# A Structural Optimization Scripted Software System

Yongyan Sun

Thesis submitted to the Faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Master of Science

In

Mechanical Engineering

Robert L. West, Chair

Jan Helge Bøhn

Robert A. Canfield

February 19th, 2019

Blacksburg, VA

Keywords: Finite Element Analysis, Optimization, Scripting

# A Structural Optimization Scripted Software System

Yongyan Sun

## Abstract

This thesis introduces an optimization software system which supports two separate optimization approaches to solve structural optimization problems with small and large-scale finite element models. The approach for solving the structural optimization problems of small-scale finite element models consists of the gradient-based optimization method and input file regeneration program. The small-scale structural optimization system, requires users only to put in the parameters of the initial design, the system will run the optimization process and generate new models automatically until the solutions are obtained. The approach for solving structural optimization problems of large-scale finite element models combines parametric finite element modeling methods executed by Python scripts with response surface optimization methods (RSM). This approach reduces the number of finite element analyses as well as reduces the optimization process time. The optimization module of the system is performed by the MATLAB optimization toolbox and the Abaqus finite element program with scripts implemented in Python.

A benchmark hollow-tube weight-minimization problem is conducted to test the optimization software system. The percent difference between the solution found by the graphical optimization method and the solution found by the 3D beam finite element model with Sequential Quadratic Programming (SQP) solver and the graphical optimization method is 1.99%. The percent difference between the results from the 3D beam finite element model with SQP solver and the result from 3D brick finite element model with

response surface method is 8.16%. The percent difference between the results from the 3D brick finite element model with RSM and the result from the graphical optimization method is 10.31%.

# A Structural Optimization Scripted Software System

Yongyan Sun


## General Audience Abstract

Commercial structural optimization software packages which integrate modeling tools, optimization and extensive computational tools such as a finite element solver were developed and pushed to the market. However, some commercial approaches to structural optimization are not very general. In addition, the commercial codes are designed for a specific-purpose, and they may not be suitable in many cases. If the commercial codes do not properly represent the structural optimization problem, users have to write custom Python scripts to assist the software system in retrieving data from the .odb files generated by FEA software.

This thesis introduces an optimization software system which supports two separate optimization approaches to solve structural optimization problems with small and large-scale finite element models. The optimization module of the system is performed by the MATLAB optimization toolbox and the Abaqus finite element program with scripts implemented in Python. This optimization software system allows users to extract and manipulate data for optimization without limitations. Furthermore, once the required parameters are input in the system, the scripting software creates the finite element model and proceeds with the optimization automatically.

# Acknowledgement

I would like to thank my committee chair, Professor Robert West, for his guidance and encouragement throughout my studies in finite element analysis and structural optimization. I would also like to thank my other committee members: Professor Jan Helge Bøhn and Professor Robert A. Canfield for serving on my committee and their suggestions and encouragement.

I would also like to express my love to my parents, my wife and other family members for their love and support throughout all of my life.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Background and Motivation

Optimization methods are applied in many areas, such as manufacturing, mathematical modeling and structural design. The objective of an optimization problem can be formulated as minimizing manufacturing cost, minimizing production weight, or achieving best performance. The optimized design parameters are determined to achieve the best measurable performance while all requirements are satisfied [1].

Essentially, an optimization process consists of a model, optimizer, and a means of evaluating the objective and constraint functions, such as a simulation program. Most real-world applications have far more complicated factors and parameters as well as constraints that impact the system behavior; most problems are nonlinear. Thus, the challenge is to create an appropriate mathematical model to represent the actual model. The next step is to choose appropriate design variables and to use the right algorithms for solving the problem and searching for optimal solutions. In most engineering designs and industrial applications, the objective functions cannot be expressed in explicit analytical forms, since the dependence of the objective function on design variables is complex and implicit [2]. In most applications, the objective function and constraint functions are evaluated hundreds or thousands of times. Thus, an efficient simulation and an appropriate optimization process tailored to use the right algorithms and appropriate design variables can significantly reduce the computing time and improve the quality of the optimization result [2].

Since Schmit [3] introduced a structural optimization approach consisting of finite element analysis and non-linear mathematical program in 1960, this approach has been used in industrial applications [4]. In recent decades, due to developments in computer science and optimization algorithms, the advance in computers able to operate on large data sets and run the finite element analysis, commercial structural optimization software packages that integrate modeling tools, optimization and extensive computational tools such as a finite element solver were developed and pushed to the market. Some notable software, such as Nastran, Abaqus, ANSYS, TOSCA, Isight and Abaqus/CAE optimization software module are used to solve structural design problems in industry [5].

Commercial optimization programs provide a suite of visual and flexible tools to set up and manage finite element analysis software, CAD/CAE software and internally developed optimization programs for the purpose of executing simulation-based design processes. In order to facilitate operation that target users from diverse fields of applications who are not familiar with design optimization, the codes are developed to be general and perform in a robust manner and easy to use [6]. These commercial finite element software packages are able to perform finite element analysis on parts with complex geometry, and subsequently use the results obtained from finite element analysis process to proceed with the structural optimization. With the help of high performance computers and fast numerical algorithms, a very impressive optimum solution solved by structural optimization with the finite element method can be achieved [2]. However, the optimization solutions solved by different software systems can be different due to a combination of factors. The optimization environment, which consists of the optimization formulation, convergence criteria, utilized optimization method, and move limit strategy, can have a significant impact on the performance of the structural optimization software system [5]. Unfortunately, commercial optimization software does not contain a wide range of analysis options able to handle large real-world industrial models. Users cannot define the optimization problem in NLP standard form. For instance, the Abaqus optimization module does not allow users to define the objective and constraint functions. Additionally, some software systems do not allow users to change the default settings, even though the default setting may not be suitable for the solution of a particular problem in some cases. In order to better understand the limitations of the commercial software system, a specific case is introduced in Section 2.1.

This thesis seeks to develop a structural optimization software system combining the Abaqus Finite Element solver with the MATLAB optimization toolbox. The software system for this research is developed in Python and MATLAB. This optimization program is designed to have the following functionalities:

1. Allow users to define the design variables, the objective and constraint functions.
2. As the required parameters are input to the system, the finite element analysis model is created as a parametric finite element model in Abaqus.
3. Allow users to control the data extraction and manipulate the data to evaluate the objective and constraint functions.
4. Automatically executes the optimization process by evaluating the objective and constraint functions repetitively until the solution is obtained.

## 1.2 Goals of the Research

The goal of the research is to develop a structural optimization software system which supports two separate optimization approaches to solve structural optimization problems with (1) small-scale finite element models and (2) large-scale finite element models. The approach for solving the structural optimization problems of small-scale finite element models consists of the gradient-based optimization method and input file regeneration program. In this system, users only need to put in the parameters of the initial design, the system will run the optimization process and generate new models automatically until the solutions are obtained. The approach for solving the structural optimization problems of large-scale finite element models combines parametric finite element modeling methods executed by Python scripts used to develop a response surface optimization method. This approach reduces the number of finite element analyses as well as reduces the optimization process time, it can also be used as an incremental optimization approach. The optimization module of the system is performed by the MATLAB optimization toolbox and the Abaqus finite element program with scripts implemented in Python. This system is not only focused on solving structural optimization problems, it also can be used in all kinds of optimization problems, such as linear statics, heat transfer, and frequency response problems [6], such as determining the number and radius of conducting tubes to maximize the surface area of the tubes in the exchanger while satisfying some addition requirements [1].

## 1.3 Research Scope

Based on business considerations, the commercial optimization software systems are designed in general-purpose and robust. These commercial optimization systems are able to handle some kinds of optimization problem under specific conditions. For a structural optimization problem, the information from the FEA can be extracted and operated on by the optimization module automatically by the embedded commercial codes. However, some commercial approaches to structural optimization are not very general. For instance, the Abaqus/CAE optimization software module does not allow users to define constraint functions for the optimization problem. In addition, the extraction codes used in Isight are designed in general-purpose, however, Isight by default extracts the stresses from the finite element in only one way, and therefore the optimization solutions calculated by using these stresses used in the optimization evaluation of the objective and constraint functions are incorrect. If the commercial codes cannot properly represent the design optimization problem, users have to write custom codes to assist the software system to appropriately represent the optimization problem and in retrieving data from the result files generated by FEA software.

This thesis introduces a structural optimization software framework, which allows users to define the optimization problem in NLP standard form, and extracts and operates on the FEA result data to be effectively used for optimization without limitations. Furthermore, once the required parameters are input in the system, the optimization software system is able to create the finite element model and proceed with the optimization automatically.

## 1.4 Thesis Organization

This thesis is divided into four main parts. The introductory part consists of Chapter 1 and Chapter 2. Chapter 1 introduces the background for the optimization framework and the structure of the optimization system, and discusses the recent issues of commercial optimization software systems. It considers solutions to some of the issues that occur in current commercial optimization software systems, and proposes a new structural optimization software system with less limitations, followed by the scope and objectives of this research. Chapter 2 reviews the literature and background of optimization numerical algorithms and development of optimization systems. The functions and issues of some commercial optimization software systems are included.

Chapter 3 introduces a structural optimization problem which is used to test the system. The Euler-Bernoulli beam equations and formulation of the structural optimization problem is developed. The graphical optimization method is introduced and used to solve the problem.

Chapter 4 introduces an optimization approach for solving the problems associated with small-scale structural models. The operation procedure of the optimization process for a 3D beam finite element model, and the details of extracting the results are introduced in this chapter.

Chapter 5 presents the approach for solving the structural optimization problem associated with a large-scale finite element model. In this chapter, a 3D continuum element model was selected as the model for representing the mechanics, and method for boundary conditions and loads used to test the structural optimization software system. The 3D continuum element model could be scaled to any large-scale finite element model. Since this approach consists of the parametric finite element modeling method and response surface optimization methodology, these methods are introduced in this chapter.

Chapter 6 summarizes the work, and presents the discussions about the results and recommendations for future work.

# Chapter 2

# Literature Review

In this chapter, the detail background of the relevant approaches and methodologies for building a structural optimization software system are introduced to help understand the motivation for this research.

