall sound attenuation of -4.1 dB; thus WSSG significantly outperforms volume velocity in this case as well. For control of this boundary condition note in particular the control of modes 6, 8, and 14, where WSSG provides improved control over volume velocity. Also note that volume velocity amplifies modes 6, 8, and 14, resulting in the negative overall attenuation mentioned. For this boundary condition one other mode of interest, a triply degenerate mode, also occurs around 235 Hz, which neither objective can control.

From the previous simulation results it can be concluded that the ribbed plate can be controlled by both volume velocity and WSSG minimization, and that WSSG minimization provides better control. It has been shown previously [1] (see Sec. 2.3 also) that structural modes which volume velocity fails to control do not contribute significantly to the sound radiation of the first radiation mode, which volume velocity was designed specifically to target. WSSG is successful at many of these structural modes because they contribute to the second, third, or fourth radiation modes, which it was designed to target. Two questions of interest at this point include why sound radiation control is achieved at some structural modes and not others for WSSG, as seen in the examples above, and what effects the ribs have on the ability of WSSG to provide control of sound radiation. The shapes of the structural modes provide some insight into these two questions.

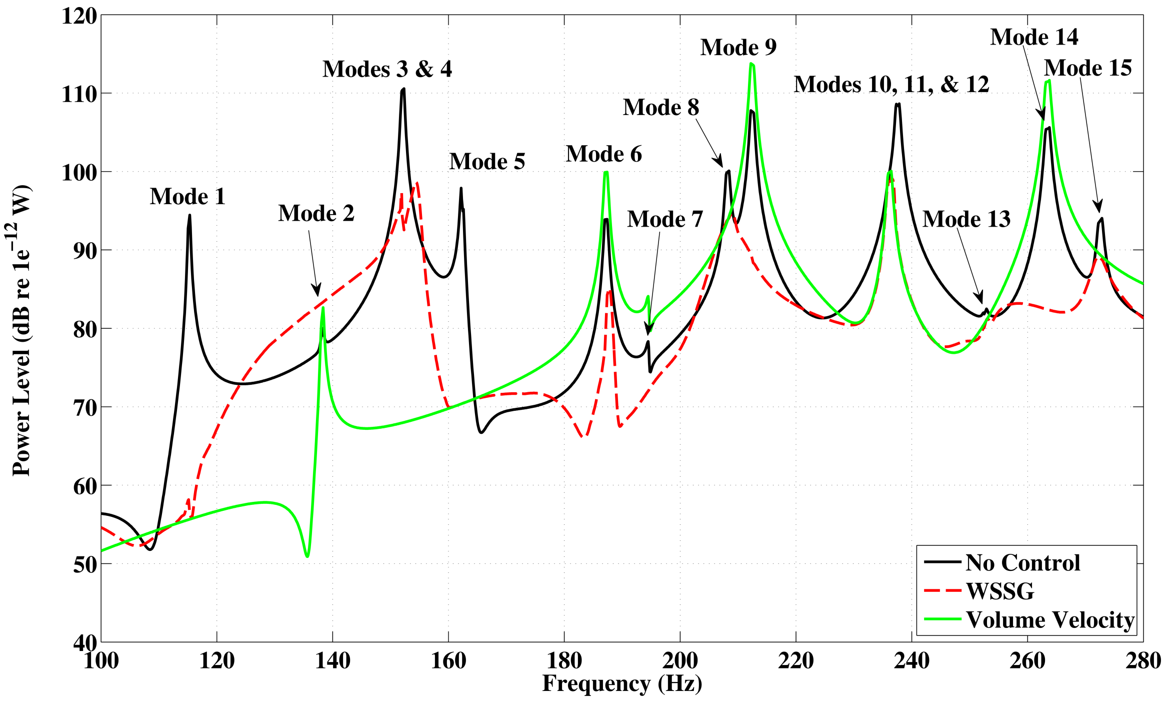


Figure 3.6: Simulated control of the ribbed plate with the clamped plate clamped ribs boundary condition.

With the plate divided into three sections by the ribs, the lower structural modes are constrained so that their nodal lines fall on the locations of the ribs. This causes the ribbed plate to look like three separate vibrating plates. On flat plates, WSSG was found to be relatively uniform over the entire plate with the correct weights chosen [1]. This is not the case for the ribbed plate. Instead, the ribbed plate tends to be uniform within each section of the plate, but not necessarily uniform for the entire plate. In particular, at some natural frequencies the middle section of the plate, between the two ribs, has extremely small amplitude vibration when compared to the outer two sections.

For example, in Fig. 3.8 the WSSG field of the simply-supported plate free ribs mode 5 is shown. For this mode, there is very little displacement in the middle section; the majority of the motion occurs between the ribs and the plate edges. For the simulations, the sensor was placed in the middle section, measuring very small values for WSSG and therefore was unable to provide control to the sections of the plate which were radiating. This was the case on the majority of the modes where WSSG was unsuccessful, irrespective of boundary conditions. When this was not the case, the mode was degenerate, as noted earlier.

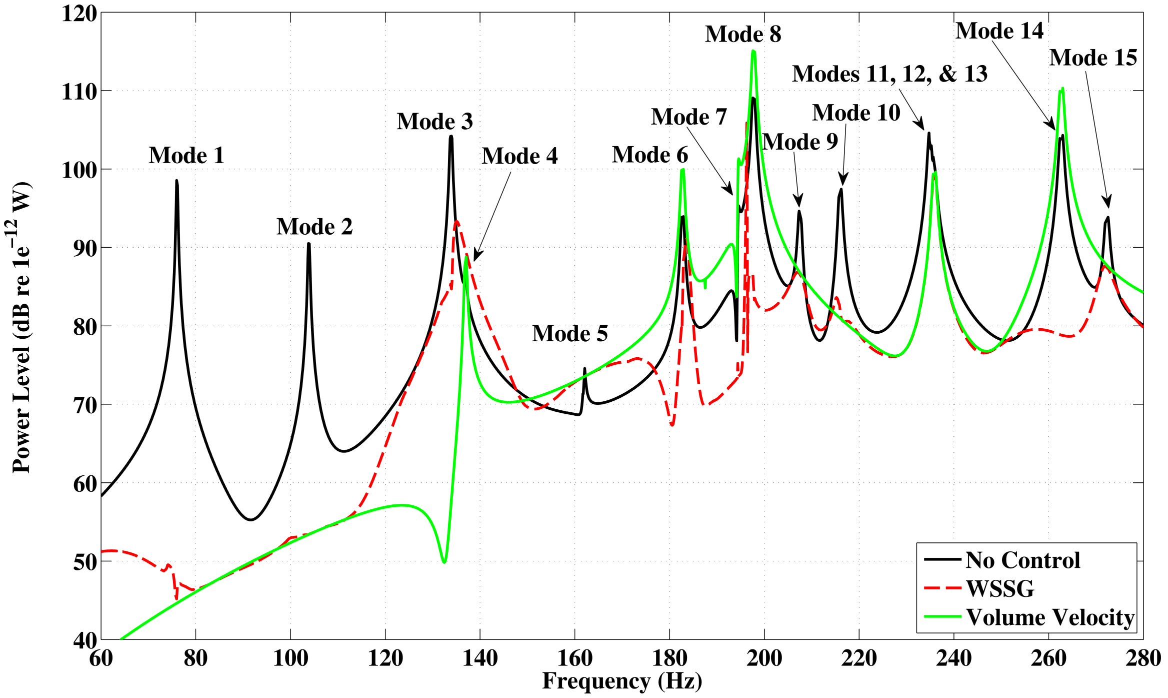


Figure 3.7: Simulated control of the ribbed plate with the clamped plate free ribs boundary condition.

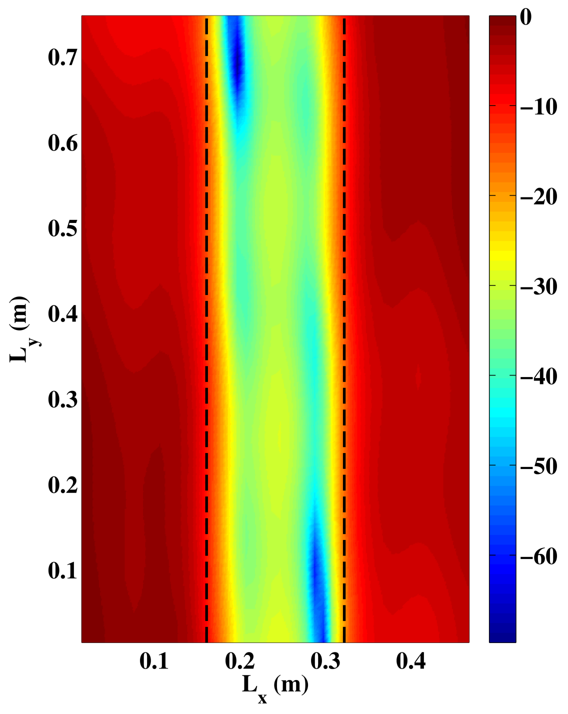


Figure 3.8: The simply-supported plate free ribs boundary condition WSSG field for $f = 130.9$ Hz, the natural frequency of the fifth mode of the plate. Values shown are in dB relative to the maximum WSSG value of the plate. The dashed lines indicate the locations of the ribs.

One final trend to discuss, for the sound attenuation of each of the cases examined, on the free ribs boundary condition the plate generally had a better overall sound attenuation for both WSSG and volume velocity. It was also true that on the simply-supported plate cases a higher attenuation could be achieved than on the clamped plate cases. These results suggest that plates with a higher overall stiffness (in this case from boundary conditions) are more difficult to control. The stiffening effect of the ribs, and the boundaries, make control more difficult.

3.3.2 WSSG Sensitivity to Sensor Location

For flat plate cases, because WSSG was relatively uniform, the control was shown to be relatively independent of sensor location [1]. This claim cannot quite be made for the ribbed plate case due to the large difference in the WSSG values between sections of the plate at some frequencies. However, each individual section is relatively uniform, as demonstrated in Fig. 3.8. Therefore, independence of sensor location would be expected within a given section of the plate. In order to investigate this effect, the disturbance and control forces were held constant at their

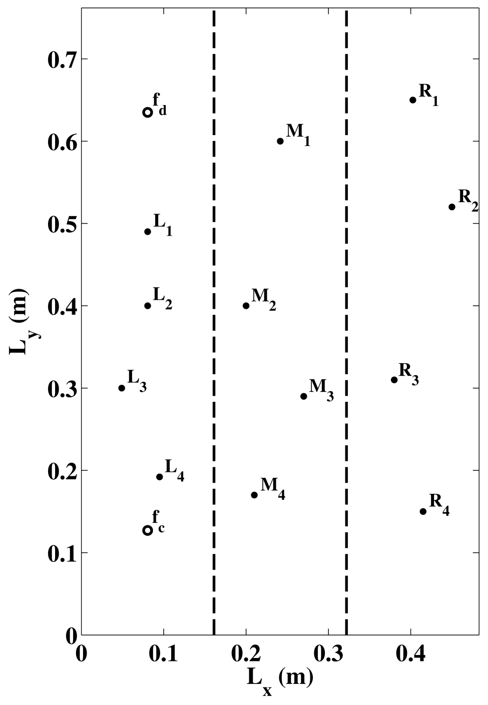


Figure 3.9: Sensor locations used in the control sensitivity studies. f_d and f_c are the disturbance and control force locations, respectively, and L_i, M_i, and R_i are the ith sensor locations in the left, middle, or right sections. The dashed lines represent the location of the ribs.

previous locations in the left section of the plate, and control was simulated with the sensor placed at four different sensor locations, chosen pseudo-randomly, within each of the three sections of the simply-supported plate, free ribs, boundary condition. The sensor locations are shown in Fig. 3.9.

Figure 3.10 shows the simulated control of the plate with the sensor at the left section locations. For this case the average of the overall sound attenuation for the four sensor locations is 7.63 dB, with a standard deviation of 0.6 dB. The control from the four sensor locations is particularly uniform above the second mode. At many frequencies the similarity in control is so close that control from each of the sensor locations is the same. Also note that the fifth mode, where control could not be achieved previously (see Fig. 3.5), is now being controlled. This is because the left section has large motion at this mode, as shown in Fig. 3.8, and the sensors are therefore measuring a quantity related to the radiated sound power.

Control for the plate with the sensor at the middle section locations is shown in Fig. 3.11. The average of the overall sound attenuations for the sensors placed in the middle section of the plate is 10.97 dB, with a standard deviation of 0.48 dB. For this case, specifically note the fifth

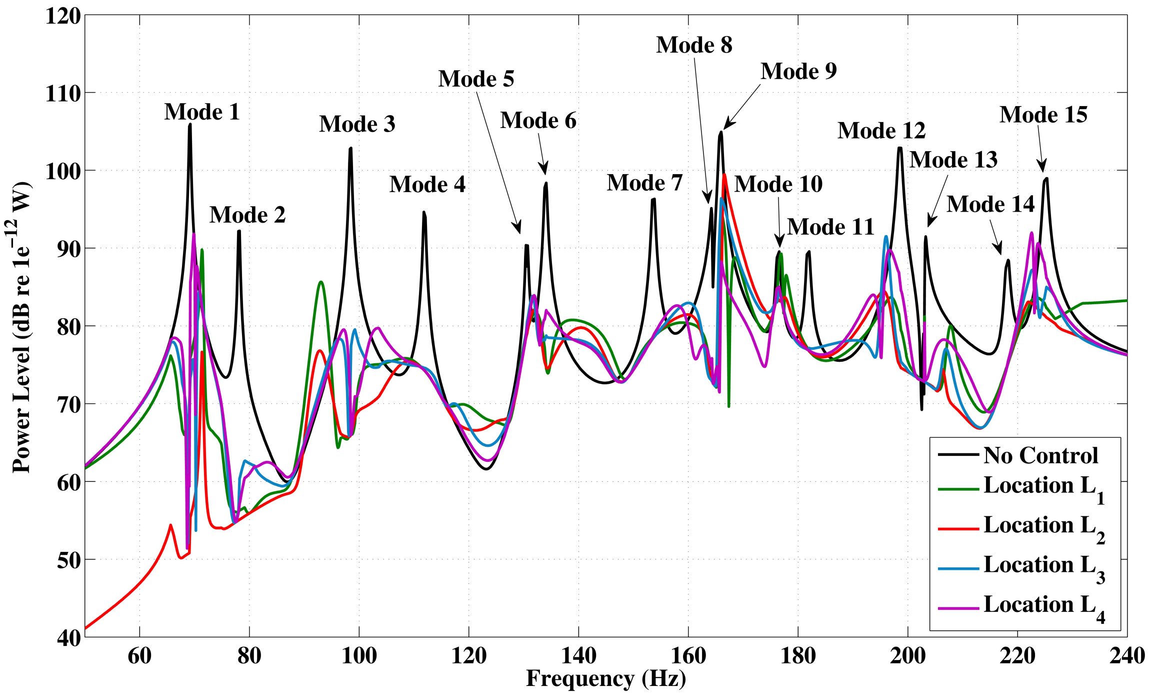


Figure 3.10: Simulated control at various sensor locations in the left section of the plate.

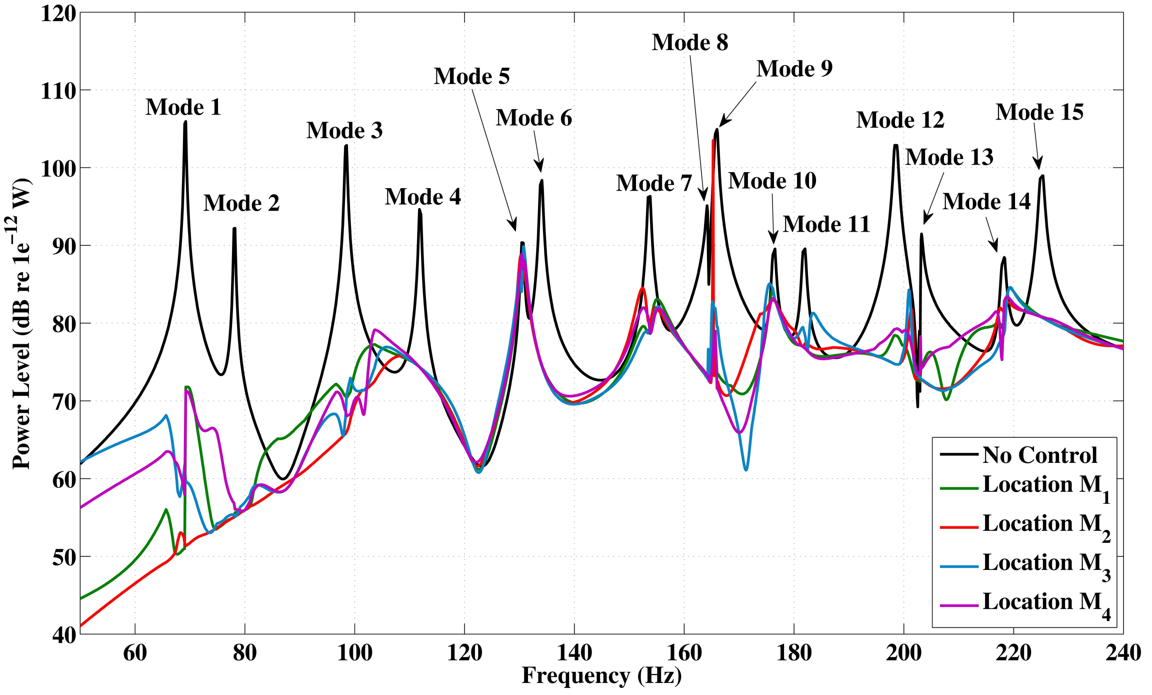


Figure 3.11: Simulated control at various sensor locations in the middle section of the plate.

mode again, where none of the sensor locations can achieve control. This is because of the extremely low amplitude of the WSSG field in the middle section at this mode, as shown in Fig. 3.8. The WSSG value in the middle section does not correlate to the radiated sound, which comes from the left and right sections. Thus minimizing WSSG measured in the middle section does not provide control at the fifth mode natural frequency.

