Technische Universität München
Fakultät für Bauingenieur- und Vermessungswesen
Lehrstuhl für Computation in Engineering
Prof. Dr. rer. nat. Ernst Rank

Validating and Updating Structural Finite Element Models for Dynamic Analysis

Master's Thesis

Adam C. Dick, BSE
November 5, 2008

Abstract

As advances in computation develop, more sophisticated Finite Element (FE) models can be generated to simulate the dynamic response of a reference model. One measure of the dynamic response of a structure is through its eigenfrequencies and mode shapes. It therefore becomes necessary to validate the FE data using correlation techniques to assure that it accurately predicts the reference data. Two standard correlation techniques are the Modal Assurance Criterion (MAC) and the Orthogonality Check (ORTHO), whose results can be used to perform an update of the FE model that minimizes the discrepancy between the FE and reference data. This thesis presents an implementation of a Java Graphical User Interface (GUI), where such FE and reference model data can be correlated and set up for optimization in MSC.Nastran.

Foreword

The presented Master's thesis was conducted during the 2008 summer semester at the *Lehrstuhl für Computation in Engineering* at the *Technische Universität München* (TUM) in cooperation with *Industrieanlagen-Betriebsgesellschaft mbH* (IABG). I would like to thank my supervisor at TUM, Dr.-Ing. Martin Ruess, for giving me his insight into standard programming practices and for thoroughly guiding me through the many difficult aspects of this work. I would also like to thank my supervisor at IABG, Dr.-Ing. Manfred Kroiss, for providing such an interesting application of *Computational Mechanics*, incorporating many of its related theoretical fields. Without their dedicated support, the success of this thesis would not have been possible.

Munich, Germany
November 5, 2008

Adam C. Dick, BSE

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The proposal for this Master's thesis was provided by *Industrieanlagen-Betriebsgesellschaft mbH* (IABG) — who for the purposes of this thesis will be referred to as *the client* — and driven by the industrial need to build simulation models that accurately predict the dynamic behavior of a structure. To ensure the accuracy of these models, they are typically correlated with test data by different methods. The correlations are used to update the properties of the simulation model, typically through an optimization process. Because the test data is usually limited to the amount and location of available measurement sensors, the correlations are also used to adjust sensor positions on the test model to assure that its complete dynamic behavior is captured. Through this synergistic approach, the results of the simulation model can be validated and further designing can be done on an already accurate model, possibly reducing the length of the design phase.

A standard method to correlate the dynamic behavior of simulation and test models is through their eigenfrequencies and mode shapes. Several correlation methods for this data exist, but the most popular are the Modal Assurance Criterion (MAC) and the Orthogonality Check (ORTHO). From these correlations, a sensitivity analysis can be performed by a Finite Element (FE) package such as MSC.Nastran to optimize one or more design parameters, including mass, stiffness or damping. The eigenfrequencies and mode shapes of the optimized simulation can then be correlated with the test data. If the correlation is still unsatisfactory, new design parameters can be selected and the procedure can be repeated iteratively as necessary.

The client has a need to perform a MAC and an ORTHO optimization procedure given the geometric and dynamic data of two models. Therefore, the client's requirements are to implement a Java program with a Graphical User Interface (GUI) for the model correlation and model update to streamline this process. Additionally, the ORTHO correlation must be designed and seamlessly integrated into the implementation. The correlations should be displayed in graphical 2D, 3D, and tabular form to easily interpret the data. Finally, an optimization deck containing the objective function derived from either the MAC or ORTHO correlation must be prepared for a parameter-based optimization analysis involving a MSC.Nastran SOL200 sensitivity analysis. To complete these tasks, the program must be able to import simulated geometry and mode shapes from MSC.Nastran and experimental mode shapes in Universal format. Additionally, for the ORTHO calculation, the mass and stiffness matrices from MSC.Nastran must be imported. All of this work has been successfully completed and supporting Java API documentation accompanies the final software package.

1.1 Industrial Objectives

The dynamic analysis of structures applies to many industrial fields, such as aerospace, naval and civil engineering. The content of this thesis focuses on, but is not limited to, the client's application of interest, automotive engineering. The following sections' content is taken from or influenced by [6] and [14] and is meant to provide an overview of the motivation and direction of automotive research concerning dynamic analysis.

Structural dynamics becomes important when considering the dynamic stiffness and vibration comfort during the automotive design phase. Figure 1.1 illustrates the difficulty in maintaining the dynamic design requirements as the total car mass increases. More effort must be spent analyzing the vibration response as the inertial effects of the system become more significant. One vibration source, the engine, creates internal forces that act on the car body at the engine mounts. The resulting vibrations vary depending on the engine type and propagate throughout the rest of the car body by means of its dynamic stiffness. Another vibration source comes from the contact of the tires to the road, and the excitation is generated by unevenness of the road surface. The critical locations where vibration comfort is examined is at the driver interface, which includes the steering wheel, steering column, interior mirror, seat mounts, and pedals.

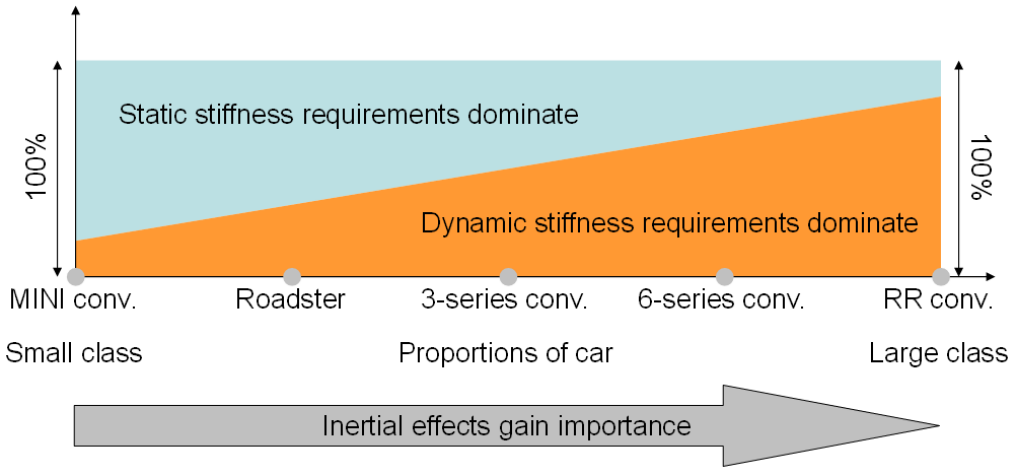

Figure 1.1: Inertial Effects Become Increasingly Significant with Greater Mass [14]

Typically, the engine frequencies associated with the highest forces are the ignition frequencies. The first order of the engine is the rotation frequency of the shaft, and every cylinder provides an additional ignition frequency, increasing the order. One strategy to improve the driver's vibration comfort is to design the car body such that its global eigenfrequencies, where resonance will occur, do not coincide with these idle ignition frequencies. Figure 1.2 shows a spectral separation of the eigenfrequencies from the excitation frequencies. It emphasizes that for a particular engine, there may be windows for the car body eigenfrequencies that will minimize the dynamic response. A further design objective is for the car body to provide an acceptable compromise for several different engines.

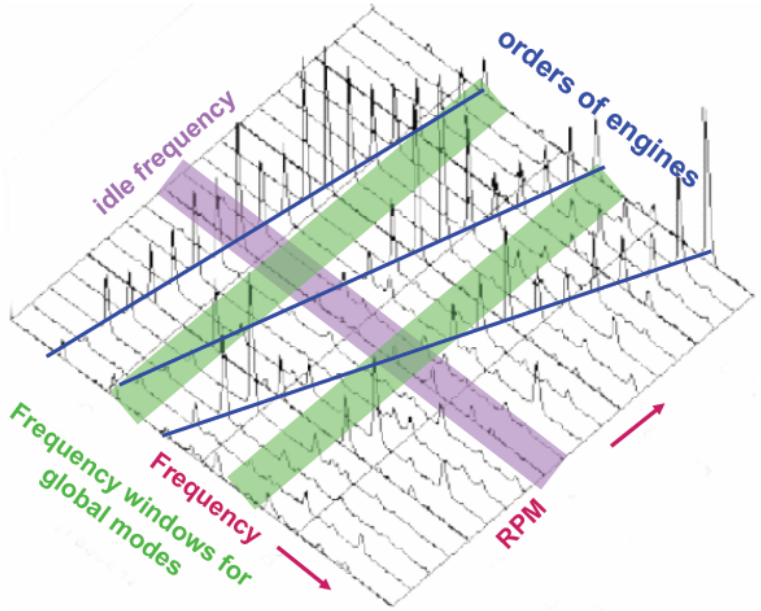

Figure 1.2: Global Frequency Windows of Car Body [14]

These design objectives facilitate the need for FE models to accurately predict the dynamic response of the car body. To this end, measurements on live test models are taken, with which the FE data can be compared. An example of the typical FE model discussed in this thesis is shown in Figure 1.3, whose geometry was rendered as a wireframe model using the current implementation. The FE geometry of this concept car is courtesy of the client and contains several thousand nodes, bar and beam connections, triangular shell elements, and quadrilateral shell elements.

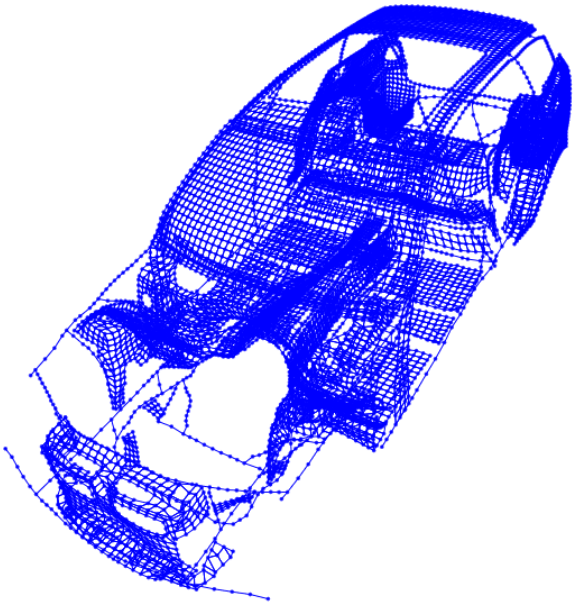

Figure 1.3: Wireframe Visualization of Finite Element Model of Station Wagon Concept Car

1.2 Model Correlation

While the use of FE models is becoming widespread, discrepancies still exist between their predicted dynamic behavior and that of dynamic measurements from test vehicles. This discrepancy is cumulative and may come from several different sources. The FE model may have modeling errors, such as a simplistic material law, inexact geometry, or incorrect boundary conditions. The problem may also stem from numerical errors, such as a deficient mesh discretization or an inappropriate numerical solution method. To account for and correct these variances, dynamic measurements of test models still provide a reference point for FE models.

Measuring test models provides an accepted method of obtaining precise dynamic response data, namely the three-dimensional displacements at various nodes along the exterior of the car body. These displacements create a coarse view of the mode shapes, discussed in **Chapter 2**, as well as the vibration response to excitation frequencies. Figure 1.4 shows a transient response measurement lab with sensors affixed to the car body and shakers positioned under each tire. Shakers can be effectively used to repeatedly simulate the same road conditions independent of weather conditions or driver. The dynamic response data from these test measurements can then be correlated with simulation data.

Figure 1.4: Transient Response Measurements and Modal Analysis in BMW Hydro-Pulse Lab [14]

There are several methods to correlate mode shapes of two models, however, the standard methods are the MAC and ORTHO correlations, discussed in **Chapter 3**. Based on the strength of the correlation between a FE model and a test model, the data from the FE model may be accepted or its properties may need to be improved. While the test model is taken to be the benchmark, its limitations at this juncture demonstrate the advantages of the FE model. The drawback of the test model is that the amount of measurement locations is restricted to the number of available sensors. The optimal amount of measurement locations, though impractical, would be equivalent to that of the full FE model. Instead, the data from the FE

model must be reduced to the sensor locations of the test model and then correlated. While the FE model can have a very fine representation of the mode shapes, the limited number of measurement locations may not be able to completely capture the mode shapes of the test model and is referred to as *spatial aliasing*. Therefore, if the resulting test data is insufficient, subsequent correlation computations may consequently produce misleading results. To guard against this, the measurement sensors must be adequately positioned and can be improved through correlation techniques.

1.3 Model Update

Given either MAC or ORTHO correlation data, the properties of the FE model may need to be improved and or the measurement locations of the test model may need to be adjusted. Improvement of sensor and shaker placement, sometimes called *pre-test analysis*, goes beyond the scope of this thesis and can be further studied by referring to [18]. Assuming a pre-test analysis has been completed and the data from the test model does not exhibit spatial aliasing, focus can be shifted to the deficiencies of the FE model.

A sensitivity analysis, as part of structural optimization, can be used to propose improvements for the properties of the FE model. The requirements to perform an optimization include defining an objective function, design variables, and inequality or equality constraints. In this case, the objective function is related to the MAC or ORTHO correlation itself and is discussed further in **Chapter 4**.

In general, the objective function can also include other correlation data, such as vibration response, mass, stiffness, and modal frequencies, resulting in a multi-objective function. When dealing with a FE model, the design variables represent the parameters which cause modeling errors as described in the previous section. These parameters can consist of material, element, connection or topology, damping, and geometric properties. Constraints are usually the bounds of these design variables or limitations they impose.

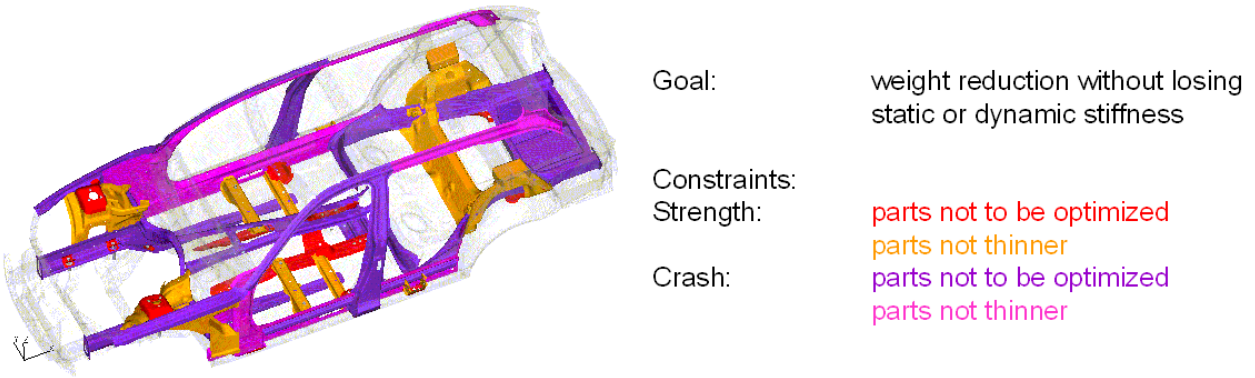

Figure 1.5: Typical Optimization for Weight Reduction of Finite Element Model [14]

Figure 1.5 provides a typical example for the optimization of a FE model. The objective is weight reduction, the design parameters are the geometric properties of the highlighted parts,

and the constraints are the static and dynamic stiffnesses, the minimum geometric thicknesses, and the strength and crash ratings of the overall structure.

The result of the optimization and sensitivity analysis are the improved values of the design variables, which best meet the objective function and constraints. These values can then replace or update the existing values in the FE model and if necessary, a pre-test analysis can also be performed to improve the positioning of the measurement instruments on the test model. After both the FE model and test model have been updated, the MAC or ORTHO correlation can be recomputed.

If the results of the new correlation show a satisfactory improvement, the process can be terminated with the updated FE model. If the correlation still does not meet the desired criteria, new design variables and or constraints can be selected in an attempt to further optimize the FE model. This process can be done iteratively until the correlation requirements are fully met. Figure 1.6 displays a standard correlation optimization procedure including the pre-test analysis, given a FE and test model.

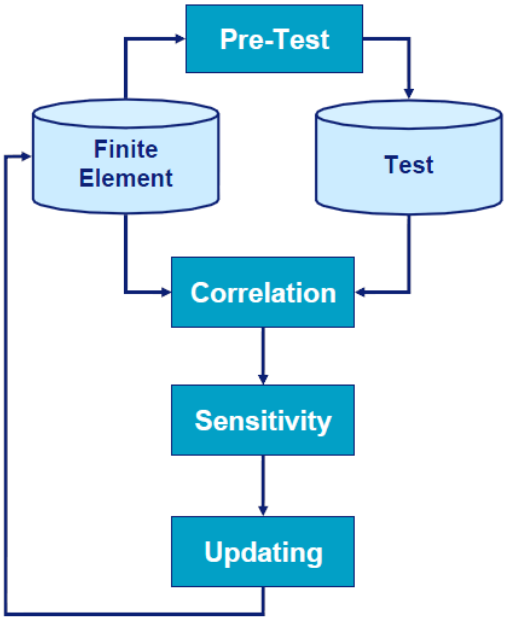

Figure 1.6: Standard Optimization Loop for Correlating Finite Element and Test Models [6]

The end result of the optimization process is a validated FE model whose dynamic behavior is considered accurate. This allows aspects of the global design to be finalized and work can be concentrated on target areas. The FE model validation can also create a freeze point that future designs may return to for a standard comparison.

Chapter 2

Structural Dynamics

For the discussion of structural dynamics in the context of this thesis, the reader is expected to be familiar with the equation of motion and its solution for a Single Degree of Freedom (SDOF) system, which is presented in detail in [8] and [13]. The extension of the SDOF equation of motion to Multi Degree of Freedom (MDOF) systems is explained in [13] and [19]. With this groundwork in place, the following sections derive the solution for the eigenfrequencies and mode shapes of undamped MDOF systems. For a given loading condition, the mode shapes of the solution can then be examined through modal analysis, producing generalized quantities for the mass, stiffness and loading. These generalized quantities then provide the fundamental theoretical basis for the correlation analysis techniques described in **Chapter 3**. The contents of the following sections are taken from [13].

2.1 Solution of the Eigenfrequencies

The extension of the SDOF equation of motion to a MDOF system is a straight-forward transition. Given are the homogeneous differential equations of motion of an undamped MDOF system, describing its free vibrations:

$$\mathbf{M}\ddot{\vec{w}} + \mathbf{K}\vec{w} = \vec{0} \tag{2.1}$$

where $\mathbf{M}$ and $\mathbf{K}$ represent the mass and stiffness matrices of the system, respectively, and $\vec{w}$ denotes the displacement vector. The free vibrations are described by the system's eigenfrequencies or natural frequencies, ω_i, and their corresponding mode shapes or eigenvectors, $\mathbf{\Psi}_i$. The general solution of the displacement vector of this second order differential equation can have the following form:

$$\vec{w} := \vec{w_0} \cdot \sin(\omega t + \phi) \tag{2.2}$$

where $\vec{w_0}$ represents the displacement amplitudes and $\tan(\phi)$ gives the phase angle of the response.

The first derivative of the displacement results in the velocity vector:

$$\dot{\vec{w}} := \omega\vec{w_0} \cdot \cos(\omega t + \phi) \tag{2.3}$$

Similarly, the second derivative results in the acceleration vector:

$$\ddot{\vec{w}} := -\omega^2\vec{w_0} \cdot \sin(\omega t + \phi) \tag{2.4}$$

Substituting the acceleration into Equation 2.1:

$$\left(-\omega^2\mathbf{M}+\mathbf{K}\right)\cdot\vec{w_0}=\vec{0} \tag{2.5}$$

One method of obtaining the non-trivial solution is formulated by applying the determinant, creating the *frequency equation*:

$$\det\left(-\omega^2\mathbf{M}+\mathbf{K}\right)=0 \tag{2.6}$$

which produces an algebraic equation of order n, where n is the dimension of the system. The roots of the resulting equation represent the n eigenfrequencies, ω_i, of the system. Each eigenfrequency has a corresponding displacement vector solution, $\mathbf{\Psi}_i$, called the mode shape. The temporal eigenfrequency, f_i, is expressed in Hertz and determined by the following relation:

$$f_i=\frac{\omega_i}{2\pi} \tag{2.7}$$

2.2 Determination of Mode Shapes

By substituting a given eigenfrequency into Equation 2.5, an indeterminate set of n equations is obtained. The linear dependence of the equations means that the amplitude of the displacements cannot be directly solved. That means that for any arbitrary scalar α, $\alpha\cdot\mathbf{\Psi}_i$ is also a valid solution. A normalization of the eigenvector by various methods, however, can produce the form or shape of the solution. Hence, the mode shape is a scaled representation of the free vibration response of the structure.

One method for normalizing a mode shape is by setting one of the displacement degrees of freedom to 1.0. A value for α can be computed and the remaining $n-1$ equations can be scaled by this constant. Some commercial FE solvers use the *Lanczos Method* to compute the mode shapes of a dynamic structure. On a HP-UX 64 bit platform, a FE model with approximately five million Degrees of Freedom (DOF) requires about 180 minutes to solve the first 40 eigenfrequencies and their corresponding mode shapes [14]. Figures 2.1 and 2.2 show two typical types of global mode shapes, torsion and bending, of an automotive FE model.

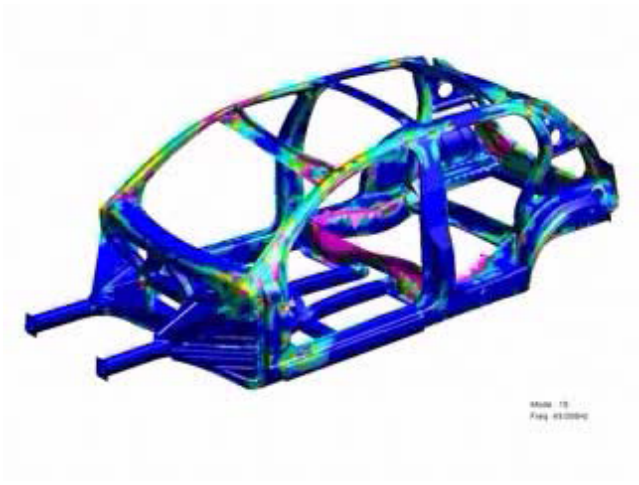

Figure 2.1: Global Torsional Mode Shape of Automotive Finite Element Model [14]

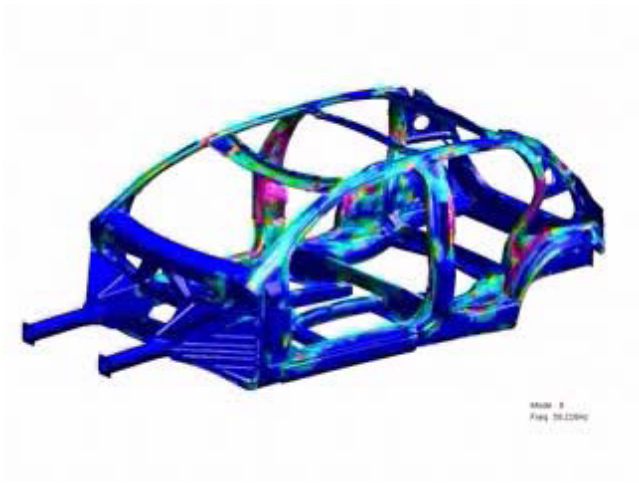

Figure 2.2: Global Bending Mode Shape of Automotive Finite Element Model [14]

As the eigenfrequency increases, the inverse relation of the period of vibration, $T = \frac{1}{\omega}$, becomes shorter. At each successive mode shape, an additional zero-crossing or sign change between subsequent displacement degrees of freedom occurs. The increasing number of zero-crossings at higher mode shapes forces the bending of the structure to also increase. This increased bending means that higher mode shapes are more energetic and consequently, suffer from higher damping effects.

2.3 Modal Analysis

For the case of a discrete system, the time-dependent solution of the displacements can be found by the superposition of the orthogonal mode shapes multiplied by modal or normal factors. Modal analysis can therefore only be applied to systems of linear vibrations. The linear equations of motion of a MDOF system, however, can then be decoupled into a set of SDOF systems.

Consider the following undamped MDOF system:

$$\mathbf{M}\ddot{\mathbf{w}} + \mathbf{K}\mathbf{w} = \mathbf{F}_0 \cdot g\,(t) \tag{2.8}$$

where $\mathbf{F}_0$ represents the amplitude of the time-dependent loading, $g\,(t)$.