## 2.1 Specific Example to Explain the Motivation

Isight, an optimization software system developed by Dassault, has some result file (.odb file) extraction options such as extract the mass properties, history outputs, all field outputs, field sets field output components, and fields at each frame from the .odb file. In order to make the software more general, Isight will only extract extreme values at integration points unless custom scripts are written and embedded into the extraction program by users. Most commercial optimization software systems choose to extract the extreme stress values at the integration points. This choice is understandable, since the stresses at the integration points are the most accurate locations to evaluate element stresses in the majority of finite element models. In general, the highest stresses can be obtained by using a mapping method, and where the mapping method requires the values from at least 2 integration points in one element. However, in some cases, this method does not work due to lack of information required for the extrapolation of the element stresses and strains to the nodes. The pipe section of the beam element is shown in Figure 2.1 [7].

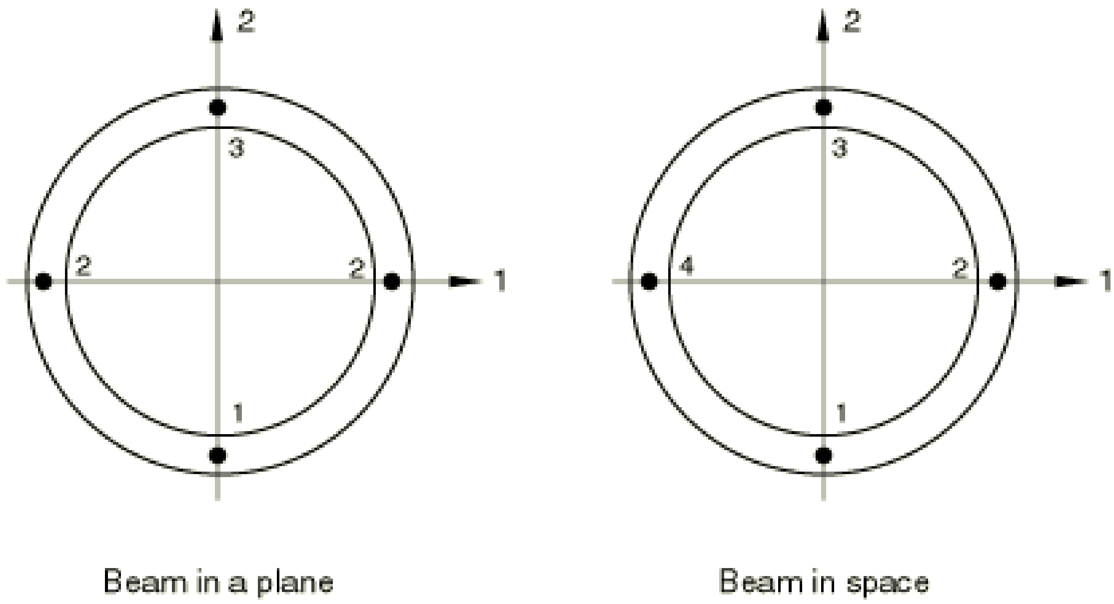


**Figure 2.1: Pipe Section used in Abaqus**

Assuming a downward transverse force is acting on the pipe, the maximum normal stress in the global x-direction, *S11*, will be located at the top section point, and the minimum normal stress in the global x-direction, *S11*, may be located at the bottom section point. Use of the extraction script provided by Isight, the *S11* values at integration points of the element are passed into the system. However, the actual extreme values cannot be calculated by using this information, since the extracted normal stress in the global x-direction, *S11*, stresses are not in the same plane. In this case, Isight extracts the maximum *S11* stresses from the interpolation point on the top fiber of the element, and it extracts the minimum *S11* stresses from the interpolation point on the bottom fiber of the element. The theoretical maximum *S11* stresses are developed at the node attached on the wall. In order to obtain the theoretical maximum *S11* stresses by using mapping, two *S11* stresses on the same fiber are required. However, the extraction script of Isight only extract the extreme stresses from the integration point in the element. Thus, the theoretical maximum *S11* stress could not be mapped. Therefore, the solutions calculated using these stresses used in the optimization evaluation of the objective and constraint functions are inappropriate.

## 2.2 Development of Commercial Optimization software Systems

The research about structural optimization has begun with the early works of Maxwell in 19$^{th}$ century [8], the structural components were optimized by using the calculus of variations before the computers were invented. The finite element analysis of structure and computer-based optimization became prevalent due to the development of computers. Numerical optimization based on finite element models was first introduced by Schmit in 1960 [9]. Schmit developed a mathematical programming approaches to solve the structural optimization problems under a multiplicity of loading conditions [10]. With the further developments computers and the increase of the performance of the computer software techniques, modern computers memory chips represent tens or hundreds of thousands of megabytes (MB) and the total memory of personal computers is measured in gigabytes (GB) or more. Digital computer circuits are capable of performing hundreds of billions of logic operations per second [11]. Thus, modern computers are capable of supporting commercial optimization software systems to perform complex calculations, such as redefinition of the finite element mesh or adding or removing parts from a complex 3D model. Additionally, due to the improvement of the optimization algorithms, such as topology optimization based on a homogenization method [12], the optimum design of complex solid models can be explored using topology optimization methods on computers. Hence, structural optimization research has focused on structures shape optimization and topology optimization. In recent years, a wide range of software tools have been introduced as commercial products. These tools provide an open system for integrating computer-aided design (CAD) software applications and optimizers to execute the optimization processes automatically. Some tools provide a standard library of components including applications, such as MATLAB and Excel, and scripting tools for integrating and running simulation or optimization analysis.

Companies such as Dassault Systemes Simulia Corporation, have a full-featured script development environment. These development environment provide tighter integration with internal software systems and from software developed by partners so that the subsystems are able to act as one system [13]. The resulting software suite is designed as an integration of CAD modeling, analysis tool and optimization solver. The designers may focus more on integrating the tools in one environment and improving the efficiency of the system, which tend to develop the extraction codes more generally and target users from diverse fields of applications [6]. However, the extraction core codes designed in general-purpose software systems may not be suitable for all cases. Thus, a structural optimization system, which allows users to fully control the optimization process, is required.

### 2.3 Parametric Modeling

Parametric modeling is a regular approach for the modeling stage in a structural optimization process. It was first developed by Sutherland in 1962 [14]. In a structural optimization problem, the objective and constraint functions were evaluated repetitively until the optimum solution is obtained. However, as the structural optimization proceeds, the independent parameters which were used to create the model keep varying.

The parametric modeling for beam finite element models is very different than parametric finite element modeling. The 3D beam finite element models are already parameterized. However, in a 3D brick model structural optimization problem, the model must be generated and remeshed if the parameters change. Without a robust parametric finite element modeling method, a large number of finite element models would have to be created and analyzed in sequence manually. In the model building scripts based on parametric modeling, the design variables can be expressed as functions of the input parameters. The parametric finite element modeling method is a connection between the finite element solver and the optimizer. The model building steps can be controlled in the optimization application, such as MATLAB, without opening the Graphical User Interface (GUI), such as Abaqus/CAE [15]. Additionally, moving the function of controlling the model building from Abaqus to MATLAB enables users to form the design variables in vectors, then use them to create finite element models and run analysis automatically with loop control statements.

## 2.4 Response Surface Methodology

Response surface methodology (RSM) was first introduced by Box and Wilson [16] in 1951[17]. This methodology was first used in the chemical industry, and later adapted for particular application areas [17], such as optimization of the design of ship structures [18].

In general, the objective functions of real-world optimization problems are unknown, and the gradient vectors to determine the search direction for design update and the step size cannot be determined. Therefore, the number of finite element analysis runs is not predictable. If a fixed step size is used in the gradient-based optimization process, a massive number of discrete brick finite element models are required to obtain the accurate optimum solution, which leads to an increase of time-consuming and computationally expensive for finite element analysis, as well as the solution may not converge while the constant step size is used in the optimization process. If RSM is used when the objective function is unknown, the number of finite element analysis runs can be controlled, and the approximation function can be generated using the function values at selected sample points.

In the traditional optimization methods, the FE model keeps updating while the updated parameters of the FE model are calculated by the gradient-based algorithm. FE model updating and gradient calculation may lead to long computational time. Involving the RSM in the optimization process, an equation which describes the behavior of the objective and constraint functions is generated base on the data from sample points in the design space [19]. Once the appropriate function, such as linear, quadratic, and cubic functions, is selected to generate the response surface, the objective and constraint function values at any points in the design space can be approximated from the response surface. Since the approximation function are typically continuous and tailored to the problem, the gradient-based algorithm can help to find the accurate optimum solution on the response surface.

In order to generate an appropriate response surface, a crude approximation of the objective function is initially developed using the function values at a few sample points from the design space. If the initial approximation functions are not able to represent the objective function adequately, a new response surface can be updated based on the regression test and the statistical analytical results from previous response surfaces, and the best response surface is generated until the regression test indicates a converged representation of the objective and constraint function exist based on statistical results. In this process, no additional FE analysis runs are made, and the response surface is updated by exploring interpolation/approximation functions that may better represent the objective and constraint functions.

In summary, the RSM is used because of the following reasons: the number of FE analysis runs can be controlled; the number of finite element analysis runs necessary to perform the optimization is much lower than traditional optimization method, which leads to a decrease of time-consuming and expenses for finite element analysis; the response surface can be checked and updated easily based on the previous response surface approximation without making addition FE analysis runs.