Finally, the simulated control for the plate with the sensor at the right section locations is shown in Fig. 3.12. The overall sound attenuation resulting from the four sensor locations in this section of the plate has an average of 8.93 dB and a standard deviation of 1.0 dB. Note that control is once again achieved at the fifth mode, due to the large magnitude of the WSSG field in the right section, as shown in Fig. 3.8. For this case the control at several off-resonance frequencies, specifically, just before the first mode, between the third and fourth modes, and between the seventh and eighth modes, is inconsistent, and causes sound amplification. A possible explanation for the loss of control, as well as the decreased ability of WSSG (or volume velocity) to provide sound attenuation at many off-resonance frequencies, is given in the next section.

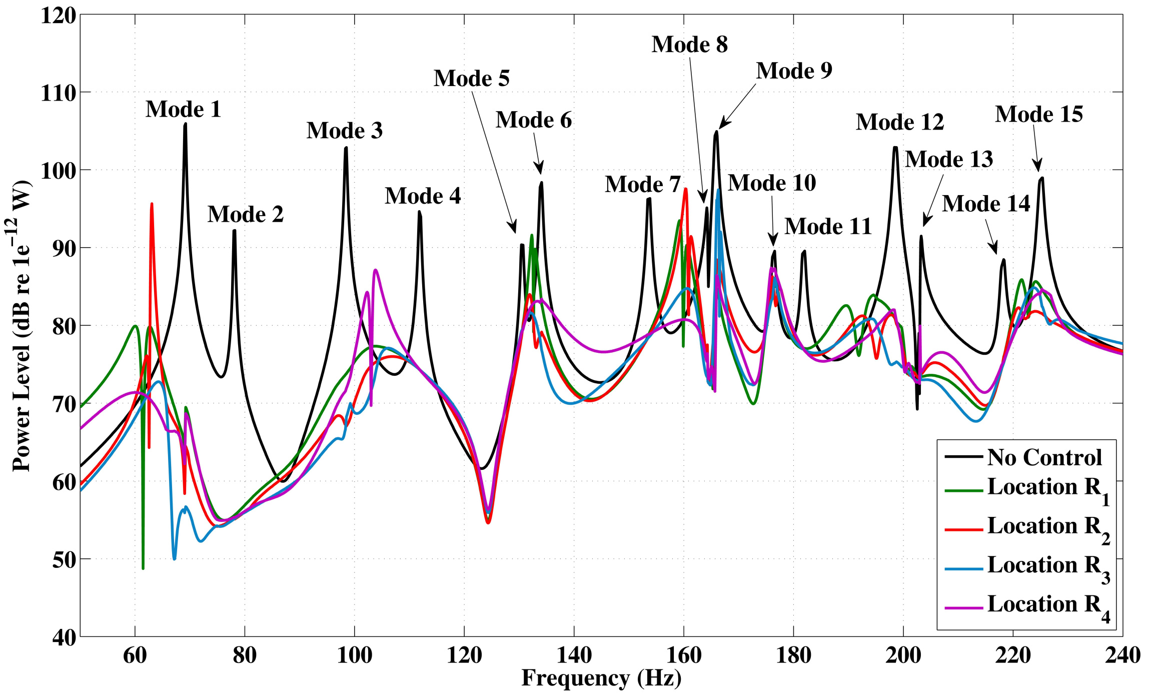


Figure 3.12: Simulated control at various sensor locations in the right section of the plate.

From the above simulations and analysis it can be concluded that the WSSG minimization control objective is largely independent of sensor location within each section of the plate, but that it varies from section to section because of the way in which the mode shapes are modified by the ribs. Thus if each individual section were considered as a separate plate these results are consistent with the simulated results for WSSG on a flat rectangular plate [1].

3.3.3 Multi-force Control

The last case of interest in examining the ability of WSSG to control the vibrating plate is applying multiple control forces in an effort to minimize the radiated sound power. It was expected that by applying multiple control forces, control could be improved at modes where poor control was originally achieved.

For the investigation of this problem, the disturbance and first control forces were kept at the same places as previously, in the left section of the plate, and the sensor was also kept at the same location in the middle of the plate. The second control force was also applied in the left section of the plate, and an optimization algorithm was used to determine the control force values that minimized the WSSG quantity at the measurement location. Figure 3.13 shows the results for the two control forces controlling the radiated sound power separately and together.

One of the important differences between the two control forces is that the original control location is unable to control the fifth mode, as stated previously, but the alternate control location is. Also of note is the ninth mode, which neither force controls well, and the twelfth mode, which the original controls well, but the alternate does not.

Control of WSSG with both forces controlling simultaneously appears similar in some aspects to control with each of the forces applied separately. There are several points of particular importance in this figure. At the 5th mode, the two combined control forces can achieve control. This is due to the effectiveness of the alternate control force here. A similar situation appears at mode 12, but the roles are reversed. At the 9th mode, because neither force could provide good control individually, no control is achieved when they are applied simultaneously.

One other point of importance is the off-resonance control at about 85 Hz. Notice that the radiated sound power is actually amplified here. In general, reduced control, or even amplification, can be observed at off-resonance frequency values in all of the previous results. For example,

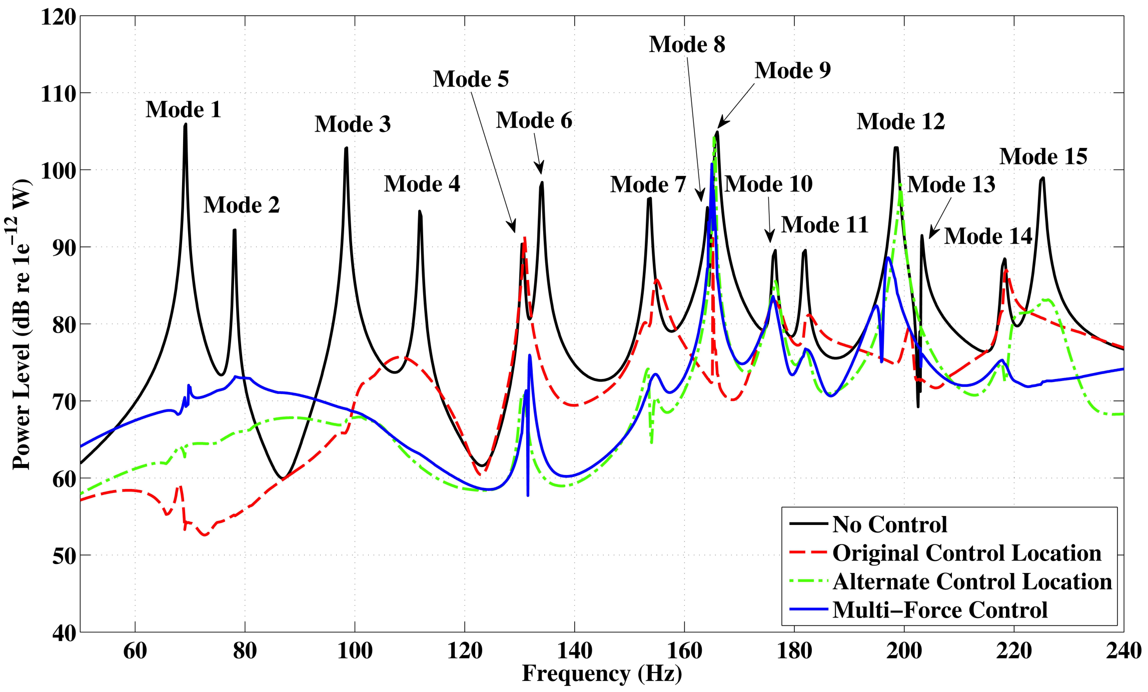


Figure 3.13: Simulated control of the plate, using the WSSG objective, with two control forces applied separately and simultaneously.

in Figs. 3.2, 3.4, and 3.5, control is generally not achieved off-resonance with either WSSG or volume velocity.

Figure 3.14 shows the minimized WSSG values (as opposed to the sound power levels resulting from minimizing WSSG, shown in all of the previous control plots), at the sensor location, on a semi-log scale, for each of the control forces applied separately, and with both applied simultaneously. At or near natural frequencies, in Fig. 3.13, by minimizing WSSG at the sensor location, the radiated sound power of the plate can be reduced. If WSSG at a particular mode cannot be minimized, control cannot be achieved at that mode. Figure 3.14 shows this effect at mode 5, where the original control force was unable to minimize WSSG; therefore, control of radiated sound was not achieved, as seen in Fig. 3.13. However, the alternate control force did attenuate WSSG at mode 5; therefore the alternate control force was able to control it with regards to radiated sound power. At off-resonances, for example at 85 Hz, it can be seen that this is not necessarily the case. The values of WSSG at this frequency were minimized by all three control cases but none of them provided control. This suggests that WSSG, which was developed to target the radiation mode

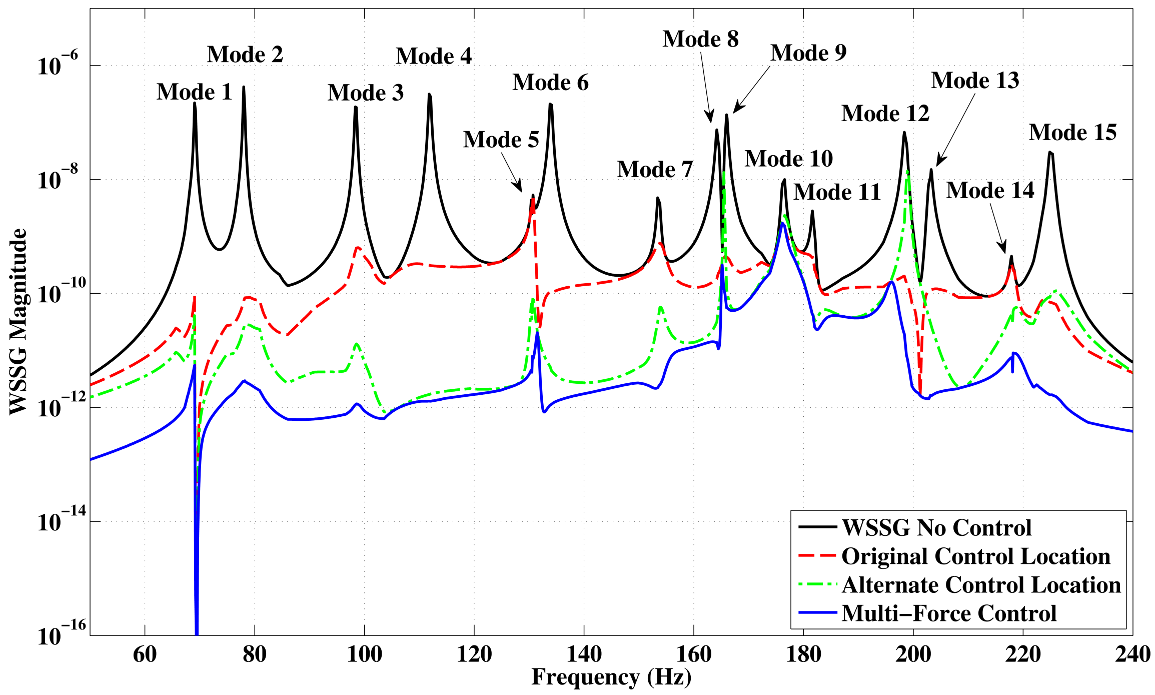


Figure 3.14: Minimized WSSG values at the sensor location, on the simply-supported plate free ribs boundary condition.

shapes of the structure, correlates well with the radiation mode shapes of the structure at or near resonance frequencies, but does not correlate as well at off-resonance frequencies.

In general, across the entire frequency spectrum the minimized value of WSSG, in the multi-force control case, can be seen to be less than or equal to the minimized value of the two control forces applied separately, verifying that improved control of WSSG is achieved with multiple control forces. However, in light of the radiated sound power plots, it can be concluded that improved control of WSSG at off-resonances does not necessarily correlate to better control of radiated sound power at off-resonances. This result suggests that WSSG (as shown in Eq. (3.1)) has a reduced correlation to the radiation mode shapes at off-resonance frequencies, where multiple modes are contributing. Also, because of its poor control at off-resonance frequencies and development in targeting the first radiation mode shape [9], volume velocity would fall prey to this problem. It is likely that to achieve effective off-resonance control the radiation mode shapes would need to be measured directly.

From the preceding analysis it can be inferred that to achieve improved control at natural frequencies complementary control force locations should be chosen so as to cover the widest range of possible natural frequencies. If the alternate control force locations are chosen carefully, control can be achieved at all of the natural frequencies of interest.

3.4 Conclusions

A modified version of composite velocity, termed the weighted sum of spatial gradients, using the assumption of time harmonicity to eliminate spatial dependence, was developed for displacement fields described by discretized data. This formulation was tested, through simulations, for effectiveness in reducing the radiated sound power of ribbed vibrating plates.

In the first set of simulations, control of the ribbed plate under four boundary conditions: simply-supported plate, simply-supported ribs; simply-supported plate, free ribs; clamped plate, clamped ribs; and clamped plate, free ribs, was shown. This set of simulations showed that control of radiated sound power on all of these plates could be achieved by using the minimization of WSSG as a control objective. The control provided by minimizing WSSG was compared to the control provided by minimizing volume velocity, the most common control objective in the literature, and was found to provide improved control in all four boundary condition cases. Through this set of simulations, it can be concluded that WSSG is applicable to a wide variety of boundary conditions and can successfully provide control.

In the second set of simulations, the simply-supported plate free ribs boundary condition was investigated for insensitivity to sensor location, as claimed in the formulation, by simulating control with the WSSG sensor placed in many locations across the plate. Results showed that although control was not consistent across the entire plate due to the mode shapes created by the ribs, control of varying sensor locations in each individual section of the plate created by the ribs was very consistent.

In the third set of simulations, a second control force was added in an attempt to improve control, particularly for natural frequencies that were uncontrolled by a single control force. These simulations showed that control of the natural frequencies could be improved by adding another control force, but at the potential cost of amplifying the radiated sound of off-resonance frequencies. The cause was suggested to be a lack of correlation between the terms of the WSSG objective and the radiation mode shapes, away from natural frequencies.

To summarize, minimizing WSSG was found to be an effective control objective for a rectangular ribbed plate. It was shown to provide comparable or better sound control to volume velocity, achieved this control with fewer sensors, independent of geometry, and to work effectively with multiple control forces. These results, on a typical ribbed plate, suggest that using WSSG to accomplish ASAC can be successfully applied to ribbed plates in general.

CHAPTER 4.   CONCLUSIONS AND FUTURE WORK

4.1 Conclusions

In this thesis, the ASAC control metric WSSG has been developed for clamped flat plates and for ribbed plates. Prior to this work, WSSG was formulated as “composite velocity”, a control metric for use on simply-supported flat plates. One of the main motivations for this research was to extend the applicability of this control metric to different boundary conditions and to increase its general applicability on flat plates. The other main motivation was to develop the control metric for the less ideal but more practical ribbed plate case.

On a flat plate with four clamped edges there is no known exact analytical solution to the wave equation. However, an analytical approximation to the wave equation under this boundary condition was found in the literature which allowed for a theoretical development. The analytical development on the clamped plate was particularly important because analytical weight expressions could be derived directly from this solution. The derived weight expressions gave insight into boundary condition independent weight expressions which correctly reduce to the simply-supported and clamped boundary condition weights already derived. Using WSSG and the derived weights, control was simulated and the results, which showed that minimizing WSSG was an effective control objective on clamped plates, were consistent with previous findings on simply-supported plates.

Because no analytical solutions are available for ribbed plates, a finite element model was used to calculate the displacement field on the plate, which allowed for the calculation of the terms in WSSG. This method did not allow for an analytical expression for the weights; however it did make possible the simulation of control with the less ideal physical geometry. In simulating control it was shown that WSSG provided effective control on a variety of boundary conditions on the ribbed plate, and was insensitive to sensor location within each section of the plate (sections were separated by the ribs). These results were similar to those for the flat plate. It was also shown that multiple control forces could provide improved control at natural frequencies.

4.2 Recommendations for Future Work

Most of the work with regards to WSSG on flat plates has been completed. The only important aspect of this research with regards to flat plates is the experimental validation of the simulated work. This is currently being carried out and will appear in future publications.

Investigation into the ribbed plate has raised a variety of interesting questions and issues which need to be addressed. Analysis has shown that WSSG works well on the specific case used in this research, a plate with two ribs subdividing it into three sections. To more fully demonstrate the ability of WSSG to control ribbed plates several steps are suggested:

1. Investigate the effect of WSSG on plates with more ribs. This case will more closely mimic ribbed plates which occur in ships in the form of hulls and bulkheads. Of interest in this would be varying the spacing between ribs (which affects the stiffness of each section) and the number of sections the plate is subdivided into.

2. Determine whether multiple WSSG sensors in separate sections would provide improved control.

3. Examine the control of ribbed plates with irregularly spaced ribs.

4. Also develop WSSG for the case of ribbed cylinders, for practical application of ASAC to aircraft.

Because of the infinite variability of ribbed plates, a more general characterization, as in the case of flat plates, would be difficult to make. The above suggestions cover a wide range of variability possible in ribbed plates, and would provide a thorough understanding of the mechanisms required to successfully provide ASAC.

The first and second items are of significance because of a phenomenon which was observed in the mode shapes of the ribbed plate. Some of the mode shapes had very small amplitude vibration in the middle section of the plate. If WSSG was measured in this section of the plate, control was not achieved at the natural frequencies corresponding to those mode shapes. An initial investigation has shown that plates with more ribs also exhibit this phenomenon, sometimes in multiple sections of the plate simultaneously. This phenomenon makes control difficult at some modes, as noted above, and an initial suggested remedy is to use two WSSG sensors in adjacent sections (adjacent dead sections initially appear to be very uncommon) to ensure that the quantity is being measured in at least one section of the plate when dead areas are present.

The third item of suggested future work is important because of its known use as a method to passively control the vibration of plates. Because of the added stiffness of the plate at the location of the ribs, the location of the ribs are nodal lines in many ribbed plate mode shapes. Irregularly spacing the ribs provides a barrier to wave propagation and vibration because the waves, which have periodic nodes, do not fit nicely into the irregularly spaced node requirement created by the ribs. It would be interesting to see if the combination of these two methods would provide improved control.

Finally, one of the important practical applications of WSSG is on ribbed cylindrical shells because of their similarity to aircraft. The same numerical methods for the ribbed plate apply in this case and can easily be used to extend the application of the control metric to this useful geometry.

In general, future research in ASAC on ribbed plates with any control metric is recommended because of the important applications available and the lack of research represented in the literature. Much groundbreaking research in this field could be easily accomplished.