Evaluating the m undamped mode shapes of the system, $\mathbf{\Psi}_i$, leads to the composition of the *mode shape matrix*, $\mathbf{\Phi}$.

$$\mathbf{\Phi} := \begin{bmatrix} \mathbf{\Psi}_1 & \mathbf{\Psi}_2 & \mathbf{\Psi}_3 & \cdots & \mathbf{\Psi}_m \end{bmatrix} = \begin{bmatrix} \Psi_{11} & \Psi_{12} & \Psi_{13} & \cdots & \Psi_{1m} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \Psi_{n1} & \Psi_{n2} & \Psi_{n3} & \cdots & \Psi_{nm} \end{bmatrix} \tag{2.9}$$

The displacement vector, $\mathbf{w}$, is obtained by superposition of the mode shapes and their corresponding modal factors, $\mathbf{y}$. The modal factors represent the contribution of each mode shape

to the overall solution:

$$\mathbf{w} = \begin{bmatrix} \boldsymbol{\Psi}_1 & \boldsymbol{\Psi}_2 & \boldsymbol{\Psi}_3 & \cdots & \boldsymbol{\Psi}_m \end{bmatrix} \mathbf{y} = \boldsymbol{\Phi}\mathbf{y} \tag{2.10}$$

Substitution of the resulting displacement vector into Equation 2.8 yields:

$$\mathbf{M}\boldsymbol{\Phi}\ddot{\mathbf{y}} + \mathbf{K}\boldsymbol{\Phi}\mathbf{y} = \mathbf{F}_0 \cdot g\left(t\right) \tag{2.11}$$

and multiplying through by $\boldsymbol{\Phi}^T$ leads to:

$$\boldsymbol{\Phi}^T\mathbf{M}\boldsymbol{\Phi}\ddot{\mathbf{y}} + \boldsymbol{\Phi}^T\mathbf{K}\boldsymbol{\Phi}\mathbf{y} = \boldsymbol{\Phi}^T\mathbf{F}_0 \cdot g\left(t\right) \tag{2.12}$$

This equation can then be simplified:

$$\mathbf{M}^*\ddot{\mathbf{y}} + \mathbf{K}^*\mathbf{y} = \mathbf{F}_0^* \cdot g\left(t\right) \tag{2.13}$$

Here $\mathbf{M}^*$ is called the generalized mass matrix:

$$\mathbf{M}^* = \begin{bmatrix} m_1^* & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & m_m^* \end{bmatrix} = \boldsymbol{\Phi}^T\mathbf{M}\boldsymbol{\Phi} \tag{2.14}$$

where the generalized mass of each mode shape can be interpreted as its kinetic energy:

$$m_i^* = \boldsymbol{\Psi}_i^T\mathbf{M}\boldsymbol{\Psi}_i = \sum_k \sum_l \Psi_{i,k} M_{kl} \Psi_{i,l} \tag{2.15}$$

Similarly, $\mathbf{K}^*$ is the generalized stiffness matrix:

$$\mathbf{K}^* = \begin{bmatrix} k_1^* & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & k_m^* \end{bmatrix} = \boldsymbol{\Phi}^T\mathbf{K}\boldsymbol{\Phi} \tag{2.16}$$

where the generalized stiffness of each mode shape is its potential energy and is related to the generalized mass by the square of the eigenfrequency:

$$k_i^* = \boldsymbol{\Psi}_i^T\mathbf{K}\boldsymbol{\Psi}_i = \sum_k \sum_l \Psi_{i,k} K_{kl} \Psi_{i,l} = \omega_i^2 m_i^* \tag{2.17}$$

$\mathbf{F}^*$ is the generalized load vector:

$$\mathbf{F}^* = \begin{bmatrix} f_1^* \\ \vdots \\ f_m^* \end{bmatrix} = \boldsymbol{\Phi}^T\mathbf{F} \tag{2.18}$$

where the generalized loading of each mode shape represents the work it does:

$$f_i^* = \boldsymbol{\Psi}_i^T\mathbf{F} = \sum_k \Psi_{i,k} F_{kl} \tag{2.19}$$

Finally, the set of uncoupled SDOF systems is derived from the MDOF system. Each resulting SDOF system can be managed using the same methods described in [8] and [13]:

$$\begin{bmatrix} m_1^* & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & m_n^* \end{bmatrix} \ddot{\mathbf{y}} + \begin{bmatrix} k_1^* & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & k_n^* \end{bmatrix} \mathbf{y} = \begin{bmatrix} f_1^* \\ \vdots \\ f_n^* \end{bmatrix} g\left(t\right) \tag{2.20}$$

The formulation and orthogonality properties of the generalized mass and stiffness matrices provide the basis of the correlation analysis methods described in **Chapter 3**.

Chapter 3

Correlation Analysis

When discussing the dynamic correlation of structural models, two distinct models are implied. The first is the *reference model*, which serves as the baseline or control and whose data is assumed to be true values. The second is the *update model*, whose data is compared to that of the reference model through any of several correlation techniques. If the correlation of the two models meets a specified tolerance or confidence level, the update model is said to be *validated*. If this criteria is not met, the update model must undergo an *update*, discussed in **Chapter 4**, until it is improved sufficiently enough for the correlation to meet the specified requirements.

In industry, there are typically two types of data. The first kind is *simulation data* or *FE data*, which is computed deterministically from numerical models. The second kind is *test data*, which is measured from sensor positions around a full scale test model or prototype. Theoretically, both FE data and test data can be used as either the reference model or update model. Therefore, the possible reference-update model combinations are FE-FE, FE-test, test-FE, and test-test. In most practical cases, the test-FE combination is of interest, because the FE model may normally have some deficiencies when compared to a full scale model. The test-test case may also be useful in determining optimal sensor placement on a test model or in monitoring any improvement in the results of the same model as it progresses through the design phase. A FE-FE combination can also be helpful in updating a coarsely discretized model with respect to a finely discretized model.

In the scope of this thesis, four correlations are discussed and implemented. The first is a spatial correlation of the nodal coordinate geometry, upon which the following correlations rely. The second is the Orthogonality Check (ORTHO), which compares the mode shapes of two structural models considering their inertial effects. The third is the Modal Assurance Criterion (MAC), which also compares the mode shapes of two structural models, but without considering their mass influence. The final correlation compares the eigenfrequencies of both models. These correlation methods provide the basis for the model optimization and update computations described in **Chapter 4**. The following sections, whose contents are taken from or influenced by [4], [6], and [18], present each correlation in more detail.

3.1 Geometric Correlation

One of the first considerations that must be addressed when examining two models is a spatial or geometric correlation. This correlation should naturally precede any dynamic correlations because it can highlight differences in the mass distribution of the two models. The mass distribution provides the inertial influence of the dynamic behavior of each model, therefore any geometric discrepancies will result in inherently different dynamic behaviors.

A second and more immediate need for the geometric correlation is to connect the DOF between the two models. The nodes from one model must be mapped to those of the other model in order to perform the dynamic correlations described in the following sections. The most obvious geometric correlation is based on the distance between any two given nodes.

3.1.1 Degree of Freedom Correlation

The DOF correlation usually focuses on the nodal coordinates and simply calculates the three-dimensional distance, $\mathbf{DOF}_{ij}$, from every node, $\mathbf{DOF}_{R,i}$, of the reference model to every node, $\mathbf{DOF}_{U,j}$, of the update model:

$$\mathbf{DOF}_{ij} = \sqrt{(\Delta x_{ij})^2 + (\Delta y_{ij})^2 + (\Delta z_{ij})^2} \tag{3.1}$$

$$\mathbf{DOF}_{ij} = \sqrt{(x_{U,j} - x_{R,i})^2 + (y_{U,j} - y_{R,i})^2 + (z_{U,j} - z_{R,i})^2} \tag{3.2}$$

This may seem like a straight-forward computation, however, the resulting correlation matrix may require a prohibitively large amount of memory. In the case of a full FE model, which may have over five million DOF, the computational effort increases proportionally with the amount of nodes in the test model. One remedy that has been incorporated into this thesis is the ability to select a subset of FE nodes for comparison with the test model, thus greatly reducing memory requirements.

3.1.2 Minimum Distance Connection

Once the DOF correlation matrix has been computed, it can be used to map the nodes of the two models. This is done by finding the minimum value, the distance, in the correlation matrix. The corresponding nodes thus represent the most likely pair and are added to the map. The search for the minimum distance is repeated, this time without the mapped nodes, and the next most likely pair is added to the map. This is done iteratively until one model is completely mapped to the other. With this DOF connection, the models are prepared for a dynamic correlation. A detailed explanation of the implemented geometric connection algorithm is provided in **Chapter 5**.

An important constraint of this connection is that the models must be spatially aligned. This means that they must be of the same scale and that no geometric transformations exist between them. Any translation, rotation, or scaling differences will result in a skewed or erroneous connection. Figure 3.1 illustrates a pair of models which must be properly oriented

prior to correlating and connecting them. It can clearly be seen that connecting the nodes in the current configuration will produce misleading results. The current implementation requires pre-aligned data to generate the proper connection.

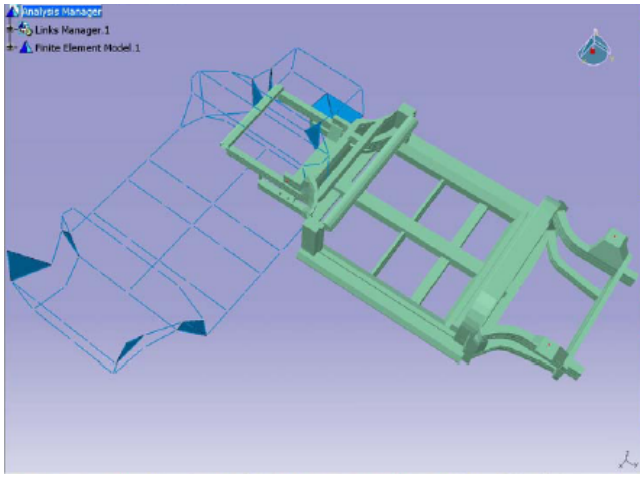

Figure 3.1: Two Models Must Be Pre-Aligned for Geometric Correlation and Connection [6]

Figure 3.2 demonstrates the geometric connection of two simple FE frame models, shown in red and green. The connection was created and visualized with the current implementation and the mapped nodes are highlighted in black. The connection was reduced by imposing a maximum distance tolerance, where only those node connections whose distance was less than the tolerance are highlighted. The connections which exceed the tolerance are not highlighted and excluded from the map.

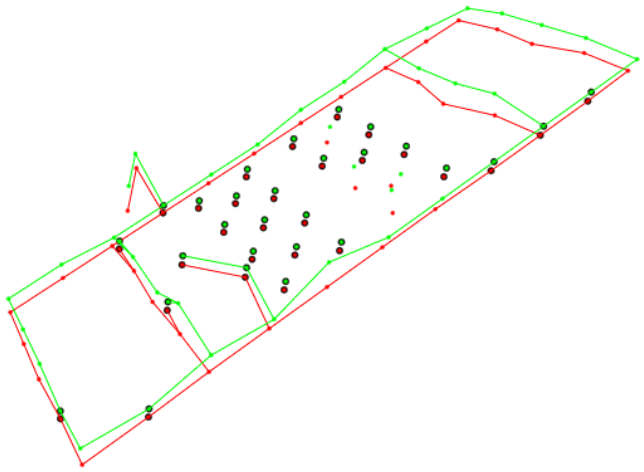

Figure 3.2: Visual Verification of Geometric Connection for Two Pre-Aligned Models

Because the node connections represent the DOF connections, this intermediate visualization provides assurance that a satisfactory DOF map has been generated. Using this DOF map, the following dynamic correlations can be computed.

3.2 Orthogonality Check

The ORTHO correlation is a quantitative and reliable method for comparing the mode shapes of two models because the calculation is rooted in the theory of structural dynamics. The basis for the calculation comes from Equations 2.14 and 2.16 for the generalized mass and stiffness matrices of a MDOF system, which decouple the original equations of motion into an orthogonal set of SDOF systems. These equations use a single mode shape matrix to linearize the MDOF system, however, they can be modified to examine the orthogonality of two distinct mode shape matrices. If, for example, a reference model and an update model should represent the same system, then inserting the mode shape matrices from both models into the formulations of the generalized quantities should also yield orthogonal results.

3.2.1 Mass Orthogonality

The most common ORTHO correlation considers the mass matrix of one of the two dynamic systems. The standard equation for a mass ORTHO correlation is as follows:

$$\mathbf{ORTHO}_{ij,M} = \mathbf{\Psi}_{A,i}^T \mathbf{M} \mathbf{\Psi}_{B,j} \tag{3.3}$$

where $\mathbf{\Psi}_{A,i}$ represents the mode shape of one system and $\mathbf{\Psi}_{B,j}$ represents the mode shape of the other. The mass matrix may belong to either system, however, it is usually taken from a FE model because measuring the mass influences of every DOF in a test model is generally not possible.

Equation 3.3, in most cases, will not produce a completely orthogonal matrix for two models. To standardize the interpretation of the data, the mass ORTHO correlation can be normalized according to the following equation:

$$\mathbf{ORTHO}_{ij,M} = \frac{\left(\mathbf{\Psi}_{A,i}^T \mathbf{M} \mathbf{\Psi}_{B,j}\right)^2}{\left(\mathbf{\Psi}_{A,i}^T \mathbf{M} \mathbf{\Psi}_{A,i}\right)\left(\mathbf{\Psi}_{B,j}^T \mathbf{M} \mathbf{\Psi}_{B,j}\right)} \tag{3.4}$$

which is the form of the ORTHO correlation in the current implementation. Figure 3.3 displays a 3D visualization, created from the current implementation, of an ideal mass ORTHO result with diagonal values of unity and off-diagonal values of zero. All correlation values will range from zero to unity, and in practice, some generally accepted limits can be applied to the resulting orthonormal matrix. A correlation value greater than 0.9 is considered a strong relationship between mode shapes, while a value less than 0.1 is considered a weak relationship.

In terms of a model correlation analysis, the quantities in Equation 3.4 can be given the following general interpretation:

$\mathbf{\Psi}_{A,i}$	mode shape of the reference model
$\mathbf{M}$	mass matrix of the reference model
$\mathbf{\Psi}_{B,j}$	mode shape of the update model

where the mass matrix typically corresponds to the FE model, even in the case of correlating two test models. Data from a FE model representing the reference test model is usually inserted into Equation 3.4 in those situations.

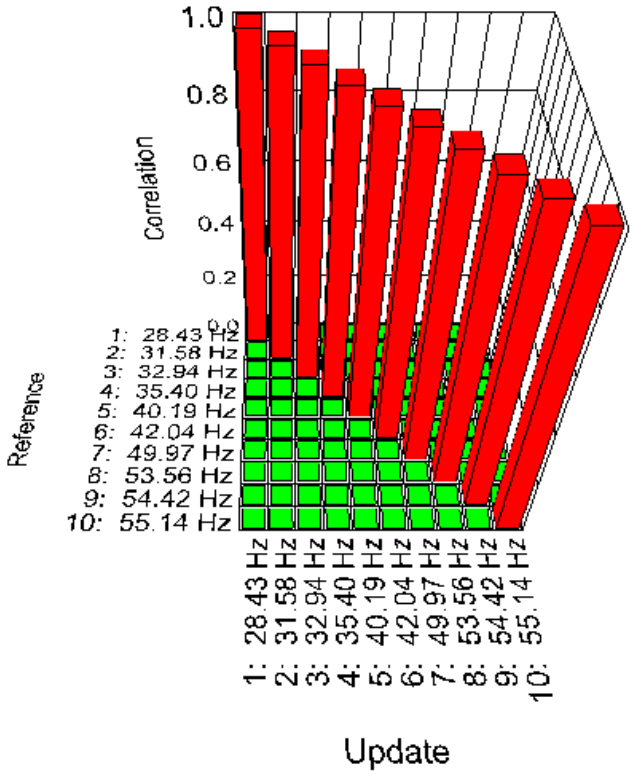

Figure 3.3: Ideal Result for Orthogonality Check Is Fully Orthogonal Matrix

A great advantage of the mass ORTHO correlation is that it accounts for the mass influence of each DOF. The displacement solution of a mode shape at a given DOF can be large relative to other DOF displacements, however, its mass influence may be relatively small. Therefore, the inertial contribution of this DOF to the global vibration response should also be relatively small. The mass ORTHO correlation uses the values of the mass matrix to damp the correlation contribution of such DOF, thus creating a sensible, weighted correlation.

This correlation also faces some challenges. As mentioned in the previous section, the computation of a dynamic mode shape correlation requires a DOF map from the reference model to the update model. However, for the mass ORTHO correlation in Equation 3.4, this map must also hold for the mass matrix. Therefore, it is critical that the DOF map is assured for both mode shape matrices and the mass matrix prior to computation. Failure to implement such a check may produce misleading results.

3.2.2 Model Reduction

In practice, a more significant challenge develops for the ORTHO correlation when considering a FE model and a test model. The ideal situation would allow every DOF of the FE model to be mapped to a DOF of the test model. Since the number of measurement locations on the test model is restricted to the number of available sensors, usually on the order of several

dozen, this ideal situation almost never occurs.

In the standard case, the test model is taken to be the reference model and the FE model is taken to be the update model. In this scenario, the reference model does not provide its own mass or stiffness matrices. They are provided by the update model, which has a much greater number of DOF than the reference model. Therefore, according to the geometric DOF map, a reduction of the unmapped DOF of the FE model is performed. For the mode shapes of the FE model, this means performing a static condensation or Guyan reduction of the full model using the measurement points as exterior DOF. Once this is completed, the mass ORTHO correlation can be performed as usual.

The drawback of this necessary reduction is that it disrupts the orthogonality of the system. This is evident because the mode shapes of the full FE model are orthogonal to the mass and stiffness matrices of the full FE model. With the condensation of the mass matrix, this orthogonality is no longer guaranteed. To illustrate this point, Figure 3.4 displays the mass ORTHO correlation of a full FE model with a reduced version of itself in order to examine this effect on the same displacement solution. The deterministic properties of the FE model assure that the resulting values are exact and only those cells rendered in green represent a value of zero.

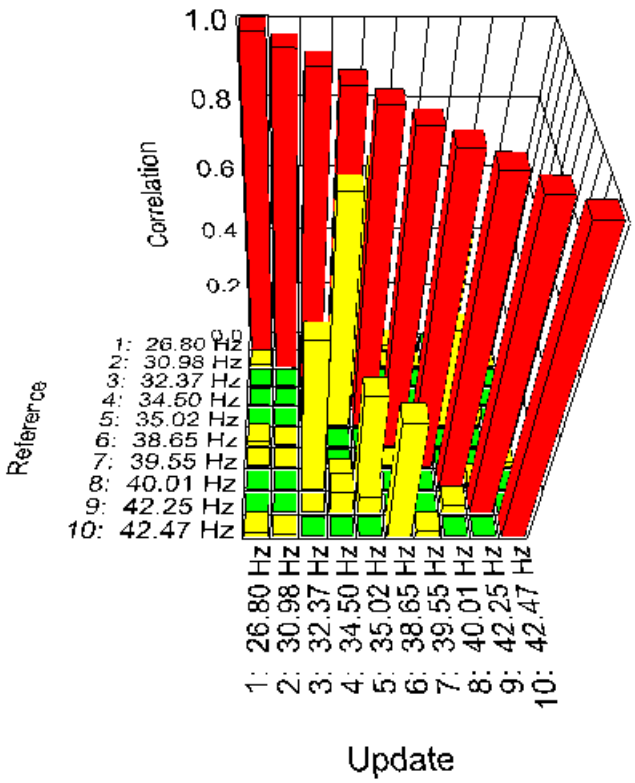

Figure 3.4: Orthogonality Check Not Fully Orthogonal with Reduced Finite Element Model

In the cases where a model reduction is necessary, it is clear that a fully orthogonal matrix should not be expected, even for an auto-ORTHO correlation. The task then is to perform

the reduction such that a minimum loss of orthogonality occurs. This can be achieved by conducting a pre-test analysis, described in detail in [18]. The concept of a pre-test analysis is to determine the optimal placement of sensors on the test model, such that the complete behavior of the mode shapes is captured with a limited amount of measurement locations. If the most influential DOF of the test model can be isolated and measured, a mass ORTHO correlation with a reduced FE model can produce reasonable results. The method and quality of the sensor placement on the test model thus becomes a crucial component of the entire correlation procedure.

3.2.3 Stiffness Orthogonality

Another possible, but less common ORTHO correlation considers the stiffness matrix of one of the dynamic systems. The formulation is similar to that of the mass ORTHO correlation, but will produce a slightly different result. This occurs due to the relationship of the mass and stiffness as seen in Equation 2.17. Therefore, the stiffness ORTHO correlation matrix is also weighted by the square of the eigenfrequencies of the reference model. The stiffness ORTHO equation reflects the form of the mass ORTHO correlation in Equation 3.3:

$$\mathbf{ORTHO}_{ij,K} = \mathbf{\Psi}_{A,i}^T \mathbf{K} \mathbf{\Psi}_{B,j} \tag{3.5}$$

where $\mathbf{\Psi}_{A,i}$ represents a mode shape of one system and $\mathbf{\Psi}_{B,j}$ represents a mode shape of the other. The normalized stiffness ORTHO correlation follows:

$$\mathbf{ORTHO}_{ij,K} = \frac{\left(\mathbf{\Psi}_{A,i}^T \mathbf{K} \mathbf{\Psi}_{B,j}\right)^2}{\left(\mathbf{\Psi}_{A,i}^T \mathbf{K} \mathbf{\Psi}_{A,i}\right)\left(\mathbf{\Psi}_{B,j}^T \mathbf{K} \mathbf{\Psi}_{B,j}\right)} \tag{3.6}$$

where $\mathbf{K}$ represents the stiffness matrix of the reference model.

The properties and considerations of the stiffness ORTHO correlation are similar as those of the mass ORTHO correlation. The main difference is that this correlation also considers the weighting of the reference model eigenfrequencies. The current implementation deals directly with the mass and stiffness matrices of each model to carry out an ORTHO calculation. Therefore, since the form of Equations 3.4 and 3.6 are identical, the selection of the mass or stiffness matrix produces the corresponding correlation with the same amount of computational effort. Furthermore, as both correlations are normalized, their results can be compared and related to each other at the same scale.

3.3 Modal Assurance Criterion

The MAC correlation, like the ORTHO correlation, is a quantitative method for comparing the mode shapes of two models. It is a simpler correlation to implement than the ORTHO correlation because it does not rely on the mass or stiffness matrices. This is one reason why it has gained popularity and is generally presented in the following form:

$$\mathbf{MAC}_{ij} = \frac{\left(\mathbf{\Psi}_{A,i}^T \mathbf{\Psi}_{B,j}\right)^2}{\left(\mathbf{\Psi}_{A,i}^T \mathbf{\Psi}_{A,i}\right)\left(\mathbf{\Psi}_{B,j}^T \mathbf{\Psi}_{B,j}\right)} \tag{3.7}$$

where $\boldsymbol{\Psi}_{A,i}$ represents a mode shape of one system and $\boldsymbol{\Psi}_{B,j}$ represents a mode shape of the other. The form of this correlation is inherently normalized, meaning that the resulting MAC values will range between zero and unity. Similar to the ORTHO correlation, a MAC value greater than 0.9 is considered a strong relationship between mode shapes, while a value less than 0.1 is considered a weak relationship. Interestingly, this correlation strongly resembles the statistical correlation function from probability theory if randomness is neglected. Further examination of the relationship between the MAC correlation and the ORTHO correlation can be seen if Equation 3.7 is expressed as follows:

$$\mathbf{MAC}_{ij} = \frac{\left(\boldsymbol{\Psi}_{A,i}^T \mathbf{E} \boldsymbol{\Psi}_{B,j}\right)^2}{\left(\boldsymbol{\Psi}_{A,i}^T \mathbf{E} \boldsymbol{\Psi}_{A,i}\right)\left(\boldsymbol{\Psi}_{B,j}^T \mathbf{E} \boldsymbol{\Psi}_{B,j}\right)} \tag{3.8}$$

where $\mathbf{E}$ represents the identity matrix. Where the ORTHO correlation utilized the mass or stiffness matrices to weight the contribution of each DOF, the MAC correlation does not apply the dynamic properties of the system nor weight the DOF in any other way. The MAC correlation, therefore, only examines the displacement independence or similarity of the mode shapes themselves.

The interpretation of the quantities in Equation 3.8 for a model correlation analysis is analogous to Equation 3.4:

$\boldsymbol{\Psi}_{A,i}$	mode shape of the reference model
$\mathbf{E}$	mathematical identity matrix
$\boldsymbol{\Psi}_{B,j}$	mode shape of the update model

The disadvantage of the MAC correlation is that it does not utilize the orthogonality properties of the mode shapes to compare two models. The effective use of the identity matrix in place of the mass or stiffness matrices means that no damping of the displacements of inertially insignificant DOF occurs. Rather, the MAC correlation simply compares the relative geometric shapes of the displacement patterns. Given the same geometry and mode shapes as in Figure 3.3 for an ideal mass ORTHO correlation, Figure 3.5 shows the corresponding ideal MAC correlation where only those cells rendered in green represent a value of zero. By omitting the dynamic properties of the system, the resulting normalized correlation only assures that the diagonal values are unity.

This contrast to the ORTHO correlation means that the MAC correlation does not encounter the same practical obstacles. While it still requires a DOF map from one model to the other, the concern over maintaining the orthogonality of the system during this procedure is lifted. In the standard case of a FE model and a test model, the FE model must once again be reduced according to the measurement locations on the test model. The computation, however, proceeds as usual and the fundamental principles of the correlation are not degraded due to the reduction. While the correlation itself is not affected by the model reduction, a proper interpretation of the results depends on the quality of the placement of the measurement instruments on the test model. Therefore, as with the ORTHO correlation, a rigorous pre-test analysis must be performed before deriving conclusions from the MAC correlation.