## 2.5 Scripting for Structural Optimization Programs

The optimization process which interfaces MATLAB with the commercial FE solver Abaqus via Python scripts has been proved to be very efficient to find a very good optimal point [20]. Based on the reliability and performance from previous work [21], MATLAB and Abaqus are selected as the optimization and FE solvers in the optimization software system. Scripting enables users to employ the functionality of the GUI of Abaqus through the programming language Python without operating on the GUI [22]. Manipulating Abaqus through Python script is accomplished using the Abaqus Scripting library modules, the Python packages from NumPy and SciPy, and other modules such as csv (Comma Separated Values) are imported into the scripts. The Abaqus Scripting library modules include Abaqus CAE Modules and other modeling modules are responsible for modeling, and running the finite element analysis. NumPy and SciPy packages allow users to write mathematical expressions as well as variables formulas for parametric modeling in the Python script. Other packages, such as job, odbAccess and csv, are used for running FEA and extract results from .odb files generated by Abaqus. The modeling and meshing scripts are responsible for building, meshing the assembled model, and submitting the jobs. After the .odb files are generated, the extracting script helps users extract results from the .odb files and pass data into MATLAB. The methods for the extraction of Abaqus results are determined by users, so that they can extract the results needed from the target nodes, elements or sets. Once the modeling, meshing and extraction scripts are developed, MATLAB is capable to call them from the command line or script interface itself. The design variables for the structural optimization problem can be stored and formed in MATLAB as matrices. As the optimization is proceeding, the design variables are updated, and the modeling, meshing, and extracting processes are executed by Python scripts automatically. MATLAB provides a powerful and reliable optimization toolbox to users. The setting for optimization, such as the selection of solvers, length of step size, and initial guess design variables can be input in the interface of the MATLAB optimization toolbox. The constraint functions and objective functions can be edited in MATLAB programs.

# Chapter 3

# Structural Optimization of the Aluminum Alloy Post

This chapter presents the statement of the structural optimization problem. The formulas used for formulating the objective and constraint functions for the post and optimization algorithm are presented and discussed.

## 3.1 Nonlinear Programming Problem

The Standard form of Nonlinear Programming design optimization model is defined as below:

find an n-vector $x = (x_1, x_2, \cdots, x_n)$ of design variables to

minimize a cost function:

$$f(x) = f(x_1, x_2, \cdots, x_n) \tag{3.1}$$

subject to the $p$ equality constraints:

$$h_j(x) = h_j(x_1, x_2, \cdots, x_n) = 0;\ j = 1 \text{ to } p \tag{3.2}$$

and the $m$ inequality constraints:

$$g_i(x) = g_i(x_1, x_2, \cdots, x_n) \leq 0;\ i = 1 \text{ to } m \tag{3.3}$$

The bounds on design variables, such as $x_i^L \leq x_i \leq x_i^U$, where $x_i^L$ and $x_i^U$ are the smallest and largest allowed values for $x_i$ are included in the Equation 3.3 [1].

### 3.1.1 Problem Description

Weight reduction is one of the most frequently discussed objectives in the design structural optimization area. The objective of this project is to minimize the weight of the hollow tube shown in Figure 3.1, while satisfying the performance and design limitations.

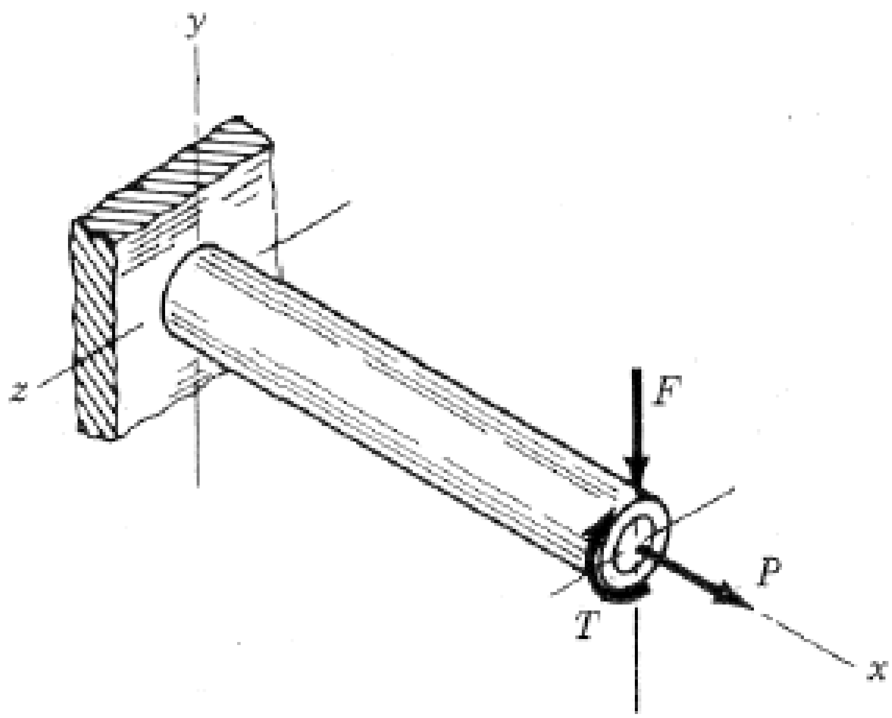


**Figure 3.1: Hollow Tube used in the Design Optimization Problem**

### 3.1.2 Data and Information Collection

In this project, a hollow tube made of UNS A92014-T4 aluminum alloy is considered as the structural optimization object. The length $L$ of the tube is 4.7 inches. The base of the tube is fixed on the wall, with three fluctuating loads acting on the end of the tube. The tube is subjected to a torsional load $T$ fluctuating from 280 to 660 in-lbf and a fluctuating downward transverse load $F$ varying from 45 to 240 $lb_{f}$. An axial load $P$ is applied to the tube and is cycled from 950 to 1680 $lb_{f}$.

**Table 3-1: Material Properties of the UNS A92014-T4 Aluminum Alloy**

| weight density γ (lb/in^3) | Modulus of Elasticity $E$ (ksi) | shear Modulus $G$ (ksi) |
|---|---|---|
| 0.101 | 10600 | 4060 |
| yield strength (ksi) | ultimate strength (ksi) | fracture strength (ksi) |
| 40 | 62 | 20 |
| allowable yield strength with 0.9 safety factor on material strength(ksi) | allowable ultimate strength with 0.9 safety factor on material strength(ksi) | allowable fracture strength with 0.9 safety factor on material strength(ksi) |
| 36 | 55.8 | 18 |

The material properties of the UNS A92014-T4 are listed in Table 3-1. Based on the ASM material data sheet, the weight density of the UNS A92014-T4 aluminum alloy $\gamma$ is 0.101 lb/in$^3$, the average value of Modulus of Elasticity $E$ under tension and compression is 10600 ksi, the shear Modulus $G$ is 4060 ksi, and the Poisson's ratio is 0.33 [23]. The minimum yield strength $S_y$ is 40 ksi, the minimum ultimate strength $S_{ut}$ is 62 ksi, and the minimum high-cycle fatigue strength $S_f$ for the aluminum alloy at 5*10$^8$ cycles is 20 ksi. Generally, allowable stresses are used as the maximum stress levels that will be allowed in the structure. As a performance requirement, it is assumed that structural failure will occur if the stress applied on the structure exceeds the allowable value [1]. The allowable stresses are defined as the material strength divided by a factor of safety on the material strength, which is written as $\sigma_{Allowable}$ = Material Strength / factor of safety. Since 10% reduction in each strength is considered as the factor of safety on material strength, the allowable stress is calculated as the material strength divided by the factor of safety which is 1.1. As a result, the allowable yield strength $S_y$ is 36 kpsi, the allowable ultimate strength $S_{ut}$ is 55.8 ksi, and the allowable fracture strength $S_f$ is 18 ksi. The information of the hollow tube design constraints are shown in Table 3-2.

**Table 3-2: Information to Design a Hollow Tube**

| **Notation** | **Data** |
|---|---|
| Minimum safety factor of fatigue fracture | $n_{AllowableFatigueFracture}$ = 1.2 |
| Minimum safety factor of fatigue yield | $n_{AllowableFatigueYield}$ = 1.1 |
| Allowable axial deflection | $\delta_{AllowableAxial}$ = 0.005 in |
| Allowable vertical deflection | $\delta_{AllowableVertical}$ = 0.1 in |
| Allowable twist deflection | $\theta_{AllowableTwist}$ = 1 degree |
| Lower limit on outer radius of hollow tube | ${r_o}^L$ = 0.375 in |
| Upper limit on outer radius of hollow tube | ${r_o}^U$ = 1 in |
| Lower limit on hollow tube outside radius | $t^L$ = 0.0625 in |
| Upper limit on hollow tube thickness | $t^U$ = 0.25 in |

### 3.1.3 Design Variables

The outside radius $r_o$ and wall thickness of the tube $t$ were identified as the design variables for the problem, since they can define the cross-sectional area $A$, which is directly related to the volume $V$ as well as the weight of the tube. To illustrate a formulation of the problem, let the design variables be defined as

$x_1$ = tube outside radius, $r_o$

$x_2$= tube wall thickness, $t$

### 3.1.4 Optimization Criterion

The design objective is to minimize the weight of the hollow tube structure subject to constraints on fatigue and deflection. For the hollow tube with a circular section, the weight was determined by its weight density and its volume, length multiplied by the cross-sectional area. The objective function to be minimized is shown in Equation 3.4. The same objective function was used for the graphical optimization method, the direct optimization approach for small-scale finite element models, and the response surface optimization approach for large-scale finite element models.

$$f(x)=\gamma LA \tag{3.4}$$

By definition, the outside radius $r_o$ is always larger than the wall thickness $t$, in terms of the variables, the cross section area $A$ is given as

$$A=\pi(2r_o t-t^2) \tag{3.5}$$

Therefore, the final form of the objective function depending on the design variables selected for the problem is given as

$$f(x)=\gamma L\pi\,(2x_1x_2-x_2^2) \tag{3.6}$$

### 3.1.5 Design Constraints

The design constraints for this problem focus on the safety factors, structural deflections and the design variables. These constraints are defined as

$$n_{AllowableFatigueYield} \leq n_{ActualFatigueYield} \tag{3.7}$$

$$n_{AllowableFatigueFracture} \leq n_{ActualFatigueFracture} \tag{3.8}$$

$$|\delta_{ActualAxial}| \leq \delta_{AllowableAxial} \tag{3.9}$$

$$|\delta_{ActualVertical}| \leq \delta_{AllowableVertical} \tag{3.10}$$

$$|\delta_{ActualTwist}| \leq \delta_{AllowableTwist} \tag{3.11}$$

$$x_1^L \leq x_1 \leq x_1^U \tag{3.12}$$

$$x_2^L \leq x_2 \leq x_2^U \tag{3.13}$$

In this research, the constraints are normalized with respect to their limit values, the constraints are defined in NLP standard form as

$$g_1(x) = 1 - \frac{nActualFatigueYield}{1.1} \leq 0 \tag{3.14}$$

$$g_2(x) = 1 - \frac{nActualFatigueFracture}{1.2} \leq 0 \tag{3.15}$$

$$g_3(x) = \frac{|\delta ActualAxial|}{0.005} - 1 \leq 0 \tag{3.16}$$

$$g_4(x) = \frac{|\delta ActualVertical|}{0.1} - 1 \leq 0 \tag{3.17}$$

$$g_5(x) = |\theta_{ActualTwist}| - 1 \leq 0 \tag{3.18}$$

$$0.375'' \leq x_1 \leq 1'' \tag{3.19}$$

$$0.0625'' \leq x_2 \leq 0.25'' \tag{3.20}$$

The details of calculation of the equations are introduced in Section 3.2.2. When the problem is solved using a numerical method, a constraint $r_o > t$, represented as $x_1 > x_2$, must be enforced, because some optimization method may take the design into the infeasible region of the design space, where $t \geq r_o$. If the thickness $t$ is equal to the magnitude of the outer radius $r_o$, meaning that the magnitude of the inner radius $r_i$ is 0, the hollow tube should be considered as a solid bar, and the equations used in this problem are not applicable any more. If $t > r_o$, the design is physically impossible. It should be noted that the upper bound of $x_2$ is less than the lower bound of $x_1$ in this problem, which means that the constraint $x_1 > x_2$ is inactive, and this constraint could be violated during the optimization solution.