REFERENCES

[1] Fisher, J. M., Blotter, J. D., Sommerfeldt, S. D., and Gee, K. L., 2012. "Development of a pseudo-uniform structural quantity for use in active structural acoustic control of simply supported plates: an analytical comparison." *Journal of the Acoustical Society of America,* **131**(5), pp. 3833 – 3840. ii, 4, 7, 10, 11, 13, 15, 16, 17, 26, 29, 30, 31, 33, 39, 40, 41, 45, 59

[2] Guicking, D., 1990. "On the invention of active noise control by Paul Lueg." *Journal of the Acoustical Society of America,* **87**(5), pp. 2251 – 2254. 1

[3] Deffayet, C., and Nelson, P. A., 1988. "Active control of low-frequency harmonic sound radiated by a finite panel." *Journal of the Acoustical Society of America,* **84**(6), pp. 2192 – 2199. 1, 8

[4] Berge, T., Pettersen, O., and Sorzdal, S., 1988. "Active cancellation of transformer noise: field measurements." *Applied Acoustics,* **23**, pp. 309 – 320. 1

[5] Ross, C. F., 1978. "Experiments on the active control of transformer noise." *Journal of Sound and Vibration,* **61**, pp. 473 – 476. 1

[6] Fuller, C. R., Hansen, C. H., and Snyder, S. D., 1991. "Active control of sound radiation from a vibrating rectangular panel by sound sources and vibration inputs: an experimental comparison." *Journal of Sound and Vibration,* **145**(2), pp. 195 – 215. 1, 8, 28

[7] Jones, J. D., and Fuller, C. R., 1989. "Active control of sound fields in elastic cylinder by multi control forces." *American Institute of Aeronautics and Astronautics Journal,* **27**(7), pp. 845 – 852. 1, 8, 28

[8] Metcalf, V. L., Fuller, C. R., Silcox, R. J., and Brown, D. E., 1992. "Active control of sound transmission/radiation from elastic plates by vibration inputs, ii: Experiments." *Journal of Sound and Vibration,* **153**(22), pp. 387 – 402. 1

[9] Elliott, S. J., and Johnson, M. E., 1993. "Radiation modes and the active control of sound power." *Journal of the Acoustical Society of America,* **94**(4), pp. 2194 – 2204. 1, 2, 8, 17, 28, 33, 47

[10] Nelson, P. A., and Elliott, S. J., 1992. *Active Control of Sound.* Academic Press, London San Diego. 1

[11] Fahy, F., and Gardonio, P., 2007. *Sound and Structural Vibration.*, 2nd ed. Academic Press, Amsterdam London. 2, 8, 17, 28

[12] Naghshineh, K., and Koopmann, G. H., 1993. "Active control of sound power using acoustic basis functions as surface velocity filters." *Journal of the Acoustical Society of America,* **93**(5), pp. 2740 – 2752. 2, 8, 28

[13] Johnson, M. E., and Elliott, S. J., 1995. "Active control of sound radiation using volume velocity cancellation." *Journal of the Acoustical Society of America,* **98**(4), pp. 2174 – 2186. 2, 28

[14] Sors, T. C., and Elliott, S. J., 2002. "Volume velocity estimation with accelerometer arrays for active structural acoustic control." *Journal of Sound and Vibration,* **258**(5), pp. 867 – 883. 2, 8, 9, 20, 28

[15] Gardonio, P., Lee, Y.-S., and Elliott, S. J., 2001. "Analysis and measurement of a matched volume velocity sensor and uniform force actuator for active structural acoustic control." *Journal of the Acoustical Society of America,* **110**(6), pp. 3025 – 3031. 2, 8, 9, 28

[16] Snyder, S. D., Tanaka, N., and Kikushima, Y., 1995. "The use of optimally shaped piezo-electric film sensors in the active control of free field structural radiation. Part 1: Feedforward control." *Journal of Vibration and Acoustics,* **117**, pp. 311 – 322. 2, 3, 9, 28

[17] Snyder, S. D., Tanaka, N., and Kikushima, Y., 1996. "The use of optimally shaped piezo-electric film sensors in the active control of free field structural radiation. Part 2: Feedback control." *Journal of Vibration and Acoustics,* **118**, pp. 112 – 121. 2, 28

[18] Sung, C.-C., and Jan, C. T., 1997. "Active control of structurally radiated sound from plates." *Journal of the Acoustical Society of America,* **102**(1), pp. 370 – 381. 2, 11, 28

[19] Manwill, D. A., Fisher, J. M., Sommerfeldt, S. D., Gee, K. L., and Blotter, J. D., 2010. "On the use of energy based metrics in active structural acoustic control." In *Proceedings of Meetings on Acoustics*, Vol. 9. 2, 9

[20] Fisher, J. M., 2010. "Development of a pseudo-uniform structural velocity metric for use in active structural acoustic control." Master's thesis, Brigham Young University, Provo, Utah. 4

[21] Fuller, C. R., 1990. "Active control of sound transmission/radiation from elastic plates by vibration inputs: I. analysis." *Journal of Sound and Vibration,* **136**, pp. 1– 15. 8, 27

[22] Rao, S. S., 2007. *Vibration of Continuous Systems*. Wiley, Hoboken, NJ. 10

[23] Leissa, A., 1993. *Vibration of Plates*. American Institute of Physics. 11

[24] Lin, T. R., and Pan, J., 2006. "A closed form solution for the dynamic response of finite ribbed plates." *Journal of the Acoustical Society of America,* **119**(2), pp. 917–925. 29

[25] Lin, T. R., 2008. "A study of modal characteristics and the control mechanism of finite periodic and irregular ribbed plates." *Journal of the Acoustical Society of America,* **123**(2), pp. 729 – 737. 29

[26] Lin, T. R., 2012. "An analytical and experimental study of the vibration response of a clamped ribbed plate." *Journal of Sound and Vibration,* **331**, pp. 902–913. 29

[27] Lin, T. R., Pan, J., O'Shea, P. J., and Mechefske, C. K., 2009. "A study of vibration and vibration control of ship structures." *Marine Structures,* **22**, pp. 730 – 743. 29

[28] Gu, Y., and Fuller, C. R., 1991. "Active control of sound radiation due to subsonic wave scattering from discontinuities on fluid-loaded plates. i: Far-field pressure." *Journal of the Acoustical Society of America,* **90**(4), pp. 2020–2026. 29

[29] Kessissoglou, N. J., and Pan, J., 1998. "Active structural acoustic control of an infinite ribbed plate under light fluid loading." *Journal of the Acoustical Society of America,* **104**(6), pp. 3398–3407. 29

[30] Johnson, W. R., Aslani, P., Sommerfeldt, S. D., Blotter, J. D., and Gee, K. L., 2013. "Acoustic radiation mode shapes for control of plates and shells." In *Proceedings of Meetings on Acoustics*, Vol. 19. 56

[31] Morse, P. M., and Ingard, U. K., 1968. *Theoretical Acoustics*. McGraw-Hill, New York. 56

APPENDIX A. WSSG FOR CIRCULAR PLATES

During the course of this research WSSG was developed for flat, circular plates. WSSG turned out to be unsuited to ASAC of circular plates. The development, simulations, and conclusions, on circular plates, published in reference [30], will be reviewed here, as an explanation for why this turned out to be the case.

A.1 Circular Plate Analytical Model

For the circular plate a clamped boundary condition was used. The displacement on the circular plate is given by the equation [31]

$$w(r,\theta) = \sum_q \frac{F_q}{\pi a^2} \sum_{m,n}^{\infty} \frac{\psi_{m,n}(r,\theta)\psi_{m,n}(r_q,\theta_q)}{\Lambda_{mn}(\gamma_{mn}^4 - \gamma^4)} \tag{A.1}$$

where F_q is the magnitude of the qth applied force, r_q and θ_q represent the location of the qth applied force, a is the radius of the plate, γ is the wave number, and γ_{mn} is the wave number of the m,nth mode. The mode shapes, $\psi_{m,n}(\cdot,\cdot)$ are given by

$$\psi_{m,n}(r,\theta) = sin(m\theta)\left[J_m\left(\frac{\pi\beta_{mn}r}{a}\right) - \frac{J_m(\pi\beta_{mn})}{I_m(\pi\beta_{mn})} I_m\left(\frac{\pi\beta_{mn}r}{a}\right)\right]. \tag{A.2}$$

For $m = 0$ the $sin(m\theta)$ is replaced by $cos(m\theta)$. β_{mn} is related to γ_{mn} by $\gamma_{mn} = \pi/a\beta_{mn}$, and $J_m(\cdot)$ and $I_m(\cdot)$ refer to Bessel and modified Bessel functions, respectively, of the mth order. Λ_{mn} is given by the expression

$$\Lambda_{mn} = \frac{2}{\varepsilon_m}\left[J_m(\pi\beta_{mn})^2 + J_m'(\pi\beta_{mn})^2\right], \tag{A.3}$$

where ε_m is a constant which is equal to 1 if $m = 0$ and 2 otherwise.

Although the plate displacement equation is given in terms of r and θ the derivative terms for WSSG, i.e. $\partial w/\partial x$, $\partial w/\partial y$, and $\partial^2 w/\partial x \partial y$ were left in terms of x and y and were not taken in polar coordinates. Instead, the solution was transformed into Cartesian coordinates using $r = \sqrt{x^2+y^2}$ and $\tan(\theta) = y/x$. To clarify the process of taking derivatives, define

$$f(x,y) = \sin\left[m\tan^{-1}\left(\frac{y}{x}\right)\right], \tag{A.4}$$

$$g(x,y) = J_m\left(\frac{\pi\beta_{mn}\sqrt{x^2+y^2}}{a}\right), \tag{A.5}$$

$$h(x,y) = I_m\left(\frac{\pi\beta_{mn}\sqrt{x^2+y^2}}{a}\right), \tag{A.6}$$

$$c = \frac{J_m(\pi\beta_{mn})}{I_m(\pi\beta_{mn})} \tag{A.7}$$

where the sine in Eq. (A.4) is replaced with cosine when $m = 0$, as described previously. Then the shape functions can be rewritten as $\psi_{mn}(x,y) = f(x,y)[g(x,y) - c \cdot h(x,y)]$, and the derivatives are given as

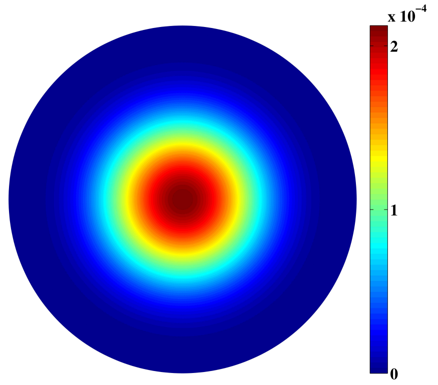
$$\frac{\partial\psi}{\partial x} = f(x,y)\left[\frac{\partial g}{\partial x} - c\frac{\partial h}{\partial x}\right] + \frac{\partial f}{\partial x}[g(x,y) - c\cdot h(x,y)] \tag{A.8}$$

$$\frac{\partial\psi}{\partial y} = f(x,y)\left[\frac{\partial g}{\partial y} - c\frac{\partial h}{\partial y}\right] + \frac{\partial f}{\partial y}[g(x,y) - c\cdot h(x,y)] \tag{A.9}$$

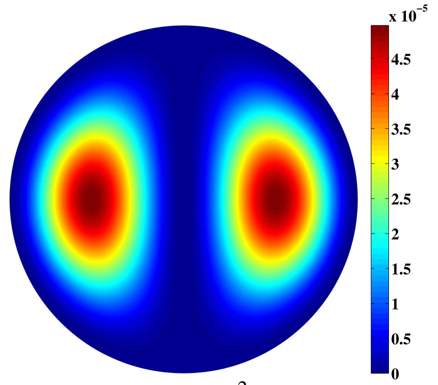
$$\begin{aligned}\frac{\partial^2\psi}{\partial x\partial y} =& f(x,y)\left[\frac{\partial^2 g}{\partial x\partial y} - c\cdot\frac{\partial^2 h}{\partial x\partial y}\right] + \frac{\partial f}{\partial y}\left[\frac{\partial g}{\partial x} - c\cdot\frac{\partial h}{\partial x}\right]\\ &+ \frac{\partial f}{\partial x}\left[\frac{\partial g}{\partial y} - c\cdot\frac{\partial h}{\partial y}\right] + \frac{\partial^2 f}{\partial x\partial y}[g(x,y) - c\cdot h(x,y)].\end{aligned} \tag{A.10}$$

Because Eq. (A.1) only depends on x and y in the shape functions, in order to obtain the derivatives $\partial w/\partial x$, $\partial w/\partial y$, and $\partial^2 w/\partial x\partial y$, the shape function $\psi_{m,n}(x,y)$ need only be replaced by its corresponding derivative term. The derivative terms, for the first mode of the plate, are shown in Figs. A.1.

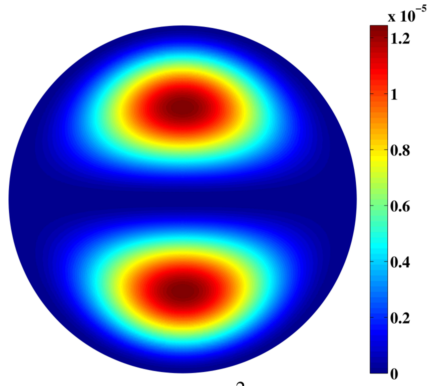
The radiation mode shapes were compared to the derivative terms on the circular plate. A formulation of the radiation mode shapes discretizing the plate in polar coordinates was originally used, however, this was found to distort the radiation mode shapes due to the large difference in element sizes at the center and outer edges of the plate. Therefore, the radiation mode shape formulation used for the circular plates is the same as was used previously, in Eq. (2.12), using



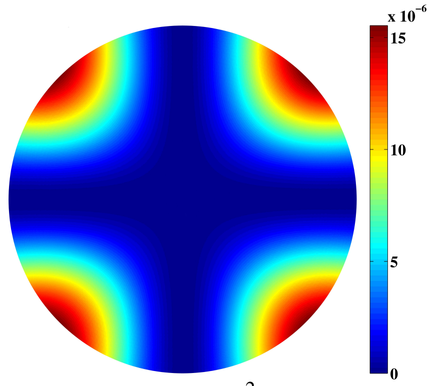
(a) w^2 term



(b) $\left(\frac{\partial w}{\partial x}\right)^2$ term



(c) $\left(\frac{\partial w}{\partial y}\right)^2$ term



(d) $\left(\frac{\partial^2 w}{\partial x \partial y}\right)^2$ term

Figure A.1: WSSG terms for the clamped circular plate first mode, on a plate with a radius of unity.

squares elements, which provided correct results, as long as the elements were sufficiently small ($\approx 1/6$ wavelength).

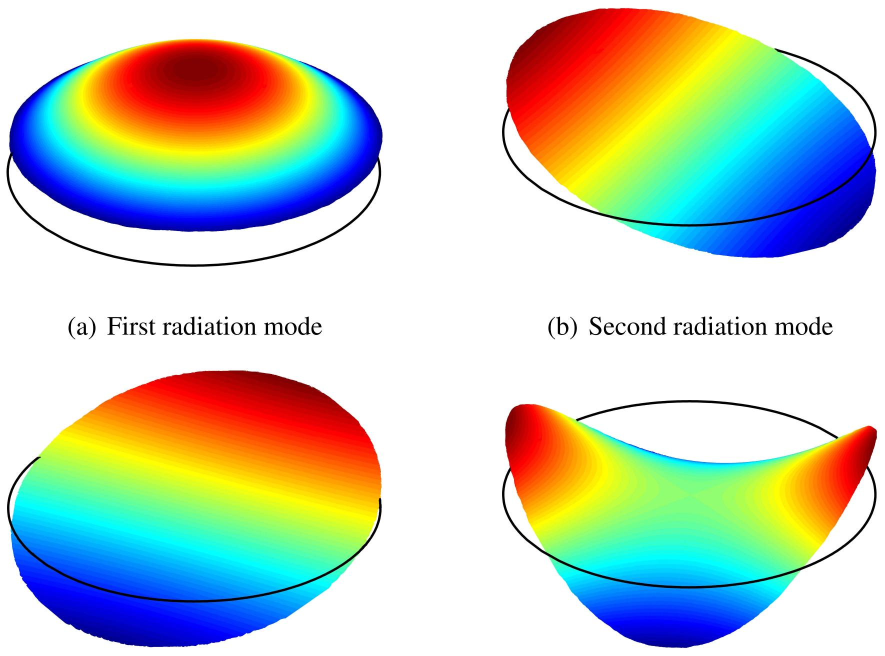
Circular Plate Weights

One of the important properties of the derivative terms used in previous developments was that a uniform field could be created through proper weighting of each of the derivative terms. The weights for each of these terms was apparent by taking the analytical derivatives and determining how each term was scaled, and then averaging the modal weights over the frequency range of interest.

In the case of the circular plate, this method did not provide an apparent weighting scheme, due to the added complications of transforming from polar to Cartesian coordinates. Thus a weighting scheme similar to the weighting scheme for the ribbed plate, described in Sec. 3.2, was used. At each frequency each of the WSSG terms, found by applying a 1 N force to the plate at the disturbance location given in Table A.2, was scaled so that it's maximum amplitude matched that of the maximum amplitude of the first term. However, the weights over the frequency range of interest were averaged, as opposed to using the frequency specific weights in the case of the ribbed plate, and are given in Table A.2. This weighting method was found to be sufficient for the circular plate.

A.1.1 Circular Plate Comparison to Radiation Modes

The four derivative terms, shown in Fig. A.1 can be compared qualitatively to the first four radiation mode shapes of the circular plate, shown in Fig. A.2. The comparison between the derivative terms and the radiation mode shapes, as in the simply-supported plate [1] and the clamped plate (see Ch. 2) was also observed for the clamped circular plate as well.


(a) First radiation mode (b) Second radiation mode

(c) Third radiation mode (d) Fourth radiation mode

Figure A.2: The first four radiation mode shapes for the circular plate.

In comparing the shapes qualitatively, note that the shape of Fig. A.1(a) is similar to the first radiation mode shape, Fig. A.2(a). Also, note that the shapes of the derivatives in x and y in Figs. A.1(b) and A.1(c) are similar to the second and third radiation mode shapes, shown in Figs. A.2(b) and A.2(c), and that the cross derivative in Fig. A.1(d) is similar to the fourth radiation mode shape, shown in Fig. A.2(d).