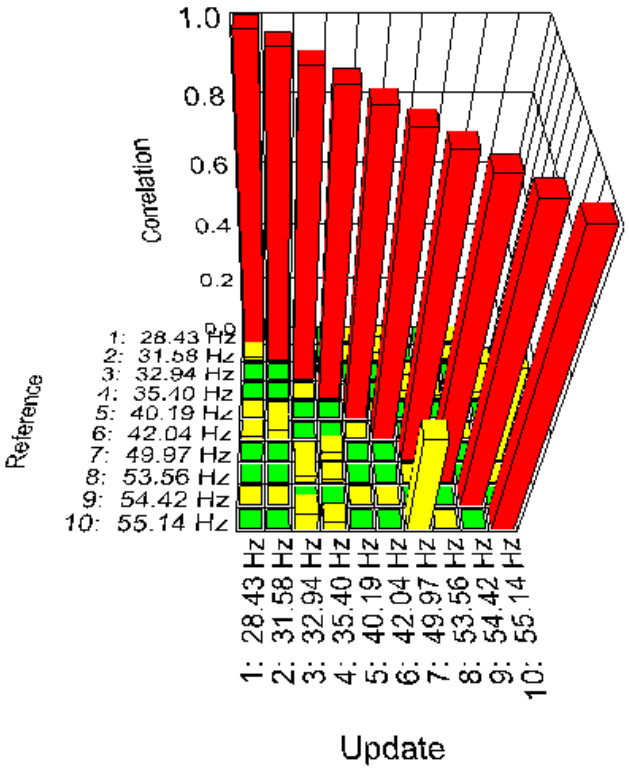

Figure 3.5: Ideal Result for Modal Assurance Criterion Is Non-Orthogonal Matrix

A numerical advantage of the MAC correlation is that it only requires the DOF map and the mode shapes of each model, not the mass and stiffness matrices. Thus, input streams and memory storage for these matrices can be omitted and the reduced amount of matrix multiplications results in less computational effort. This provides one reason why the MAC correlation is often used as an initial comparison of the global mode shapes of two models. Detailed information regarding the interpretation, development, and extensions of the MAC correlation can be explored by referring to [1].

3.4 Frequency Correlation

The frequency correlation, FREQ, is the final correlation presented in this thesis and is a comparison of the temporal eigenfrequencies of two mode shapes. This correlation simply calculates the difference of the eigenfrequencies of each mode:

$$\mathbf{FREQ}_{ij} = f_{R,i} - f_{U,j} \tag{3.9}$$

where, in the current implementation, $f_{R,i}$ represents a temporal eigenfrequency of the reference model and $f_{U,j}$ represents a temporal eigenfrequency of the update model. The normalized FREQ correlation follows:

$$\mathbf{FREQ}_{ij} = \frac{f_{R,i}}{f_{U,j}} - 1.0 \tag{3.10}$$

The FREQ correlation, like the DOF correlation, may seem like a straight-forward correlation, however, it has some significant strengths. A major advantage of the correlation is that the eigenfrequency of a FE model or a test model is likely to be the most accurate data of either system. The benefit of this accuracy is enhanced by the correlation in that neither spatial nor mode shape information is required in the computation. The FREQ correlation, therefore, is independent of the DOF map between the two models and is not limited to the amount of measurement locations on the test model. Consequently, any reduction of the FE model will also not impact the correlation results.

Ideally, there should be no difference in the eigenfrequencies of the two models, however, this is generally not the case. A significant variance in the eigenfrequencies of two models can indicate that the corresponding mode shapes should not have the same form. This is a valuable tool in identifying and handling a phenomenon commonly called *mode switching*.

3.4.1 Mode Switching

When determining the eigenfrequencies and mode shapes of a FE model, the ordering of the mode shapes may not always coincide with that of a test model. A frequent occurrence, which may be due to different mass distributions or symmetric mode shapes, is that a mapping of the eigenfrequencies and mode shapes based on the FREQ, MAC, and ORTHO correlations must be performed.

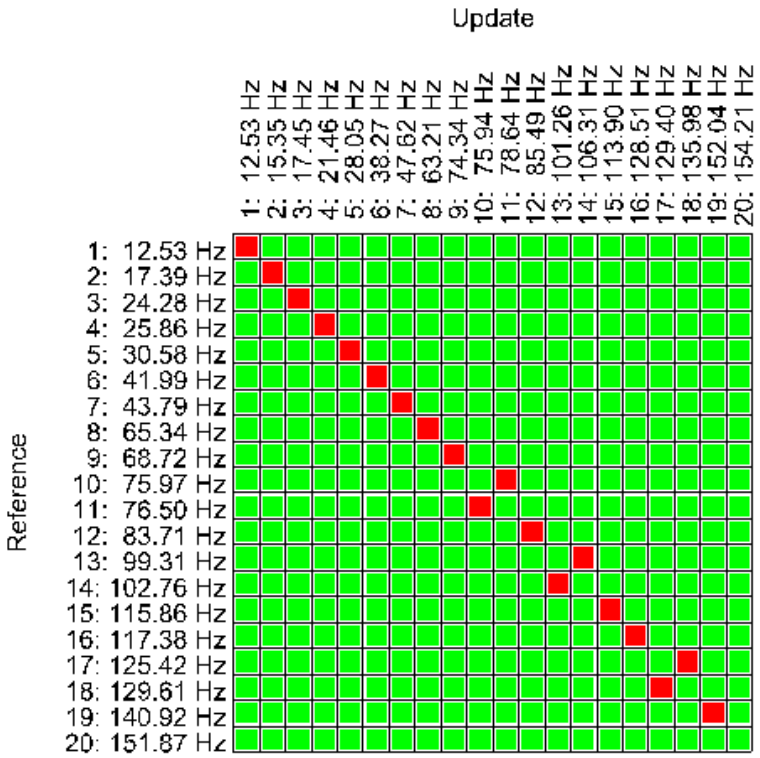

Figure 3.6: Evidence of Mode Switching for Two Comparable Finite Element Models

Figure 3.6 illustrates this effect with a MAC correlation of the first 20 mode shapes of two similar FE models. The 2D visualization, which can also be rendered in 3D, was created using the current implementation. To clearly emphasize mode switching in this example, those MAC correlation values less than 0.7 are displayed in green, while those greater than this threshold are shown in red. Careful consideration of mode switching must be made in order to properly perform a model update, described in **Chapter 4**, where the modes of one model must be mapped to those of the other.

In the previous example, such a mode map can be readily inferred from a visual inspection of the MAC correlation alone. As the amount of correlated mode shapes increases or as the variance in the dynamic behavior of the two models grows, however, this mapping method becomes impractical. Figure 3.7 shows a MAC correlation of the first 40 mode shapes of another pair of similar FE models. The ambiguity of the mode relationships becomes apparent as their statistical similarities mingle with the effects of mode switching. A threshold MAC correlation tolerance may now return multiple viable mappings for a particular mode shape.

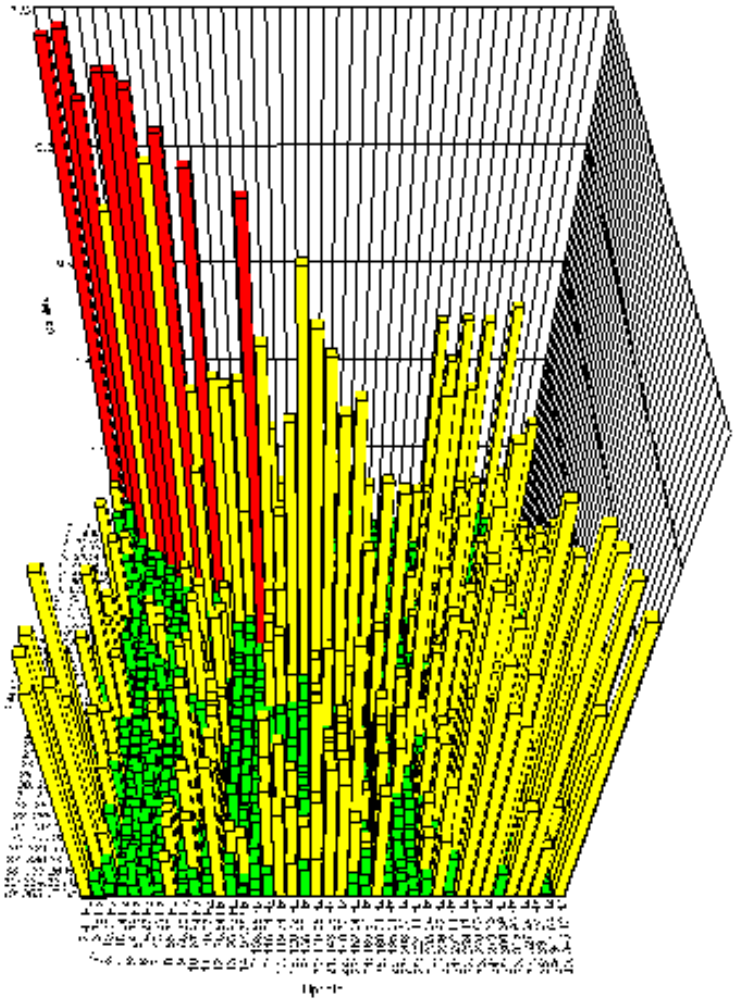

Figure 3.7: Dynamic and Frequency Correlations Must Be Used Together to Map Modes

To supplement the MAC and ORTHO correlations in these situations, the FREQ correlation provides additional information about the eigenfrequencies helpful in eliminating unlikely mode mappings. For a given mode shape from one model, for instance, the MAC and ORTHO correlations may suggest that it correlates to two or more mode shapes from the other model. In many cases, these suggested mode shapes have significantly different eigenfrequencies. Thus, by applying the FREQ correlation, the mode shapes with the strongest MAC, ORTHO and eigenfrequency correlations can be mapped to each other. The combined use of all available correlations, therefore, leads to the optimal modal mapping between two models.

3.4.2 Maximum Dynamic Connection

The modal connection in the current implementation strongly resembles the minimum distance connection described earlier and can be initiated once a MAC or ORTHO correlation matrix has been computed. The only major difference is that the geometric DOF connection sought a minimum correlation, whereas this dynamic connection seeks a maximum correlation. The modal map is created by first finding the maximum dynamic correlation in the matrix. The corresponding modes represent the most likely pair and are added to the modal map. The search for the maximum dynamic correlation is repeated without the previously mapped modes and the next most likely pair is added to the map. This is done iteratively until one model is completely mapped to the other.

The current implementation does not directly use the FREQ correlation to determine the modal map. Rather, the map based on the MAC or ORTHO correlation is returned in tabular form along with the corresponding FREQ correlation for each pair. The user then has the flexibility to switch any mode pair based on the information provided by both correlations. This dynamic mapping is critical for a model update, where an improvement of the overall dynamic correlation consists of minimizing the discrepancy of both the eigenfrequencies and the mode shapes. Further discussion of the implemented modal connection algorithm is presented in **Chapter 5**.

Chapter 4

Model Update

Once data from a correlation analysis of a reference model and an update model has been retrieved, confidence limits can be applied to examine the validity of the update model. In the following sections, the discussion primarily focuses on the case where the reference model is a test model and the update model is a FE model. Within the scope of this thesis, a discussion of pre-test analysis or optimization of the positioning of measurement instruments on test models is not covered. In reality, though, a rigorous pre-test analysis may need to be performed in conjunction with the following FE model optimization techniques. Further reading regarding pre-test analysis can be found in [18].

If the confidence limits of the correlation analysis are met, the predicted behavior of the FE model can be accepted as valid. If, however, the behavior of the FE model deviates significantly from that of the test model, one or more of its parameters must be updated to improve the correlation. This can be done in a variety of ways, however, a comprehensive and efficient method is usually preferred. To be comprehensive, the update method should use all or as much of the available correlation data to determine if an overall improvement has been achieved. Efficiency can be attained by utilizing a systematic method for determining the optimal value or setting for a particular parameter. The method presented in the following sections is an application of the *Bayesian parameter estimation* and solved numerically using the MSC.Nastran SOL 200 command for design optimization.

The standard elements of structural optimization include an objective function, design variables, system responses, inequality constraints and equality constraints. The Bayesian parameter estimation incorporates all of these aspects into one equation that represents an error that should be minimized through an iterative process. As the aim of the following sections is to present the client's requirements regarding FE model update and optimization, the solution of this error minimization is not described here in detail. Rather, determination of the solution and optimized parameters is performed by the MSC.Nastran SOL 200 design optimization tool, which is further documented and referenced in [12]. A broader interpretation of the general solution procedure as well as other applications of structural optimization pertaining to the automotive industry can be found by referring to [7].

Given the breadth of the topic of structural optimization, it is important to clearly outline the client's requirements and existing capabilities within the framework of this thesis. The essential

components of the Bayesian parameter estimation come in two parts, the objective function and the design variables. The client has pre-existing software to independently generate a MSC.Nastran optimization deck for the design variables. The corresponding MSC.Nastran deck for the objective function, which in this case can be formulated from any combination of the specific correlation analysis methods described in **Chapter 3**, is the primary focus of the following sections. The client also has a Fortran implementation to strictly create an optimization deck for the MAC and frequency correlations of two models, which serves as a reference, but requires some intermediate processing of the model data.

Part of the client's specifications, therefore, are to provide the same functionality in the current Java implementation without a significant reduction in computational efficiency. An additional optimization deck should be similarly fashioned for the ORTHO and FREQ correlations of two models. Finally, the correlation data should be given 2D, 3D, and tabular representations to properly assess the improvement of the FE model before and after an optimization trial. The discussion regarding these particular representations is continued in **Chapter 5**. The contents of the following sections strive to provide an overview of the theoretical basis necessary for fulfilling the client's requirements and are taken from [2] and [11].

4.1 Bayesian Parameter Estimation

An objective function represents a quantity or quantities whose final value should be optimized, typically through an iterative process where it is minimized or maximized. Common examples of such definitions in industry are the minimization of weight or the maximization of strength of a particular design. In this application, the objective function represents the minimization of the error in the correlation of two design models. Thus, it is dependent on the correlation methods, such as the MAC, ORTHO, and FREQ correlations, which are computed from the responses of each respective model.

The design variables generate the space of all possible combinations of the modifiable parameters of the system. This design space is usually broken into two types, called the feasible domain or the infeasible domain. The separation of these two domains is typically delineated by inequality or equality constraints. These constraints are sometimes derived from the possible bounds of the design variables, for instance, a geometric constraint might prevent a design from incorporating a negative thickness for any part. Constraints may also be formulated from relationships of the design variables, for instance, a minimum allowable surface area calculated from the geometric properties of a part. The feasible design space is then limited to the combinations of admissible values of the design parameters.

With the proper definition of the objective function, the design variables, and their constraints, a sensitivity analysis can be performed. This is accomplished by varying the values of the design variables within the feasible domain until the optimal result of the objective function is determined. While there exist a myriad of methods to efficiently and accurately locate this point for a whole host of situations and configurations of varying complexity, which can be examined further in [7], the method used in this application is implemented by the MSC.Nastran SOL 200 command.

The Bayesian parameter estimation combines the objective function and design variables into one equation representing the error between the results of a reference model and an update model. The purpose of the estimation procedure is to vary the desired parameters until this error is minimized. The following general form is presented in matrix notation:

$$E = wt \cdot (\mathbf{RT} - \mathbf{RA})^T \mathbf{WR} (\mathbf{RT} - \mathbf{RA}) + wp \cdot (\mathbf{PF} - \mathbf{PO})^T \mathbf{WP} (\mathbf{PF} - \mathbf{PO}) \tag{4.1}$$

where the variables are defined as:

RT	responses from the test
RA	responses from the analysis
WR	weighting factors or confidences for the responses
PF	parameters of the final model
PO	parameters of the original model
WP	weighting factors or confidences of the parameters
wt	scalar weighting for the test data as a whole
wp	scalar weighting for the model parameters as a whole
E	the error to be minimized

Some viable parameters include element, damping, or geometric properties. As previously stated, the client already has the capabilities to generate the expression for the design parameters, rendering the latter portion of Equation 4.1 unnecessary:

$$wp \cdot (\mathbf{PF} - \mathbf{PO})^T \mathbf{WP} (\mathbf{PF} - \mathbf{PO}) \tag{4.2}$$

which focuses the work of this thesis on the expression for the objective function:

$$wt \cdot (\mathbf{RT} - \mathbf{RA})^T \mathbf{WR} (\mathbf{RT} - \mathbf{RA}) \tag{4.3}$$

The formulation of the Bayesian parameter estimation provides constraints by directly coupling the variance of the parameters into the objective function. In some typical cases of structural optimization, the optimal values of the design variables may be quite different from their original value and may be limited only by the extent of the feasible domain. In this case, there is a balance between the optimal response of the FE model and the influence of the original global design in the resulting optimization. The client's current optimization procedure, however, does not aim to maintain this balance.

The incorporation of weight factors or confidences, a sign of multi-objective functions, provides a large degree of customization for the optimization. The resemblance of the final and original structures, based on these confidences, can be significantly varied based on the client's requirements. This can be seen because the weight factors do not have absolute restrictions. The test data, for example, can be assumed to be accurate and its corresponding weight factor may be assigned as unity. The model parameters, on the other hand, may be given freedom to radically change by applying a weight factor many orders less than unity. To further emphasize the customization options of this estimation, each parameter and response also has the flexibility to be assigned individual weight factors. The client's requirements, therefore, are to apply this flexible estimation scheme to the dynamic correlation of two models.

4.2 Objective Functions

As the essential deliverables of this thesis, the two objective functions presented in this section allow the client to efficiently optimize a dynamic FE structure based on quantitative information. The optimization solution itself is computed using the MSC.Nastran SOL 200 design optimization tool, however, the ability to generate the optimization deck from raw test and FE data is the culmination of the principles presented in the previous chapters. The implemented Java GUI, elaborated upon in **Chapter 5**, provides an intuitive interface to streamline the optimization process and interpret the results of the initial, intermediate, and final designs, which was not previously possible.

4.2.1 Modal Assurance Criterion Optimization

Equation 4.3 outlines the expression to be implemented in the current Java implementation. The client has an existing Fortran program that implements the desired format of this expression, which uses the MAC and FREQ correlations and serves as a reference. The following equation represents this reference objective function and substitutes the simple difference of the dynamic responses from the test and FE models with the difference of their optimal and current correlations:

$$f_{MAC} = \sum_{j=1}^{n} \left\{ W_{freq,j} \cdot \left(\frac{f_{reference,j}}{f_{update,j}} - 1.0 \right)^2 + W_{MAC,j} \cdot (MAC_j - 1.0)^2 \right\} \tag{4.4}$$

where the variables are defined as:

$f_{reference,j}$	eigenfrequency of the connected mode shape from the reference model
$f_{update,j}$	eigenfrequency of the connected mode shape from the update model
MAC_j	MAC correlation of the connected mode shapes from each model
$W_{freq,j}$	scalar weighting for the FREQ correlation
$W_{MAC,j}$	scalar weighting for the MAC correlation
n	the size of the modal map
f_{MAC}	the error of the MAC objective function

This expression for the objective function utilizes two of the four correlation techniques described in **Chapter 3**. The exact form of the normalized FREQ correlation from Equation 3.10 is placed in the first parentheses as it represents a simple difference, whose optimal value is inherently zero. The optimal value for a MAC correlation between two modes, however, is unity. In order to standardize the form of the objective function such that the optimal error is zero, not simply a local minimum, unity is subtracted from the existing MAC value. The current implementation of this objective function is discussed further in **Chapter 5**.

An important note regarding this objective function is that the mode shapes of interest are those which are connected to each other. Neither the properties of the MAC nor FREQ correlations guarantee an optimal value between unconnected mode shapes, meaning that MAC correlation values of unconnected mode shapes must not necessarily be zero nor does there exist a general relationship between their eigenfrequencies. For this reason, the MAC and FREQ correlations of the unconnected mode shapes are not considered.

4.2.2 Orthogonality Check Optimization

Using the form of Equation 4.4, a similar objective function can be written utilizing the ORTHO and FREQ correlations. The normalized FREQ correlation remains unchanged, while the ORTHO correlation replaces the MAC correlation, leading to:

$$f_{ORTHO} = \sum_{j=1}^{n} \left\{ W_{freq,j} \cdot \left(\frac{f_{reference,j}}{f_{update,j}} - 1.0 \right)^2 + W_{ORTHO,j} \cdot (ORTHO_j - 1.0)^2 \right\} \tag{4.5}$$

where the definitions of the variables are analogous to Equation 4.4:

$f_{reference,j}$	eigenfrequency of the connected mode shape from the reference model
$f_{update,j}$	eigenfrequency of the connected mode shape from the update model
$ORTHO_j$	ORTHO correlation of the connected mode shapes from each model
$W_{freq,j}$	scalar weighting for the FREQ correlation
$W_{ORTHO,j}$	scalar weighting for the ORTHO correlation
n	the size of the modal map
f_{ORTHO}	the error of the ORTHO objective function

While the notation may be quite identical to the previous objective function, the necessary output of the mass matrix for the ORTHO correlation creates some logistical difficulties. The returned MSC.Nastran ASCII text-formatted optimization deck must explicitly state the entire ORTHO correlation to determine the optimized solution of the objective function. The inclusion of the mass matrix in the ORTHO correlation multiplies the output stream by approximately the number of connected DOF, which can reach 3000 for 1000 nodes in the current implementation. Typical usage, however, should generally be one-tenth or less of this maximum, which should be manageable in most cases.

This leads directly to another issue, which is that the ORTHO correlation guarantees that the off-diagonal values must be zero whereas the MAC correlation did not. Therefore, Equation 4.5 should include an error for every pair of unconnected mode shapes whose ORTHO correlation does not produce zero. The objective function in the current implementation, though, like Equation 4.4, does not consider these unconnected mode shapes for optimization. The reason for this is again to guard against an extreme size of the output stream as well as imperfect model reduction of test models as described in **Chapter 3**, which can produce a loss of orthogonality.

In the current implementation, the maximum amount of mode shapes that can be safely connected for an optimization deck is 100. Considering all unconnected mode shapes would thus multiply the output stream by approximately 100 times. When compounded with the previous increase from the mass matrix computation, the extreme scenario would produce an output stream approximately 30,000 times larger than the corresponding MAC optimization deck. Considering this, combined with the client's specification that in practice the correlations of the connected mode shapes are the primary concern, the current implementation does not take into account the unconnected mode shapes for the creation of this optimization deck.

Chapter 5

Software Development

The implementation of a stand-alone Java program representing the ideas presented in the previous chapters constitutes a significant portion of this thesis and the deliverable end product per the client's requirements. The selection of Java over other programming languages was based in large part on its portability and available libraries for generating a GUI. Java is platform-independent, thus allowing the program to be transferred between various operating systems, which is one concern the client has expressed. A standard 2D library, *Swing*, is commonly used for generating a GUI with the same standard *Look and Feel* as other familiar and intuitive Java programs. An extended 3D library, *Java 3D*, can be utilized to provide an interactive visualization for many applications, namely to examine the geometry of either a test model or a FE model and the correlation analyses between them.

Java is also an Object-Oriented Programming (OOP) language with a strong emphasis on the ability to reuse code in different applications and for multiple purposes. This has led to the creation of a large number of standard libraries useful for most users, including the Swing package especially oriented to GUI design. A user interface is essential in almost all programs, whether it is a text-based command line or an elaborate graphical display. This necessitates an internal structure to handle the interactivity between the user and the program. Thus, an important design concept to be familiar with is the *Model-View-Controller* (MVC) architecture pattern, which is maintained throughout the Swing package. To adhere to the principles of MVC and OOP design, significant effort has been made to structure the current implementation according to these programming practices.

The Java 3D package is a useful library for rendering three-dimensional scenes, however, it is not yet standard to Java and requires an understanding of scene graph theory to be effectively applied. The diverse capabilities of Java 3D include geometric rendering, interaction, animation, texturing, lighting, and sound features. Therefore, a carefully detailed scene can fully be realized with a comparable amount of effort. For the various purposes of this thesis, geometric rendering and simple interaction are sufficient to generate the appropriate visualizations. It is important to note that the visualizations rendered in Java 3D are dependent on the graphics card of the computing hardware and generally comprise the bulk of the processing execution time of the software. When loading FE models with many nodes, connections, and elements on computers with different graphics cards, the client may experience changes in execution speed due to the native rendering process.

Another advantage of Java is the amount and standardization of its documentation, for which extensive support exists online. Reference queries for the source code of the standard libraries were almost exclusively provided by [16]. Additional resources regarding the structure and MVC architecture of typical Java GUI programs were found in [9] and [17], while the standard practices for implementing the graphical interface itself were presented in [15]. Finally, information regarding basic visualization theory pertaining to the particular Java 3D implementation was obtained through [5]. The contents of the subsequent sections regarding the fundamental aspects of Java were taken from, influenced by, or derived from these sources. To perpetuate Java's standard style of documentation, the *Javadoc*, an API of the same form is provided to the client fully detailing the implemented source code.