## 3.2 Graphical Optimization

### 3.2.1 Graphical Optimization Concepts Overview

Since the optimization problem has only two design variables, graphical optimization is an appropriate method to represent the optimization problem by plotting the objective functions and constraints. After all of the constraint and objective function contours are plotted in the design space, the feasible region is identified and the optimum solution can be determined by visual inspection. Figure 3.2 demonstrates the design space of the optimization problem for this project. The details of the graphical solution process and the result analysis will be discussed in the following sections. The values of $g_4(x)$, which is the constraint function on the vertical displacement, are less than 0 in the whole optimization process. It means that the absolute values of vertical displacement of the hollow tube are always less than the absolute value of allowable vertical displacement, thus, the vertical displacement constraint line does not show up on the plot.

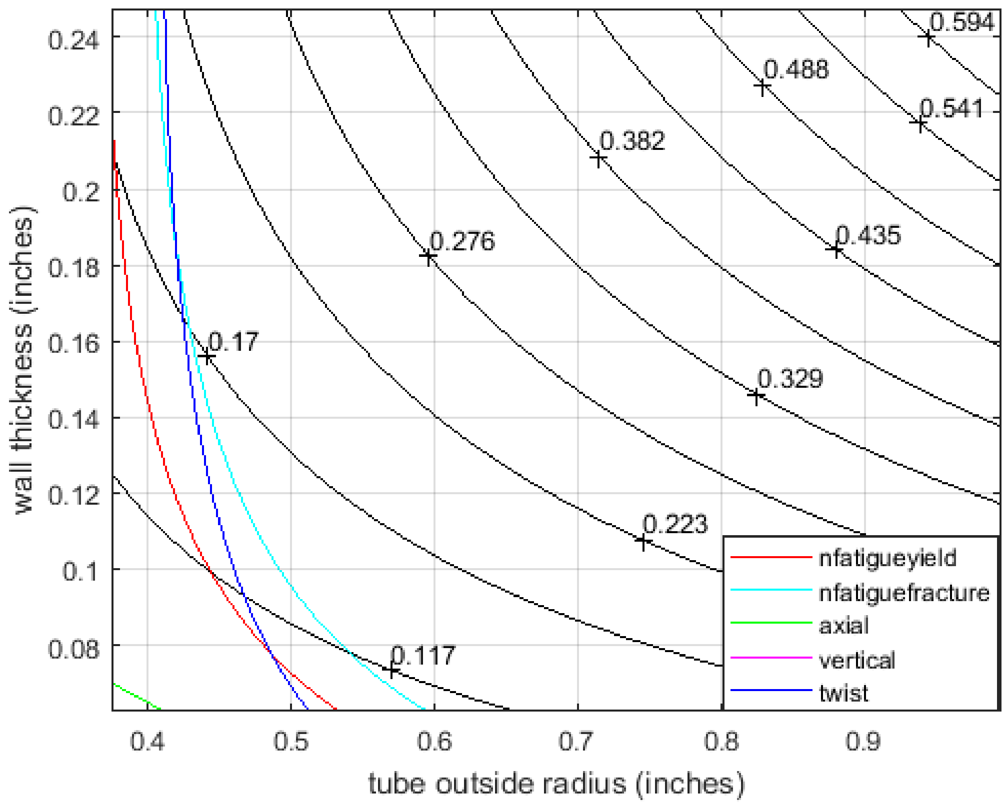


**Figure 3.2: Design Space for the Optimization Problem**

The procedure of graphical optimization includes the following 6 steps: coordinate system set-up, inequality constraint boundary plot, identification of the feasible region for an inequality, identification of the feasible region, plotting of objective function contours and identification of the optimum solution.

**Step 1: Coordinate system set-up:** The boundaries of the design space are set up based on the upper bounds and lower bounds of the design variables.

**Step 2: Inequality constraint boundary plot:** To represent the constraint graphically, the constraints' boundaries, which consist of the points that satisfy the constraints as an equality, are plotted.

**Step 3: Identification of the feasible region for an inequality constraint:** To accomplish this step, points on either side of the constraints are selected to evaluate the constraint functions. The feasible region for safety factor for fatigue fracture constraint $g_2(x)$ is shown in Figure 3.3. At point A, the $g_2(x)$ constraint function value is less than 0, the variables at point A satisfy the constraint. The region where the point lies on is the feasible region, the opposite side of the constraint line is the infeasible region for this constraint.

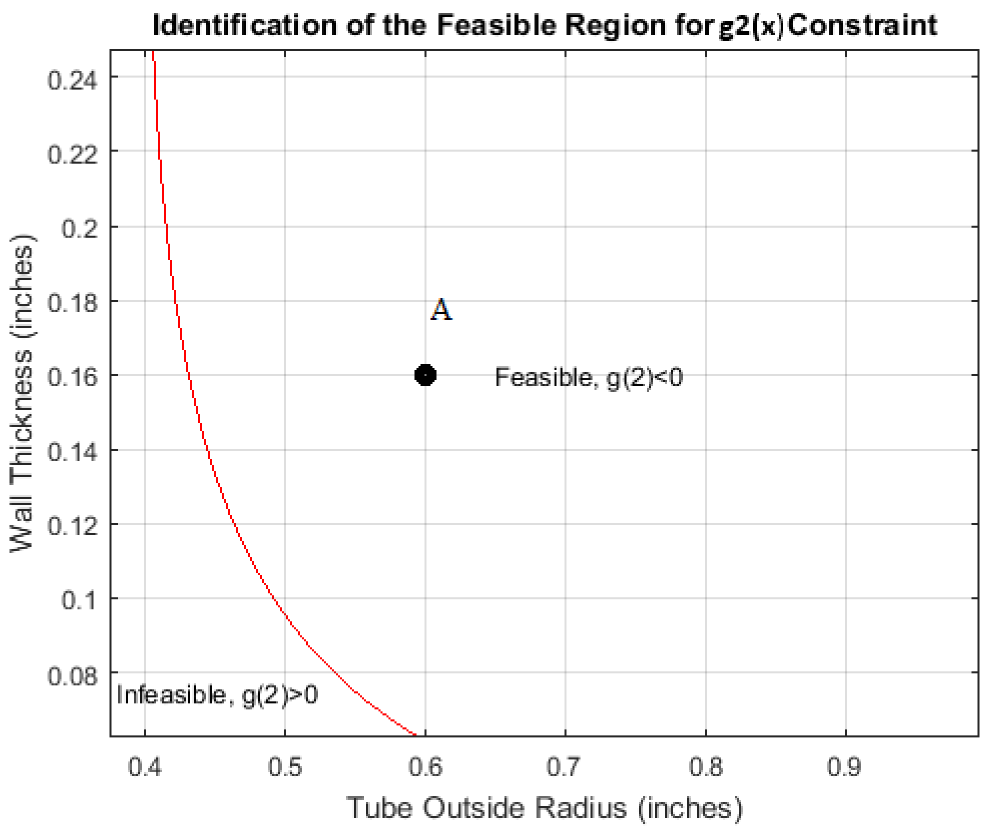


**Figure 3.3: Feasible Region for $g_2(x)$ Constraint Function**

**Step 4: Identification of the feasible region:** The fourth step is to identify the feasible region for the optimization problem. The intersection of feasible regions for all constraints is the feasible region for the optimization problem.

**Step 5: Plotting the contours of the objective function:** The objective function contours are the curves that connect all points having the same objective function value.

**Step 6: Identification of the optimum solution:** To locate the optimum point for the objective function, several contours that pass through the feasible region are plotted. The points in the region or on the feasible region's boundary provides a feasible solution for the problem. Since the objective of this project is to minimize the weight of the hollow tube, the best solution point locates at the intersection of the feasible region and the contour with the minimum objective function value.

### 3.2.2 Beam Theory Hand Calculation

The Euler-Bernoulli beam theory is used in the design analysis of a wide range of structures. In order to find an approximate solution for the objective function, the beam theory formulas for stresses and deflections are used to obtain the constraints. The equations used in the project are introduced below, the detailed explanation and derivation of the equations can be found in [24] and [25].

The assumption of a uniform axial stress distribution requires that the tube should be straight and of a homogeneous material and the line of action of the force contains the centroid of the section. As shown in Figure 3.1, the axial load $P$ is applied at the center of the cross section at the end of the hollow tube. The axial stress $\sigma_{axial}$ is calculated from the equation

$$\sigma_{axial} = \frac{P}{A} \tag{3.21}$$

The axial deflection caused by the axial stress is given by

$$\delta_{axial} = \frac{PL}{AE} \tag{3.22}$$

The axial deflection equation does not apply if there is a possibility of buckling. In general, the tube is considered a short tube if the slenderness ratio does not exceed 50. The slenderness ratio is $\frac{L}{k}$, where the radius of gyration is $k = \sqrt{\frac{I}{A}}$. The maximum slenderness ratio for this problem is less than 50, so that the axial deflection equation is appropriate to use for this problem. Additionally, the hollow tube only experience tension in this problem, hence, buckling will not occur on the tube under the defined loading.