There are some differences between the derivative terms and the radiation mode shapes. In particular, because the circular plate used here has a clamped boundary, the derivatives go to zero at the edges of the plate in the x and y directions, unlike the radiation mode shapes, which are boundary condition independent. Interestingly, this is not carried to the cross derivative term, because the coordinate system used is not well suited to the geometry.

This difference in boundaries between the radiation modes and the derivative terms initially suggest that these terms may not provide good control. However, if a small area of the plate were examined, it would be observed that each of the derivative terms, locally, was comparable to the radiation mode shapes, without the boundary condition caveat. This suggests that a local measurement of the derivative terms, on the circular plate, would provide control of the first four radiation mode shapes, and is the justification for the continued examination of this case.

A.2 Simulations

For the control simulations, a plate with the properties shown in Table A.1, was used. The weights and disturbance and control locations are given in Table A.2, where the subscript d represents the disturbance force, the subscript c represents the control force, and the subscript s represents the WSSG sensor. The simulation of control of the clamped ribbed plate was done with

Table A.1: Geometric and material properties used in the clamped circular plate simulations

Quantity	Value
Radius	0.46 m
Plate thickness	0.001 m
Young's modulus	207 GPa
Poisson's ratio	0.29
Density	7800 kg/m^3

Table A.2: Disturbance, control, and sensor locations used in the clamped circular plate simulations

Quantity	Value	Weight	Value
r_d	0.0955 m	α	1
θ_d	197.4°	β	0.0242
r_c	0.1996 m	γ	0.0198
θ_c	346.4°	δ	7.8042e-04
r_s	0.2898 m		
θ_s	127.7°		

the typical objectives of minimizing volume velocity across the plate and minimizing WSSG at a point location. The volume velocity was calculated using Eq. (1.6). The simulated control is shown in Fig. A.3.

For this simulation overall sound attenuation for the frequency range shown was 20.5 dB for WSSG and only 2.7 dB for volume velocity. This seems odd considering that at the natural frequencies of the system volume velocity has an average attenuation of 20.3 dB compared to

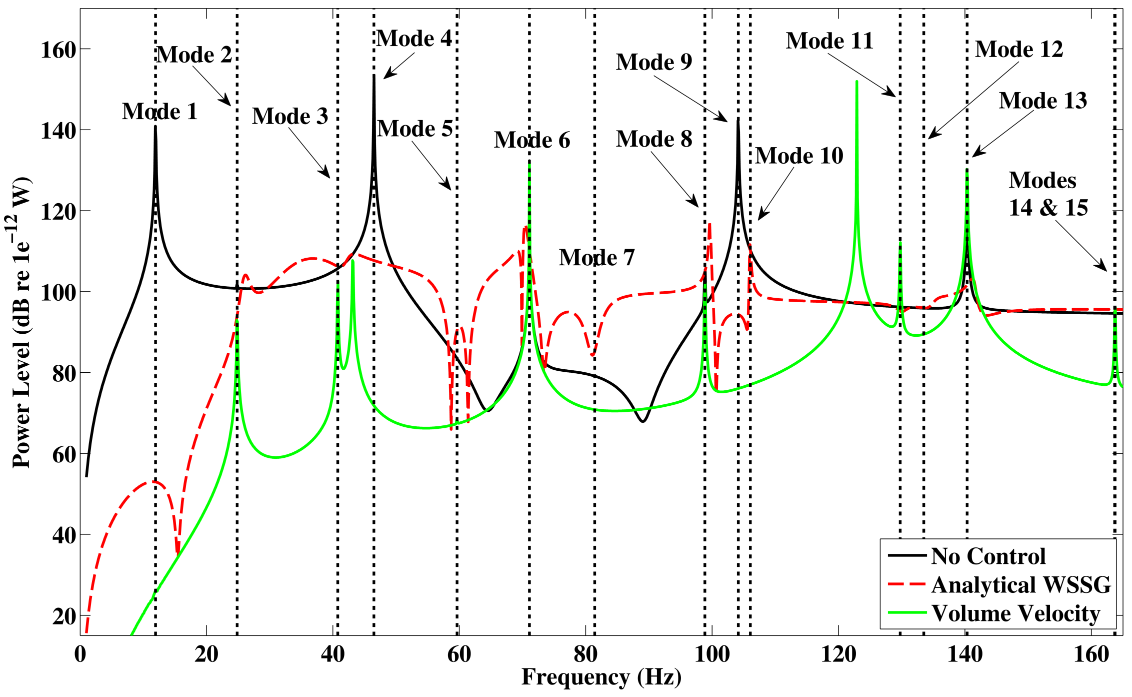


Figure A.3: Simulated control of the circular plate using minimization of volume velocity and WSSG as the control objectives. The dashed lines represent the structural modes of the plate.

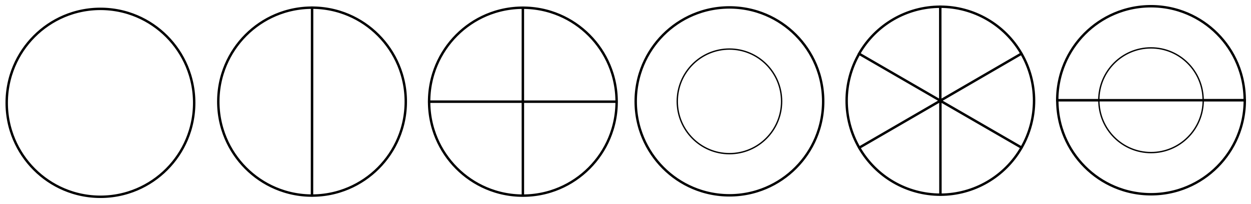


Figure A.4: The first six structural mode shapes of the clamped circular plate.

WSSG's 14.9 dB. However, volume velocity amplifies the control significantly at the off frequency of 123 Hz and at the 13th mode. Other than these two points, volume velocity generally did a better job of minimizing the sound radiation when compared to WSSG.

This raises the question of why, when WSSG has been shown to provide improved control in the clamped and ribbed plate cases, does it get outperformed by volume velocity across most of the frequency range shown for the circular plate. The answer lies in the fact that the majority of the modes which were found to radiate sound over the frequency range examined had a net volume velocity, while the modes which didn't radiate sound had a net volume velocity of zero, due to inter-cellular cancellation. The first 6 structural modes are shown in Figs. A.4, where it can be seen that modes 1 and 4, which radiate sound, have a net volume velocity, and modes 2, 3, and 5, which didn't radiate sound, have net zero volume velocity. Mode 6, which is the first mode with a radial nodal line, is the first mode which has a net volume velocity of zero, but also radiates sound. In this case the volume velocity control objective neither attenuates nor amplifies this mode, but WSSG does provide some minimal control. These results show that the volume velocity control function is ideally suited for controlling the circular plate, because the majority of the modes which have no volume velocity don't radiate, whereas the modes with a net volume velocity do. This also suggests that the first radiation mode radiates sound more effectively than the other three examined, because volume velocity was designed to target this radiation mode.

A.3 Conclusions

WSSG was formulated for a clamped circular plate. The formulation did not allow for an apparent weighting mechanism, therefore numerical weights were used. Using these numerical weights control was simulated, and shown to provide good control using WSSG but much better control over the majority of the frequency range of interest using volume velocity. This is due to the fact that when volume velocity was zero on the circular plate sound radiation frequently did not occur, and suggests that the first radiation mode radiates sound more effectively than the other three modes.

APPENDIX B. MATLAB CODE FOR COMPUTER SIMULATIONS

B.1 Clamped Plate Code

The code to run the clamped plate simulations is included below. Each piece of code was written as a series of functions and subfunctions, which significantly simplified the coding. Each subsection represents a different m-file in Matlab, to be saved with the corresponding name. In order to rerun these simulations, recreate each of the m-files, and then call the function 'control_clamped_Vcomp(15)'. The argument '15' represents the number of modes used for the typical simulations.

B.1.1 control_clamped_Vcomp

```
 % Date:     4/19/2012
 %           8/6/2012    Modified
 %           8/13/2012   Modified for frequency dependent weights
 % Purpose:  This program to simulate controllability of a clamped plate, in
 %           the same manner as was tested for the simply supported.
 % Notes: 1.


 function control_clamped_Vcomp(num_modes)

clc;

% Get the top modes
modes1     = top_modes_clamped(num_modes);
m          = modes1(:,1);
n          = modes1(:,2);
omega_mode = 2*pi*modes1(:,3);

% Define the values for lambda
lambda   = [3.011 5:2:5*max(m)*max(n)];
lambda   = lambda*pi/2; % This was found from the transcendental
                        % relationship cosh(lambda)cos(lambda) = 1,
                        % from the clamped clamped beam relationship.
lambda_m = lambda(m);
lambda_n = lambda(n);

% Preallocate
I2 = zeros(length(m),1);
I6 = zeros(length(m),1);

% Plate parameters
Lx = 0.483;
```

```
Ly = 0.762;

% Calculate I2 and I6
for i = 1:length(m)
    I2(i) = quadgk(@(x)I2_integrand(x,lambda_n(i),Ly),0,Ly);
    I6(i) = quadgk(@(x)I6_integrand(x,lambda_m(i),Lx),0,Lx);
end

% Element spacing
el_x      = 20;
el_y      = 20;
space_x   = Lx/el_x;
space_y   = Ly/el_y;
x         = space_x/2:space_x:Lx;
y         = space_y/2:space_y:Ly;
distances = distance_mat(x,y);

% Volume velocity sensor struct
sensor_el_x = 6;
sensor_el_y = 10;

space_x = Lx/sensor_el_x;
space_y = Ly/sensor_el_y;

sensor_vv.x = space_x/2:space_x:Lx-space_x/2;
sensor_vv.y = space_y/2:space_y:Ly-space_y/2;

[sensor_vv.x sensor_vv.y] = meshgrid(sensor_vv.x,sensor_vv.y);
sensor_vv.Ae              = space_x*space_y;

% Force parameters
f_mag         = 1;
f_phase       = -180;
f_disturbance = 1;
x_disturbance = 0.083;
y_disturbance = 0.629;
x_control     = 0.083;
y_control     = 0.127;
x_sense1      = 0.286;
y_sense1      = 0.432;
xq            = [x_disturbance x_control];
yq            = [y_disturbance y_control];

% Call the optimization routine
freq_inc           = 1;
freq               = 1:freq_inc:630;
omega              = 2*pi*freq;
f_ave_mag1         = zeros(length(freq), 1);
f_ave_phase1       = zeros(length(freq), 1);
f_ave_control1     = zeros(length(freq), 1);
f_vv_control       = zeros(length(freq), 1);
power_nc           = zeros(length(freq), 5);
power_ave_control1 = zeros(length(freq), 5);
power_vv_control   = zeros(length(freq), 5);

% fminsearch options
options = optimset('MaxFunEvals',400);

tic
for i = 1:length(omega)
    disp(['Frequency ' num2str(freq(i))])

    % Control over 15 modes
    f_ave1=fminsearch(@(f_control)clamped_Vcomp_objective(x_disturbance,...
        y_disturbance, f_disturbance, x_control, y_control, f_control,...
        x_sense1, y_sense1, m, n, omega_mode, omega(i), I2, I6),...
        [f_mag f_phase], options);
   f_ave_mag1(i)     = f_ave1(1);
```

```
    f_ave_phase1(i)   = f_ave1(2)*pi/180;
    f_ave_control1(i) = f_ave_mag1(i)*exp(1i*f_ave_phase1(i));
    fq_ave1           = [f_disturbance f_ave_control1(i)];

    % Control over 15 modes, frequency specific weights
    f_vv = fminsearch(@(f_control)clamped_VV_objective(x_disturbance,...
        y_disturbance, f_disturbance, x_control, y_control, f_control,...
        sensor_vv, m, n, omega_mode, omega(i), I2, I6), [f_mag f_phase],...
        options);
    f_vv_mag        = f_vv(1);
    f_vv_phase      = f_vv(2)*pi/180;
    f_vv_control(i) = f_vv_mag*exp(1i*f_vv_phase);
    fq_vv           = [f_disturbance f_vv_control(i)];

    % Calculate the power
    power_nc(i, :) = myPower(el_x, el_y, x_disturbance, y_disturbance,...
        f_disturbance, lambda_m, lambda_n,omega_mode, omega(i), I2, I6,...
        Lx, Ly, distances);
    power_ave_control1(i, :) = myPower(el_x, el_y, xq, yq, fq_ave1,...
        lambda_m, lambda_n, omega_mode,omega(i),I2,I6,Lx,Ly,distances);
    power_vv_control(i, :) = myPower(el_x, el_y, xq, yq, fq_vv,lambda_m,...
        lambda_n, omega_mode,omega(i),I2,I6,Lx,Ly,distances);

end
toc

% Create plots
lw = 2;
fs = 18;

log_power_nc = 10*log10(power_nc/(1*10^-12));
log_ave_con1 = 10*log10(power_ave_control1/(1*10^-12));
log_vv_con   = 10*log10(power_vv_control/(1*10^-12));

figure()
nc = plot(freq, log_power_nc(:, 1), 'k', 'LineWidth', lw);
hold('on')
cl_ave = plot(freq, log_ave_con1(:, 1), '--r', 'LineWidth', lw);
vv     = plot(freq, log_vv_con(:, 1), 'g', 'LineWidth', lw);
hold('off')
grid('on')
set(gca,'FontSize', fs)
xlabel('Frequency (Hz)', 'FontSize', fs)
ylabel('Power level (dB re 1e^{-12} W)', 'FontSize', fs)
legend([nc cl_ave vv], 'No Control', 'WSSG', 'Volume Velocity')
xlim([freq(1) freq(end)])
ylim([0  160])

figure()
plot(freq,log_power_nc(:, 1) - log_ave_con1(:, 1), 'k','LineWidth', lw)
hold('on')
plot(freq, log_power_nc(:, 1) - log_vv_con(:, 1), 'g', 'LineWidth', lw)
hold('off')
grid('on')
set(gca,'FontSize',fs)
xlabel('Frequency (Hz)','FontSize',fs)
ylabel('Power (dB)','FontSize',fs)
legend('w/ Clamped Ave Weights','w/ Frequency Dependent Weights')
xlim([freq(1) freq(end)])

% Figures of the sound radiation of the radiation mode shapes
figure()
nc = plot(freq, log_power_nc(:, 2), 'k', 'LineWidth', lw);
hold('on')
ave = plot(freq, log_ave_con1(:, 2), '--r', 'LineWidth', lw);
vv  = plot(freq, log_vv_con(:, 2), 'g', 'LineWidth', lw);
hold('off')
grid('on')
```

```
set(gca, 'FontSize', fs)
xlabel('Frequency (Hz)', 'FontSize', fs)
ylabel('Power level (dB re 1e^{-12})', 'FontSize', fs)
legend([nc ave vv], 'No Control', 'WSSG', 'Volume Velocity')
xlim([freq(1) freq(end)])
ylim([0 160])

figure()
nc = plot(freq, log_power_nc(:, 3), 'k', 'LineWidth', lw);
hold('on')
ave = plot(freq, log_ave_con1(:, 3), '--r', 'LineWidth', lw);
vv  = plot(freq, log_vv_con(:, 3), 'g', 'LineWidth', lw);
hold('off')
grid('on')
set(gca, 'FontSize', fs)
xlabel('Frequency (Hz)', 'FontSize', fs)
ylabel('Power level (dB re 1e^{-12})', 'FontSize', fs)
legend([nc ave vv], 'No Control', 'WSSG', 'Volume Velocity')
xlim([freq(1) freq(end)])
ylim([0 160])

figure()
nc = plot(freq, log_power_nc(:, 4), 'k', 'LineWidth', lw);
hold('on')
ave = plot(freq, log_ave_con1(:, 4), '--r', 'LineWidth', lw);
vv  = plot(freq, log_vv_con(:, 4), 'g', 'LineWidth', lw);
hold('off')
grid('on')
set(gca, 'FontSize', fs)
xlabel('Frequency (Hz)', 'FontSize', fs)
ylabel('Power level (dB re 1e^{-12})', 'FontSize', fs)
legend([nc ave vv], 'No Control', 'WSSG', 'Volume Velocity')
xlim([freq(1) freq(end)])
ylim([0 160])

figure()
nc = plot(freq, log_power_nc(:, 5), 'k', 'LineWidth', lw);
hold('on')
ave = plot(freq, log_ave_con1(:, 5), '--r', 'LineWidth', lw);
vv  = plot(freq, log_vv_con(:, 5), 'g', 'LineWidth', lw);
hold('off')
grid('on')
set(gca, 'FontSize', fs)
xlabel('Frequency (Hz)', 'FontSize', fs)
ylabel('Power level (dB re 1e^{-12})', 'FontSize', fs)
legend([nc ave vv], 'No Control', 'WSSG', 'Volume Velocity')
xlim([freq(1) freq(end)])
ylim([0 160])

% Calculate the overall attenuation
overall_nc  = freq_inc*trapz(power_nc);
overall_ave = freq_inc*trapz(power_ave_control1);
overall_vv  = freq_inc*trapz(power_vv_control);

overall_nc  = 10*log10(overall_nc/(1*10^-12));
overall_ave = 10*log10(overall_ave/(1*10^-12));
overall_vv  = 10*log10(overall_vv/(1*10^-12));

disp(['Overall attenuation, average weights: '...
    num2str(overall_nc - overall_ave)])
disp(['Overall attenuation, volume velocity: '...
    num2str(overall_nc - overall_vv)])

%-----------------------------------------%
% I checked this on 5/1/2012 and it is correct
%-----------------------------------------%

end
```

```
% Function to calculate the total sound power
function out = myPower(el_x, el_y, xq, yq, fq, lambda_m, lambda_n,...
    omega_mode, omega, I2, I6, Lx, Ly, distances)
% Receives inputs of number of elements in x (el_x), number of elements in
% y (el_y), the positions of applied forces and their magnitudes, the modes
% to look at power over, and the frequency to evaluate the power at.
% xq,yq,fq,m,n can all be vectors the others are expected as scalars.

% Plate parameters                % Units: m
space_x = Lx/el_x;
x       = space_x/2:space_x:Lx;            % Units: m
space_y = Ly/el_y;
y       = space_y/2:space_y:Ly;
Ae      = (space_x)*(space_y);  % Elemental area of an individual element
rho_air = 1.204;                % Density of air, used only to calculate
                                % coeff of radiation resistance matrix
c       = 343;                  % Speed of sound in air at room temp,
                                % Units: m/s
k       = omega/c;
[x y]   = meshgrid(x,y);


% Calculate the radiation resistance matrix
coeff = omega^2*rho_air*Ae^2/(4*pi*c);
RR    = coeff*sinc(k*distances/pi);

% Get the transverse velocity of each element
vel_el = zeros(el_x,el_y);

for i = 1:length(lambda_m)

    % Elemental velocity
    vel_el = vel_el + transverse_vel(x,y,xq,yq,fq,lambda_m(i),...
        lambda_n(i),omega_mode(i),omega,I2(i),I6(i),Lx,Ly);
end

vel_el = reshape(vel_el,numel(vel_el),1);

% Evaluate power for individual radiation modes, see page 170 of Fahy.
[Q, Lambda] = eig(RR);
y_tilde     = Q'*vel_el;
lambda      = diag(Lambda);

first_mode_power = lambda(end)*(y_tilde(end)*conj(y_tilde(end)));
second_mode_power= lambda(end-1)*(y_tilde(end-1)*conj(y_tilde(end-1)));
third_mode_power = lambda(end-2)*(y_tilde(end-2)*conj(y_tilde(end-2)));
fourth_mode_power= lambda(end-3)*(y_tilde(end-3)*conj(y_tilde(end-3)));

% Evaluate power
out = zeros(1, 5);

out(1) = real(vel_el'*RR*vel_el); % The real is not necessary, but
                                  % many of the numbers contain an
                                  % imaginary portion near machine
                                  % epsilon, and I got sick of the
                                  % warning.
out(2) = first_mode_power;
out(3) = second_mode_power;
out(4) = third_mode_power;
out(5) = fourth_mode_power;

end


% Function to calculate the matrix for radiation resistance. This was
% originally the distance function, but modified to calculate each value in
% RR while it goes
function dd = distance_mat(x_pos,y_pos)
% Accepts the wave number and two vectors containing the x and y positions,
% where each value in x corresponds to an entire column, and each value in
% y corresponds to an entire row of elements.
% With x and y vector of lengths three there will be 9 total elements,
% the 1,1 1,2 1,3 2,1 2,2 etc, and the output will be a 9x9 matrix
% containing the radiation resistance coupling between each element, with
% the top row containing resistances from the first element to all others,
% 2nd row corresponding to the second element, and it's resistances, and so
% on.

% Preallocate
dd = ones(length(x_pos)*length(y_pos));

% This calculates D. The inner two loops loop over the second point for
% the distance calculation, the outer two loops loop over the first
% point in the distance calculation.
m = 1;
for i = 1:length(x_pos)
    for j = 1:length(y_pos)

        n = 1;

        % For loops to loop over the second point
        for k = 1:length(x_pos)
            for l = 1:length(y_pos)

                % Calculate distance
                dd(m,n) = sqrt((x_pos(i) - x_pos(k))^2 + (y_pos(j) - ...
                    y_pos(l))^2);
                n = n + 1;

            end
        end
        m = m + 1;
    end
end

end


% Function to calculate the transverse velocity, to be used in the sound
% power calculation
function out = transverse_vel(x,y,xq,yq,fq,lambda_m,lambda_n,omega_mode,...
    omega,I2,I6,Lx,Ly)

% Plate parameters, plate material used is steel here
h        = 0.0030988;     % Plate thickness, Units: m
eta      = 0.02;     % Damping ratio, Unitless,I believe
rho      = 2700;      % Density, Units: kg/m^3
m_bar    = rho*h;
lambda_m = lambda_m*(1 + 1i*eta);   % Add damping to the system
lambda_n = lambda_n*(1 + 1i*eta);

vel_trans = 0;

% Loop for multiple input forces
for i = 1:length(fq)

    % Define coefficient
    C = fq(i)*shape(Lx,lambda_m,xq(i)).*shape(Ly,lambda_n,yq(i))./...
        (m_bar*I2.*I6.*(omega_mode.^2 - omega^2));

    % Get transverse velocity
    vel_trans = vel_trans + C.*shape(Lx,lambda_m,x).*...
        shape(Ly,lambda_n,y)*1i*omega;
end
```

```
out = vel_trans;

end


% Function for the shape function X
function out = shape(L,lambda,q)
% Function requires input of x, a position on the plate, the output is the
% evaluation. This is written so that it can accept a vector for lambda, q,
% or L individually, but not two or three at a time.

    out = J1(lambda*q/L) - J1(lambda)./H1(lambda).*H1(lambda*q/L);

end


% The integrand for I2
function out = I2_integrand(y,lambda_n,Ly)
% Function requires input of y, the position along the curve, for
% integration, will return the integrand operation. This is for the
% performance of Gauss Quadrature

    out = shape(Ly,lambda_n,y).^2;

end


% The integrand for I6
function out = I6_integrand(x,lambda_m,Lx)
% Function requires input of x, the position along the curve, for
% integration, will return the integrand operation. This is for the
% performance of Gauss Quadrature

    out = shape(Lx,lambda_m,x).^2;

end


% Smaller functions for the shape functions. I know they're simple, this
% just seemed the easiest way.
function out = J1(in)

    out = cosh(in) - cos(in);

end

function out = H1(in)

    out = sinh(in) - sin(in);

end
```