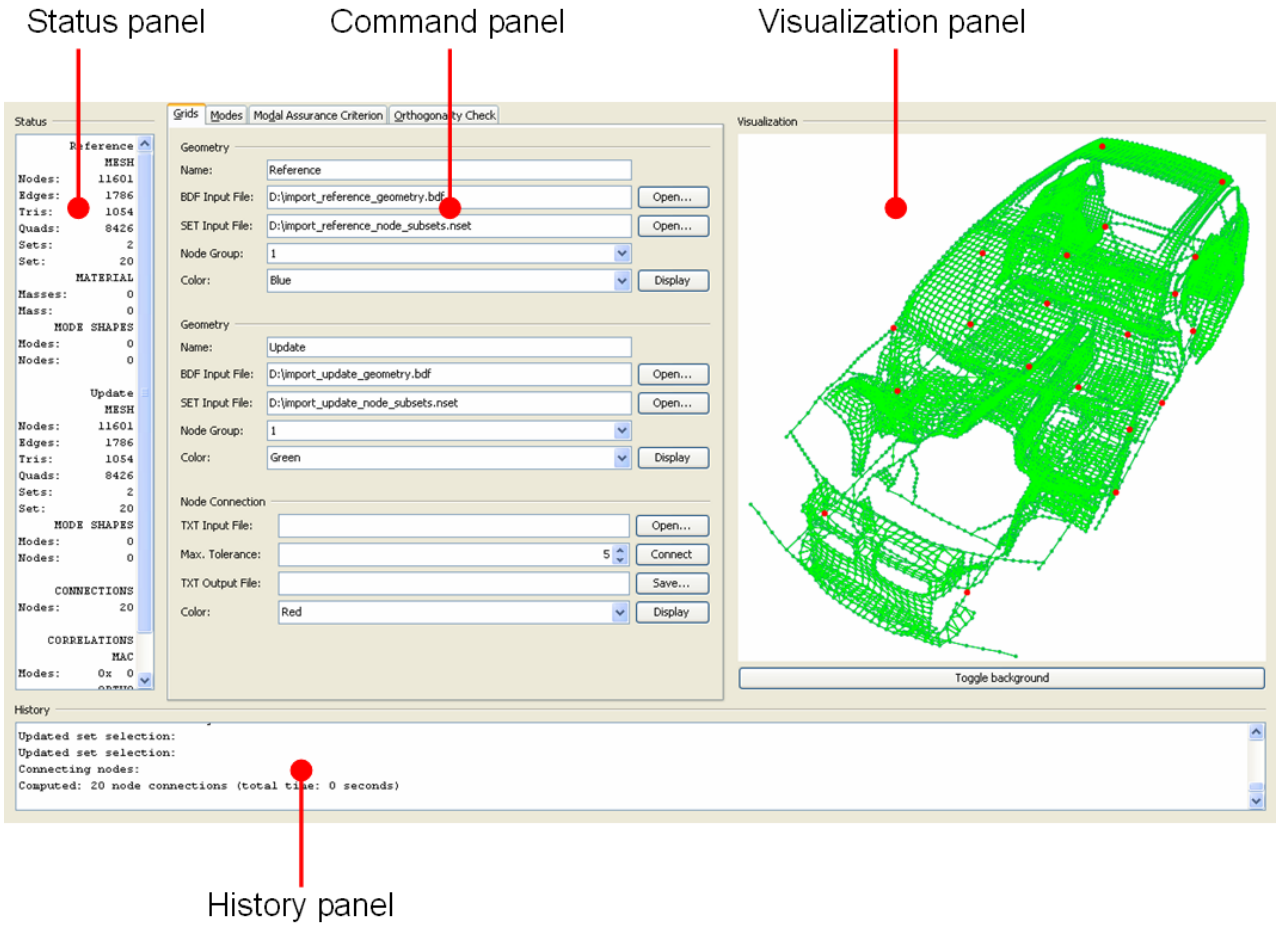

Figure 5.1: Software Screenshot Displaying Geometric Connection of Two Finite Element Models

To give the reader an impression of the final product for the discussion in the following sections, a screenshot of the Java GUI is displayed in Figure 5.1 where the geometries of two similar FE models have been connected. The main modules consisting of the command, visualization, status, and history panels are highlighted. The command panel in the center of the GUI provides all the functionality for performing the operations detailed throughout

this thesis. The implementation is discussed in the following sections and more information regarding the proper use of these capabilities is demonstrated in **Chapter 6**.

The final presentation of the software package given to the client is composed of 120 source files, most of which are necessary to maintain a Swing MVC and OOP design. These source files include both classes and interfaces, which provide the framework for a modular structure in almost all aspects of the current implementation, promoting the reuse of source code in other applications as well as additional future work pertaining to dynamic model validation and update. The main ideas and concepts behind generating the source code are presented in the following sections, though detailed documentation for each class and interface can be found in the Javadoc accompanying the software code distributed to the client.

5.1 Model-View-Controller Architecture

In the fundamental stages of designing an interactive software package, the concept of the MVC architecture pattern has gained recognition and is used in the core of the Java Swing package. This section aims to provide a brief overview of the guiding principles used in the current implementation and to illustrate its global design. According to [9], the MVC pattern separates a visual application into three main parts:

- A *model* that represents the data for the application
- The *view* that is the visual representation of the data
- *A controller* that takes user input on the view and translates that to changes in the model

The simplified diagram in Figure 5.2 interprets the idea of the MVC pattern, where the arrows indicate references from one component to another. These reference connections allow each component to notify the other components that its state has been updated and to enact responses based on user-driven events. In order to *listen* for notifications from a particular component, the listener must register itself to the target. In general, each component of the MVC pattern is subdivided into many subcomponents, thus creating an intricate network of references and notification pathways to customize the program behavior. It is therefore important to ensure that the final design is *acyclic*, meaning that there are no loops where two or more component actions are dependent upon each other, which would create a vicious cycle of unending notifications that may stall the program.

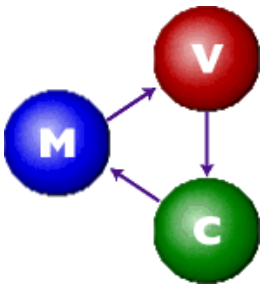

Figure 5.2: Simplified Diagram of Model-View-Controller Architecture [9]

When the user initiates an event or action such as a mouse-click on a button, for example, the controller responds and makes appropriate changes to the model. The model notifies the view that its state has been updated and the view then queries the model for the new information it needs to display. Depending on the nature of the action, these notifications and queries may bounce back and forth between components until all components represent the most recent data. In this way, there is a level of decoupling between different components, allowing software developments to be contained within one area of the design. This concept prevents the effect of minor source code changes rippling through the entire implementation causing other source code changes, which could easily become a tedious and time consuming process.

5.1.1 The Model

The model, sometimes called *business logic* or simply *logic*, of a visual application is where data storage, manipulation, and retrieval occurs. Generally, the model can be thought of as independent of the view and controller, but it does allow access for the controller to affect changes to it and provides methods for the view to obtain the currently stored data. Typically, the view first registers itself as a listener of the model. The model then *fires* notifications informing the view to retrieve the current information and update the display.

In terms of this thesis, the initial data collected from any FE or test model combination consists of the following for each model:

- nodes, bars, beams, triangles, and quadrilaterals of the geometry
- lists of subsets of nodes
- eigenfrequencies and corresponding mode shapes of the dynamic solution
- mass and stiffness matrices

where each group of data is read in from a MSC.Nastran file or, in the case of the eigenfrequencies and mode shapes of a test model, a Universal file. In the current implementation, the input and output streams necessary for importing and exporting data are considered part of the model and are explained further in Section 5.4. Most of the imported data can be stored internally in matrix form of varying types, with which the model can perform the correlation analyses described in **Chapter 3**.

The first correlation that must be performed is the DOF correlation and connection, which requires the geometries of two models and the desired subsets of nodes to connect. This connection can be imported from an ASCII file or computed from the geometries and exported to an ASCII file. After a DOF connection has been computed, the eigenfrequencies and mode shapes of both models can be imported. Using the DOF connection, eigenfrequencies, and mode shapes, the MAC and FREQ correlations can be successfully computed and exported to a tabular Excel file. Using these two correlations, the MAC optimization deck can be generated and exported to a MSC.Nastran file to be solved. The procedure to generate an ORTHO optimization deck is similar, except that the mass matrix must be imported prior to the calculation of the ORTHO can FREQ correlations. A simple flow chart illustrating the necessary steps to create an optimization deck is presented in Figure 5.3.

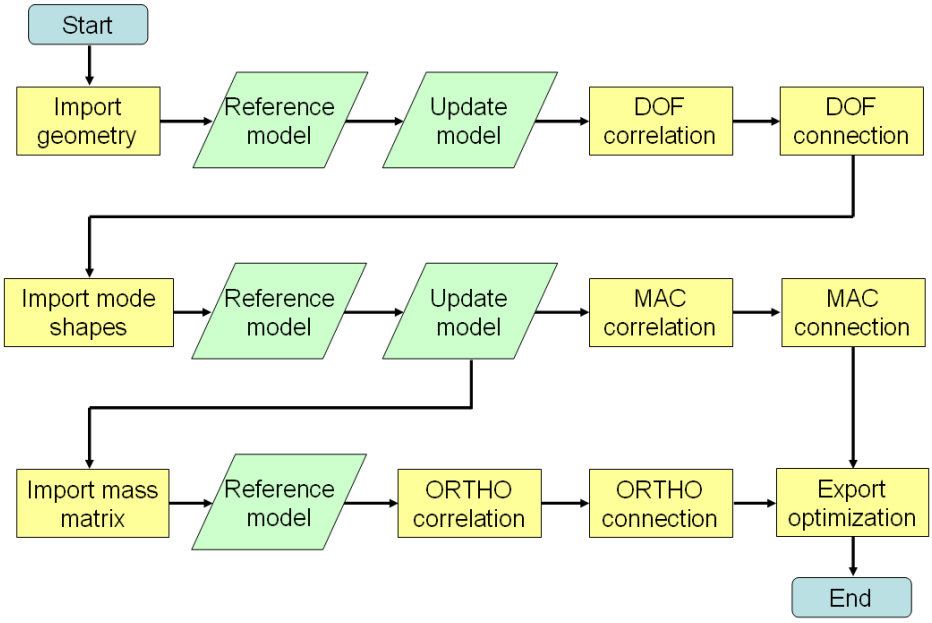

Figure 5.3: Procedure to Generate and Export an Optimization Deck

The model must have the capacity to store all the data that is imported and intermediately computed. To accomplish this, the model is subdivided into several parts, the first of which is a class called JMeshPanel, in part because it extends Swing's JPanel. JMeshPanel is used to store the geometry of one FE or test model and its structure is represented by the simple Unified Modeling Language (UML) class diagram shown in Figure 5.4.

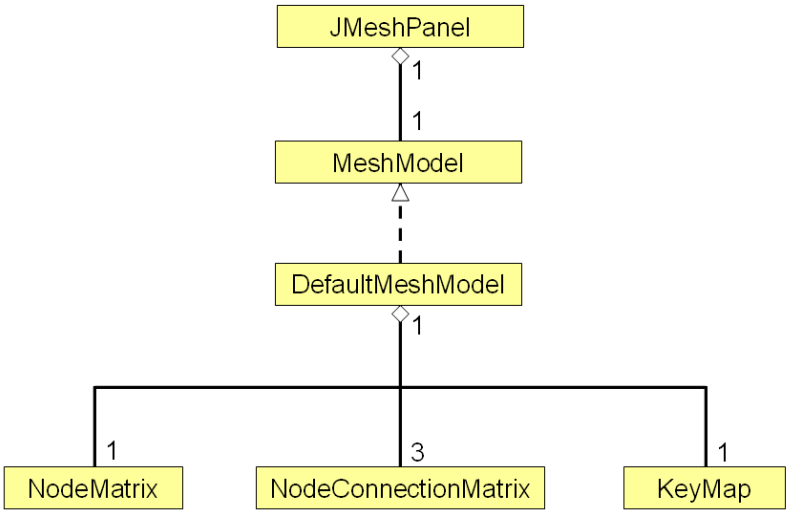

Figure 5.4: UML Class Diagram for a JMeshPanel

The components of a JMeshPanel have the following general interpretation:

MeshModel	an interface that all mesh models must implement
DefaultMeshModel	the default model of a JMeshPanel
NodeMatrix	a matrix to store nodal coordinate data
NodeConnectionMatrix	three matrices to store edges, triangles, and quadrilaterals
KeyMap	subsets of nodes to be connected

For the discussion of the model, it is sufficient to understand the purpose of the classes and interfaces presented, however, more information regarding the mathematical implementation of these classes is given in Section 5.2. When prompted to import geometric or nodal subset data, JMeshPanel calls two readers that parse the respective MSC.Nastran files and store all the available data into this scheme.

With a self-contained object to import and store the geometry of a dynamic model, an object to compute, store, import, and export the DOF connection must be created. This leads to the JMeshConnectionPanel, which is composed of two JMeshPanel objects. The computation of the DOF correlation and connection is performed in one step as an intermediate view of the correlation itself is unnecessary. Figure 5.5 displays the corresponding UML class diagram:

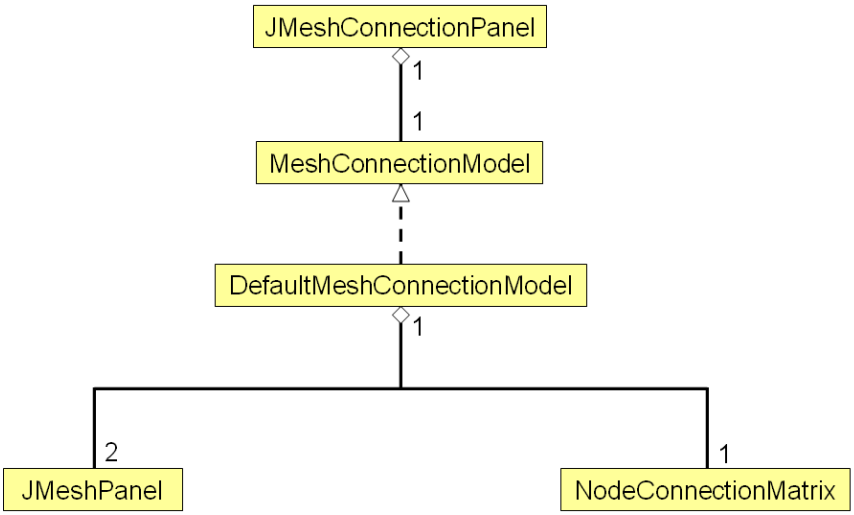

Figure 5.5: UML Class Diagram for a JMeshConnectionPanel

The components of a JMeshConnectionPanel are described as follows:

MeshConnectionModel	an interface that all connection models must implement
DefaultMeshConnectionModel	the default model of a JMeshConnectionPanel
JMeshPanel	storage for the geometry of each dynamic model
NodeConnectionMatrix	storage for the DOF connection of the two models

Similar to the JMeshPanel, the JMeshConnectionPanel calls an ASCII reader or writer to import or export the DOF connection. The algorithm of the DOF connection itself is discussed

further in Section 5.2.

Once a geometric DOF connection has been achieved, storage for the mode shapes of each model must be considered. This is handled by a JModePanel, which calls the respective MSC.Nastran or Universal readers for a FE model or a test model. The relative simplicity of its structure is shown in Figure 5.6:

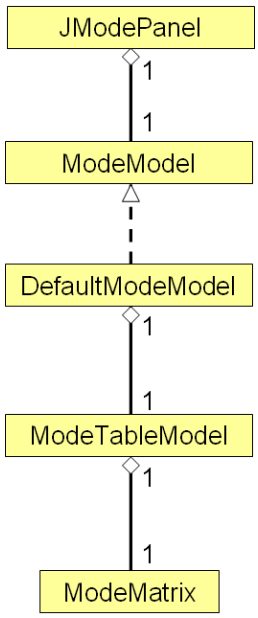

Figure 5.6: UML Class Diagram for a JModePanel

The corresponding components of a JModePanel are outlined as follows:

ModeModel	an interface that all mode models must implement
DefaultModeModel	the default model of a JModePanel
ModeTableModel	a wrapper class for tabular representation of the mode shapes
ModeMatrix	storage for the mode shapes of a dynamic model

Given the previous models for handling geometric storage, dynamic storage, input streams, and output streams, a dynamic correlation can be computed and stored. The strength of the MVC pattern can be seen in the resulting JModeCorrelationPanel, which must have the ability to represent either a MAC correlation or an ORTHO correlation. The computation of each of these correlations differs by the inclusion of the mass matrix in the ORTHO correlation, which also affects its storage scheme. To cope with these differences, two instantiable classes implement the model of the JModeCorrelationPanel, which can simply be interchanged according to the desired application. The use of the model interface, which determines the functionality of the JModeCorrelationPanel, localizes source code changes to the implementing classes and allows multiple types of dynamic correlations to be employed under the same basic framework. The differences between the UML class diagrams of the MAC correlation and ORTHO correlation can be seen in Figures 5.7 and 5.8:

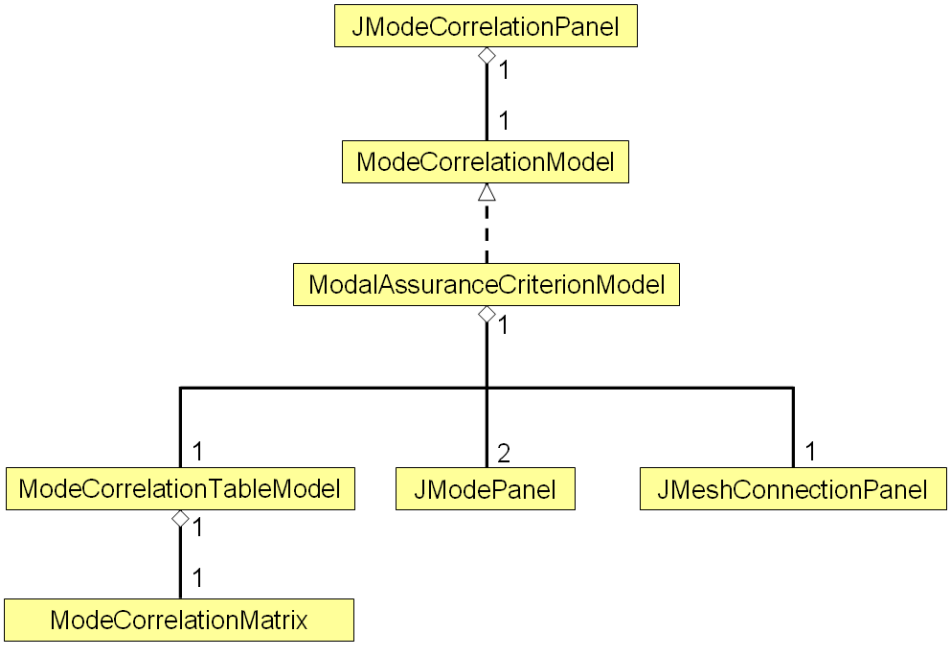

Figure 5.7: UML Class Diagram for a Modal Assurance Criterion JModeCorrelationPanel

A JModeCorrelationPanel can compute and export the dynamic correlation to an Excel file. Following a correlation, the dynamic connection can be made for the MSC.Nastran optimization deck, which can be exported as the final step in the procedure.

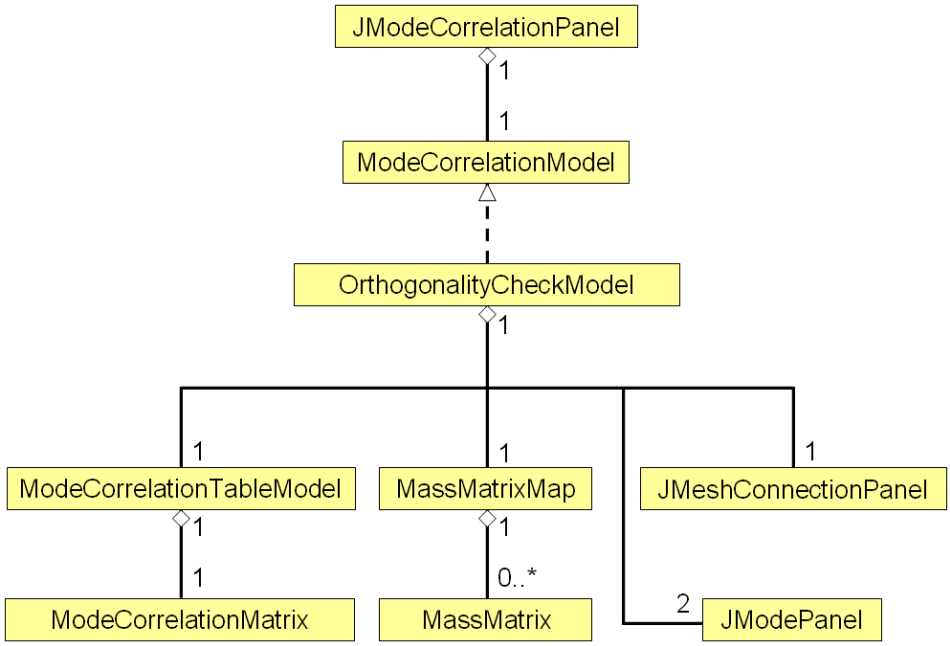

Figure 5.8: UML Class Diagram for an Orthogonality Check JModeCorrelationPanel

The components of a JModeCorrelationPanel for either a MAC or ORTHO correlation are described as follows:

ModeCorrelationModel	an interface that correlation models must implement
ModalAssuranceCriterionModel	implementation structure for a MAC correlation
OrthogonalityCheckModel	implementation structure for an ORTHO correlation
ModeCorrelationTableModel	wrapper class for tabular correlation representation
JModePanel	storage for mode shapes of each dynamic model
JMeshConnectionPanel	storage for DOF connection of the two dynamic models
ModeCorrelationMatrix	storage for mode shape correlation of the two models
MassMatrixMap	storage for multiple mass or stiffness matrices
MassMatrix	storage for an individual mass or stiffness matrix

After the individual logical components have been defined and implemented, they must be *wired* together. The JModeCorrelationPanel can be considered a top level container and when it is created, the objects that it is composed of are instantiated as well. It is important to recall that there are two JModeCorrelationPanel objects in the final software package, one for the MAC correlation and one for the ORTHO correlation. They would normally create their own storage schemes for geometric and dynamic data, however, in this case they should ideally share the same data. Therefore, the ModalAssuranceCriterionModel and OrthogonalityCheckModel constructors allow the same lower level objects to be referenced in both correlation models. To accomplish this final task, a JTabbedPane instantiates each object in the proper sequence and wires the references to each component accordingly.

5.1.2 The View

The most fundamental part of an interactive OOP software program is the model, where the capabilities of the entire program are declared and implemented. An interactive model, however, must receive commands and input data from the user to perform particular functions or methods. The user thus interacts with the model through the view, which is also sometimes termed as the User Interface (UI) in Swing. The UI can come in different forms, but in Swing as well as in the current implementation, it is a graphical 2D display. Figure 5.9 shows a sample screenshot of a UI of a JMeshPanel, in this case the DefaultMeshUI:

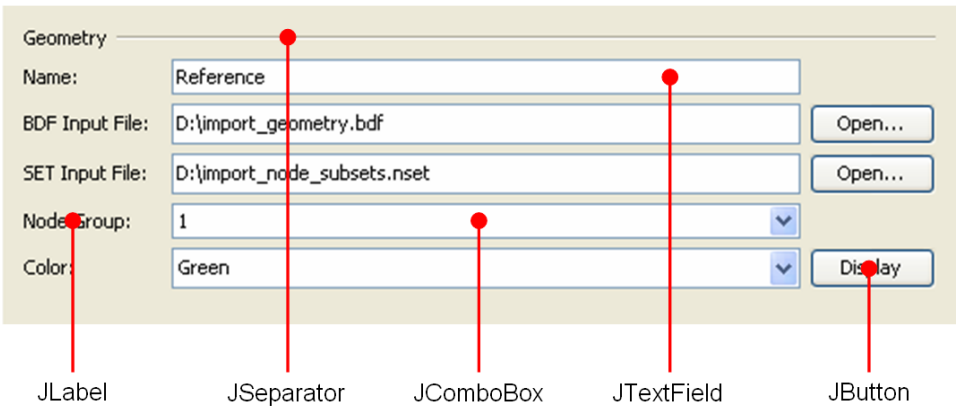

Figure 5.9: User Interface of a JMeshPanel

The view can display information stored in the model and must therefore be aware of any data changes. In this respect, the view depends on the model to provide access to any relevant information. The manner in which the view displays this data, however, is independent of the model. The DefaultMeshUI contains buttons, labels, separators, text fields, and combo boxes. There are a number of other types of components that a UI may have, though, which are extensively documented in [16]. Figure 5.10 shows the view structure of a JMeshPanel, which has a noticeable resemblance to the corresponding UML model diagram in Figure 5.4.