A downward transverse load $F$ which is applied at the end of the hollow tube generates deformation and tension on the tube. Therefore, the bending moment $M$ is created by the transverse load $F$. The bending moment $M$ for this cantilever post is given by

$$M = FL \tag{3.23}$$

The maximum bending stress $\sigma_{bend}$ is a consequence of the bending moment $M$, and $\sigma_{bend}$ varies linearly with the distance from the neutral axis, $c$. The bending stress is given by

$$\sigma_{bend} = \frac{Mc}{I} \tag{3.24}$$

where $I$ is the area moment of inertia about the axis through the centroid of the cross-sectional area. The area moment of inertia for a hollow cylindrical section can be calculated as

$$I = \frac{\pi\,(r_o^4 - (r_o - t)^4)}{4} \tag{3.25}$$

The maximum magnitude of $\sigma_{bend}$ will occur where $c$ has the greatest magnitude, thus, for the hollow cylindrical section the maximum magnitude of $c$ is the outside radius $r_o$. Equation 3.24 is often written as

$$\sigma_{bend} = \frac{M}{Z} \tag{3.26}$$

where $Z = \frac{I}{c}$ is called the section modulus.

The vertical deflection $\delta_{vertical}$ caused by the transverse force $F$ applied to this cantilever post is given by

$$\delta_{vertical} = \frac{FL^3}{3EI} \tag{3.27}$$

The torsional load $T$ develops a shear stress throughout the cross section, and the moment generated by $T$ causes the mechanical element of the tube to be twisted about the x-axis. The polar second moment of area for the hollow cylindrical cross section is $J$. The shear stress is given as

$$\tau = \frac{Tc}{J} = \frac{T}{2Z} \tag{3.28}$$

The angle of twist in radians for the hollow tube is

$$\theta = \frac{LT}{GJ} \tag{3.29}$$

Rotational degrees of freedom are expressed in radians in Abaqus. The maximum allowable twist deflection angle was converted from degrees to radians since the constraint of twist deflection was formulated using degrees as units. The angle of twist was converted as

$$\theta_{degree} = \frac{57.2958LT}{GJ} \tag{3.30}$$

In order to calculate the factor of safety for fatigue fracture and fatigue yield, the fluctuating forces applying on the tube were characterized into the steady component and the alternating component. The midrange steady component and the alternating component were constructed as follows:

$$F_m = \frac{F_{max} + F_{min}}{2} \tag{3.31}$$

$$F_a = \frac{|F_{max} - F_{min}|}{2} \tag{3.32}$$

The midrange component and alternating component of fluctuating loads can be constructed by using the Equation 3.31 and Equation 3.32, the results are shown in Table 3-3.

**Table 3-3: The Components Loads of the Fluctuating Loads Applying on the Hollow Tube**

| Loads | Maximum value | Minimum value | Midrange steady component | Alternating component |
|---|---|---|---|---|
| Axial Load P (lbf) | 1680 | 950 | 1315 | 365 |
| Transverse Force F (lbf) | -240 | -45 | -142.5 | -97.5 |
| Torsional Load T (in-lbf) | -660 | -280 | -470 | -190 |

The relevant components of stress are computed as

$$\sigma_{axiala} = K_{fAxial}\frac{P_a}{A}, \sigma_{axialm} = K_{fAxial}\frac{P_m}{A}$$

$$\sigma_{benda} = K_{fBending}\frac{F_a L}{Z}, \sigma_{bendm} = K_{fBending}\frac{F_m L}{Z} \tag{3.33}$$

$$\tau_a = K_{fsTorsion}\frac{T_a c}{J}, \tau_m = K_{fsTorsion}\frac{T_m c}{J}$$

where $K_{fBending}$ denotes the fatigue stress-concentration factor for bending stress, $K_{fAxial}$ denotes the fatigue stress-concentration factor for axial stress, and $K_{fsTorsion}$ is the fatigue stress-concentration factor for torsional stress. Assuming that the hollow tube is perfectly constructed, and there is no notch on the tube or stress concentration at the wall, the stress concentration factors are taken as $K_{fAxial} = K_{fBending} = K_{fsTorsion}$= 1 as a consequence.

Since the type of loads applied to the cantilever post result in a multiaxial stress state with alternating and midrange values, the von Mises stress method was used to combine the multiple stresses into a single equivalent effective stress when determining the endurance limit. The von Mises stress for a biaxial state of stress is given as

$$\sigma' = \sqrt{{\sigma_x}^2 + 3{\tau_{xy}}^2} \tag{3.34}$$

Considering that the bending, torsional, and axial stresses have alternating and midrange components, the alternating and mean von Mises stresses were written as

$$\sigma'_a = \sqrt{(\sigma_{axiala} + \sigma_{benda})^2 + 3(\tau_a)^2} \tag{3.35}$$

$$\sigma'_m = \sqrt{(\sigma_{axialm} + \sigma_{bendm})^2 + 3(\tau_m)^2} \tag{3.36}$$

The modified Goodman equation and Goodman line were used to estimate the overall factor of safety for fatigue fracture. The Yield line, also known as the Langer first cycle yielding line, was used to estimate the factor of safety for fatigue yield. Figure 3.4 shows the various criteria of failure. The Goodman Fracture line is highlighted in blue, and the Yield line is highlighted in red. $S_a$ and $S_m$ represent the limit values of the alternating and midrange stresses. For the case of proportional overload, the limit values of mean and alternating stress can be replaced by $n\sigma_a$ and $n\sigma_m$, where n is the factor of safety.

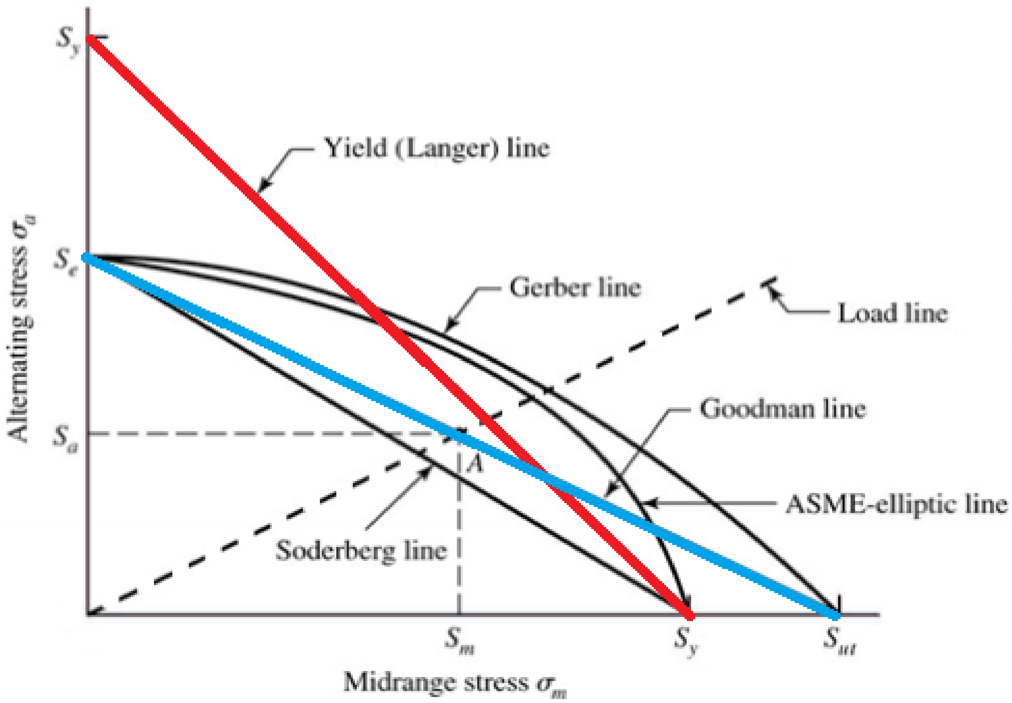


**Figure 3.4: Fatigue Diagram Showing Various Criteria of Failure. Reprinted from Shigley's Mechanical Engineering Design (p.305) by Budynas. R., 2014, MCGRAW-HILL EDUCATION**

The criterion for the modified Goodman line becomes

$$\frac{\sigma_a'}{S_f} + \frac{\sigma_m'}{S_{ut}} = \frac{1}{n_{FatigueFracture}} \tag{3.37}$$

The Langer static yield criterion for a 1D state of stress used in connection with the fatigue curve in the $\sigma_m - \sigma_a$ plane is given as

$$\frac{\sigma_a' + \sigma_m'}{S_y} = \frac{1}{n_{FatigueYield}} \tag{3.38}$$

For first-cycle yielding, Equation 3.38 could be written as Equation 3.40, since the sum of $\sigma_a'$ and $\sigma_m'$ is always greater than or equal to $\sigma_{max}'$ and will be conservative [25]. Thus, the maximum von Mises stress and the factor of safety for yield were written as

$$\sigma_{max}' = \left[(\sigma_{axialmax} + \sigma_{bendingmax})^2 + 3(\tau_{max})^2\right]^{\frac{1}{2}} \tag{3.39}$$

$$n_{FatigueYield} = \frac{S_y}{\sigma'_{max}} \tag{3.40}$$

Substituting the expressions 3.22, 3.27, 3.30, 3.37, and 3.40 into the constraint functions 3.14-3.18 with the information in Table 3-2, the constraints were defined as

$$1.1 \leq \frac{S_y}{\sigma'_{max}} \tag{3.41}$$

$$1.2 \leq \frac{1}{\left(\frac{\sigma_a'}{S_f}+\frac{\sigma_m'}{S_{ut}}\right)} \tag{3.42}$$

$$\frac{PL}{AE} \leq 0.005 \tag{3.43}$$

$$\frac{FL^3}{3EI} \leq 0.1 \tag{3.44}$$

$$\frac{57.2958LT}{GJ} \leq 1 \tag{3.45}$$

$$x_1^L \leq x_1 \leq x_1^U \tag{3.46}$$

$$x_2^L \leq x_2 \leq x_2^U \tag{3.47}$$

The constraints were normalized with respect to their limit values, the constraint functions were transformed as following:

Safety Factor for Fatigue Yield Constraint:

$$g_1(x) = 1 - \frac{S_y}{1.1\,\sigma'_{max}} \leq 0 \tag{3.48}$$

Safety Factor for Fatigue Fracture Constraint:

$$g_2(x) = 1 - \frac{1}{1.2(\frac{\sigma'_a}{S_f}+\frac{\sigma'_m}{S_{ut}})} \leq 0 \tag{3.49}$$

Axial Deflection Constraint:

$$g_3(x) = \frac{PL}{0.005AE}\text{-}1 \leq 0 \tag{3.50}$$

Vertical Deflection Constraint:

$$g_4(x) = \frac{FL^3}{0.3EI}\text{-}1 \leq 0 \tag{3.51}$$

Twist Deflection Constraint:

$$g_5(x) = \frac{57.2958LT}{GJ} - 1 \leq 0 \tag{3.52}$$

Figure 3.5 to 3.9 characterize a representative run for constraints functions. To provide analytical estimates of the constraint functions, the constraint functions are plotted as reference surfaces where the values of the constraint functions represent the z-coordinates.