B.1.2 top_modes_clamped

```
 % Date:     3/27/2012
 % Purpose:  Function to determine the first n modes of the clamped-clamped
 %           plate using the formulation for clamped-clamped plates found in
 %           the paper "Active control of Structurally Radiated Sound from
 %           Plates" by Chia-Chi sung and C. T. Jan, published in JASA, July
 %           1997. This will calculate the frequencies and then order the
 %           modes from lowest to highest.
 % Notes: 1. This code gives a warning for max number of intervals reached,
 %           however it has been validated and the results are accurate for
%           the first 10 modes.
```

```
function out = top_modes_clamped(num_modes)

% Plate parameters, plate material used is steel here
Lx    = 0.483;     % Units: m
Ly    = 0.762;     % Units: m
h     = 0.0030988;     % Plate thickness, Units: m
E     = 68.9*10^9;  % Young's Modulus, Units: Pa
nu    = 0.33;      % Poissson's Ratio
rho   = 2700;      % Density, Units: kg/m^3
D     = E*h^3/(12*(1 - nu^2));
m_bar = rho*h;      % Density per unit area, Units: kg/m^2

% Get the values for lambda
lambda   = 3:2:5*num_modes^2;
lambda   = lambda*pi/2;   % This was found from the transcendental
                          % relationship cosh(lambda)cos(lambda) = 1

% Find the first n modes
freq = zeros(num_modes^2,3);
i    = 1;

for m = 1:num_modes
    for n = 1:num_modes

        I1 = quadgk(@(x)I1_integrand(x,lambda(m),Lx),0,Lx);
        I2 = quadgk(@(x)I2_integrand(x,lambda(n),Ly),0,Ly);
        I3 = quadgk(@(x)I3_integrand(x,lambda(m),Lx),0,Lx);
        I4 = quadgk(@(x)I4_integrand(x,lambda(n),Ly),0,Ly);
        I5 = quadgk(@(x)I5_integrand(x,lambda(n),Ly),0,Ly);
        I6 = quadgk(@(x)I6_integrand(x,lambda(m),Lx),0,Lx);

        % Get the natural frequency (in rad/s)
        omega_mn = sqrt(D*(I1*I2 + 2*I3*I4 + I5*I6)/(m_bar*I2*I6));
        % I compared the solutions to this with my FEA results and they are
        % very close, about 0.35% percent difference

        % Build output matrix of frequency and mode number
        freq(i,1) = m;
        freq(i,2) = n;
        freq(i,3) = 1/(2*pi)*omega_mn;    % Conversion to Hz

        i = i + 1;
    end
end

% Sort the rows
freq = sortrows(freq,3);

% Save top 30 frequencies
out = freq(1:num_modes,:);

end


% Function for the shape function X
function out = shape(l,lambda,q)
% Function requires input of x, a position on the plate, the output is the
% evaluation. This is written so that it can accept a vector for lambda, q,
% or l individually, but not two or three at a time.

out = cosh_cos_minus(lambda*q/l) - cosh_cos_minus(lambda)./...
    sinh_sin_minus(lambda).*sinh_sin_minus(lambda*q/l);

end


% Function for the second derivative of the shape function X
function out = shape_double_prime(l,lambda,q)
```

```
 % Function requires input of x, a position on the plate, the output is the
 % evaluation.

 out = (lambda/l).^2*(cosh_cos_plus(lambda*q/l) - cosh_cos_minus(lambda)...
     ./sinh_sin_minus(lambda).*sinh_sin_plus(lambda*q/l));

 end


 % Function for the fourth derivative of the shape function X
 function out = shape_quadruple_prime(l,lambda,q)
 % Function requires input of q, a generalized coordinate on the plate,
 % the output is the evaluation.

 out = (lambda/l).^4*(cosh_cos_minus(lambda*q/l) - cosh_cos_minus(lambda)...
     ./sinh_sin_minus(lambda).*sinh_sin_minus(lambda*q/l));

 end


% The integrand for I1
function out = I1_integrand(x,lambda_m,Lx)
% Function requires input of x, the position along the curve, for
% integration, will return the integrand operation. This is for the
% performance of Gauss Quadrature

out = shape_quadruple_prime(Lx,lambda_m,x).*shape(Lx,lambda_m,x);

end


% The integrand for I2
function out = I2_integrand(y,lambda_n,Ly)
% Function requires input of y, the position along the curve, for
% integration, will return the integrand operation. This is for the
% performance of Gauss Quadrature

out = shape(Ly,lambda_n,y).^2;

end


% The integrand for I3
function out = I3_integrand(x,lambda_m,Lx)
% Function requires input of x, the position along the curve, for
% integration, will return the integrand operation. This is for the
% performance of Gauss Quadrature.

out = shape_double_prime(Lx,lambda_m,x).*shape(Lx,lambda_m,x);

end


% The integrand for I4
function out = I4_integrand(y,lambda_n,Ly)
% Function requires input of y, the position along the curve, for
% integration, will return the integrand operation. This is for the
% performance of Gauss Quadrature.

out = shape_double_prime(Ly,lambda_n,y).*shape(Ly,lambda_n,y);

end


% The integrand for I5
function out = I5_integrand(y,lambda_n,Ly)
% Function requires input of y, the position along the curve, for
% integration, will return the integrand operation. This is for the
% performance of Gauss Quadrature.

out = shape_quadruple_prime(Ly,lambda_n,y).*shape(Ly,lambda_n,y);

end


% The integrand for I6
function out = I6_integrand(x,lambda_m,Lx)
% Function requires input of x, the position along the curve, for
% integration, will return the integrand operation. This is for the
% performance of Gauss Quadrature

out = shape(Lx,lambda_m,x).^2;

end


% Smaller functions for the shape functions. I know their simple, this just
% seemed the easiest way.
function out = cosh_cos_minus(in)

out = cosh(in) - cos(in);

end

function out = sinh_sin_minus(in)

out = sinh(in) - sin(in);

end

function out = cosh_cos_plus(in)

out = cosh(in) + cos(in);

end

function out = sinh_sin_plus(in)

out = sinh(in) + sin(in);

end
```

B.1.3 clamped_Vcomp_objective

```
% Date:     4/25/2012
% Purpose:  Program to calculate Vcomp for the clamped plate equation to
%           determine uniformity over the entire plate. Derivation of
%           solution of a clamped plate was found in the paper "Active
%           control of structurally radiated sound from plates" by Chia-Chi
%           Sung and C. T. Jan. Derivations here refer back to this.
% Notes: 1. The number of functions are a little out of control here, I
%           know, but there were so many small pieces that were used over
%           and overs, so functions seemed the best way. Otherwise the code
%           would have ended up with lines with an infinitely large number
%           of characters (approximately).
%        2. Properties:
%           Lx    = 0.483;     % Units: m
%           Ly    = 0.762;     % Units: m
%           h     = 0.0030988;;     % Plate thickness, Units: m
%           E     = 68.9*10^9;  % Young's Modulus, Units: Pa
%           nu    = 0.33;      % Poissson's Ratio
%           rho   = 2700;      % Density, Units: kg/m^3
%           D     = E*h^3/(12*(1 - nu^2));    % Bending stiffness, Units:Nm
%           m_bar = rho*h;      % Density per unit area, Units: kg/m^2
%           P     = 1;          % Magnitude of the applied force.
%        3. This has been modified from the 4_18 program to be vectorized
%           w.r.t. x and y, and to sum over the modes in a for loop, to
%           check the code. That was the mistake that I made on the simply
%           supported plate.


function out = clamped_Vcomp_objective(x_disturbance, y_disturbance,...
    f_disturbance, x_control, y_control, f_control, x_sense, y_sense, m,...
    n, omega_mode, omega, I2, I6)
% The objective function to calculate Vcomp for a clamped plate. Requires
% inputs listed above, the names are descriptive enough I think, and
% returns the value of Vcomp at the sensor location.

f_mag     = f_control(1);
f_phase   = f_control(2)*pi/180;
f_control = f_mag*exp(1i*f_phase);

% Plate geometry
Lx = 0.483;
Ly = 0.762;

% Organize inputs
xq = [x_disturbance x_control];
yq = [y_disturbance y_control];
fq = [f_disturbance f_control];

% Define the values for lambda
lambda   = [3.011 5:2:5*max(m)*max(n)];
lambda   = lambda*pi/2;   % This was found from the transcendental
                          % relationship cosh(lambda)cos(lambda) = 1
lambda_m = lambda(m);
lambda_n = lambda(n);

% Get Vcomp terms
velocities = Vcomp(x_sense,y_sense,xq,yq,fq,lambda_m,lambda_n,...
    omega_mode,omega,I2,I6,Lx,Ly);
vel_trans  = velocities(1);
vel_rockx  = velocities(2);
vel_rocky  = velocities(3);
vel_twist  = velocities(4);

% Square Vcomp terms
vel_trans = vel_trans.*conj(vel_trans);
vel_rockx = vel_rockx.*conj(vel_rockx);
vel_rocky = vel_rocky.*conj(vel_rocky);
vel_twist = vel_twist.*conj(vel_twist);

% Calculate the weights
alphas = weights(lambda_m,lambda_n,Lx,Ly);
alpha  = alphas(1);
beta   = alphas(2);
gamma  = alphas(3);
delta  = alphas(4);

% Vcomp!
out = alpha*vel_trans + beta*vel_rockx + gamma*vel_rocky + ...
    delta*vel_twist;

end


% Function to calculate the different terms of Vcomp
function out = Vcomp(x,y,xq,yq,fq,lambda_m,lambda_n,omega_mode,omega,I2,...
I6,Lx,Ly)
% The inputs to this function are in the following order: x and y
% coordinates of a single point on the plate, xq,yq are points of applied
% force. fq the magnitude of applied force (can be complex), lambda_m and
```

```
 % lambda_n the values for transcendental equation given in paper that
 % define given modes. omega_mode a vector of natural frequencies of given
 % modes, omega the frequency of vibration (I guess this is the frequency of
 % the applied force). I2 and I6 as defined in the paper. I know this is a
 % lot, some of these things are passed in to avoid multiple calculations
 % thus reducing computation time.
 %
 % The output is a vector containing values of Vcomp terms at a given point

 % Plate parameters, plate material used is steel here
 h        = 0.0030988;     % Plate thickness, Units: m
eta      = 0.02;     % Damping ratio, Unitless,I believe
rho      = 2700;      % Density, Units: kg/m^3
m_bar    = rho*h;
lambda_m = lambda_m*(1 + 1i*eta);   % Add damping to the system
lambda_n = lambda_n*(1 + 1i*eta);

% Preallocate
vel_trans = 0;
vel_rockx = 0;
vel_rocky = 0;
vel_twist = 0;

den = (m_bar*I2.*I6.*(omega_mode.^2 - omega^2))';

% Loop for multiple input forces
for i = 1:length(fq)

    % Define coefficient
    C = fq(i)*shape(Lx,lambda_m,xq(i)).*shape(Ly,lambda_n,yq(i))./den;

    % Define velocity terms
    vel_trans = vel_trans + C.*shape(Lx,lambda_m,x).*...
        shape(Ly,lambda_n,y)*1i*omega;
    vel_rockx = vel_rockx + C.*(H2(lambda_m*x/Lx) - J1(lambda_m)./...
        H1(lambda_m).*J1(lambda_m*x/Lx)).*shape(Ly,lambda_n,y).*...
        (lambda_m/Lx)*1i*omega;
    vel_rocky = vel_rocky + C.*shape(Lx,lambda_m,x).*...
        (H2(lambda_n*y/Ly) - J1(lambda_n)./H1(lambda_n).*...
        J1(lambda_n*y/Ly)).*(lambda_n/Ly)*1i*omega;
    vel_twist = vel_twist + C.*(H2(lambda_m*x/Lx) - J1(lambda_m)./...
        H1(lambda_m).*J1(lambda_m*x/Lx)).*(H2(lambda_n*y/Ly) - ...
        J1(lambda_n)./H1(lambda_n).*J1(lambda_n*y/Ly)).*...
        (lambda_m.*lambda_n/(Lx*Ly))*1i*omega;

end

out(1) = sum(vel_trans);
out(2) = sum(vel_rockx);
out(3) = sum(vel_rocky);
out(4) = sum(vel_twist);

%--------------------------------------------------------%
% I checked this on 5/1/2012 and it is correct.
%--------------------------------------------------------%

end

% Function for the shape function X
function out = shape(L,lambda,q)
% Function requires input of x, a position on the plate, the output is the
% evaluation. This is written so that it can accept a vector for lambda, q,
% or l individually, but not two or three at a time.

out = J1(lambda*q/L) - J1(lambda)./H1(lambda).*H1(lambda*q/L);

end
```

```
% Smaller functions for the shape functions. I know they're simple, this
% just seemed the easiest way.
function out = J1(in)

out = cosh(in) - cos(in);

end

function out = H1(in)

out = sinh(in) - sin(in);

end

function out = H2(in)

out = sinh(in) + sin(in);

end


% Calculate the average weights over the given modes
function out = weights(lambda_m,lambda_n,Lx,Ly)
% Required inputs are the lambda values, output is the average weights, in
% a vector w/ 4 components.

% Calculate weights
alpha = 1;
beta  = (Lx./lambda_m).^2;
gamma = (Ly./lambda_n).^2;
delta = (Lx*Ly./(lambda_m.*lambda_n)).^2;

beta  = mean(beta);
gamma = mean(gamma);
delta = mean(delta);

out = [alpha beta gamma delta];

%---------------------------------------------------------%
% I checked this on 4/19/2012 and it is correct.
%---------------------------------------------------------%
end
```