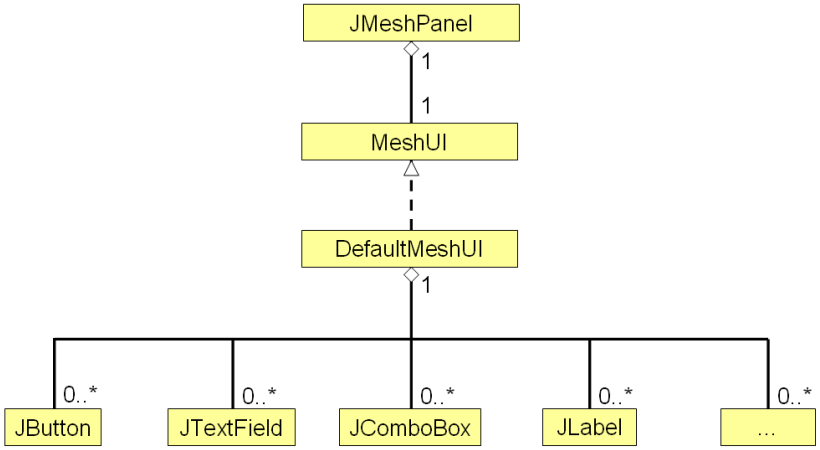

Figure 5.10: UML Class Diagram for the User Interface of a JMeshPanel

The UI components of a JMeshPanel mirror that of its model components:

MeshUI	an interface that all mesh UI must implement
DefaultMeshUI	the default UI of the JMeshPanel

The DefaultMeshUI, like the other panels discussed, is composed directly out of Swing components such as JButton or JLabel. The UML class diagram for these other panels, therefore, is analogous to the diagram outlined in Figure 5.10. Given an operational UI, the user may see a display with a look and feel that seems intuitive, but will find that it bears no actual functionality. The UI properties must be merged with those of the model to produce what most users understand as a program. In this case, these two separate entities are connected in the JMeshPanel, which can be thought of as a *facade*.

The JMeshPanel thus wraps the functionality of the MeshModel, delegating responsibility to it. To obtain the properties of the MeshUI, the MeshUI must be *installed* onto the JMeshPanel. This includes laying out the subcomponents, registering listeners, creating keyboard mnemonics, and setting mouse-over tool tips among other responsibilities. The user then deals directly with the JMeshPanel, while the MeshModel and MeshUI are hidden behind it. Changes to the logic or representation of the JMeshPanel can be made, respectively, by swapping the MeshModel or MeshUI with another one. While this MVC design is beneficial to integrate future work, the JModeCorrelationPanel already makes use of this capability to switch between the model and UI properties of the MAC or ORTHO correlations.

As with the individual model components, the corresponding UI components must be wired together or *layered*. The concept of layering in Java is naturally conveyed in Figure 5.11, which gives further examples of standard Swing components:

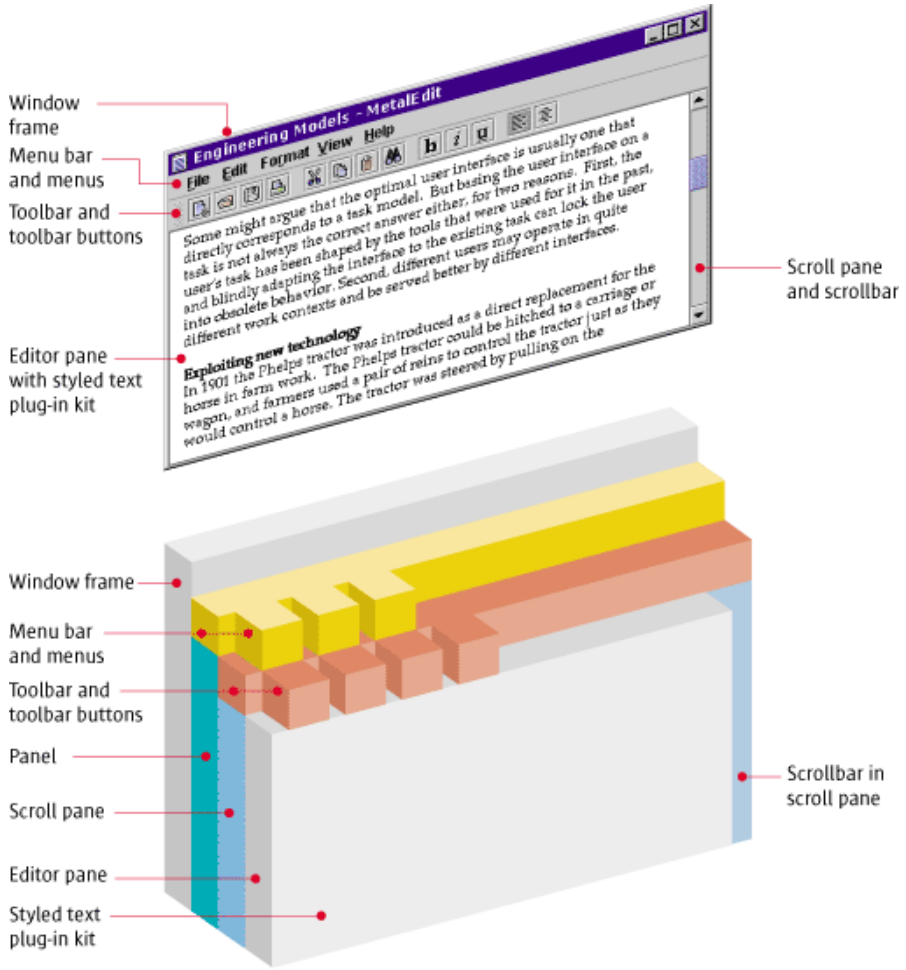

Figure 5.11: Anatomy of a Java Swing Primary Window [15]

For the UI of each implemented component, Swing's standard JFrame, JTabbedPane, and JPanel components are extended or subclassed, providing the broad base upon which subcomponents can be installed and contained. The approach taken in the current implementation is a straight-forward application of the basic principles of Swing UI design, the specifics of which are explored in detail in [9], [15], and [16]. To emphasize a previously stated objective, the effort to develop this MVC and OOP framework increases the initial amount of source code, but is meant to increase the scalability and reduce the maintenance of the global design.

5.1.3 The Controller

The final component that completes the MVC triad is the controller, whose responsibilities are generally subdivided into compact units and handled by *listeners* in Swing. As outlined in the previous example, the MeshModel and MeshUI are linked to each other through a facade, the

JMeshPanel. The details of this connection are defined, in general, by one or more listeners. Listeners have the ability to react to user-defined events, such as mouse-clicks, mouse-rollovers and keyboard strokes. To listen for events, they register themselves to a UI subcomponent, which notifies them that the user has acted. The listener, in turn, calls the appropriate method from the model to provide the intended behavior.

An intuitive example of the MVC pattern can be seen in a simple button. A button is given a visual representation by its UI and an event is fired to the listener whenever the user clicks on the button. The listener then performs a particular sequence of actions, usually calling one or more methods in the button model. As can be seen from this example, the features of a listener are generally tailored towards a specific UI component, therefore the controller and view have a strong relationship. For this reason, Swing integrates the two into a *UI Delegate* as seen in Figure 5.12:

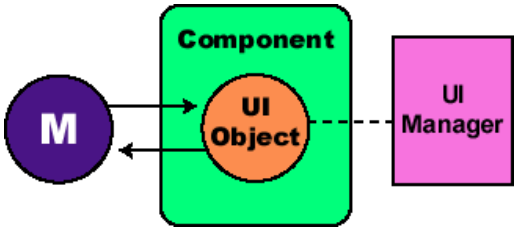

Figure 5.12: Separable Model Architecture in Swing Employs User Interface Delegate [9]

The Swing modification of the MVC pattern is called *Separable Model Architecture* and produces two main components, the model and the UI Delegate. The typical responsibilities of the UI are handled by a manager, while the controller is usually hidden by making it an inner class of the UI Delegate, which is further explained in [9]. Contrary to this, the objective of the current implementation is to maintain a high degree of visibility and scalability, thus adhering to a stricter definition of the MVC pattern. Therefore, the listeners are implemented as normal outer classes, which promotes maintainability through an organized packaging of the overall structure.

The visibility of each listener is also one reason why the number of source files of the software package may seem substantial, where at least 44 of the 120 classes are directly related to controlling the behavior of the GUI. A second reason for the source file count is the inclusion of interfaces, which are necessary to follow MVC principles. Continuing with the example of the JMeshPanel, which can again be applied to other components in the current implementation, the general MVC structure is shown in Figure 5.13. In the diagram, the implementing instances of the MeshUI may have any number of EventListener objects, each defined to react to a particular user-driven event and provide specific behavior. The UML diagram also introduces two interfaces, *Loggable* and *Renderable*, which most of the components implement.

Both the Loggable and Renderable interfaces are implemented by the facades, which delegate this responsibility to their corresponding models. The Loggable interface ensures that the facades provide the necessary functionality for the status and history panels, which serve as

Loggers. This means that whenever a method within the model is called, it fires notifications in the form of events that contain information, which the Logger can use to update its view. The same principles apply to the Renderable interface, which requires an implementing class to be able to return a Java 3D scene graph representing itself when called. The Renderable object fires notifications to a *Renderer*, in this case the visualization panel, which requests the scene graph and displays it.

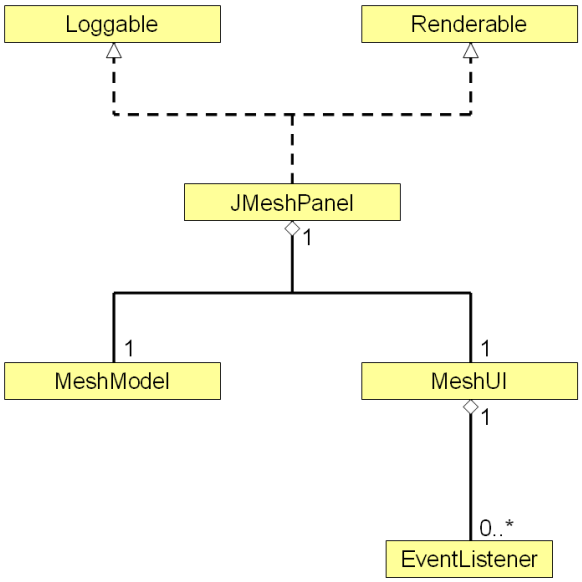

Figure 5.13: UML Class Diagram for the Adapted MVC Pattern

The concepts of the MVC pattern described in this section provide the mission for organizing the global design of the current implementation. This insight is crucial to understanding the necessary and significant amount of communication between the generated models, views, and controllers. A reiteration of these concepts as well as the specific details of each class and interface are fully documented in the Javadoc accompanying the software package.

5.2 Algorithms and Data Structure

The previous sections reference several schemes for storing and accessing data representing mathematical matrices. As depicted in the UML diagram in Figure 5.14, the matrices can be divided into three categories, which are rectangular, symmetric, and diagonal. As these matrices are part of the internal MVC model, an abstract class implements their fundamental features, which are defined by an interface. Each matrix inherits the attributes of this abstract class, allowing these subclasses to focus on the unique aspects of their storage schemes.

The objective of this section is to provide an overview of the design concepts utilized in forming the basic model structure, as influenced by all of the elements of the problem description. For the client, each class or interface described in this section is fully documented in the Javadoc accompanying the software package.

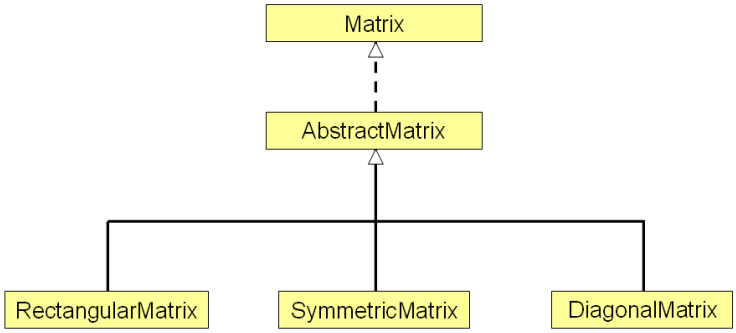

Figure 5.14: UML Class Diagram for the Mathematical Matrix Structure

The components of the diagram have the following general meaning:

Matrix	an interface that all matrices must implement
AbstractMatrix	an abstract class providing default behavior for matrices
RectangularMatrix	a standard storage scheme for a fully populated matrix
SymmetrixMatrix	a lower-triangular storage scheme for a symmetric matrix
DiagonalMatrix	a diagonal storage scheme for an orthogonal matrix

where the rectangular, symmetric, and diagonal matrices implement most of the methods and algorithms required to perform the DOF, MAC, and ORTHO correlations and connections. These classes are further subclassed into the convenience classes for the specific data types mentioned in previous sections.

5.2.1 Rectangular Matrices

An implementation of a 2D rectangular matrix typically involves initializing an array of arrays, where each element can be accessed by two indices. In Java, each index is checked against the bounds of the array, increasing security as well as access time. To reduce access time, the current implementation instantiates a single array, requiring only one index check per access, which is accomplished by adding each row to the end of the previous row. The elements of this array can be accessed by calculating the index corresponding to the given row and column with the relation:

$$index_{rc} = (index_{row} \cdot n_{column}) + index_{column} \tag{5.1}$$

where n_{column} represents the total number of columns in the matrix.

The utility of this rectangular matrix is subclassed four times for the following purposes:

ModeMatrix	a matrix to store the mode shapes of a dynamic model
NodeMatrix	a matrix to store the nodal coordinates of a dynamic model
ModeCorrelationMatrix	a matrix to store the MAC and ORTHO correlations
NodeCorrelationMatrix	a matrix to store the DOF correlations

The ModeMatrix and NodeMatrix convenience classes provide additional attributes and methods particular to their application. A visual interpretation of these classes is shown in Figure

5.15, where both can have an arbitrary number of nodes, while the ModeMatrix class can also contain an arbitrary number of mode shapes:

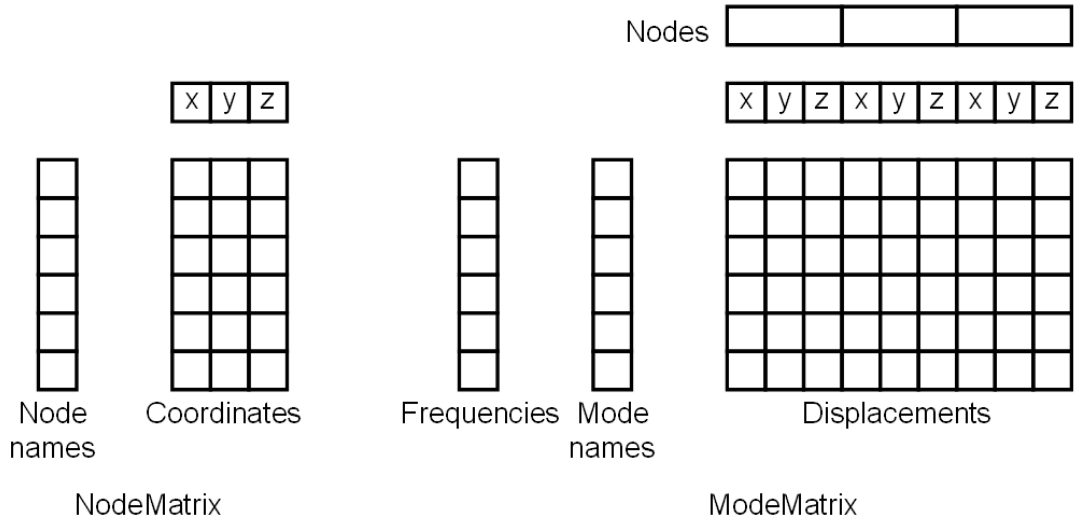

Figure 5.15: ModeMatrix and NodeMatrix Extensions of RectangularMatrix

Similarly, Figure 5.16 shows the ModeCorrelationMatrix and NodeCorrelationMatrix classes:

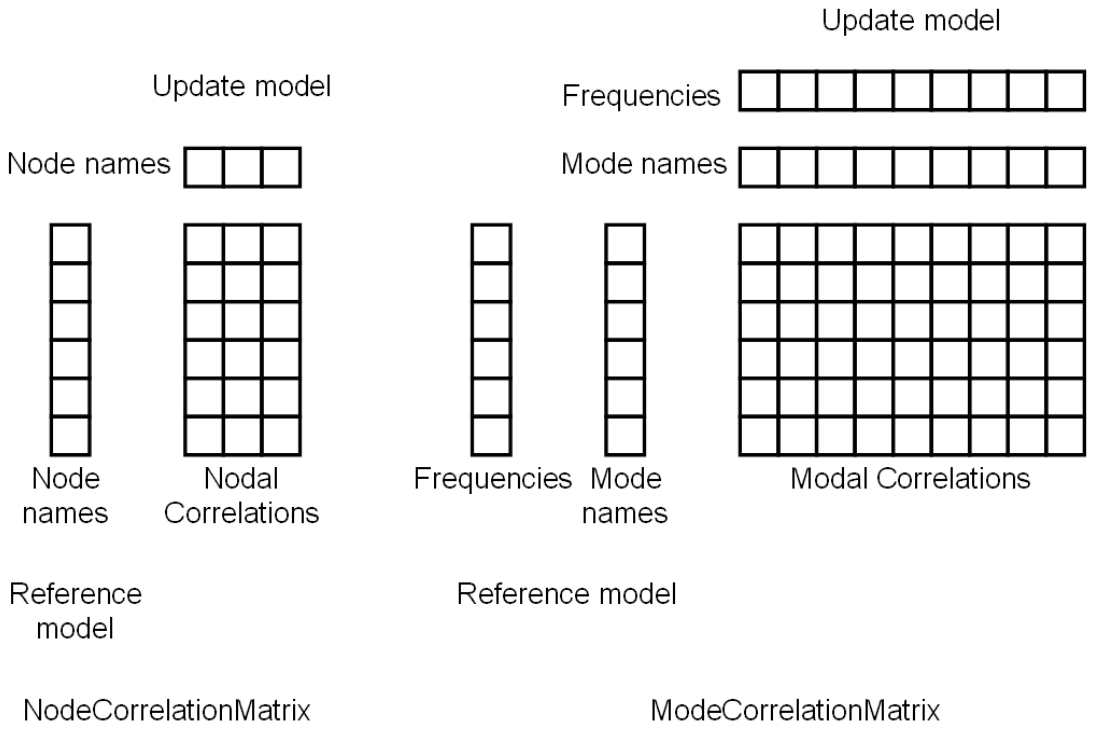

Figure 5.16: ModeCorrelationMatrix and NodeCorrelationMatrix Extensions of RectangularMatrix

where the rows and columns represent data from the reference model and update model, respectively. The cells of the NodeCorrelationMatrix contain the DOF correlation values of the corresponding nodes from the reference and update models. Analogously, the cells of the ModeCorrelationMatrix contain the values of the MAC and ORTHO correlations of the corresponding mode shapes. The computation of these correlation values is a direct implementation of the equations described in **Chapter 3**.

5.2.2 Connection Algorithm

The DOF, MAC, and ORTHO correlation matrices must each be connected at some point during the validation and update procedure. The DOF correlation must be sorted to produce a minimum connection, while the MAC and ORTHO correlations must be sorted to produce a maximum connection. The sorting procedure for each connection, however, is essentially the same and is illustrated in Figure 5.17:

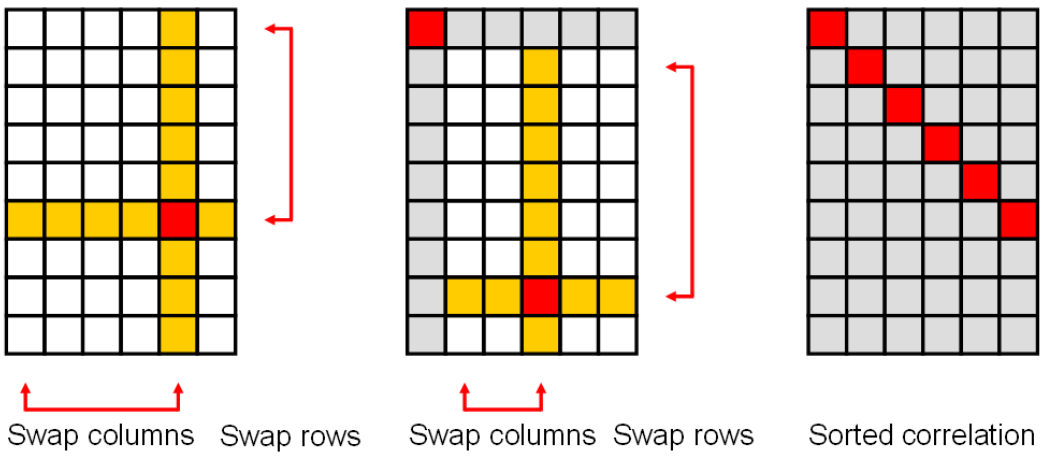

Figure 5.17: Connection Algorithm of a ModeCorrelationMatrix or a NodeCorrelationMatrix

where the optimal connection is formed along the diagonal of the correlation matrix. The resulting connection is a one-to-one mapping, whose values are sorted in ascending or descending order for a minimum or maximum connection, respectively. This robust algorithm assures that a many-to-one mapping is not possible, where a single mode or node of one model can be mapped to more than one mode or node of the other model. To provide this assurance, the order of the algorithm is increased, requiring significant memory storage and computing resources for large systems. In most applications specified by the client, however, the typical computation time is on the order of a few seconds. Outlining the connection algorithm:

1. Search the entire matrix for the minimum or maximum value.
2. Swap the corresponding row and column with the first row and column, which now represents a connection.
3. Search the remaining unconnected rows and columns for the minimum or maximum value.

4. Swap the corresponding row and column with the first remaining unconnected row and column.

5. Repeat steps 3 and 4 until the reference model or update model is completely connected.

The resulting diagonal values of the correlation matrix can then be extracted and stored more efficiently in a diagonal matrix.

5.2.3 Diagonal Matrices

A diagonal matrix represents an orthogonal matrix whose diagonal values are stored and whose off-diagonal values are taken as zero. The diagonal values are stored as a single primitive array where each connection can be incrementally accessed. This diagonal matrix is subclassed twice, once for each connection matrix:

ModeConnectionMatrix	a matrix to store the MAC and ORTHO connections
NodeConnectionMatrix	a matrix to store the DOF connections

As the connection values are sorted in ascending or descending order in most cases, the connection can be cropped according to a maximum or minimum tolerance for a minimum or maximum connection, respectively. In the current implementation, the technique to obtain this final form is called the *preferred connection* and specifically used in modifying the precision of the DOF connection. The two subclasses of the implemented diagonal matrix are visualized in Figure 5.18:

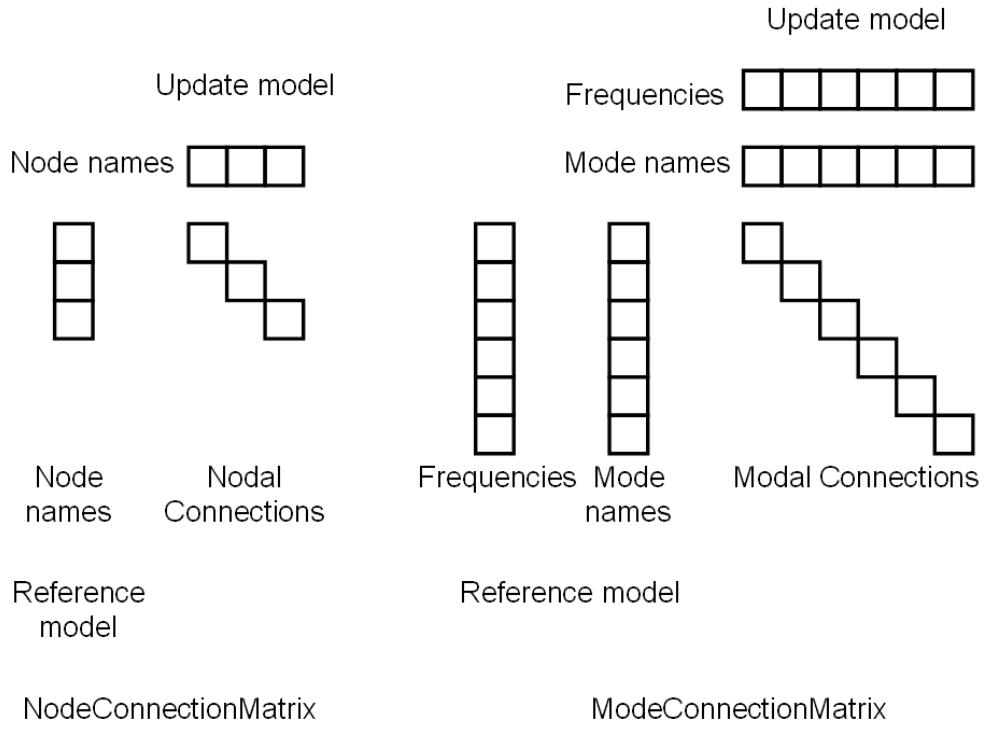

Figure 5.18: ModeConnectionMatrix and NodeConnectionMatrix Extensions of DiagonalMatrix

where the rows and columns represent data from the reference and update models, respectively. Naturally, the number of connected mode shapes and nodes must be the same for each model, even if the dimensions of the correlation matrix are not.