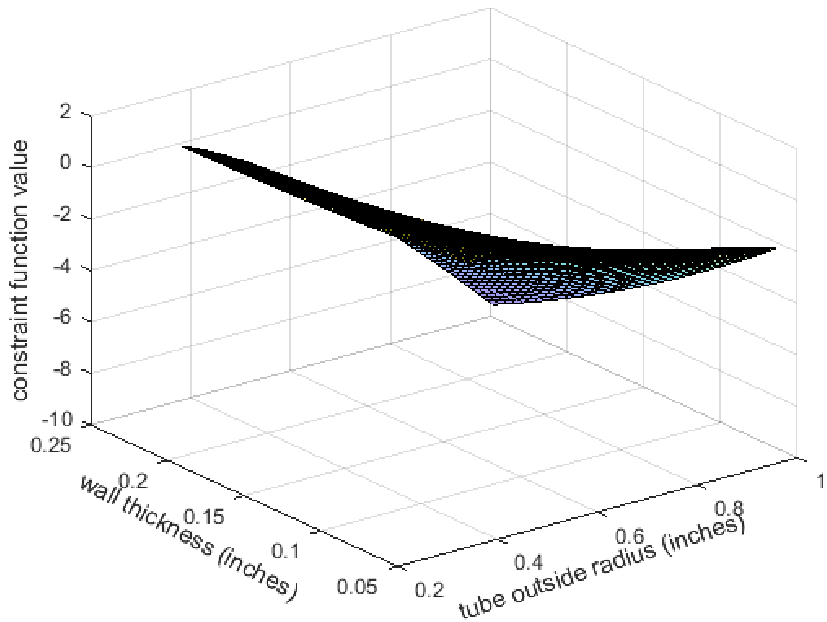


**Figure 3.5: The *$g_1(x)$* Safety Factor for Fatigue Yield Constraint Function Value**

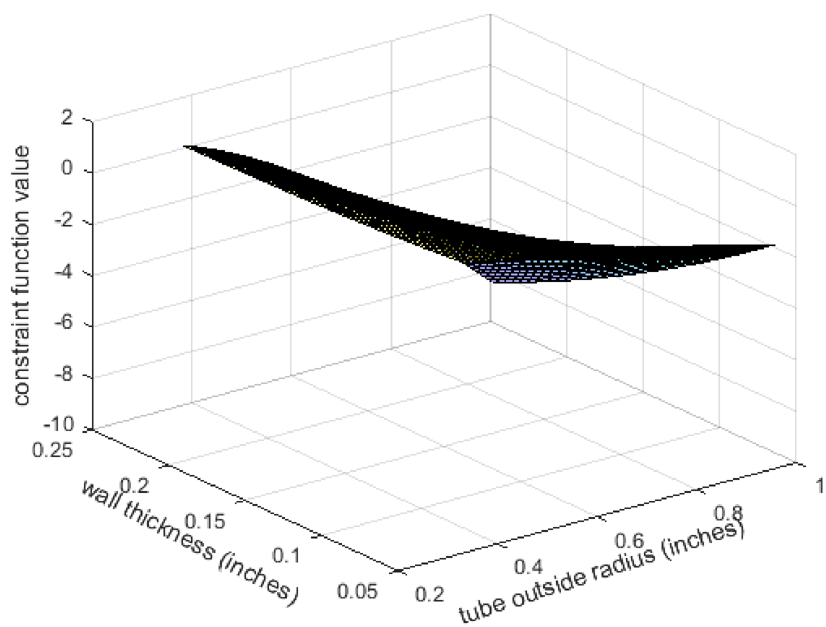


**Figure 3.6: The *$g_2(x)$* Safety Factor for Fatigue Fracture Constraint Function Value**

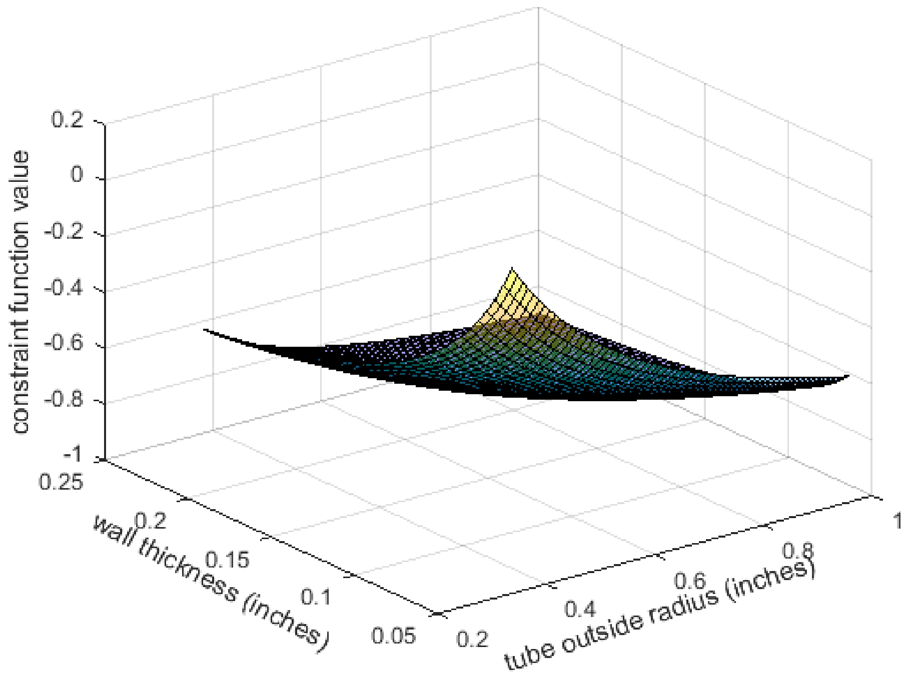


**Figure 3.7: The *$g_3(x)$* Axial Deflection Constraint Function Value**

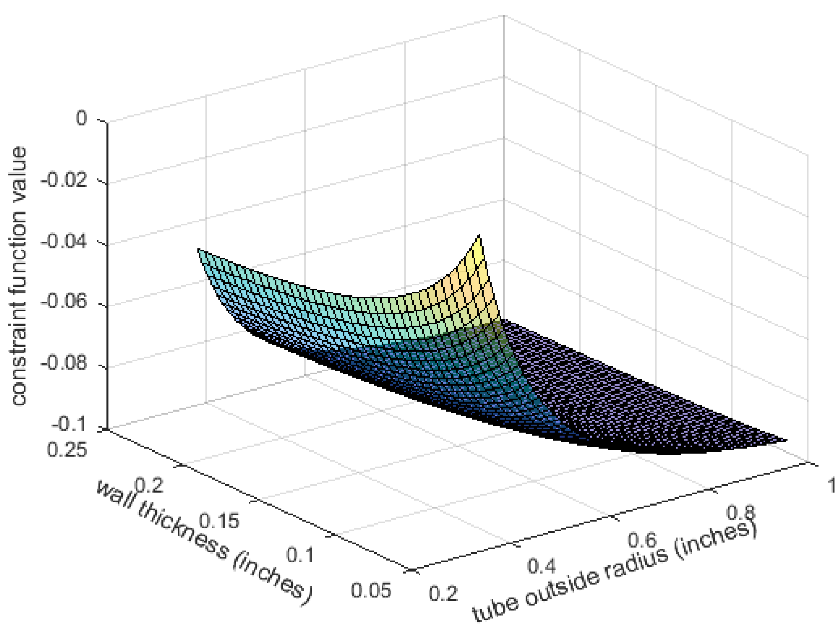


**Figure 3.8: The *$g_4(x)$* Vertical Deflection Constraint Function Value**

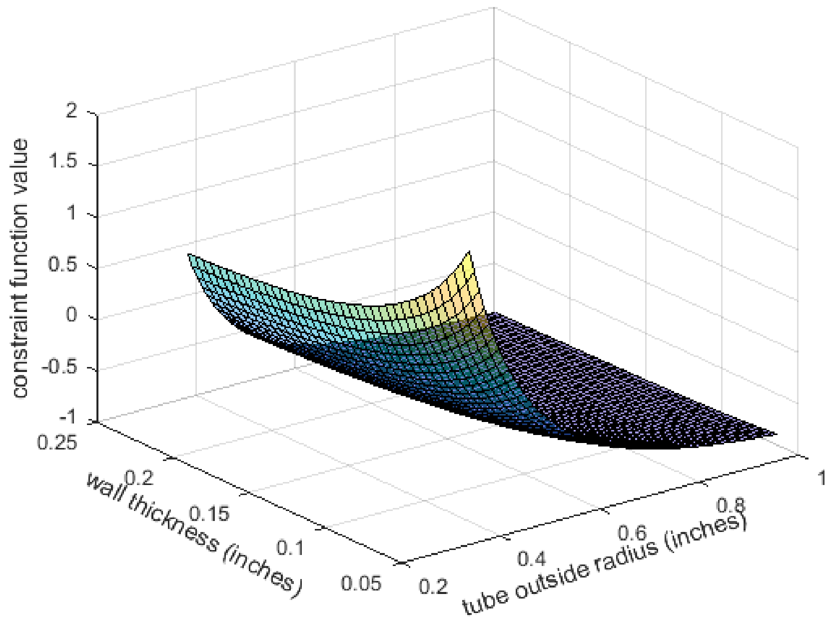


**Figure 3.9: The $g_5(x)$ Twist Deflection Constraint Function Value**

### 3.2.3 Graphical Optimization Results

To demonstrate the best solution of the optimization problem by graphical optimization method, the calculation and plotting for the objective and constraint functions were implemented in MATLAB. The accuracy of the graphical optimization is closely related to the size of the scale of x and y axis. The size for each unit grid of the x-axis was set as 0.01 with the range from 0.375 to 1. The size for each unit grid of the y-axis was set as 0.005 with the range from 0.0625 to 0.25. By using the foregoing procedure, all constraint functions contours values were set to a level of 0. Objective function contour levels for the problem are shown in Figure 3.10. The contours were plotted by using the *contour* command in MATLAB. The constraint functions were named *nfatigueyield*, *nfatiguefracture*, *axial*, *vertical*, and *twist*. The colors of the constraint functions plots were specified in red, cyan, green, maroon, and blue as shown in the legend of Figure 3.10.