B.1.4 clamped_VV_objective

```
% Date:     5/24/2013
% Purpose:  This code is the objective function to be minimized with volume
%           velocity, to compare WSSG to alternative objective functions.
%           This is based on the volume velocity code from the rectangular
%           plate.

function out = clamped_VV_objective(x_disturbance, y_disturbance, ...
    f_disturbance, x_control, y_control, f_control, sensor_vv, m, n, ...
    omega_mode, omega, I2, I6)

f_control_mag   = f_control(1);
f_control_phase = f_control(2)*pi/180;
f_control       = f_control_mag*exp(1i*f_control_phase);

% Organize inputs
xq = [x_disturbance x_control];
yq = [y_disturbance y_control];
fq = [f_disturbance f_control];

% Define the values for lambda
lambda   = [3.011 5:2:5*m(end)*n(end)];
lambda   = lambda*pi/2;   % This was found from the transcendental
                          % relationship cosh(lambda)cos(lambda) = 1
lambda_m = lambda(m);
lambda_n = lambda(n);

% Break out sensor parameters
x_sense = sensor_vv.x;
y_sense = sensor_vv.y;

% Preallocate matrices for FOR loops
el_velocity = zeros(size(x_sense));

% Calculate the volume velocity
for i = 1:length(m)

    el_velocity = el_velocity + 1i*omega*displacement(x_sense, y_sense,...
        xq, yq, fq, lambda_m(i), lambda_n(i), omega_mode(i), omega,...
        I2(i), I6(i));

end

out = sensor_vv.Ae*sum(sum(el_velocity));
out = out.*conj(out);

end


% Function to calculate the transverse velocity. The equation was taken
% from the transverse velocity term used for Vcomp.
function out = displacement(x, y, xq, yq, fq, lambda_m, lambda_n,...
    omega_mode, omega, I2, I6)

% Define material properties (steel) and other parameters
Lx       = 0.483;     % Units: m
Ly       = 0.762;     % Units: m
h        = 0.0030988;     % Plate thickness, Units: m
eta      = 0.02;
rho      = 2700;      % Density, Units: kg/m^3
m_bar    = rho*h;
lambda_m = lambda_m*(1 + 1i*eta);    % Add damping to the system
lambda_n = lambda_n*(1 + 1i*eta);

% Preallocate
out = 0;

% Loop for multiple input forces
for i = 1:length(fq)

    % Define coefficient
    C = fq(i)*shape(Lx, lambda_m, xq(i)).*shape(Ly, lambda_n, yq(i))./...
        (m_bar*I2.*I6.*(omega_mode.^2 - omega^2));

    % Define velocity terms
    out = out + C.*shape(Lx, lambda_m, x).*shape(Ly, lambda_n, y);

end


end


% Function for the shape function X
function out = shape(l, lambda, q)
% Function requires input of x, a position on the plate, the output is the
% evaluation. This is written so that it can accept a vector for lambda, q,
% or l individually, but not two or three at a time.

out = cosh_cos_minus(lambda*q/l) - cosh_cos_minus(lambda)./...
```

```
    sinh_sin_minus(lambda).*sinh_sin_minus(lambda*q/l);

end


% Smaller functions for the shape functions. I know their simple, this just
% seemed the easiest way.
function out = cosh_cos_minus(in)

out = cosh(in) - cos(in);

end

function out = sinh_sin_minus(in)

out = sinh(in) - sin(in);

end
```

B.2 Ribbed Plate Code

The ribbed plate code was written using a Matlab class definition. The class definition file, along with an example script of how it is run, is included. In order to run this code the data files which it was written for must be used, contact Jon Blotter to obtain this data.