The NodeConnectionMatrix has an important second utility, which is the ability to store the geometric connectivity of a dynamic model, including information for bar, beam, triangular, and quadrilateral elements. Bars and beams can be called *edges* and are defined by two nodes. Triangular and quadrilateral elements can similarly be defined by three and four consecutive nodes, respectively. By using the same model for the rows and columns of the NodeConnectionMatrix, this connectivity can be effectively stored as seen in Figure 5.19:

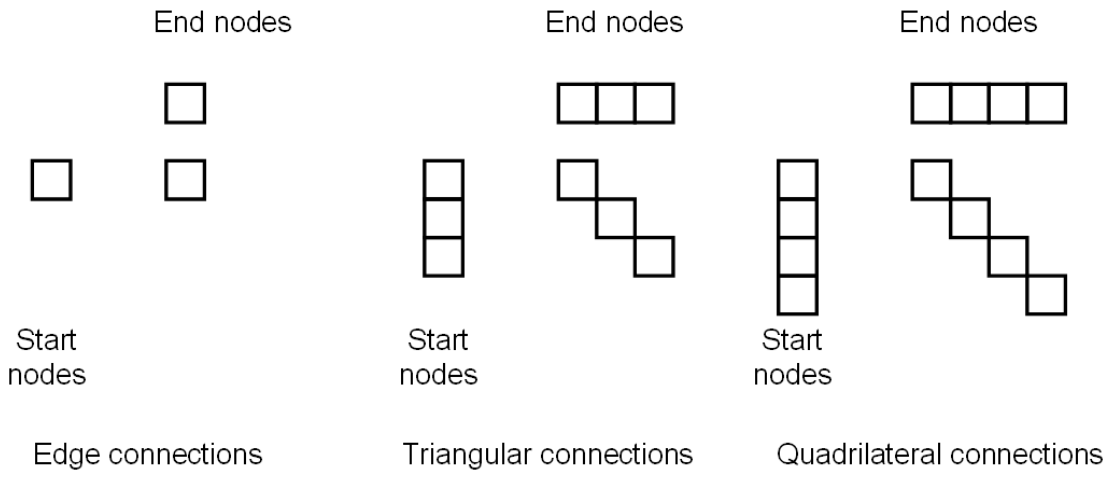

Figure 5.19: Edge, Triangle, and Quadrilateral Storage Using NodeConnectionMatrix

where the dimension of an arbitrary edge, triangle, and quadrilateral NodeConnectionMatrix is a multiple of one, three, and four, respectively. The diagonal connection values have no role in this geometric connection and can be taken as zero. An advantage of this scheme is that the corresponding coordinate data in a NodeMatrix can be sorted according to these node connections. By recalling that the current implementation of NodeMatrix stores coordinate data in a single array, these values can be exported directly to Java 3D for rendering. This is possible because Java 3D's LineArray, TriangleArray, and QuadArray classes are designed to receive and render coordinate data in this exact form. The reuse of classes for multiple applications such as this is a fundamental goal of OOP design.

5.2.4 Symmetric Matrices

The final type of implemented mathematical matrix is a lower-triangular matrix class, whose aim is to reduce the storage requirements of a symmetric matrix by approximately half. The mass and stiffness matrices that are used in most practical applications are symmetric and typically provide the largest quantity of data in comparison to the previously described matrices. Therefore, it is imperative to take full advantage of their symmetry, especially considering the impact to element access time during matrix multiplication in the ORTHO correlation.

The implemented symmetric matrix is subclassed as a MassMatrix, which can be used for both mass and stiffness matrices and whose structure is depicted in Figure 5.20:

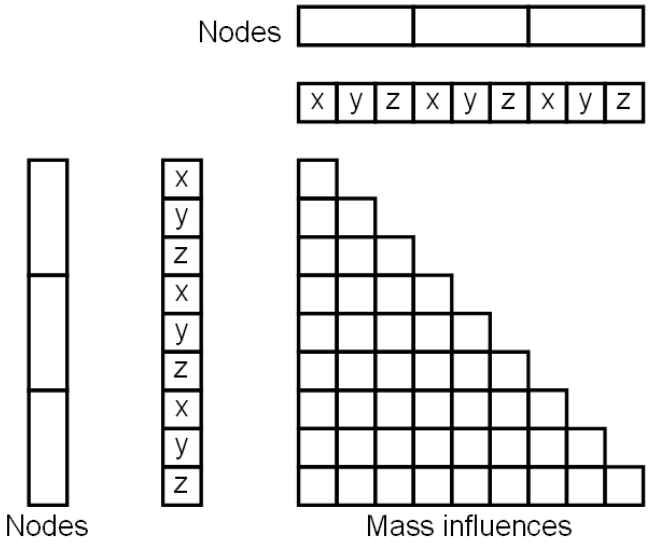

Figure 5.20: MassMatrix Extension of SymmetricMatrix

The storage reduction is achieved by implementing an array of arrays, where the length of each row is incremented to produce the lower-triangular portion of the matrix. The values of the upper-triangular portion of the matrix can be accessed by transposing the indices of the given row and column.

An important note regarding the ORTHO correlation is that the DOF of the mass or stiffness matrices must be mapped to those of the mode shapes of the reference model. As symmetric matrices have an inherent sorting, the DOF ordering of the mass matrix is used to sort the mode shapes of the reference model. The DOF connection must consequently be used to sort the mode shapes of the update model. Therefore, it is paramount that the DOF ordering of each matrix is correct, otherwise the computation may produce unexpected results, which may be difficult to notice since it will neither produce compile-time nor runtime errors.

5.3 Java 3D Visualization

In addition to the familiar Look and Feel of a two-dimensional Java Swing program, the use of the Java 3D library allows some of the model data to be represented in three-dimensions during runtime. A three-dimensional visualization can provide the user with added insight into the given problem and may indicate possible sources of error during the process, which may not have otherwise been possible. The theoretical background employed by Java 3D encompasses a range of topics within the field of *Computational Visualization*, which is out of the scope of this thesis. Knowledge of particular aspects of Computational Visualization such as scene graph theory, however, are essential to successfully utilize the Java 3D library. The

reader should be familiar with the principles discussed in [5] pertaining to Java 3D. While this section does not go into significant detail regarding the rendering process, information particular to the implementation of the thesis is highlighted.

The procedure to integrate Java 3D into the Swing GUI is similar to that of the UI installation process. In Swing, subcomponents are installed into containers, while in Java 3D, the installation process can be done on a Canvas3D. In the current implementation, a DrawingCanvas3D extends a Canvas3D and acts as a 3D object that can be added to a Swing component. The installation of a Java 3D scene includes registering listeners, creating the view branch graph and content branch graph, and defining behaviors. The listeners have a similar purpose as in Swing, and mainly provide the functionality to add or remove a scene from the DrawingCanvas3D. The view branch graph, content branch graph, and behaviors are discussed further in [5]. In this way, the 2D and 3D visual components of the application are merged.

Examples of the three main types of visualizations utilized in the current implementation have already been presented in Figures 1.3, 3.6, and 3.3 for the geometric coordinate data of an automotive structure and the 2D and 3D representations of a 2D matrix, respectively. The majority of the objects in each of these scenes is generated by using subclasses of the Java 3D GeometryArray class in order to optimize the efficiency of the renderer. These subclasses used include PointArray, LineArray, TriangleArray, QuadArray, and IndexedQuadArray. Attributes such as PointAttributes, LineAttributes, PolygonAttributes, and ColorAttributes are then assigned to the various aspects of each GeometryArray to produce the final scene.

For the automotive geometry, which may contain a large quantity of points, lines, triangles, and quadrilaterals, the polygons are rendered as wireframe lines rather than as filled solids. One reason for this is that on some workstations, the graphics card may not be fast enough to smoothly render the scene in real time. Further, by rendering the polygons as solids, lighting effects would need to be integrated into the scene to provide depth perception for the user. While this feature may produce a convenient image, it would require more effort from the renderer, further degrading runtime performance without providing significant benefits. This aspect of the program has been given much attention because the ability of the workstation to perform the rendering process is, in most cases, the most time-consuming component during program execution.

5.4 File Types

After the GUI framework has been prepared through a MVC architecture and the internal data can be visualized with Java 3D, focus can shift to importing and exporting data into and out of the storage scheme. For each input file type, a parser or *reader* is designed to retrieve the data from a file and store it in the internal database of the model. Similarly, for each output file type, a *writer* is created to format and print the internal data to a file. As the importing and exporting of data usually does not need to persist during runtime, the classes representing the readers and writers do not need to be maintained as attributes of the model, but rather can be instantiated within the appropriate method calls.

To further promote the scalability of the program software, the functionality of these input readers and output writers are declared as interfaces so that additional file types can be later supported. This pattern is similar to the previously presented UML structures and its advantage can be seen in the case of the DefaultModeModel as seen in Figure 5.21:

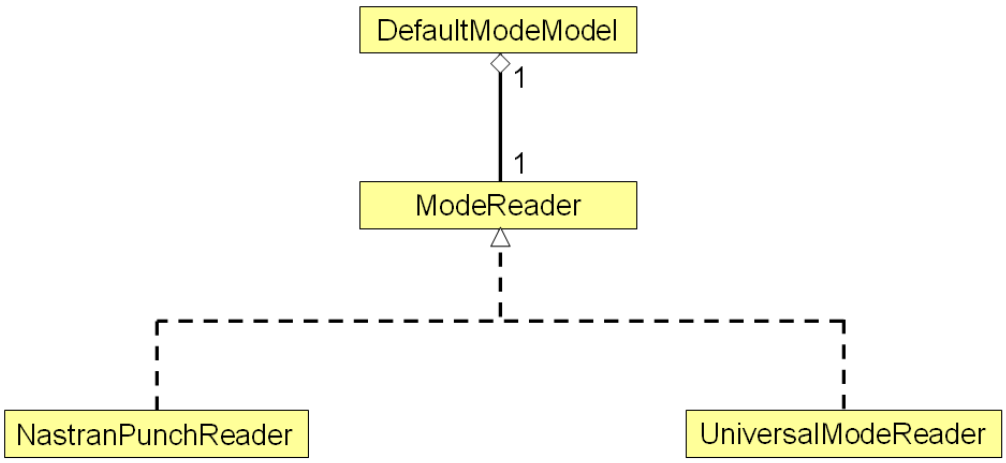

Figure 5.21: UML Class Diagram for the Input Stream Reader of Eigenfrequencies and Mode Shapes

where the eigenfrequencies and mode shapes must be read from either a MSC.Nastran or Universal file format. When the method to import the data is called, the model can determine the file type and instantiate the appropriate reader at runtime. This example is analogous for the other readers and writers and can be readily extended to accommodate further file types.

The client has expressed that the raw data must be handled in MSC.Nastran and Universal file formats, while the correlation and connection data should generally be delivered in ASCII or Excel format. The MSC.Nastran and Universal sources more completely document their use and can be found in [12] and [10], respectively. Guided by these sources, the implemented readers and writers are designed to handle the particular properties of each necessary command. As these properties can be quite specific, this section provides only a brief overview of each file type. Detailed information regarding each reader and writer can be found in the Javadoc accompanying the software package.

5.4.1 Model Geometries

The geometry of dynamic models is supported in MSC.Nastran *bulk data* format, whose file extension may either be *.BDF, *.DAT, or *.NAS. The bulk data contains information for the nodal coordinates as well as the bar, beam, triangular, and quadrilateral elements of the dynamic model. The corresponding MSC.Nastran cards are GRID, CBAR, CBEAM, CTRIA3, and CQUAD4, which themselves may come in different formats called *free*, *large* and *small*. The NastranBulkDataReader class thus supports each of these cards in each format and stores the nodal coordinates in instances of NodeMatrix, while storing the connectivity information in instances of NodeConnectionMatrix.

5.4.2 Nodal Subsets

Subsets or *groups* of the nodes of a dynamic model can be stored in MSC.Nastran *set* format, whose file extension is generally *.SET or *.NSET. Using the MSC.Nastran SET card, lists of nodes of the entire model can be stored and a particular node group can be selected to reduce the number of nodes correlated in the DOF correlation. The NastranSetReader class handles this file type and stores this information in instances of KeyMap.

5.4.3 Eigenfrequencies and Mode Shapes

As previously stated, the eigenfrequencies and mode shapes of a dynamic model can be imported from either MSC.Nastran or Universal format. MSC.Nastran usually provides this data in *punch* format, whose file extension is typically *.PCH. Universal *data set 55* provides the same information and bears a *.UNV file extension. The NastranPunchReader and UniversalModeReader classes parse the MSC.Nastran and Universal files, respectively, and store the eigenfrequencies and mode shapes in instances of ModeMatrix.

5.4.4 Mass and Stiffness Matrices

As the mass and stiffness matrices are typically provided by FE models, they are supported only in MSC.Nastran punch format for *direct matrix input*, whose file extension is *.PCH. These files generally store the mass matrix, stiffness matrix, and load vector with the MSC.Nastran DMIG card. Therefore, the NastranDirectMatrixReader class that handles these files stores the mass and stiffness matrices in instances of MassMatrix, while filtering out the load vector. Then either the mass or stiffness matrix can be selected for computation in the ORTHO correlation. It is important to note that the mode shapes and mass matrices have the same file extension, though they represent different data. Considering this, if the user attempts to load a punch file containing the incorrect type of data, generally no data will be imported.

5.4.5 Nodal Connections

The nodal connections, represented as a textual ASCII format, are the only data that can be both imported and exported. The file type has an extension of *.TXT and contains a list of all nodal connections between the reference model and update model. Each connection simply states a nodal identifier from the reference model, the identifier of the connected node from the update model, and the distance between these two nodes. The ASCIIMeshConnectionReader class imports this data to instances of NodeConnectionMatrix. The user can then use this data as the DOF connection or recompute it, which can be exported back to a file using the ASCIIMeshConnectionWriter class.

5.4.6 Modal Correlations

The MAC and ORTHO correlations produce the same kind of matrix data and can thus be exported using the same writer. In order to usefully obtain the numerical data, the information is exported to a semicolon-separated Excel format, which is convenient for the client. The file

extension, therefore, is *.CSV and the ExcelModeCorrelationWriter class that handles these files exports the tabular data stored in instances of ModeCorrelationMatrix.

5.4.7 Optimization Decks

The generation of the MSC.Nastran MAC and ORTHO optimization decks is the primary objective of the client, which have file extensions of *.NAS. The MAC and ORTHO optimization decks are individually handled by the ModalAssuranceCriterionUpdateWriter and OrthogonalityCheckUpdateWriter classes, respectively. The resulting output streams create the most complex file types of the entire implementation since they utilize several MSC.Nastran cards such as DEQATN, DRESP1, DRESP2, and DTABLE and can consist of tens of thousands of lines, even for relatively small systems. These MSC.Nastran cards define equations, expressions, and variables, which provide the tools to implement the MAC and ORTHO objective functions described in **Chapter 4**.

One major reason for the file size is that every mathematical operation of the MAC and normalized ORTHO correlations described in **Chapter 3** must be explicitly output to the optimization deck. As a result of this task, the 30,000 non-whitespace character limit of the DEQATN card for defining equations is typically exceeded when exporting the ORTHO correlation, even for a DOF connection of twenty nodes. Therefore, the equation describing the ORTHO correlation must be subdivided into shorter DEQATN equations and aggregated to accommodate this restriction.

The complexity of the optimization decks stems from the need to incorporate data taken from all aspects of the internal database, including:

- the nodes of the DOF connection
- the eigenfrequencies and mode shapes of the reference and update models
- the mass or stiffness influences of the reference model
- the MAC or ORTHO modal connections
- the weight factors of the connected mode shapes and their eigenfrequencies

The resulting MAC and ORTHO optimization decks can be combined with the client's existing files for defining design parameters to successfully execute a MSC.Nastran SOL 200 design optimization trial.

Chapter 6

User's Manual

In order to run the software package that has been distributed to the client, the target workstation should have the latest versions of the Java Runtime Environment (JRE), Java Development Kit (JDK), and Java 3D installed on it. If these are not correctly installed, the executable JAR file of the current distribution may not properly function. The program can be started by double-clicking on the JAR file or by accessing it through a shell or the console. When the program successfully initializes itself after a few seconds, the GUI resembling Figure 5.1 appears on the screen.

The objective of this tutorial is to visually guide the reader through the basic steps for generating the MAC and ORTHO MSC.Nastran optimization decks through the command panel. The history, status, and visualization panels are not explicitly shown throughout the tutorial, but are discussed as necessary. The procedure presented in this section is appropriate for creating the examples shown in **Chapter 7**, which focus on presenting the tabular and visual results from a model update.

6.1 Peripheral Panels

The history panel at the bottom of the GUI provides the user with a standard capability of many engineering software programs. Any significant actions that the user makes can be tracked and referred to in the scrollable history panel. This allows the user to verify which files have been imported or exported and provides approximate execution times for each action. Additionally, any computations of the correlations or connections are logged in this panel. Finally, brief warnings or errors are logged when invalid settings prevent the software from performing a requested action.

Two possible errors of note, however, are not specifically logged in the history panel. The first comes from a Java heap size exception which can occur if the memory requirements of the geometry or dynamic mode shapes of a model exceeds the maximum heap size of the Java Virtual Machine (JVM). If this occurs, the software may appear to do nothing. This may be solved by closing the application and running a script in a shell or from the console that increases the maximum heap size and executes the JAR file. A second unreported error may occur if a given file is partially corrupt or does not strictly adhere to its file format. As an

unforeseeable amount of eventualities may produce this kind of error, the responding parser may load all, some, or none of the data. Both the Java heap space exception and file format exceptions, however, should not produce a fatal error and the program may be able to recover by importing data from different files.

The status panel on the left-hand side of the GUI displays a textual view of the data currently stored in the database. The content labels provide an intuitive and quick glimpse of the size of each relevant data from two dynamic models. For each model, this includes the geometric data such as the amount of nodes, edges, triangles, and quadrilaterals. The number of mode shapes and nodes of the imported dynamic solutions is also displayed. The mass matrix is provided for the upper model, which is taken to be the reference during computation of the ORTHO correlation. With two imported models, the size of the DOF connection and dimensions of the MAC and ORTHO correlation matrices are finally displayed.

The visualization panel on the right-hand side of the GUI renders available data for each panel with the exception of the mode shapes of each model. The background of the visualization can be toggled between black and white, which automatically flips the axis or text colors. The 2D and 3D scenes can be manipulated with the mouse buttons similar to other 3D applications. Holding down the left-click button allows the user to rotate the objects, while holding down the right-click button provides a panning capability. Further, scrolling with the wheel zooms the objects in and out.

6.2 Importing Geometry

The first step that should be performed is importing the geometry of both the reference model and the update model. The procedure is the same for each model and the corresponding panel is found in the *Grids* tab as seen in Figure 6.1:

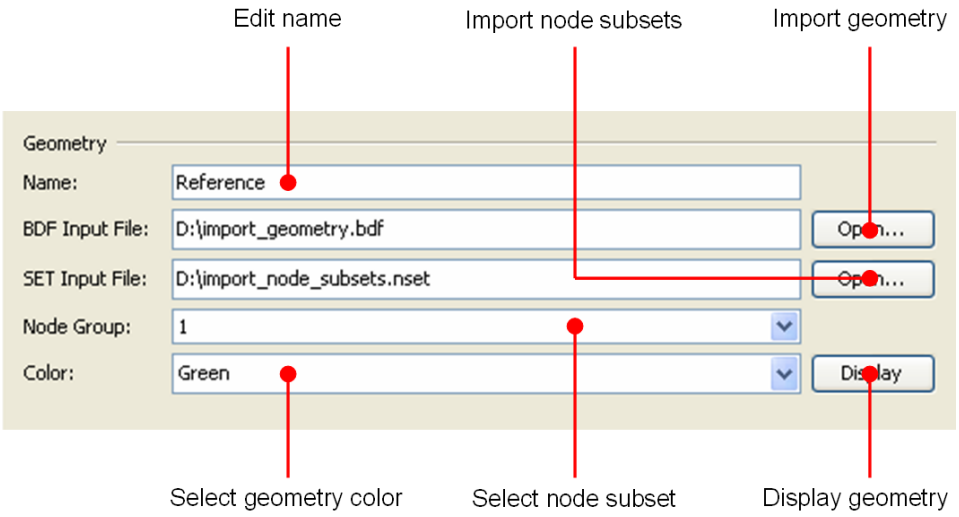

Figure 6.1: Importing Reference Model or Update Model Geometry

1. Edit the model name and press ENTER.
2. Load the geometry file by searching for it or entering its path.
3. Load the node group file by searching for it or entering its path.
4. Select the node group to be connected from the drop-down menu.
5. Select the color of the geometry from the drop-down menu.

Importing a node group file is not obligatory since a default group of all nodes in the geometry file can be selected for each model. Upon completing these steps, the user can verify that the data has correctly been loaded by checking the history and status panels. The geometry should be loaded for both the reference model and the update model, keeping in mind that the reference model will be assigned the mass matrix for a subsequent ORTHO correlation. The model is automatically rendered when its geometry is imported, however, the user can switch the visualization back to this view by clicking the Display button.

6.3 Connecting the Degrees of Freedom

Once the reference model and the update model have been imported, a DOF connection can be made between their nodes. The panel to do this is found at the bottom of the *Grids* tab and shown in Figure 6.2:

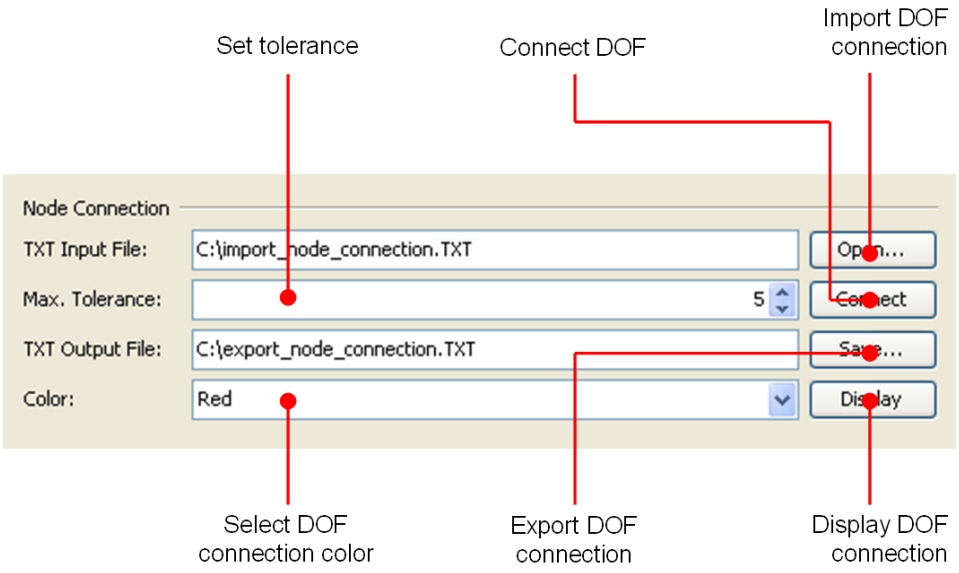

Figure 6.2: Connecting Reference Model and Update Model Degrees Of Freedom

1. Set the maximum node connection tolerance.
2. Compute the DOF connection by clicking the Connect button.

3. Select the color of the connected nodes from the drop-down menu.

Alternatively, a previously saved DOF connection can be imported by searching for it or entering its path. The size of the resulting DOF connection can be verified by checking the history and status panels. If the number of connected nodes is insufficient, the DOF connection can be refined by recomputing it with a different tolerance. A satisfactory connection can then be exported for later use. Like the geometry for a single model, the visualization of the connected nodes can be refreshed by clicking the Display button.

6.4 Importing Eigenfrequencies and Mode Shapes

The eigenfrequencies and mode shapes of both the reference model and update model can be imported from the *Modes* tab displayed in Figure 6.3:

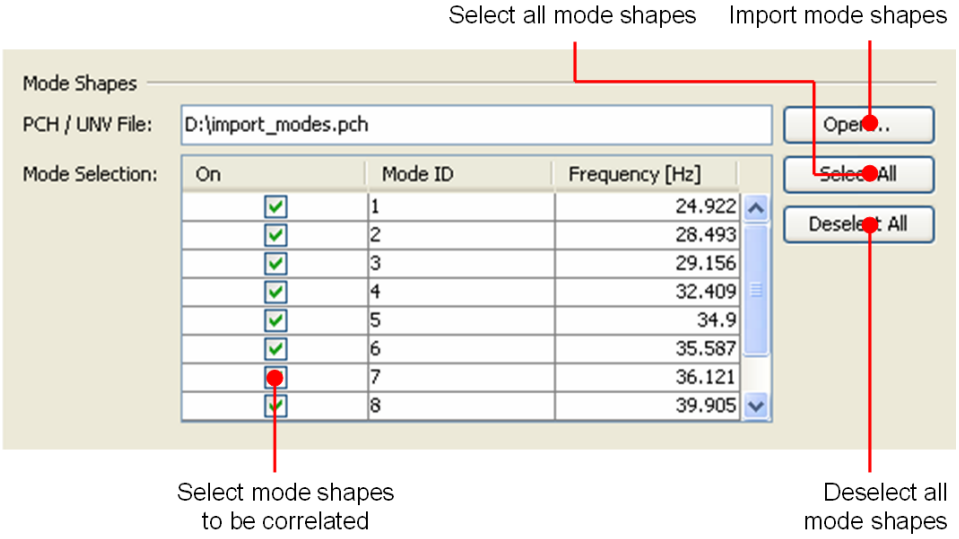

Figure 6.3: Importing Reference Model and Update Model Eigenfrequencies and Mode Shapes

1. Load the mode shape file by searching for it or entering its path.
2. Select the mode shapes to be correlated with the check boxes.