Since the absolute values of the actual vertical deflection are less than the allowable vertical deflection, $g_4(x)$ constraint functions contour at a level of 0 does not appear on the design space. Thus, this constraint could be violated during the optimization solution. The values of 10 contours of objective functions are 0.594, 0.541, 0.488, 0.435, 0.382, 0.329, 0.276, 0.223, 0.17, and 0.117, and the direction of decrease for the objective function is going towards the original point of the coordinate system of the design space. The points marked as A, B, and C on Figure 3.10 are the boundary points of the feasible region. A is the intersection point of twist deflection constraint 0 level line and the upper bound of wall thickness variables, B is the intersection point of the line of twist deflection constraint and the line of safety factor for fatigue fracture constraint, and C is the intersection point of the lower bound of wall thickness variable and the line of safety factor of fatigue fracture constraint. To determine the feasible regions for each constraint, the point (0.375, 0.0625) is selected to evaluate the constraint functions. At this point, the constraint function values are greater than 0, the region on the right side of constraint function 0 level contour is feasible for each constraint function. Therefore, the intersection of feasible regions for all constraints provides the feasible region for the weight minimization problem, indicated as the region on the right side of boundary A-B-C. Thus, the potential optimum solution is on the boundary A-B-C. Since the objective function value at point C is the lowest on the boundary A-B-C, the optimum solution is at point C. In general, the optimum solution solved by using the graphical optimization method is determined by visual inspection. The optimum solution is near the point (0.6, 0.0625) as indicated in Figure 3.10. In order to get a solution to the problem to within a tolerance of 1%, the *fzero* command in MATLAB was used to numerically solve the system of nonlinear equations. The constraint of safety factor for fatigue fracture was active, thus, the x-coordinate of point C can be solved by setting the wall thickness to 0.0625 as well as the active constraint function value is equal to 0. The result solved by *fzero* is (0.5953, 0.0625), the optimum weight is 0.1057 lbs. The optimum weight found by the visual inspection is 0.1066 lbs., the percent difference between the solutions solved by visual inspection and by the graphical optimization method with using MATLAB *fzero* solver is 0.85%.

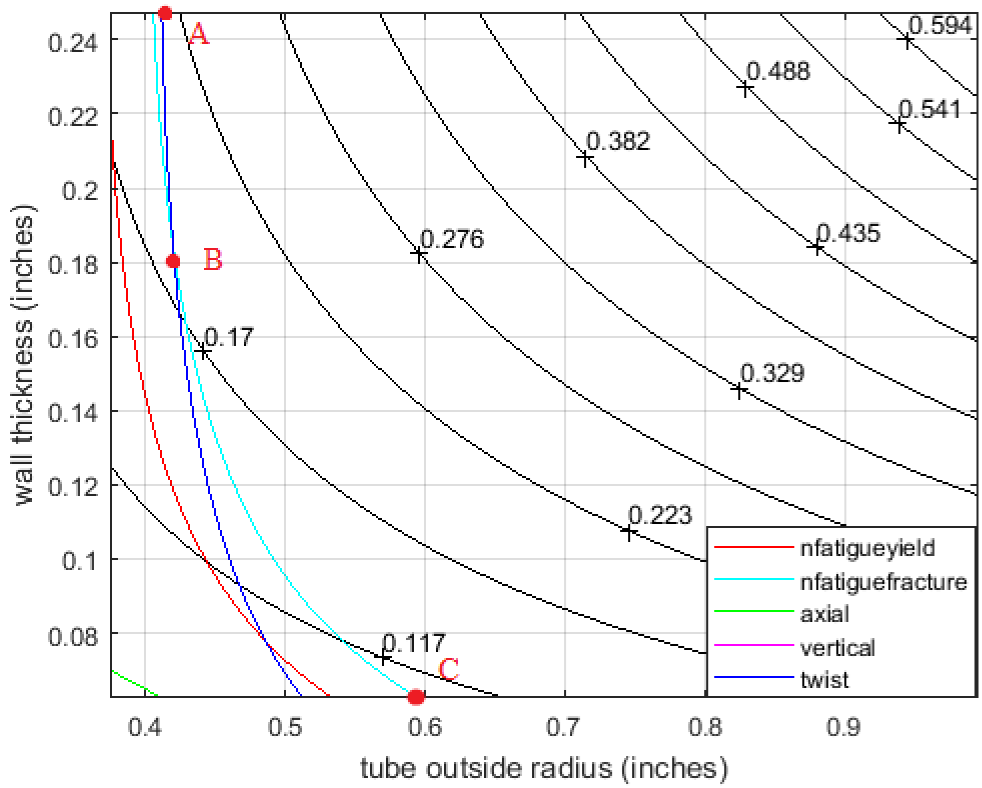


**Figure 3.10: Design Space for the Optimization Problem**

# Chapter 4

# Gradient-based optimization method combined with beam element model

This chapter represents an approach combined with a gradient-based optimization method and 3D beam finite element model for solving the structural optimization problem. An interface that enable the user to create finite element models through the finite element solver, Abaqus and process the results of FEM through the optimization toolbox in MATLAB is presented and discussed.

## 4.1 Numerical Optimization method

### 4.1.1 Sequential Quadratic Programming concept overview

In this project, since the objective function and the nonlinear constraints functions are smooth with two continuous variables, a gradient-based algorithm is available to solve this problem. Sequential Quadratic Programming (SQP) is an effective gradient-based method for solving constrained nonlinear optimization problems with single objective function and two design variables. Thus, the SQP method is appropriate for the weight minimization problem. A description and explanation of the SQP method can be found from in [1]. The SQP method implements the iterative concepts in several steps:

**Step 1.** Initialize the design: Set $k = 0$ at the initial guess point $x^{(k)}$, initialize the Hessian matrix $H$ to identity matrix $I$.

**Step 2**: Evaluate the objective function $f$ and constraint functions, $g$, at current values of the design variables $x^{(k)}$ at iteration $k$.

**Step 3:** Compute the gradient of the functions $\nabla f, \quad \nabla g$ at $x^{(k)}$

**Step 4:** Estimate the Lagrange multipliers

**Step 5:** Once the Lagrange multipliers are calculated, check convergence by observing if the norm of the gradient of the Lagrangian is less than the tolerance $\varepsilon$ where $\varepsilon$ is close to 0 as well as check if the values of the constraints functions $g$ are less than 0, the condition is written as

$$\|\nabla L\| < \varepsilon$$

$$g \leq 0$$

If the condition is true, the process will stop. If the condition is false, the optimization will proceed to next step.

**Step 6:** Solve the Quadratic Programming (QP) subproblem for the search direction. The QP sub problem is defined as

Minimize: $$\bar{f} = c^T d + \frac{1}{2} d^T d \tag{4.1}$$

Subject to: $$A^T d - b \leq 0 \tag{4.2}$$

where $c$ is the gradient vector of objective function, $b$ is the equality constraint value, $A$ is the gradient coefficient matrix for the inequality constraint, and $d$ is the search direction.

**Step 7:** if the method does not converge, perform a line search for the step size that satisfies fraction of Cauchy Decrease (FCD) condition. The condition is given as

$$f(\alpha) \leq f(0) + \alpha\rho f'(0) \tag{4.3}$$

where $\rho$ is a fixed number between 0 and 1, and $\alpha$ is the step size. Armijo Bisection is chosen to do line search since this method works well for SQP.

**Step 8:** update the design variables as

$$x^{(k+1)} = x^{(k)} + \alpha_k d^{(k)} \tag{4.4}$$

**Step 9:** update the Hessian matrix as

$$H^{(k+1)} = H^{(k)} + B + C \tag{4.5}$$

where $B$ and $C$ are updated Hessian matrixes. Step 9 is only available for the SQP method.

**Step 10:** set $k = k+1$ and go to step 3.

Once convergence is obtained when the norm of the gradient of the Lagrangian is less than the tolerance after a number of SQP loops, and the constraints and complementarity conditions are satisfied, the optimum point is achieved.

### 4.1.2 Optimization Toolbox Embedded in MATLAB Interfaced with Abaqus

Based on the documentation provided by MathWorks®, the optimizer embedded in MATLAB named Optimization Toolbox™ provides functions for finding parameters that minimize or maximize objectives functions while satisfying constraints [26]. In order to utilize the constrained nonlinear minimization algorithms, the *fmincon* command was used. For this project, the SQP implementation was selected to solve the optimization problem. As a finite element solver, Abaqus was used for creating the finite element hollow tube model and performing the finite element analysis. To integrate the finite element analysis with the constrained nonlinear optimization process, Abaqus was interfaced with the SQP solver in MATLAB. Figure 4.1 shows the approach which is able to combine Abaqus and MATLAB to solve the optimization problems.