B.2.1 WSSGTransFunc

```
% Date:     5/22/2013
% Purpose:  This is the class definition file for the wssg transfer
%           function class, to define the methods, and allow for increased
%           flexibility in how I run the FEA simulations for multiple
%           control, disturbance, and sensor locations. I think that this
%           is a brilliant piece of code, almost like real programming!
% Notes: 1.

classdef WSSGTransFunc < handle

% Define class properties
properties (SetAccess = private)

    ModelName
    Displacement
    XNode
    YNode
    CurrentFrequency
    Frequencies

    % WSSG sensor location (default)
    XSensor = 0.286;     % Units: m
    YSensor = 0.432;     % Units: m

    % WSSG sensor spacing for calculation of finite difference
    % derivatives
    XSpace  = 0.0254*1;  % Units: m
    YSpace  = 0.0254*1;  % Units: m

    ControlModelName
    Objective
    ControlMode = 'single'; % Default control mode is single, most of
                            % the simulations have been done this way
    OptimizedForces
```

```
        RadiatedPowerWSSG
        RadiatedPowerVV
        RadiatedPowerNC
        OverallAttenuation

    end

    properties (SetAccess = private, Hidden)

        % This is a matrix which stores the transfer functions for the
        % current model and all of the models to run the optimization
        % routine with. It is 4 dimensional with the first dimension
        % being the same length as the number of frequencies the FEA model
        % obtained, the second dimesion being 4, for the four sensor
        % locations required to get WSSG, the third being the number of
        % models to optimize over, and the fourth being the number of WSSG
        % sensors on the plate.
        TransferFuncsSense
        TransferFuncsDisplacement
        NumSensors       = 1;
        NumControlModels
        NumFrames
        Distances
        XSenseLoc % This gives the locations for the four sensors required
                  % for the FD derivatives
        YSenseLoc % This gives the locations for the four sensors required for
                  % the FD derivatives
        DisturbanceForce = 1;
        ObjectiveFunction % Variable to contain the function handle to be
                          % optimized
        ObjectiveFunctionStruct     % This is a struct to contain all of the
                                    % random data required for the various
                                    % objective functions
        OptimizedForcesVV
        OptimizedForcesWSSG
        FreqInc
        RadiatedPowerWSSGnodB
        RadiatedPowerVVnodB
        RadiatedPowerNCnodB

    end


    % Define class methods
    methods

        % Constructor, requires an input of one of the model names, with
        % '.mat' suffix attached.
        function obj = WSSGTransFunc(name)

            data = obj.getData(name);

            % Break out the public data
            obj.ModelName    = name;
            obj.XNode        = data.node_x;
            obj.YNode        = data.node_y;
            obj.Displacement = data.displacement;
            obj.Frequencies  = data.frequencies;
            obj.Distances    = obj.getDistanceMat(reshape(obj.XNode,...
                numel(obj.XNode), 1), reshape(obj.YNode, numel(obj.YNode), 1));

            % Break out the private data
            [~, ~, obj.NumFrames] = size(obj.Displacement);
            obj.FreqInc           = obj.Frequencies(3) - obj.Frequencies(2);

           % Define the default weights to be the simply supported weights
           obj.ObjectiveFunctionStruct.Weights = [1 0.012104891442120 ...
               0.017173361543038 1.865392869919163e-04];
```

```
end


% Method to change the model name. I will probably delete this, I
% think it will be confusing, not useful
function changeModel(obj, name)

    data = obj.getData(name);

    % Break out the data
    obj.ModelName    = name;
    obj.XNode        = data.node_x;
    obj.YNode        = data.node_y;
    obj.Displacement = data.displacement;
    obj.Frequencies  = data.frequencies;

    % Change the current frequency
    if ~isempty(obj.CurrentFrequency)
        setFreq(obj, obj.CurrentFrequency)
    end

end


% Method to set the frequency of interest. The function will find
% the closest available frequency.
function setFreq(obj, freq)

    % Get the frequencies
    frequencies = obj.Frequencies;

    % Find closest available frequency to desired (within 0.1 Hz)
    index = find(frequencies < freq);
    index = index(end);

    if abs(freq - frequencies(index)) > abs(freq - frequencies...
            (index + 1))
        index = index + 1;
    end

    % Set the current frequency
    obj.CurrentFrequency = frequencies(index);

end


% Method to set the sensor locations of interest. The sensor_x and
% sensor_y values can be vectors. Be aware that the optimization
% routine will run over all of the sensor locations
function setSensor(obj, sensor_x, sensor_y)

    % Exception handling if the sensors vector lengths don't match
    if length(sensor_x) ~= length(sensor_y)

        % Create the MException object to represent the error
        err = MException('Sensors:vectorLengthMismatch', ...
            'Argument vector lengths must match');

        % Throw the error
        throw(err)
    end

    % Define the sensor locations, correct to column vectors for
    % the getTransferFunction routine.
    if isrow(sensor_x)
        sensor_x = sensor_x';
    end
    if isrow(sensor_y)
        sensor_y = sensor_y';
    end

    obj.XSensor    = sensor_x;
    obj.YSensor    = sensor_y;
    obj.NumSensors = length(sensor_x);

    % Check to see if the objective function has been set. If it
    % has reset it so that it will reacquire the data
    if ~isempty(obj.Objective)
        obj.getTransferFunctions
    end

end


% Method to set the sensor spacing
function setSpacing(obj, space_x, space_y)

    obj.XSpace = space_x;
    obj.YSpace = space_y;

end


% Method to define what other models will be used for control
function setControlModel(obj, varargin)

    obj.ControlModelName = varargin;
    obj.NumControlModels = length(varargin);

    % Correct/replace the data if the transfer functions have
    % already been found
    if ~isempty(obj.Objective)
        obj.getTransferFunctions
    end

end


% Method to set the mode of optimization, single or multi-force
% optimization. Inputs should be either 'single' or 'multi'
function setControlMode(obj, mode)

    obj.ControlMode = mode;

end


% Method to set the objective function to be used in the
% optimization routine. Right now it accepts flags which toggle
% between the objective function options. The flags are as follows:
%   1. WSSG with clamped plate average weights
%   2. WSSG with frequency dependent weights
%   3. WSSG with simply supported plate average weights
%   4. Volume velocity
function setObjectiveFunction(obj, flag)

    switch(flag)
        case 1
            obj.Objective         = 'WSSG with clamped plate weights';
            obj.NumSensors        = length(obj.XSensor);
            obj.ObjectiveFunction = @WSSGObjective;

            % Get the transfer functions
            obj.getTransferFunctions
            % Create the objective function struct
            obj.ObjectiveFunctionStruct.Weights = ...
                [1 0.005946702877783   0.008967650137259...
                0.000049162812180];

        case 2
            obj.Objective    = 'WSSG with frequency dependent weights';
            obj.NumSensors   = length(obj.XSensor);
            obj.ObjectiveFunction = @WSSGObjective;

            % Get the transfer functions
            obj.getTransferFunctions

            % Create the objective function struct
            obj.getFreqDepWeights();

        case 3
            obj.Objective = 'WSSG with simply-supported plate weights';
            obj.NumSensors        = length(obj.XSensor);
            obj.ObjectiveFunction = @WSSGObjective;

            % Get the transfer functions
            obj.getTransferFunctions

            % Create the objective function struct
            obj.ObjectiveFunctionStruct.Weights = [1 ...
                0.012104891442120 0.017173361543038 ...
                1.865392869919163e-04];

        case 4
            obj.Objective         = 'Volume velocity';
            obj.NumSensors        = 1;
            obj.ObjectiveFunction = @VolVelObjective;

            % Get the transfer functions
            obj.getTransferFunctions

            % Create the objective function struct
            obj.ObjectiveFunctionStruct.Ae = 9.830288461538463e-05;
            obj.ObjectiveFunctionStruct.Frequencies = obj.Frequencies;
            obj.ObjectiveFunctionStruct.TransferFuncsDisplacement = ...
                obj.TransferFuncsDisplacement;

        case 5
            obj.Objective =...
          'WSSG with simply-supported plate free ribs average weights';
            obj.NumSensors        = length(obj.XSensor);
            obj.ObjectiveFunction = @WSSGObjective;

            % Get the transfer functions
            obj.getTransferFunctions

            % Create the objective function struct
            obj.ObjectiveFunctionStruct.Weights = [1 ...
                0.017548728339548 0.002711484661583 0.000016626340845];

    end

end


% Method to run the control optimization routine using the current
% model as the disturbance and the submitted models as the control.
% Extra arguments are the names of the control models
function runOptimizer(obj)

    % Breakout some values
    frequencies = obj.Frequencies;
```

```
% Define the disturbance force
fd = obj.DisturbanceForce;

% The optimization routine
if strcmp(obj.ControlMode, 'single')

    % Preallocate
    optimized_forces = zeros(obj.NumFrames,obj.NumControlModels,...
        obj.NumSensors);

    % Run the optimization routine to get the optimal control force
    for i = 1:obj.NumSensors

        % Get the disturbance transfer function
        transfer_funcs.tf_disturbance    =...
            obj.TransferFuncsSense(:, :, 1, i);
        transfer_funcs.sensor_location_x =...
            obj.XSenseLoc(2*i - 1:2*i, 1:2);
        transfer_funcs.sensor_location_y =...
            obj.YSenseLoc(2*i - 1:2*i, 1:2);

        for j = 1:obj.NumControlModels

            % Get the control transfer function
            transfer_funcs.tf_control = obj.TransferFuncsSense...
                (:, :, j + 1, i); % The 'j+1' because the first
                                  % model is the disturbance

            % Predefine initial guesses for magnitude and
            % phase
            f_control_mag   = 1;
            f_control_phase = pi;

            % To check for minimization
            val = zeros(obj.NumFrames, 1);

            for k = 1:obj.NumFrames

                disp(['Frequency ' num2str(frequencies(k))])

                % Control
                [fc, val(k)]              = fminsearch(@(fc)...
                    FunctionShell(obj, transfer_funcs, fd, fc,...
                    k), [f_control_mag, f_control_phase]);
                f_control_mag             = fc(1);
                f_control_phase           = fc(2);
                optimized_forces(k, j, i) = f_control_mag*...
                    exp(1i*f_control_phase);

            end
        end
    end

elseif strcmp(obj.ControlMode, 'multi')

    % Preallocate
    optimized_forces = zeros(obj.NumFrames,...
        obj.NumControlModels, obj.NumSensors);

    % Run the optimization routine to get the optimal control force
    for i = 1:obj.NumSensors

        % Get the disturbance transfer function
        transfer_funcs.tf_disturbance    =...
            obj.TransferFuncsSense(:, :, 1, i);
        transfer_funcs.sensor_location_x =...
            obj.XSenseLoc(2*i - 1:2*i, 1:2);
            transfer_funcs.sensor_location_y =...
                obj.YSenseLoc(2*i - 1:2*i, 1:2);

            % Get the control transfer functions
            transfer_funcs.tf_control = obj.TransferFuncsSense...
                (:, :, 2:end, i);

            % Prediefine initial guesses for magnitude and phase
            f_control_mag   = 1;
            f_control_phase = -pi;
            f_control_mag   = f_control_mag*ones...
                (1, obj.NumControlModels);
            f_control_phase = f_control_phase*ones...
                (1, obj.NumControlModels);

            % To check for minimization
            val = zeros(obj.NumFrames, 1);

            for j = 1:obj.NumFrames

                disp(['Frequency ' num2str(frequencies(j))])

                % Control
                [fc, val(j)]              = fminsearch(@(fc)...
                    FunctionShell(obj, transfer_funcs, fd, fc, j),...
                    [f_control_mag, f_control_phase]);
                f_control_mag             = fc(1:end/2);
                f_control_phase           = fc(end/2+1:end);
                optimized_forces(j, :, i) = f_control_mag.*...
                    exp(1i*f_control_phase);

            end
        end

    else
        % Create the MException object to represent the error
        err = MException('ControlMode:incorrectControlMode',...
            'Check control mode settings');

        % Throw the error
        throw(err)
    end

    % Assign the optimized forces. If previous values have already
    % been placed here this will overwrite them. To maintain
    % continuity power will need to be calculated between runs, and
    % power values will be saved.
    if ~strcmp(obj.Objective, 'Volume velocity')
        obj.OptimizedForcesWSSG = optimized_forces;
    end

    obj.OptimizedForces = optimized_forces;

end


% Method to calculate the radiated sound power given a series of
% forces.
function getRadiatedSoundPower(obj)

    % Frequency increment
    freq_inc = obj.FreqInc;

    % Break out the transfer function struct
    tf_displacement = obj.TransferFuncsDisplacement;
    node_x          = obj.XNode;
    node_y          = obj.YNode;
```

```
% Get the disturbance and control forces
fd = obj.DisturbanceForce;
fc = obj.OptimizedForces;

% Material properties
rho_air = 1.204;     % Density of air, used only to calculate coeff
                     % of radiation resistance matrix
c       = 343;       % Speed of sound in air at room temp,Units:m/s

% Get element spacing and locations
delta_x = node_x(1, 2);
delta_y = node_y(2, 1);

Ae = delta_x*delta_y;

% Calculate the radiated power without control
radiated_power_nc    = zeros(obj.NumFrames, 1);
radiated_power_nc_db = zeros(obj.NumFrames, 1);

% Calculate the radiated power
if strcmp(obj.ControlMode, 'single')

    % Preallocate
    radiated_power    = zeros(size(obj.OptimizedForces));
    radiated_power_db = zeros(size(radiated_power));
    overall_atten    = zeros(obj.NumControlModels, obj.NumSensors);

    for i = 1:obj.NumSensors
        for j = 1:obj.NumControlModels
            for k = 1:obj.NumFrames

                disp(['Sound Power for Frequency: '...
                    num2str(obj.Frequencies(k))])

                % Calculate wavenumber
                omega       = 2*pi*obj.Frequencies(k);
                wave_number = omega/c;

                % Calculate the radiation resistance matrix
                coeff = omega^2*rho_air*Ae^2/(4*pi*c);
                RR    = coeff*sinc(wave_number*obj.Distances/pi);

                % Get the transverse velocity of each element
                vel_el_disturbance = 1i*omega*fd*reshape(...
                    tf_displacement(:, :, k, 1), numel(...
                    tf_displacement(:, :, k, 1)), 1);
                vel_el_control     = 1i*omega*fc(k, j, i)*...
                    reshape(tf_displacement(:, :, k, j+1),...
                    numel(tf_displacement(:, :, k, j+1)), 1);

                vel_el = vel_el_disturbance + vel_el_control;

                % Evaluate power
                radiated_power(k, j, i) = real(vel_el'*RR*vel_el);
                % The real is not necessary, but many of the
                % numbers contain an imaginary portion near machine
                % epsilon, and I got sick of the warning.

                % Evaluate power without control
                if (j == 1) && (i == 1)
                    radiated_power_nc(k) = real(...
                        vel_el_disturbance'*RR*vel_el_disturbance);
                end

            end

            % Calculate the average attenuation over the
            % frequency range of interest
            overall_atten(j, i) = 10*log10(freq_inc*trapz...
                (radiated_power_nc(2:end))/(1*10^-12)) -...
                10*log10(freq_inc*trapz(radiated_power...
                (2:end, j, i))/(1*10^-12));

            % Get the radiated power in dB
            if (j == 1) && (i == 1)
                radiated_power_nc_db = 10*log10(...
                    radiated_power_nc(:)/(1*10^-12));
            end

            radiated_power_db(:, j, i) = 10*log10(...
                radiated_power(:, j, i)/(1*10^-12));

        end
    end

elseif strcmp(obj.ControlMode, 'multi')

    % Preallocate
    radiated_power    = zeros(obj.NumFrames, obj.NumSensors);
    radiated_power_db = zeros(size(radiated_power));
    overall_atten     = zeros(1, obj.NumSensors);

    for i = 1:obj.NumSensors
        for j = 1:obj.NumFrames

            disp(['Sound Power for Frequency: '...
                num2str(obj.Frequencies(j))])

            % Calculate wavenumber
            omega       = 2*pi*obj.Frequencies(j);
            wave_number = omega/c;

            % Calculate the radiation resistance matrix
            coeff = omega^2*rho_air*Ae^2/(4*pi*c);
            RR    = coeff*sinc(wave_number*obj.Distances/pi);

            % Get the transverse velocity of each element. THE
            % INDICES ARE INTENTIONALLY DIFFERENT FROM THE
            % PREVIOUS 'IF' STATEMENT, ALTHOUGH IT IS A LITTLE
            % CONFUSING. I MAY COME BACK LATER AND MAKE ALL THE
            % INDICES UNIFORM SO IT IS CLEAR.
            vel_el = 1i*omega*fd*reshape(tf_displacement(:, :,...
                j, 1), numel(tf_displacement(:, :, j, 1)), 1);

            % Evaluate power without control
            if i == 1
                radiated_power_nc(j) = real(vel_el'*RR*vel_el);
            end

            % Get velocity with control forces applied
            for k = 1:obj.NumControlModels
                vel_el = vel_el + 1i*omega*fc(j, k, i)*reshape...
                    (tf_displacement(:, :, j, k+1), numel(...
                    tf_displacement(:, :, j, k+1)), 1);
            end

            % Evaluate power
            radiated_power(j, i) = real(vel_el'*RR*vel_el);
            % The real is not necessary, but many of the numbers
            % contain an imaginary portion near machine epsilon,
            % and I got sick of the warning.

        end

        % Calculate the average attenuation over the
        % frequency range of interest
```

```
            overall_atten(i) = 10*log10(freq_inc*trapz(...
                radiated_power_nc(2:end))/(1*10^-12)) - 10*log10...
                (freq_inc*trapz(radiated_power(2:end, i))/(1*10^-12));

            % Get the radiated power in dB
            if i == 1
                radiated_power_nc_db = 10*log10(radiated_power_nc/...
                    (1*10^-12));
            end

            radiated_power_db(:, i) = 10*log10(radiated_power(:, i)/...
                (1*10^-12));

        end

    end

    % Assign the output, radiated power in dB
    if ~strcmp(obj.Objective, 'Volume velocity')
        obj.RadiatedPowerWSSG     = radiated_power_db;
        obj.RadiatedPowerWSSGnodB = radiated_power;
    else
        obj.RadiatedPowerVV     = radiated_power_db;
        obj.RadiatedPowerVVnodB = radiated_power;
    end
    obj.RadiatedPowerNC     = radiated_power_nc_db;
    obj.RadiatedPowerNCnodB = radiated_power_nc;
    obj.OverallAttenuation = [obj.OverallAttenuation(:) overall_atten];

end


% Function to plot the radiated sound power
function plotSoundPower(obj)

    % Set the plotting parameters
    fs = 18;
    lw = 2;

    % Get the matrix dimensions
    [~, n, p] = size(obj.RadiatedPowerWSSG); % m - number of frames,
    % n - number of control models for single, number of sensors for
    % multi, p - number of sensors for single

    if strcmp(obj.ControlMode, 'single')

        for i = 1:p

            % Preallocate
            plot_handles   = zeros(1, n+1);
            legend_entries = cell(1, n+1);

            % Create the figures
            figure()
            line_colors = hsv(n);
            plot_handles(1)   = plot(obj.Frequencies, ...
                obj.RadiatedPowerNC, 'LineWidth', lw, 'Color', 'k');
            legend_entries{1} = 'No Control';
            hold('on')
            for j = 1:n
                plot_handles(j+1)   = plot(obj.Frequencies, ...
                    obj.RadiatedPowerWSSG(:, j, i), 'LineWidth', lw,...
                    'Color', line_colors(j, :));
                legend_entries{j+1} = ['Model ' num2str(j)];
            end
            hold('off')
            set(gca, 'FontSize', fs)
            xlabel('Frequency (Hz)', 'FontSize', fs)
            ylabel('Power (dB)', 'FontSize', fs)
            title(['Radiated Power vs. Frequency at Sensor '...
                num2str(i)], 'FontSize', fs)
            legend(plot_handles, legend_entries{:})
            grid('on')
            xlim([0 obj.Frequencies(end)])
            ylim([0 160])

        end

    elseif strcmp(obj.ControlMode, 'multi')

        % Preallocate
        plot_handles   = zeros(1, n+1);
        legend_entries = cell(1, n+1);

        % Create the figure
        figure()
        line_colors = hsv(n);
        plot_handles(1)   = plot(obj.Frequencies, ...
            obj.RadiatedPowerNC, 'LineWidth', lw, 'Color', 'k');
        legend_entries{1} = 'No Control';
        hold('on')
        for i = 1:n
            plot_handles(i+1)   = plot(obj.Frequencies,...
                obj.RadiatedPowerWSSG(:, i), 'LineWidth', lw,...
                'Color', line_colors(i, :));
            legend_entries{i+1} = ['Model ' num2str(i)];
        end
        hold('off')
        set(gca, 'FontSize', fs)
        xlabel('Frequency (Hz)', 'FontSize', fs)
        ylabel('Power (dB)', 'FontSize', fs)
       title('Radiated Power vs. Frequency for Multi-force Control',...
           'FontSize', fs)
        legend(plot_handles, legend_entries{:})
        grid('on')
        xlim([0 obj.Frequencies(end)])
        ylim([0 160])

    end

end


% Function because I'm sick of writing out the three main functions
% to create the control plots
function doIt(obj)

    obj.runOptimizer
    obj.getRadiatedSoundPower
    obj.plotSoundPower

end


% Method to Plot Displacement at any frequency
function plotDisplacement(obj, varargin)

    % Check for current frequency
    if isempty(varargin)

        % Check to see if current frequency is set
        if isempty(obj.CurrentFrequency)
            err = MException('FrequencyError:noFrequencySet', ...
                'No frequency specified');

            % Throw the error
            throw(err)
        end

    else
        obj.setFreq(varargin{1});
    end

    % Set the frequency
    freq = obj.CurrentFrequency;

    % Plot parameters
    fs      = 18;
    frindex = find(obj.Frequencies == freq, 1, 'last');

    % Create plots
    figure()
    pcolor(obj.XNode, obj.YNode, obj.Displacement(:, :, frindex))
    colorbar('FontSize', fs)
    set(gca, 'FontSize', fs)
    xlabel('L_x', 'FontSize', fs)
    ylabel('L_y', 'FontSize', fs)
    zlabel('Displacement(m)', 'FontSize',fs)
    title(['Displacement at ', num2str(freq), ' Hz'], 'FontSize', fs)
    shading('interp')
    axis('image')
    xlim([0 0.483])
    ylim([0 0.762])

    if ~isempty(obj.OptimizedForces)
        if strcmp(obj.ControlMode, 'single')
            for i = 1:obj.NumSensors
                for j = 1:obj.NumControlModels

                    controlDisplacement = obj.OptimizedForces(...
                        frindex, j, i) * ...
                       obj.TransferFuncsDisplacement(:, :, frindex, j);
                    controlPlot = abs(controlDisplacement).*cos(...
                        angle(controlDisplacement))+ ...
                        obj.DisturbanceForce * obj.Displacement(:,:,...
                        frindex);