It is important to remember that the placement of the panels in the *Modes* tab is analogous to that of the *Grids* tab, where the upper and lower panels are reserved for the reference model and update model, respectively. Convenient buttons are also available to select or deselect all mode shapes for correlation. The number of nodes and mode shapes of the dynamic solution can be verified in the history and status panels.

6.5 Computing the Modal Assurance Criterion

After the DOF connection has been set and the mode shapes of the reference model and update model have been imported, the MAC correlation can be directly performed. Figure 6.4 depicts

the upper panel in the *Modal Assurance Criterion* tab for performing this operation, where only those mode shapes selected in the *Modes* tab will be correlated:

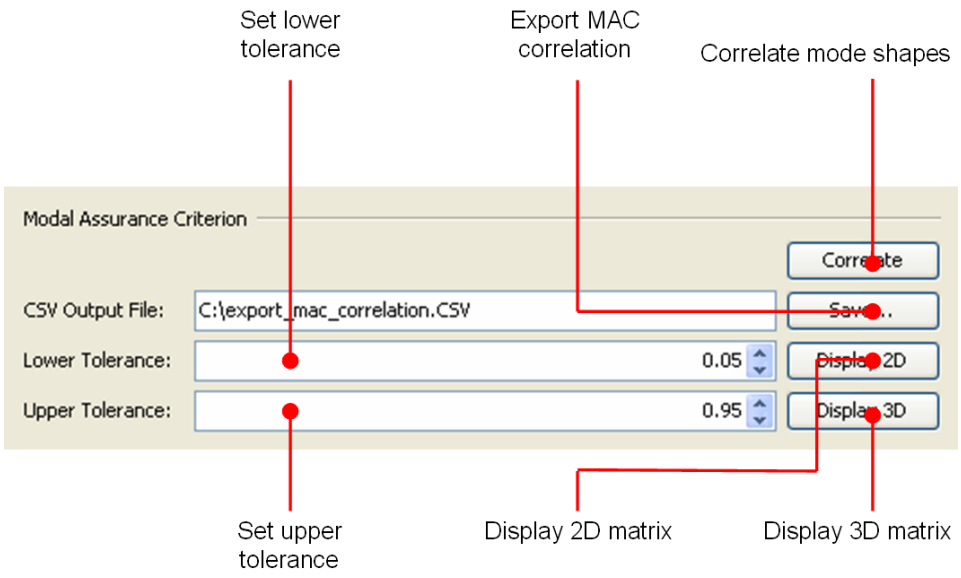

Figure 6.4: Computing the Modal Assurance Criterion of Two Dynamic Models

1. Compute the MAC correlation by clicking the Correlate button.
2. Set the lower MAC correlation tolerance to customize the visualization.
3. Set the upper MAC correlation tolerance to customize the visualization.

The resulting correlation matrix can be exported in tabular form to review the numerical results. A 3D visual representation of the matrix is also automatically displayed in the visualization panel. The user can switch between a 2D and 3D rendering of the correlation matrix by clicking either the Display 2D or Display 3D buttons, respectively.

6.6 Computing the Orthogonality Check

The procedure to generate the ORTHO correlation requires all of the same steps and considerations as the MAC correlation, but must additionally import data for the mass matrices. The components to accomplish this are reflected in Figure 6.5, which displays the upper panel in the *Orthogonality Check* tab. The modified correlation procedure includes two extra steps:

1. Load the mass matrix file by searching for it or entering its path.
2. Select the mass matrix to be assigned to the reference model.
3. Compute the ORTHO correlation by clicking the Correlate button.
4. Set the lower ORTHO correlation tolerance to customize the visualization.

5. Set the upper ORTHO correlation tolerance to customize the visualization.

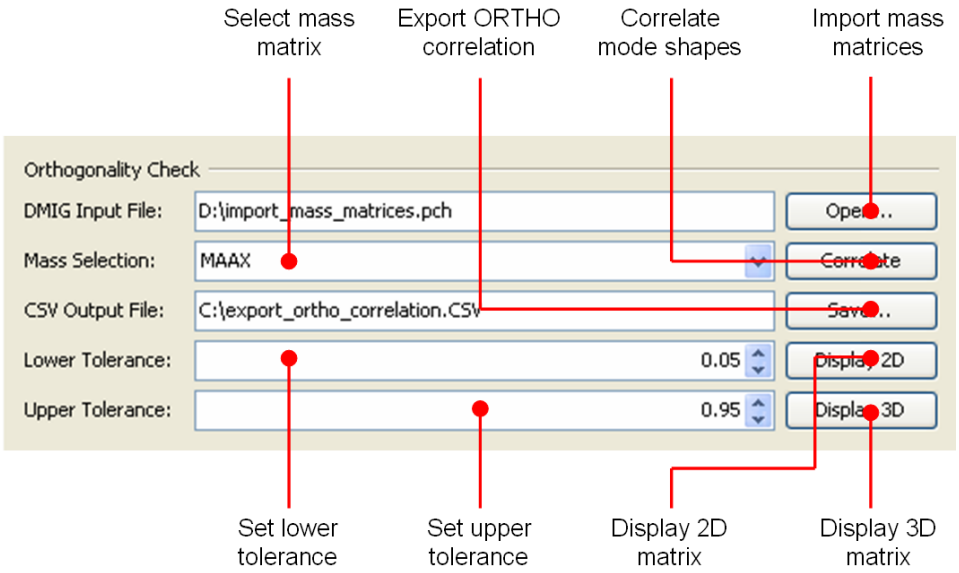

Figure 6.5: Computing the Orthogonality Check of Two Dynamic Models

It is crucial to remember that the selected mass matrix is assigned to the reference model. When an ORTHO correlation is attempted, a check is performed to ensure that the connected DOF of the reference model are contained in the selected mass matrix. If the check is valid, the ORTHO correlation is computed and the subsequent tasks are analogous to those of the MAC correlation.

6.7 Generating an Optimization Deck

After the MAC or ORTHO correlation has successfully been computed, the final step is to generate the optimization deck. The panel shown in Figure 6.6 provides the necessary tools to complete this task and can be found at the bottom of both the *Modal Assurance Criterion* and *Orthogonality Check* tabs to create their respective optimization decks. The considerations that must be handled include the following:

1. Select the connected mode shapes of the reference model from the check boxes.
2. Edit the modal weight factor of each connected mode shape in the third column.
3. Edit the frequency weight factor of each connected mode shape in the fifth column.
4. Select the mode shapes of the update model to be connected with the corresponding mode shapes of the reference model in the sixth column from the drop-down box.
5. Export the optimization deck with the current settings.

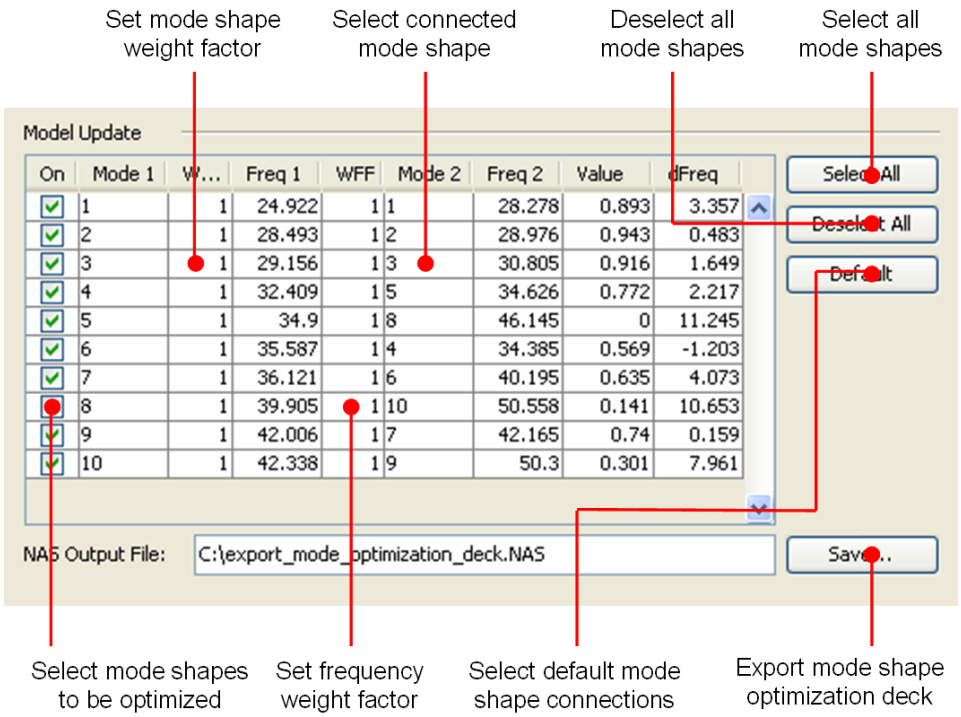

Figure 6.6: Generating an Optimization Deck from Two Dynamic Models

The table is generated immediately after the corresponding MAC or ORTHO correlation has been computed. The dynamic mode shape connection is likewise computed and displays the numerically optimal connection based on this correlation. To provide the user with more information, the corresponding modal and FREQ correlations for each pair of mode shapes is displayed in the last two columns. As the connected mode shapes can be revised, clicking the Default button resets the connection to its original state.

There are also convenience buttons to select and deselect all reference mode shapes for connection in the optimization deck. This may not be necessary in all cases, especially where the number of mode shapes of the reference model differs from that of the update model. It is also possible to connect the same mode shape from the update model to more than one mode shape from the reference model. Therefore, the user must be careful when manually connecting the mode shapes. When the desired settings have been chosen, the optimization deck can finally be exported by creating the file or by entering its path.

The resulting optimization deck can be combined with the client's design parameters to optimize the dynamic structure. After the optimization run is complete, the optimized update model may include a new geometry file and a new dynamic solution for its eigenfrequencies and mode shapes. To view the new correlation, the steps of this tutorial can be repeated using the reference model and the optimized update model. The results of the improved correlations can therefore be reviewed with the visualization and tabular tools or a further optimization deck can be generated if the correlations are still unsatisfactory. In this way, an optimization run can be efficiently setup, verified, and repeated if necessary.

Chapter 7

Numerical Examples

After grasping the theoretical concepts and methods presented throughout this thesis, the GUI procedure for model validation and update described in **Chapter 6** can be used to demonstrate the capabilities of the implemented software package. The following examples provide numerical and visual results from sample data that were created using this procedure. While the raw data for the dynamic models was supplied by the client, the preparation of the optimization deck, the intermediate and final correlation computations, and the visualizations were produced by the software package unless otherwise noted.

To perform a structural design optimization in MSC.Nastran, the client correlated a reference model with an update model to generate the appropriate optimization decks using the current implementation. These optimization decks were then combined with design parameters created from the client's pre-existing software to perform the design optimization trial. The resulting optimized update model was then correlated with the reference model using the software package to determine the level of improvement over the previous update model.

The practical examples presented in this chapter are meant to familiarize the reader with typical numerical results and to gauge the types of dynamic improvements that are possible through an optimized model update. The examples demonstrate that the update models may improve significantly through the optimization process, even after manual updating techniques have been exhausted. Both MAC and ORTHO optimization trials are performed in the following sections to contrast their results.

7.1 A Simple Beam

Considering a variation of a simple pre-test beam example described in [4], the beam shown in Figure 7.1 consists of 11 nodes and ten beam elements and was rendered by the client in ANSA. A mass with a value of $0.01t$ is located at each node of this simple beam, which represents the reference model. The eigenfrequencies and mode shapes computed by MSC.Nastran thus provide a reference dynamic response. Similarly, the mass and stiffness matrices of the system provide a reference for correlations with update models. The objective of this example is to perform the MAC correlation and optimization of the dynamic responses of this reference model and a perturbed or unevenly-distributed mass system.

Figure 7.1: Reference Model of a Simple Beam with an Even Mass Distribution

The update model shown in Figure 7.2, also rendered by the client in ANSA, is a beam with an uneven mass distribution, where the mass at the first five nodes is $0.01t$ and the mass at the last six nodes is $0.05t$. By a simple visual examination of the two models, it can be inferred that each will produce a different dynamic behavior. To quantitatively measure the dynamic correlation of the first ten mode shapes and eigenfrequencies of the reference and update models, a visualization of the computed MAC correlation matrix is displayed in Figure 7.3.

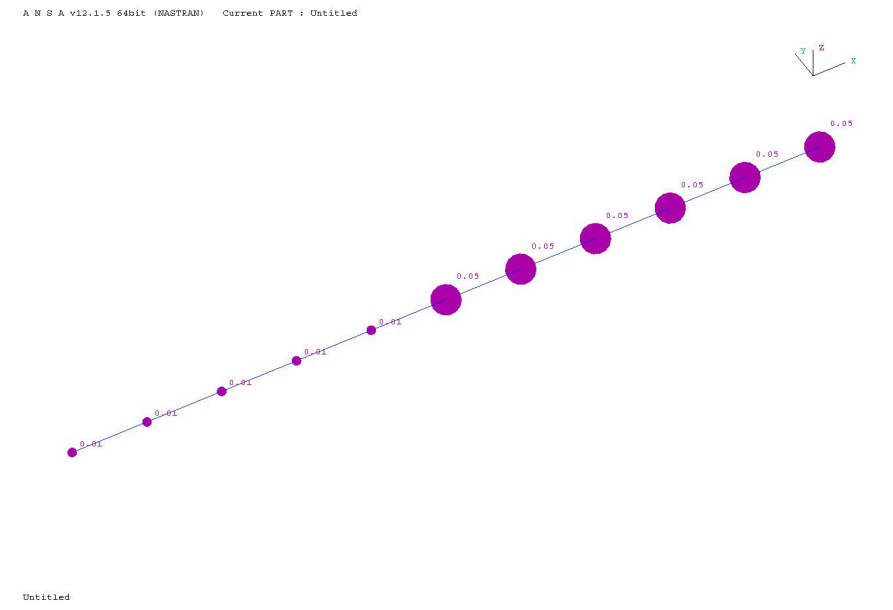

Figure 7.2: Update Model of a Simple Beam with an Uneven Mass Distribution

A glance at the 3D visualization gives the impression that the reference and update models are indeed weakly correlated. The cells rendered in green indicate MAC values less than 0.05, while those rendered in yellow indicate values between 0.05 and 0.95. These color tolerances illustrate standard confidence intervals employed by the client, where a MAC value greater than 0.95 signifies that two mode shapes are strongly correlated.

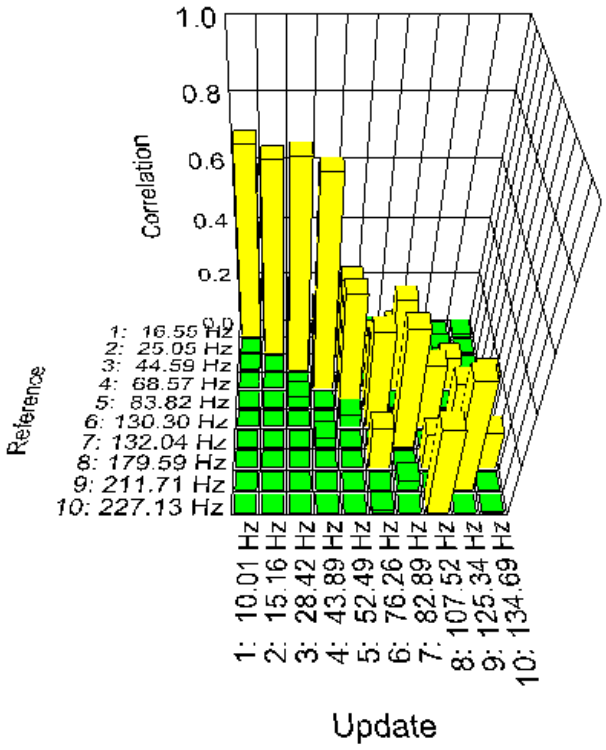

Figure 7.3: Visualization of Modal Assurance Criterion of Beam Models Before Optimization

The MAC correlation data is also represented in tabular form in Table 7.1, where the rows and columns represent mode shapes from the reference and update models, respectively:

Mode	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	0.688	0.000	0.186	0.000	0.118	0.025	0.000	0.001	0.000	0.010
2	0.000	0.689	0.000	0.187	0.000	0.000	0.115	0.000	0.018	0.000
3	0.003	0.000	0.750	0.000	0.353	0.147	0.000	0.011	0.000	0.000
4	0.000	0.003	0.000	0.755	0.000	0.000	0.348	0.000	0.126	0.000
5	0.000	0.000	0.044	0.000	0.436	0.262	0.000	0.121	0.000	0.068
6	0.000	0.000	0.001	0.000	0.048	0.370	0.000	0.247	0.000	0.215
7	0.000	0.000	0.000	0.039	0.000	0.000	0.453	0.000	0.258	0.000
8	0.000	0.000	0.000	0.000	0.002	0.163	0.000	0.138	0.000	0.147
9	0.000	0.000	0.000	0.001	0.000	0.000	0.040	0.000	0.428	0.000
10	0.000	0.000	0.000	0.000	0.000	0.010	0.000	0.335	0.000	0.001

Table 7.1: Modal Assurance Criterion of Beam Models Before Optimization

The next task is to generate the MAC optimization deck since the dynamic behaviors of the update model and reference model are not strongly correlated. To create the MAC optimization deck, the optimal modal connection described in **Chapter 3** must be performed, whose result is displayed in Table 7.2:

Reference Mode	Update Mode	MAC Value	Frequency Difference
1	**1**	0.688	-6.543
2	**2**	0.689	-9.894
3	**3**	0.750	-16.172
4	**4**	0.755	-24.684
5	**5**	0.436	-31.331
6	**6**	0.370	-54.037
7	**7**	0.453	-49.152
8	**10**	0.147	-44.896
9	**9**	0.428	-86.372
10	**8**	0.335	-119.612

Table 7.2: Mode Shape Connection of Beam Models Before Optimization

Using the mass magnitudes of the update model as design variables, a MAC optimization trial can be performed to improve the dynamic correlation of the two models. The natural solution for a perfect MAC correlation would be to replace the mass magnitudes of the update model with those of the reference model. In this case, that would mean setting those values to $0.01t$. Figure 7.2 displays the results of the optimized update model rendered by the client in ANSA, which shows the attempt of the sensitivity analysis to evenly redistribute the mass of the system:

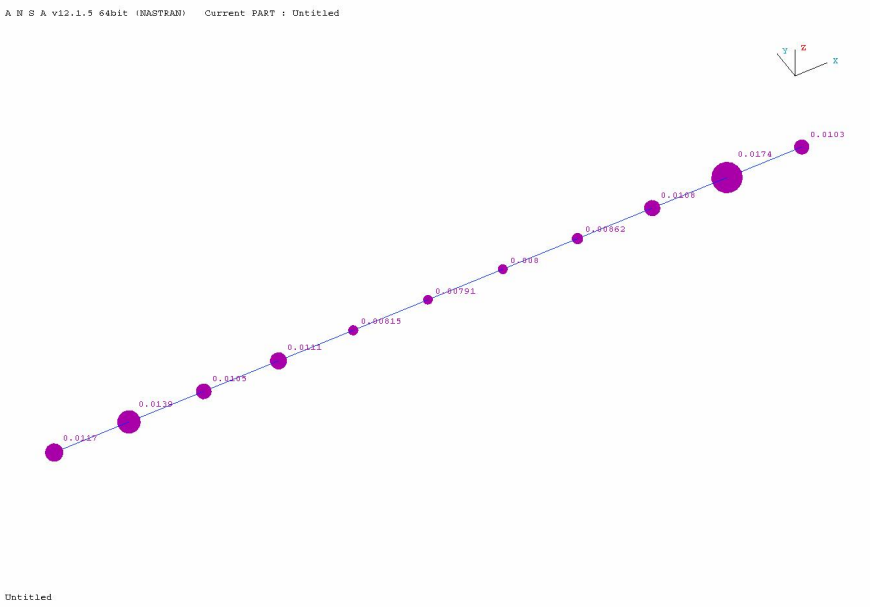

Figure 7.4: Optimized Update Model of the Simple Beam with an Uneven Mass Distribution

This mass redistribution of the update model is evident in Figure 7.5, which shows the convergence history of the sensitivity analysis performed by the client in MSC.Nastran:

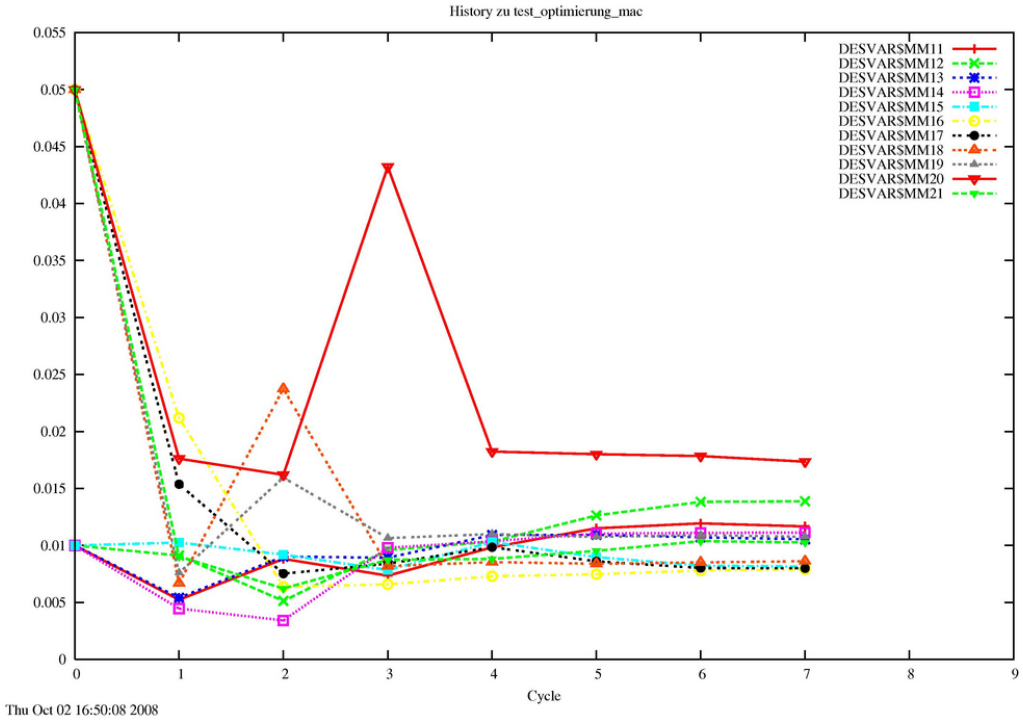

Figure 7.5: Convergence History of Mass Magnitudes of Update Model of the Simple Beam

The improvement of the optimized update model can be measured by correlating it with the reference model and computing the modal connection as seen in Table 7.3:

Reference Mode	Update Mode	MAC Value	Frequency Difference
1	**1**	0.976	-0.076
2	**2**	0.976	-0.115
3	**3**	0.995	-0.159
4	**4**	0.995	-0.260
5	**5**	0.979	-0.365
6	**6**	0.962	-3.953
7	**7**	0.979	-0.647
8	**8**	0.927	-9.951
9	**9**	0.966	-6.359
10	**10**	0.000	-20.755

Table 7.3: Mode Shape Connection of Beam Models After Optimization

A comparison of Tables 7.2 and 7.3 shows a significant improvement in the correlations of the connected eigenfrequencies and mode shapes between the reference and update models, which is the primary concern of the client. Figure 7.6 shows that most of the connected mode shapes meet the client's confidence requirements:

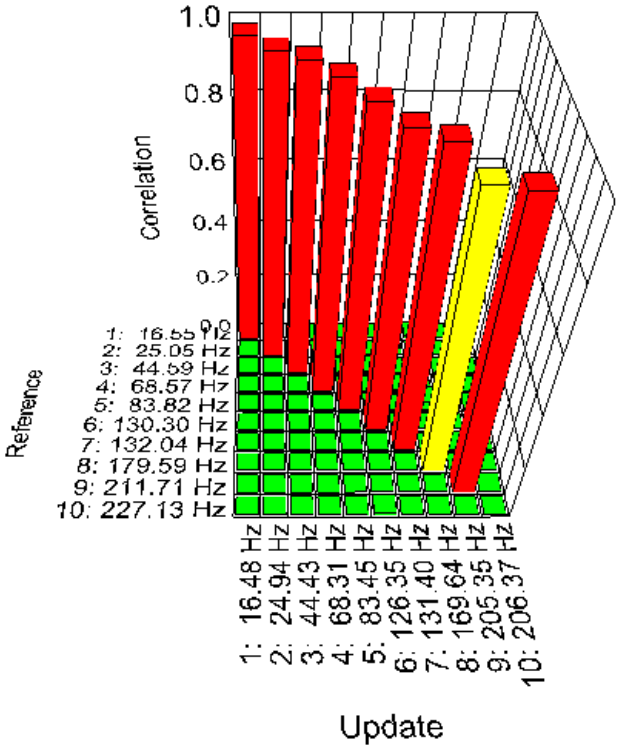

Figure 7.6: Visualization of Modal Assurance Criterion of Beam Models After Optimization

The data is also represented in tabular form in Table 7.4, which also incidentally indicates that many of the off-diagonal values have been improved as a consequence of the MAC optimization, reaching values less than the client's off-diagonal confidence level of 0.05:

Mode	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	0.976	0.000	0.004	0.000	0.003	0.000	0.000	0.013	0.000	0.000
2	0.000	0.976	0.000	0.003	0.000	0.000	0.003	0.000	0.000	0.000
3	0.000	0.000	0.995	0.000	0.001	0.008	0.000	0.001	0.000	0.000
4	0.000	0.000	0.000	0.995	0.000	0.000	0.001	0.000	0.008	0.000
5	0.000	0.000	0.000	0.000	0.979	0.001	0.000	0.031	0.000	0.000
6	0.000	0.000	0.000	0.000	0.000	0.962	0.000	0.000	0.000	0.000
7	0.000	0.000	0.000	0.000	0.000	0.000	0.979	0.000	0.000	0.000
8	0.000	0.000	0.000	0.000	0.002	0.000	0.000	0.927	0.000	0.000
9	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.966	0.000
10	0.000	0.000	0.000	0.000	0.000	0.009	0.000	0.000	0.000	0.000

Table 7.4: Modal Assurance Criterion of Beam Models After Optimization

This relatively simple example demonstrates the fundamental ability of the software package to complete a structural design optimization to improve the MAC correlation of two models. With an understanding of the steps and components necessary to perform a simple beam optimization, larger and more complex examples can be considered.