                    % Create control plots
                    figure()
                    pcolor(obj.XNode, obj.YNode, controlPlot)
                    colorbar('FontSize', fs)
                    set(gca, 'FontSize', fs)
                    xlabel('L_x', 'FontSize', fs)
                    ylabel('L_y', 'FontSize', fs)
                    zlabel('Displacement(m)', 'FontSize',fs)
                    title(['Displacement with model ' num2str(j) ...
                        ' at sensor ', num2str(i),' and' num2str...
                        (freq), ' Hz'], 'FontSize', fs)
                    shading('interp')
                    axis('image')
                    xlim([0 0.483])
                    ylim([0 0.762])

                end
            end

        elseif strcmp(obj.ControlMode, 'multi')
            for i = 1:obj.NumSensors

                % Preallocate for controlDisplacement
                controlDisplacement = zeros(size(obj.Displacement...
                    (:, :, frindex)));

                % Calculate displacement field
```

```
                for j = 1:obj.NumControlModels
                    controlDisplacement = controlDisplacement + ...
                        obj.OptimizedForces(frindex, j, i) * ...
                       obj.TransferFuncsDisplacement(:, :, frindex, j);
                end
                controlPlot = abs(controlDisplacement).*cos(...
                    angle(controlDisplacement))+obj.DisturbanceForce...
                    * obj.Displacement(:, :, frindex);

                % Create control plots
                figure()

                pcolor(obj.XNode, obj.YNode, controlPlot)
                colorbar('FontSize', fs)
                set(gca, 'FontSize', fs)
                xlabel('L_x', 'FontSize', fs)
                ylabel('L_y', 'FontSize', fs)
                zlabel('Displacement(m)', 'FontSize',fs)
                title(['Displacement with Multi Control at sensor ',...
                    num2str(i),' and ' num2str(freq), ' Hz'],...
                    'FontSize', fs)
                shading('interp')
                axis('image')
                xlim([0 0.483])
                ylim([0 0.762])

            end
        end
    end

end


% Method to Plot WSSG
function plotWSSG(obj, varargin)

    % Check for current frequency
    if isempty(varargin)

        % Check to see if current frequency is set
        if isempty(obj.CurrentFrequency)
            err = MException('FrequencyError:noFrequencySet',...
                'No frequency specified');

            % Throw the error
            throw(err)
        end

    else
        obj.setFreq(varargin{1});
    end

    % Set the frequency
    freq = obj.CurrentFrequency;

    % Plot parameters
    fs      = 18;
    frindex = find(obj.Frequencies == freq, 1, 'last');

    % Correct node locations from numerical error
    delta_x = 0.483/48;
    delta_y = 0.762/78;

    % Calculate number of samples to skip
    num_samples_skip_x = obj.XSpace/delta_x;
    num_samples_skip_y = obj.YSpace/delta_y;

    num_samples_skip_x = round(num_samples_skip_x);
```

```
num_samples_skip_y = round(num_samples_skip_y);

% Actual spacing in x and y
act_x = num_samples_skip_x*delta_x;
act_y = num_samples_skip_y*delta_y;

% Get the weights
if strcmp(obj.Objective, 'WSSG with frequency dependent weights')
    weights = obj.ObjectiveFunctionStruct.Weights(frindex, :);
else
    weights = obj.ObjectiveFunctionStruct.Weights(:);
end

% Weights
alpha = weights(1);
beta  = weights(2);
gamma = weights(3);
delta = weights(4);

% Preallocate
x_loc          = zeros(79 - num_samples_skip_y, 49 - ...
    num_samples_skip_x);
y_loc          = zeros(size(x_loc));
WSSG          = zeros(size(x_loc));
w              = zeros(size(x_loc));
dw_dx          = zeros(size(x_loc));
dw_dy          = zeros(size(x_loc));
d2w_dx_dy      = zeros(size(x_loc));

% For loop for the calculation of Vcomp at desired points
for i = 1:(79 - num_samples_skip_y)
    for j = 1:(49 - num_samples_skip_x)

        % Displacement data
        p1 = obj.Displacement(i, j, frindex);
        p2 = obj.Displacement(i + num_samples_skip_x, j, frindex);
        p3 = obj.Displacement(i, j + num_samples_skip_y, frindex);
        p4 = obj.Displacement(i + num_samples_skip_x, j + ...
            num_samples_skip_y, frindex);

        % Spatial derivatives
        w(i, j)         = (p1 + p2 + p3 + p4)/4;
        dw_dx(i, j)     = (p2 - p1 + p4 - p3)/(2*act_x);
        dw_dy(i, j)     = (p1 - p3 + p2 - p4)/(2*act_y);
        d2w_dx_dy(i, j) = (p2 - p1 + p3 - p4)/(act_x*act_y);

        % Locations of Vcomp calculations
        x_loc(i, j) = obj.XNode(i, j) + (obj.XNode(i, j + ...
            num_samples_skip_x) - obj.XNode(i, j))/2;
        y_loc(i, j) = obj.YNode(i, j) + (obj.YNode(i + ...
            num_samples_skip_y, j) - obj.YNode(i, j))/2;

        % WSSG
        WSSG(i, j) = alpha*w(i, j)*conj(w(i, j)) + beta*dw_dx...
            (i, j)*conj(dw_dx(i, j)) + gamma*dw_dy(i, j)*conj...
            (dw_dy(i, j)) + delta*d2w_dx_dy(i, j)*conj...
            (d2w_dx_dy(i, j));

    end
end

% Calculate WSSG in dB
WSSG_dB = 20*log10(WSSG/max(max(WSSG)));

% Plot the data
figure()
subplot(2,2,1)
pcolor(x_loc, y_loc, w.^2)
```

```
xlabel('L_x (m)', 'FontSize', fs)
ylabel('L_y (m)', 'FontSize', fs)
title('w^2', 'FontSize', fs)
shading('interp')
xlim([0 0.483])
ylim([0 0.762])
colorbar('FontSize', fs)
set(gca, 'FontSize', fs)
axis('image')

subplot(2,2,2)
pcolor(x_loc, y_loc, dw_dx.^2)
xlabel('L_x (m)', 'FontSize', fs)
ylabel('L_y (m)', 'FontSize', fs)
title('(\partial w/\partial x)^2', 'FontSize', fs)
shading('interp')
xlim([0 0.483])
ylim([0 0.762])
colorbar('FontSize', fs)
set(gca, 'FontSize', fs)
axis('image')

subplot(2,2,3)
pcolor(x_loc, y_loc, dw_dy.^2)
xlabel('L_x (m)', 'FontSize', fs)
ylabel('L_y (m)', 'FontSize', fs)
title('(\partial w/\partial y)^2', 'FontSize', fs)
shading('interp')
xlim([0 0.483])
ylim([0 0.762])
colorbar('FontSize', fs)
set(gca, 'FontSize', fs)
axis('image')

subplot(2,2,4)
pcolor(x_loc, y_loc, d2w_dx_dy.^2)
xlabel('L_x (m)', 'FontSize', fs)
ylabel('L_y (m)', 'FontSize', fs)
title('(\partial^2 w/\partial x\partial y)^2', 'FontSize', fs)
shading('interp')
xlim([0 0.483])
ylim([0 0.762])
colorbar('FontSize', fs)
set(gca, 'FontSize', fs)
axis('image')

figure()
pcolor(x_loc, y_loc, WSSG_dB)
xlabel('L_x (m)', 'FontSize', fs)
ylabel('L_y (m)', 'FontSize', fs)
title(['Plate WSSG at ' num2str(freq) 'Hz'], 'FontSize', fs)
shading('interp')
xlim([0 0.483])
ylim([0 0.762])
colorbar('FontSize', fs);
set(gca, 'FontSize', fs)
axis('image')

if ~isempty(obj.OptimizedForcesWSSG)
if strcmp(obj.ControlMode, 'single')

w_calc = w;
dw_dx_calc = dw_dx;
dw_dy_calc = dw_dy;
d2w_dx_dy_calc = d2w_dx_dy;

for i = 1:obj.NumSensors
for j = 1:obj.NumControlModels
```

```
 % For loop for the calculation of Vcomp at desired points
 for m = 1:(79 - num_samples_skip_y)
 for n = 1:(49 - num_samples_skip_x)

 % Displacement data
 p1 = obj.TransferFuncsDisplacement(m, n, frindex, j);
 p2 = obj.TransferFuncsDisplacement(m + num_samples_skip_x, n,...
     frindex, j);
 p3 = obj.TransferFuncsDisplacement(m, n + num_samples_skip_y,...
     frindex, j);
 p4 = obj.TransferFuncsDisplacement(m + num_samples_skip_x, n +...
     num_samples_skip_y, frindex, j);

% Spatial derivatives
w_calc(m, n)          = w_calc(m, n) + obj.OptimizedForcesWSSG...
    (frindex, j, i)*(p1 + p2 + p3 + p4)/4;
dw_dx_calc(m, n)      = dw_dx_calc(m, n) +obj.OptimizedForcesWSSG...
    (frindex, j, i)*(p2 - p1 + p4 - p3)/(2*act_x);
dw_dy_calc(m, n)      = dw_dy_calc(m, n) +obj.OptimizedForcesWSSG...
    (frindex, j, i)*(p1 - p3 + p2 - p4)/(2*act_y);
d2w_dx_dy_calc(m, n) = d2w_dx_dy_calc(m, n) + ...
    obj.OptimizedForcesWSSG(frindex, j, i)*(p2 - p1 + p3 - p4)/...
    (act_x*act_y);

% Vcomp
WSSG(m, n) = alpha*w(m, n)*conj(w(m, n)) + beta*dw_dx(m, n)*...
    conj(dw_dx(m, n)) + gamma*dw_dy(m, n)*conj(dw_dy(m, n)) +...
    delta*d2w_dx_dy(m, n)*conj(d2w_dx_dy(m, n));

end
end

% Get WSSG in dB
WSSG_dB = 20*log10(WSSG/max(max(WSSG)));

% Plot the data
figure()
subplot(2,2,1)
pcolor(x_loc, y_loc, real(w.*conj(w)))
xlabel('L_x (m)', 'FontSize', fs)
ylabel('L_y (m)', 'FontSize', fs)
title('w^2', 'FontSize', fs)
shading('interp')
xlim([0 0.483])
ylim([0 0.762])
colorbar('FontSize', fs)
set(gca, 'FontSize', fs)
axis('image')

subplot(2,2,2)
pcolor(x_loc, y_loc, real(dw_dx.*conj(dw_dx)))
xlabel('L_x (m)', 'FontSize', fs)
ylabel('L_y (m)', 'FontSize', fs)
title('(\partial w/\partial x)^2', 'FontSize', fs)
shading('interp')
xlim([0 0.483])
ylim([0 0.762])
colorbar('FontSize', fs)
set(gca, 'FontSize', fs)
axis('image')

subplot(2,2,3)
pcolor(x_loc, y_loc, real(dw_dy.*conj(dw_dy)))
xlabel('L_x (m)', 'FontSize', fs)
ylabel('L_y (m)', 'FontSize', fs)
title('(\partial w/\partial y)^2', 'FontSize', fs)
shading('interp')
```

```
    xlim([0 0.483])
    ylim([0 0.762])
    colorbar('FontSize', fs)
    set(gca, 'FontSize', fs)
    axis('image')

    subplot(2,2,4)
    pcolor(x_loc, y_loc, real(d2w_dx_dy.*conj(d2w_dx_dy)))
    xlabel('L_x (m)', 'FontSize', fs)
    ylabel('L_y (m)', 'FontSize', fs)
    title('(\partial^2 w/\partial x\partial y)^2', 'FontSize', fs)
    shading('interp')
    xlim([0 0.483])
    ylim([0 0.762])
    colorbar('FontSize', fs)
    set(gca, 'FontSize', fs)
    axis('image')

    figure()
    pcolor(x_loc, y_loc, real(WSSG_dB))
    xlabel('L_x (m)', 'FontSize', fs)
    ylabel('L_y (m)', 'FontSize', fs)
    title(['Plate WSSG Single Controlled at ' num2str(freq)...
        'Hz'], 'FontSize', fs)
    shading('interp')
    xlim([0 0.483])
    ylim([0 0.762])
    colorbar('FontSize', fs)
    set(gca, 'FontSize', fs)
    axis('image')

end
end


elseif strcmp(obj.ControlMode, 'multi')

w_calc = w;
dw_dx_calc = dw_dx;
dw_dy_calc = dw_dy;
d2w_dx_dy_calc = d2w_dx_dy;

for i = 1:obj.NumSensors

% For loop for the calculation of Vcomp at desired points
for m = 1:(79 - num_samples_skip_y)
for n = 1:(49 - num_samples_skip_x)

for j = 1:obj.NumControlModels

    % Displacement data
    p1 = obj.TransferFuncsDisplacement(m, n, frindex, j);
    p2 = obj.TransferFuncsDisplacement(m + num_samples_skip_x, n,...
        frindex, j);
    p3 = obj.TransferFuncsDisplacement(m, n + num_samples_skip_y,...
        frindex, j);
    p4 = obj.TransferFuncsDisplacement(m + num_samples_skip_x, n +...
        num_samples_skip_y, frindex, j);

    % Spatial derivatives
    w_calc(m, n)         = w_calc(m, n) + obj.OptimizedForcesWSSG...
        (frindex, j, i)*(p1 + p2 + p3 + p4)/4;
    dw_dx_calc(m, n)     = dw_dx_calc(m, n) +obj.OptimizedForcesWSSG...
        (frindex, j, i)*(p2 - p1 + p4 - p3)/(2*act_x);
    dw_dy_calc(m, n)     = dw_dy_calc(m, n) +obj.OptimizedForcesWSSG...
        (frindex, j, i)*(p1 - p3 + p2 - p4)/(2*act_y);
   d2w_dx_dy_calc(m, n)=d2w_dx_dy_calc(m, n)+obj.OptimizedForcesWSSG...
       (frindex, j, i)*(p2 - p1 + p3 - p4)/(act_x*act_y);
```

```
end


    % Vcomp
    WSSG(m, n) = alpha*w(m, n)*conj(w(m, n)) + beta*dw_dx(m, n)*...
        conj(dw_dx(m, n)) + gamma*dw_dy(m, n)*conj(dw_dy(m, n)) +...
        delta*d2w_dx_dy(m, n)*conj(d2w_dx_dy(m, n));

end
end

% Get WSSG in dB
WSSG_dB = 20*log10(WSSG/max(max(WSSG)));

% Plot the data
figure()
subplot(2,2,1)
pcolor(x_loc, y_loc, real(w.*conj(w)))
xlabel('L_x (m)', 'FontSize', fs)
ylabel('L_y (m)', 'FontSize', fs)
title('w^2', 'FontSize', fs)
shading('interp')
xlim([0 0.483])
ylim([0 0.762])
colorbar('FontSize', fs)
set(gca, 'FontSize', fs)
axis('image')

subplot(2,2,2)
pcolor(x_loc, y_loc, real(dw_dx.*conj(dw_dx)))
xlabel('L_x (m)', 'FontSize', fs)
ylabel('L_y (m)', 'FontSize', fs)
title('(\partial w/\partial x)^2', 'FontSize', fs)
shading('interp')
xlim([0 0.483])
ylim([0 0.762])
colorbar('FontSize', fs)
set(gca, 'FontSize', fs)
axis('image')

subplot(2,2,3)
pcolor(x_loc, y_loc, real(dw_dy.*conj(dw_dy)))
xlabel('L_x (m)', 'FontSize', fs)
ylabel('L_y (m)', 'FontSize', fs)
title('(\partial w/\partial y)^2', 'FontSize', fs)
shading('interp')
xlim([0 0.483])
ylim([0 0.762])
colorbar('FontSize', fs)
set(gca, 'FontSize', fs)
axis('image')

subplot(2,2,4)
pcolor(x_loc, y_loc, real(d2w_dx_dy.*conj(d2w_dx_dy)))
xlabel('L_x (m)', 'FontSize', fs)
ylabel('L_y (m)', 'FontSize', fs)
title('(\partial^2 w/\partial x\partial y)^2', 'FontSize', fs)
shading('interp')
xlim([0 0.483])
ylim([0 0.762])
colorbar('FontSize', fs)
set(gca, 'FontSize', fs)
axis('image')

figure()
pcolor(x_loc, y_loc, real(WSSG_dB))
xlabel('L_x (m)', 'FontSize', fs)
```

```
    ylabel('L_y (m)', 'FontSize', fs)
    title(['Plate WSSG multi controlled at ' num2str(freq) 'Hz'], ...
        'FontSize', fs)
    shading('interp')
    xlim([0 0.483])
    ylim([0 0.762])
    colorbar('FontSize', fs)
    set(gca, 'FontSize', fs)
    axis('image')

    end
    end
    end


    end

end

% Hidden methods for only the optimization routine to call
methods (Access = private)

    % Function to get the transfer functions to be used
    function getTransferFunctions(obj)

        % Calculate number of sensors and models
        num_sensors        = obj.NumSensors;
        num_control_models = length(obj.ControlModelName);
        num_models         = num_control_models + 1;

        % Preallocate the length of the models vectors.
        models    = cell(num_control_models + 1, 1);
        models{1} = obj.ModelName;

        % Put the models together into a single cell
        if num_control_models ~= 0
            for i = 1:num_control_models
                models{i+1} = obj.ControlModelName{i};
            end
        end

        % Find the closest points to calculate Vcomp with the given
        % sensor locations
        des_x = obj.XSpace;
        des_y = obj.YSpace;

        sensor_x = obj.XSensor;
        sensor_y = obj.YSensor;

        delta_x = obj.XNode(1, 2);
        delta_y = obj.YNode(2, 1);

        x_sense_loc = [sensor_x - des_x/2 sensor_x + des_x/2];
        y_sense_loc = [sensor_y + des_y/2 sensor_y - des_y/2];

        node_index_x = x_sense_loc/delta_x + 1; % If the location = 0 then
                                                % the node_indexcorresponds
                                                %to the first value in the
                                                % vector, so need +1 for
                                                % index to be at correct
                                                % spot.
        node_index_y = y_sense_loc/delta_y + 1;

        node_index_x = round(node_index_x);
        node_index_y = round(node_index_y);

        % Matrix of sensor locations
        x_sense         = zeros(2*num_sensors, 2);
        y_sense         = zeros(2*num_sensors, 2);

        for i = 1:num_sensors
            [x_sense(2*i-1:2*i, 1:2), y_sense(2*i-1:2*i, 1:2)] =meshgrid...
                (node_index_x(i, :)*delta_x, node_index_y(i, :)*delta_y);
        end

        % Preallocate output size. It will be a four dimensional matrix
        % with a row for each frequency the FEA model analyzed, a
        % column for displacement at each of the 4 displacement
        % sensors, a level (3rd dimension) for each of models to be
        % optimized over, and the 4th dimension for the total number of
        % sensor locations. That matrix will hold a lot of data, to be
        % broken out later.
        [m, n, p]                   = size(obj.Displacement);
        transfer_funcs_sense        = zeros(p, 4, num_models, num_sensors);
        transfer_funcs_displacement = zeros(m, n, p, num_models);

        % Get the FEA data for the disturbance models in varargin
        for i = 1:num_models

            % Access the data to calculate the transfer functions
            if i ~= 1
                model_data = obj.getData(models{i});
                model_data = model_data.displacement;
            else
                model_data = obj.Displacement;
            end

            % Save the transfer functions for each model
            transfer_funcs_displacement(:, :, :, i) = model_data;

            % Get the transfer functions for each individual sensor
            for j = 1:num_sensors
              transfer_funcs_sense(:, :, i, j)=obj.getSensorDisplacement...
                  (node_index_x(j, :), node_index_y(j, :), model_data);
            end

        end

        % Assign the output
        obj.TransferFuncsSense        = transfer_funcs_sense;
        % These are the transfer functions only to the location of the
        % sensor(s) on the plate.
        obj.TransferFuncsDisplacement = transfer_funcs_displacement;
        obj.XSenseLoc                 = x_sense;
        obj.YSenseLoc                 = y_sense;

        %-------------------------------------------------------------%
        % The results regardless of number of models and sensors is
        % consistent with the getTransferFunctions function this method
        % is based on.
        %-------------------------------------------------------------%
    end


    % Get the frequency dependent weights
    function getFreqDepWeights(obj)

        % Load the data
        weights = load('frequency_dependent_weights.mat');

        % Assign output
        obj.ObjectiveFunctionStruct.Weights = weights.weights;
        % Get the value out of the struct

    end
```

```
end


% Static methods for setting up the data. Static methods are used when
% you don't want to have a specific instance of the class associated
% with the method, so these methods can be called before the class is
% created.
methods (Static = true)

    % Method to get the data for the model database
    function out = getData(name)

        % Load the data file
        data = load(name);

        % Break out the data from the struct
        out.displacement = data.node_displacement;
        out.node_x       = data.node_x;
        out.node_y       = data.node_y;

        % Get the frequencies
        frequencies = zeros(size(data.frequencies));

        for i = 1:length(data.frequencies)
            frequencies(i) = str2double(data.frequencies{i});
        end

        out.frequencies = frequencies;

    end

end


% Static, private methods
methods (Static = true, Access = private)

    % Function to be used for the getTransferFunctions function
    function out = getSensorDisplacement(node_index_x, node_index_y,...
            displacement)

        % Get sizes for indices
        [~, ~, p] = size(displacement);

        % Get displacement at the desired points. The index values very
        % important here, also chosen to follow order in Jeff's thesis.
        p1 = displacement(node_index_y(1), node_index_x(1), :);
        p2 = displacement(node_index_y(2), node_index_x(1), :);
        p3 = displacement(node_index_y(1), node_index_x(2), :);
        p4 = displacement(node_index_y(2), node_index_x(2), :);

        p1 = reshape(p1, p, 1);
        p2 = reshape(p2, p, 1);
        p3 = reshape(p3, p, 1);
        p4 = reshape(p4, p, 1);

        % Define the output
        out = [p1 p2 p3 p4];

    end


% Function to calculate the distance between each element in the radiation
% resistance matrix. This is modeled after the code radiation_mode_shapes.m
% which I wrote previously and is more general than the version used here.
% This code has been optimized for speed.
function out = getDistanceMat(x_pos, y_pos)
% The required inputs are the x and y positions of the centers of each
% element. These are used to create the distances required for the
% radiation resistance matrix, and must be given as vectors (this is most
% general way).

% Preallocate
out = zeros(length(x_pos),length(y_pos));

% Get distances
for i = 1:length(x_pos)
    out(i,:) = sqrt((x_pos(i) - x_pos).^2 + (y_pos(i) - y_pos).^2);
end

%-----------------------------------------------------------------------%
% Checked on 09/21/2012 and is consistent with previous versions of
% code that do the same thing (and by the way kind of clever I think,
% due to it's simplicity).
%-----------------------------------------------------------------------%

end

end


end


% Dummy function to allow me to switch between objective functions.
function out = FunctionShell(obj, transfer_funcs, fd, fc, freq_index)

% Break out the objective struct
objective_function = obj.ObjectiveFunction;

% Using the function handle in Objective call the function
out = objective_function(obj.ObjectiveFunctionStruct, transfer_funcs,fd,...
    fc, freq_index);

end


% Objective function for the optimizer to use
function out = WSSGObjective(objective_struct, transfer_funcs, fd, fc,...
    freq_index)

% Get the correct weights
weights = objective_struct.Weights;
if length(weights) > 4
    weights = weights(freq_index, :);
end

% Get control as a complex number
if length(fc) == 2
    amp   = fc(1);
    phase = fc(2);
    fc    = amp*exp(1i*phase);
else
    amp   = fc(1:end/2);
    phase = fc(end/2+1:end);
    fc    = amp.*exp(1i*phase);
end

% Break out the transfer function struct
tf_disturbance = transfer_funcs.tf_disturbance;
tf_control     = transfer_funcs.tf_control;
x_sense_loc    = transfer_funcs.sensor_location_x;
y_sense_loc    = transfer_funcs.sensor_location_y;

% Break out the transfer function
h_d1 = tf_disturbance(freq_index, 1);
```

```
h_d2 = tf_disturbance(freq_index, 2);
h_d3 = tf_disturbance(freq_index, 3);
h_d4 = tf_disturbance(freq_index, 4);

% Spacing for finite difference derivative points
space_x = x_sense_loc(1, 2) - x_sense_loc(1, 1);
space_y = y_sense_loc(1, 1) - y_sense_loc(2, 1);

% Define the weights (Average analytical weights)
alpha = weights(1);
beta  = weights(2);
gamma = weights(3);
delta = weights(4);

% Calculate WSSG
w        = (h_d1 + h_d2 + h_d3 + h_d4)*fd;
dw_dx    = (h_d2 - h_d1 + h_d4 - h_d3)*fd;
dw_dy    = (h_d1 - h_d3 + h_d2 - h_d4)*fd;
d2w_dxdy = (h_d2 - h_d1 + h_d3 - h_d4)*fd;

for i = 1:length(fc)

    % Get the control transfer function
    h_c1 = tf_control(freq_index, 1, i);
    h_c2 = tf_control(freq_index, 2, i);
    h_c3 = tf_control(freq_index, 3, i);
    h_c4 = tf_control(freq_index, 4, i);

    % Calculate
    w        = w + (h_c1 + h_c2 + h_c3 + h_c4)*fc(i);
    dw_dx    = dw_dx + (h_c2 - h_c1 + h_c4 - h_c3)*fc(i);
    dw_dy    = dw_dy + (h_c1 - h_c3 + h_c2 - h_c4)*fc(i);
    d2w_dxdy = d2w_dxdy + (h_c2 - h_c1 + h_c3 - h_c4)*fc(i);

end

w        = w/4;
dw_dx    = dw_dx/(2*space_x);
dw_dy    = dw_dy/(2*space_y);
d2w_dxdy = d2w_dxdy/(space_x*space_y);

out = alpha*(w.*conj(w)) + beta*(dw_dx.*conj(dw_dx)) + gamma*(dw_dy.*...
    conj(dw_dy)) + delta*(d2w_dxdy.*conj(d2w_dxdy));

end


% Objective function for volume velocity
function out = VolVelObjective(objective_struct, ~, fd, fc, freq_index)

% Get control as a complex number
if length(fc) == 2
    amp   = fc(1);
    phase = fc(2);
    fc    = amp*exp(1i*phase);
else
    amp   = fc(1:end/2);
    phase = fc(end/2+1:end);
    fc    = amp.*exp(1i*phase);
end

% Break out the objective struct
Ae                  = objective_struct.Ae;
model_displacements = objective_struct.TransferFuncsDisplacement;
omega               = 2*pi*objective_struct.Frequencies(freq_index);

% Calculate the displacement
displacement = model_displacements(:, :, freq_index, 1)*fd;
```

```
for i = 1:length(fc)
    displacement = displacement + fc(i)*model_displacements(:, :,...
        freq_index, i+1);
end

% Calculate the volume velocity
el_velocity = 1i*omega*displacement;

out = Ae*sum(sum(el_velocity));
out = out.*conj(out);

end
```

B.2.2 Simulation_Example

```
 % Run the control simulations for multi-force control
 x = WSSGTransFunc('nodes_and_displacements_disturbance.mat');
 x.setControlModel('nodes_and_displacements_control.mat',...
     'nodes_and_displacements_control_mod2.mat')
 % x.setControlMode('single')
 % x.setObjectiveFunction(2)
 % x.runOptimizer
 x.setControlMode('multi')
 x.setObjectiveFunction(2)
x.doIt

% mod6 = WSSGTransFunc('nodes_and_displacements_disturbance.mat');
% mod6.setControlModel('nodes_and_displacements_control_mod6.mat')
% mod6.setControlMode('single')
% mod6.setObjectiveFunction(2)
% mod6.doIt

% both_2_6 = WSSGTransFunc('nodes_and_displacements_disturbance.mat');
% both_2_6.setControlModel('nodes_and_displacements_control.mat',...
% 'nodes_and_displacements_control_mod2.mat',...
% 'nodes_and_displacements_control_mod6.mat')
% both_2_6.setControlMode('multi')
% both_2_6.setObjectiveFunction(2)
% both_2_6.doIt
```