7.2 Body in White and Trimmed Body

A more elaborate example of model validation and update that is practical for the client involves different phases of the car body design process, specifically the *body in white* and the *trimmed body*. Both of these FE models represent the same car body at different stages of its development. The body in white is an earlier phase that includes the formation of the geometry of the sheet metal frame without items such as the engine, chassis, windshield, seats, and electronics. The trimmed body represents a later phase where many of these components have been added to the final car body design.

The client is interested in designing towards the dynamic response of the trimmed body while still working on the body in white. Changes in the mass distribution between these two phases generate discrepancies between the eigenfrequencies and mode shapes of the two models. A method to overcome these differences is to temporarily add external point masses at various locations on the body in white so that its dynamic behavior more closely resembles that of the trimmed body. Work can then be done on the body in white to perform intermediate design and analyses, while utilizing the projected dynamic behavior of the final design. The client has access to proprietary methods that manually determine the locations and magnitudes of these point masses, but would like to take advantage of optimization to further improve the correlation of these models.

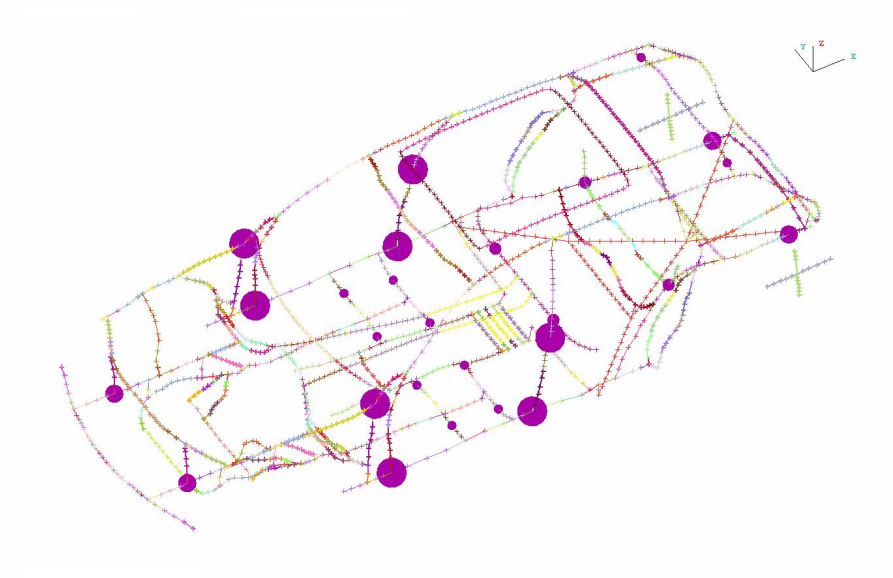

Figure 7.7: Initial Distribution of External Point Masses on Body in White

A sample trimmed body, therefore, can be used to improve the dynamic response of a body in

white. Thus, in this example, the trimmed body represents the reference model and the body in white represents the update model. The design parameters selected for the presented MAC and ORTHO optimization trials are the mass magnitudes of the external point masses. The reference model, rendered by the client in ANSA, can be seen in Figure 7.7. The purple dots along the wireframe indicate a graphical representation of the relative size and location of the external point masses. The geometry, eigenfrequencies, mode shapes, and mass matrices of the reference and update models in the following sections represent a station wagon concept car and were supplied by the client.

As the geometry of the body in white and the trimmed body are identical, a presentation of the trimmed body is not given. The visualization of the DOF connection, however, is provided in Figure 7.8 and can be used to verify that the geometric mapping is accurate:

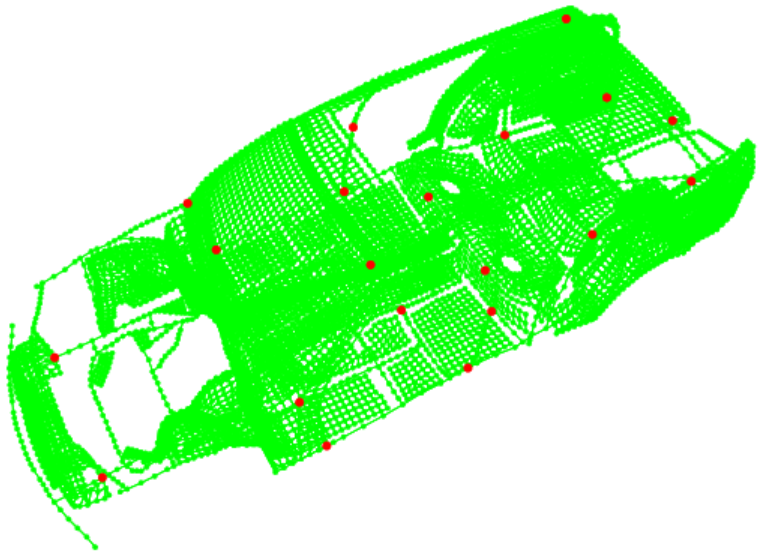

Figure 7.8: Visualization of the Geometric Connection of the Car Bodies

Given the first five mode shapes of both the reference model and the update model as well as the DOF connection, the MAC correlation can be computed and is displayed numerically in Table 7.5. It is important to note the presence of mode switching as the fourth mode shape of both models produce a MAC correlation value of zero.

Mode	**1**	**2**	**3**	**4**	**5**
1	0.893	0.007	0.000	0.091	0.000
2	0.020	0.943	0.000	0.004	0.000
3	0.000	0.000	0.916	0.000	0.055
4	0.000	0.000	0.046	0.000	0.772
5	0.000	0.000	0.068	0.000	0.318

Table 7.5: Modal Assurance Criterion of Car Bodies Before Optimization

By also accounting for the mass matrix of the reference model, the ORTHO correlation can be computed as shown in Table 7.6:

Mode	1	2	3	4	5
1	0.893	0.006	0.000	0.102	0.000
2	0.034	0.933	0.000	0.003	0.000
3	0.000	0.000	0.909	0.000	0.095
4	0.000	0.000	0.025	0.000	0.800
5	0.000	0.000	0.103	0.000	0.330

Table 7.6: Orthogonality Check of Car Bodies Before Optimization

The relative similarity of the results from the MAC and ORTHO correlations is illustrated in Figure 7.9, where both correlations indicate that no combination of mode shapes meet the client's confidence requirements:

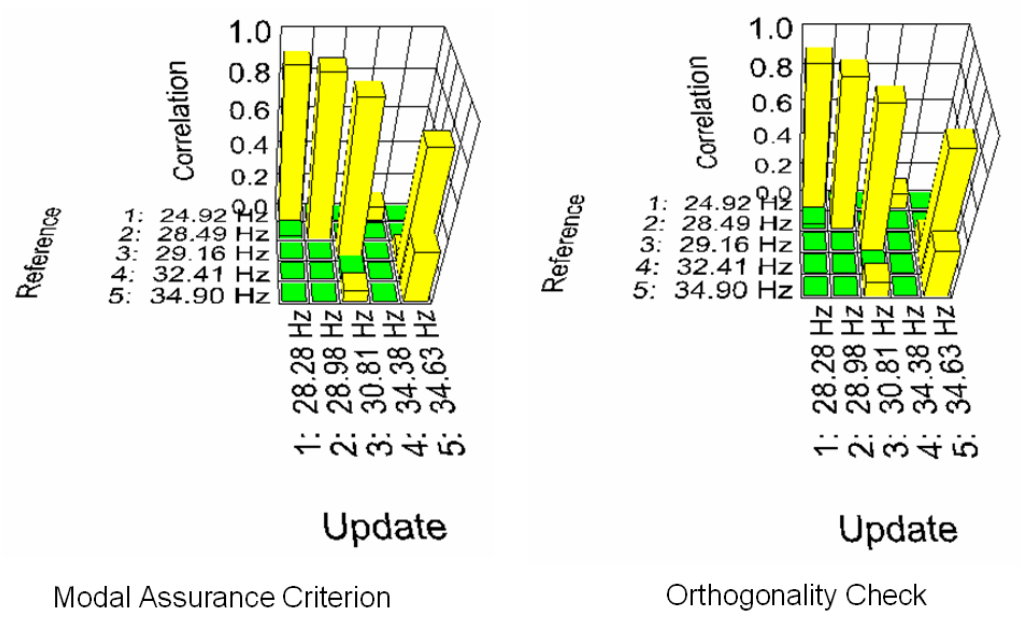

Figure 7.9: Visualization of Car Body Correlations Before Optimization

The ordering of the optimal modal connections according to the MAC and ORTHO correlations are the same, therefore, a numerical representation of these connections is displayed together in Table 7.7 for the MAC, ORTHO, and FREQ correlations. The numerical similarity of the MAC and ORTHO correlations is evident and mode switching can be identified in both cases. To account for the fifth mode connection that produces MAC and ORTHO correlation values of zero, the subsequent optimization decks omit this connection. If this omission had not occurred, the sensitivity analysis would attempt to produce a strong correlation for the connection, most likely sacrificing the correlation optimization of the other modal connections. Thus, it is prudent to emphasize that an examination of the modal connections must be made

to ensure that all connections are viable. In some cases, such as this one with only five mode shapes, it may be incorrect to attempt to connect all available mode shapes.

Reference	Update	MAC	ORTHO	Frequency Difference
1	**1**	0.893	0.893	3.357
2	**2**	0.943	0.933	0.483
3	**3**	0.916	0.909	1.649
4	**5**	0.772	0.800	2.217
5	**4**	0.000	0.000	-0.515

Table 7.7: Mode Shape Connection of Car Bodies Before Optimization

Using the first four modal connections, MAC and ORTHO optimization decks were generated and combined with the client's definitions for the design parameters. Figure 7.10 shows the convergence history for the ORTHO sensitivity analysis, where the magnitudes of the external point masses on the body in white are optimized. The plot of the convergence history of the MAC sensitivity analysis strongly resembles that of the ORTHO sensitivity analysis and is not displayed.

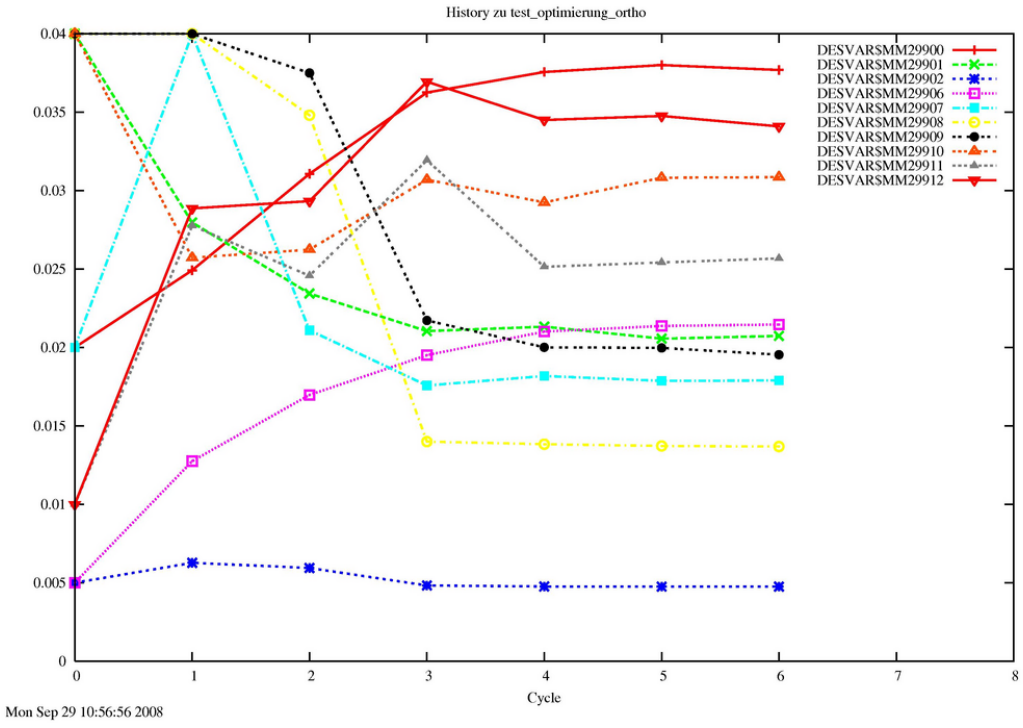

Figure 7.10: Orthogonality Check Convergence History of External Point Masses on Body in White

As the results of the MAC and ORTHO optimization trials are similar, the visual representation of the optimized update model for the MAC correlation is shown in Figure 7.11. In general, it can be seen that the optimization results in a more even distribution of the point masses along the update model.

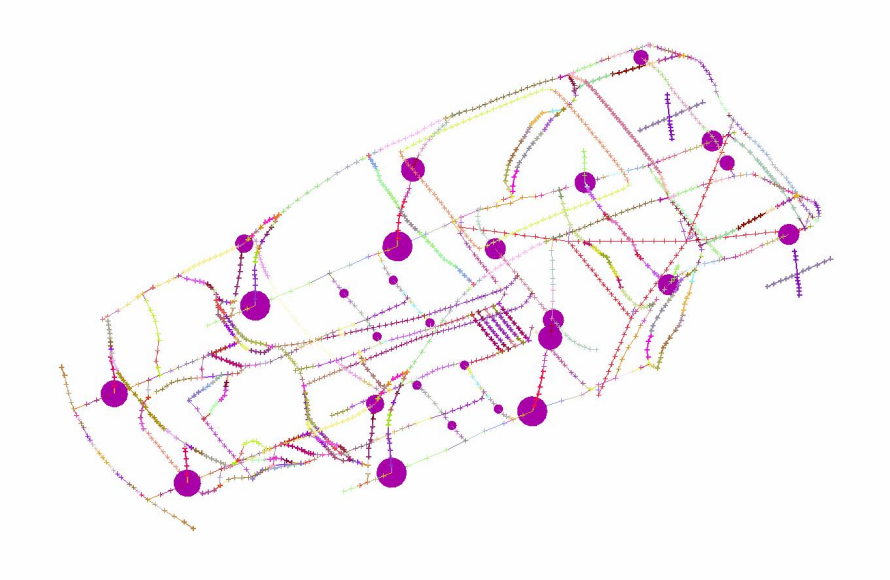

Figure 7.11: Modal Assurance Criterion Optimized Mass Distribution on Body in White

To determine if the update model has been improved, a MAC correlation was performed between the reference model and the optimized update model, provided in Table 7.8:

Mode	**1**	**2**	**3**	**4**	**5**
1	0.983	0.000	0.000	0.003	0.000
2	0.004	0.990	0.000	0.008	0.000
3	0.000	0.000	0.989	0.000	0.001
4	0.000	0.000	0.034	0.000	0.954
5	0.000	0.000	0.053	0.000	0.132

Table 7.8: Modal Assurance Criterion of Car Bodies After Optimization

Similarly, the results of the ORTHO correlation are presented in Table 7.9:

Mode	**1**	**2**	**3**	**4**	**5**
1	0.984	0.009	0.000	0.012	0.000
2	0.037	0.944	0.000	0.001	0.000
3	0.000	0.000	0.993	0.000	0.001
4	0.000	0.000	0.002	0.000	0.994
5	0.000	0.000	0.057	0.000	0.098

Table 7.9: Orthogonality Check of Car Bodies After Optimization

As can be visually concluded from Figure 7.12, the MAC and ORTHO optimization trials result in significant improvements of the correlations of the selected modal connections. Additionally, some reduction of the error of the off-diagonal correlation values occurs even without direct consideration in the objective function of the optimization decks.

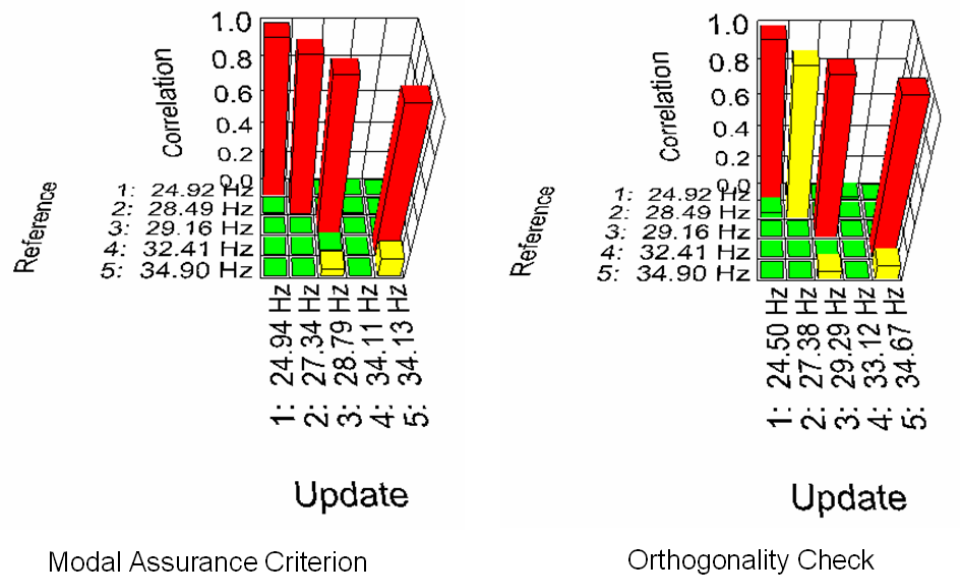

Figure 7.12: Visualization of Car Body Correlations After Optimization

The numerical results of the optimized MAC connection are presented in Table 7.10:

Reference Mode	Update Mode	MAC Value	Frequency Difference
1	**1**	0.983	0.017
2	**2**	0.990	-1.154
3	**3**	0.989	-0.362
4	**5**	0.954	1.718
5	**4**	0.000	-0.789

Table 7.10: Modal Assurance Criterion Connection After Optimization

Similar results for the optimized ORTHO connection are shown in Table 7.11:

Reference Mode	Update Mode	ORTHO Value	Frequency Difference
1	**1**	0.984	-0.417
2	**2**	0.944	-1.113
3	**3**	0.993	0.131
4	**5**	0.994	2.264
5	**4**	0.000	-1.784

Table 7.11: Orthogonality Check Connection After Optimization

The results of the modal connections demonstrate that the optimization trials improved the MAC, ORTHO, and FREQ correlations of the reference and update models. They also show that in some cases, the effects of the MAC and ORTHO correlations are quite similar. One possible reason for this is that the number of DOF incorporated into most practical applications is relatively low. As the number of compared and correlated DOF increases, a divergence of the results from the MAC and ORTHO correlations may start to emerge.

Considering this, another important observation is that a relatively low number of design parameters can have a significant impact on the global response of a dynamic FE structure. In this example, the proper placement of point masses at several locations along the body in white was able to improve the MAC and ORTHO correlations of almost all modal connections beyond the client's confidence requirements. This suggests that the wide range of design parameters available to designers can provide a high degree of customization and optimization of the dynamic response of the structure. As the dynamic correlation of two models consists of comparing both the mode shapes and the eigenfrequencies, consideration of the MAC, ORTHO, and FREQ correlations as a whole is essential in finally interpreting the improvement generated by a model update.

Chapter 8

Summary

The contents of this Master's thesis focus on the industrial need to validate and update FE models to accurately predict the dynamic behavior of the structures for which they are simulated. This process includes iteratively computing the MAC, ORTHO, and FREQ correlations of two models and conducting a parameter-based optimization analysis to improve their results. Further, a discussion of the Java software implementation of these techniques is provided to document the work done for the client, which is the deliverable objective of this thesis.

The presented MAC and ORTHO correlations each have benefits and drawbacks. The ORTHO correlation is fundamentally derived from the theory of structural dynamics, however, the data required to properly conduct an ORTHO correlation are seldom available. Therefore, a model reduction must be made to successfully perform the computation. This reduction, however, is generally not perfect and may distort the final result. The MAC correlation does not encounter this problem and can be readily scaled according to the data set, however, its fundamental background is statistical and may be prone to providing misleading results.

The form of the implemented objective function, based on the Bayesian parameter estimation, provides a high degree of customization, taking into account correlation data and weight factors for each design parameter. As demonstrated in the presented examples, the dynamic results of an update model can be significantly improved with this optimization procedure. While the optimization process is relatively reliable for most applications, the maximum size of the optimization deck that can be generated may be a limiting factor in some scenarios.

Given the scope of this work, the following future improvements may prove useful:

- A more comprehensive algorithm for determining the DOF correlation, where the geometries of two models do not have to be pre-aligned to each other.
- The addition of other modal correlations, detailed in [18], which focus on more specific goals such as targeting mode shapes and considering load cases.
- A correlation and optimization procedure, based on the frequency response of a dynamic structure as described in [3]. Combined with the eigenfrequency and modal correlations, the frequency response provides a complete overview of the entire dynamic response of two structural models.

Bibliography

[1] R. J. Allemang. The modal assurance criterion – twenty years of use and abuse. *Sound and Vibration*, 2003.

[2] K. Blakely. Updating msc/nastran models to match test data. In *MSC 1991 World Users' Conference Proceedings*, 1991.

[3] K. Blakely. Matching frequency response test data with msc/nastran. In *MSC 1994 World Users' Conference Proceedings*. The MacNeal-Schwendler Corporation, 1994.

[4] K. Blakely and T. Rose. Cross-orthogonality calculations for pre-test planning and model verification. In *MSC 1993 World Users' Conference Proceedings*. The MacNeal-Schwendler Corporation, 1993.

[5] D. J. Bouvier. Getting started with the java 3d api. Tutorial, K Computing, 1999.

[6] K. De Langhe. Validating and updating structural fe models for dynamic analysis. LMS International, 2007.

[7] F. Duddeck. Car body design – optimization. Lecture notes, Department of Engineering, Queen Mary College, London University, 2006.

[8] C. H. Edwards and D. E. Penney. *Differential Equations, Computing and Modeling*. Prentice-Hall, Inc., 2 edition, 2000.

[9] A. Fowler. A swing architecture overview. *Sun Developer Network*, 2003.

[10] I-DEAS Test. *Universal Dataset Number 55*, 1997.

[11] C. Katzenschwanz. Parameteridentifikation an einer nasa-experimentalstruktur. IABG mbH, 2006.

[12] MSC.Software Corporation. *MSC.Nastran 2004 Quick Reference Guide*, 2003.

[13] G. Müller. Structural dynamics. Lecture notes, Lehrstuhl für Baumechanik, Bauingenieur- und Vermessungswesen, Technische Universität München, 2007.

[14] G. Pflanz. Nvh related car body design at bmw. BMW, 2006.

[15] Sun Microsystems, Inc. *Java Look and Feel Design Guidelines*, 2 edition, 2001.

[16] Sun Microsystems, Inc. *Java 2 Platform API Standard Edition 5.0 Specification*, 2004.

[17] T. Sundsted. Mvc meets swing. *JavaWorld*, 1998.

[18] T. Van Langenhove and M. Brughmans. Using msc/nastran and lms/pretest to find an optimal sensor placement for modal identification and correlation of aerospace structures. In *MSC 1999 Aerospace Users' Conference Proceedings*. LMS International, 1999.

[19] D. A. Wells. *Schaum's Outline of Lagrangian Dynamics*. McGraw-Hill, Inc., 1967.