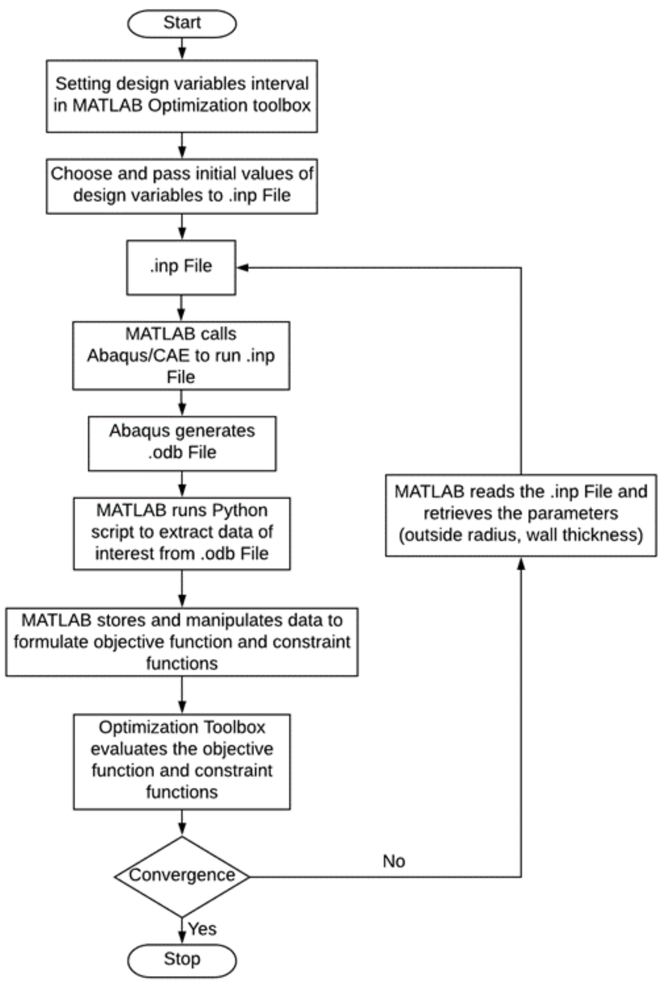


**Figure 4.1**: **Procedures of the Approach which combines Abaqus and MATLAB to solve Structural Optimization Problems**

In the .inp file, the design variables are defined as element section property parameters. The finite element analysis (FEA) is executed based on the information from the .inp file. As the FEA is completed by Abaqus, the analysis results will be written to the .odb files including the original and restart output database files. Python scripts are written to extract data from the .odb file and write the required result data to a .csv file. In order to perform the optimization processes through the optimization toolbox, the required Abaqus results in the .csv files are read and formed into a matrix in MATLAB. The optimization toolbox is responsible for checking convergence of the optimization solution, and will determine if the process is to continue or stop. If the optimization does not converge, MATLAB will start the next iteration optimization loop. The program will read the input file and catch the text line that includes the variables. Then, the new design variables created by optimization toolbox will be written to the new .inp file and subsequently the process will restart from a new pair of geometry parameters. Once the optimization converges at the optimum parameters, MATLAB will stop the process.

## 4.2 Abaqus Beam Finite Element Model

### 4.2.1 Beam Element Model

The material selected for this beam element model is UNS A92014-T4 aluminum alloy, the material properties are listed in Table 3-1. The beam element model of the post is shown in Figure 4.2. A one element finite element model was created to adequately represent the structural response of the post. This wire-shape base-feature deformable model was created in a 3D modeling space. The profile of the cross-section shape assigned to the model was a pipe with thick-walled formulation. The pipe shape profile, was assigned to a beam-type under beam section, is shown in Figure 2.1. In the simulation stage, the analysis steps, which involves process selection, load selection, and field output request selection, are required. An Abaqus static-general type analysis step was created for applying the loads, setting up the boundary conditions, and requesting the field outputs. The torsional load, the concentrated axial load and the concentrated transverse load were applied to the end of the tube. The maximum values of the fluctuating loads were used in the initial model. Since the stresses generated by the fluctuating loads were linearly related to the applied forces, the mean and alternating stresses were able to be represented by scaling the loads. The scaling method enables the optimization analysis to proceed with the results from a single model run. The details for the scaling method will be introduced in this chapter. Since one end of the hollow tube was fixed on the wall, ENCASTRE, defined as fully build-in (all available degrees of freedom are constrained), was selected for the boundary condition. The Abaqus 2-node cubic beam in space elements, as Abaqus Element type “B33”, was used for the static analysis.

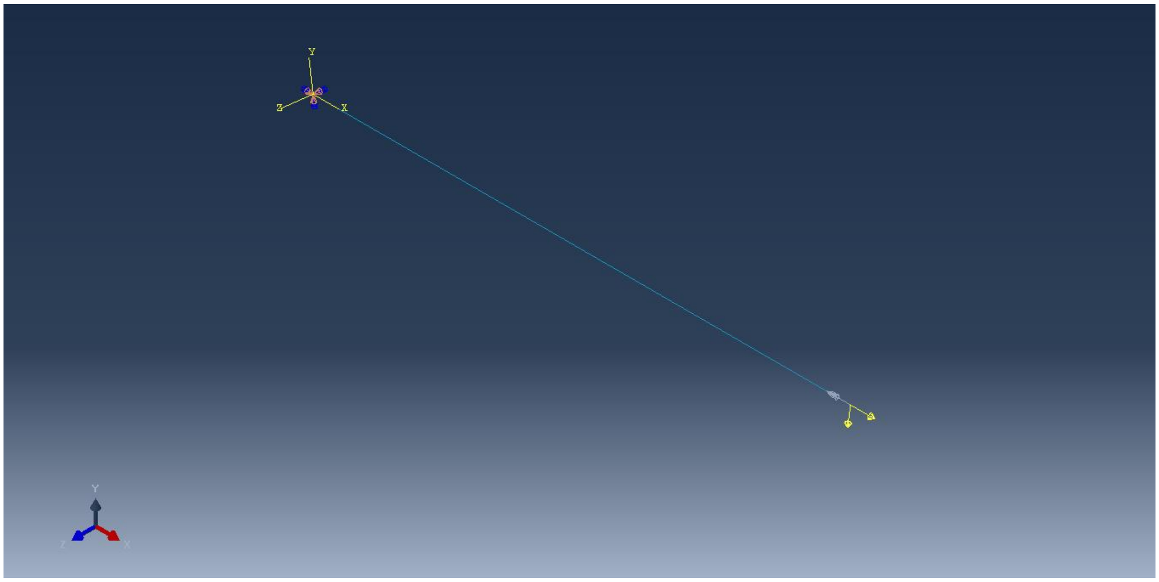

**Figure 4.2: 3D Beam Element Model with Loads and Boundary Condition**

### 4.2.2 Scaling Method

In order to simplify the optimization process, the scaling method is used for building constraint functions. Since the stresses are linear functions of the applied loads, the fluctuating loads can be represented with the maximum loads, minimum loads and scale factor $l$. By using the scaling method, only one single model run is required to represent the stresses for the max and min load cases.

The difference between the peaks values for each load are defined as

$$\Delta P = P_{max} - P_{min} \tag{4.6}$$

$$\Delta F = F_{max} - F_{min} \tag{4.7}$$

$$\Delta T = T_{max} - T_{min} \tag{4.8}$$

Using Equations 4.6 – 4.8 and the data from Table 3-2, the differences about each load are $\Delta P$=730 lbf, $\Delta F$=195 lbf, $\Delta T$=380 in-lbf. The difference values for each loads will be used to calculate the relevant stresses.

The midrange and alternating stresses are given as

$$\sigma_{bendm} = \frac{\sigma_{bendmax} + \sigma_{bendmin}}{2} \tag{4.9}$$

$$\sigma_{benda} = \frac{\sigma_{bendmax} - \sigma_{bendmin}}{2} \tag{4.10}$$

$$\sigma_{axialm} = \frac{\sigma_{axialmax} + \sigma_{axialmin}}{2} \tag{4.11}$$

$$\sigma_{axiala} = \frac{\sigma_{baxialmax} - \sigma_{axialmin}}{2} \tag{4.12}$$

$$\tau_{m} = \frac{\tau_{max} + \tau_{min}}{2} \tag{4.13}$$

$$\tau_{a} = \frac{\tau_{max} - \tau_{min}}{2} \tag{4.14}$$

where

$$\sigma_{bend} = \sigma_{bendmax} - t\Delta\sigma_{bend} \tag{4.15}$$

$$\sigma_{axial} = \sigma_{axialmax} - t\,\Delta\sigma_{axial} \tag{4.16}$$

$$\tau = \tau_{max} - t\,\Delta\tau \tag{4.17}$$

Therefore, using the parametric form for the equation of a line, the equations 4.9 - 4.14 can be transformed to

$$\sigma_{bendm} = \sigma_{bendmax} - \frac{t}{2}\Delta\sigma_{bend} \tag{4.18}$$

$$\sigma_{benda} = \frac{t}{2}\Delta\sigma_{bend} \tag{4.19}$$

$$\sigma_{axialm} = \sigma_{axialmax} - \frac{t}{2}\Delta\sigma_{axial} \tag{4.20}$$

$$\sigma_{axiala} = \frac{t}{2}\Delta\sigma_{axial} \tag{4.21}$$

$$\tau_m = \tau_{max} - \frac{t}{2}\,\Delta\tau \tag{4.22}$$

$$\tau_a = \frac{t}{2}\,\Delta\tau \tag{4.23}$$

The scale factor $t$ varies from 0 to 1. At $t = 0$, the scaled stresses are equal to the results generated by the maximum loads, at $t=1$, the scaled stresses are equal to the results generated by the minimum loads. In this project, the maximum loads are applied to the model.

### 4.2.3 Data Extraction

To calculate the values of objective and constraint functions, the axial and vertical displacement of the free end, the twist deflection of the free end, the S11, S12, and von Mises stress are extracted through a custom Python script. The Abaqus results extracted are stored in a matrix and are used to form the constraint functions in MATLAB. As shown in Figure 4.3, the maximum von Mises stress is developed at the fixed end. Due to the bending force, the critical section is located at the 90° section point, point 3 in Figure 4.4, of the pipe section at the fixed end face. At the 0° section point, point 2 in Figure 4.4, the S11 stress is the axial stress generated by the maximum axial load.

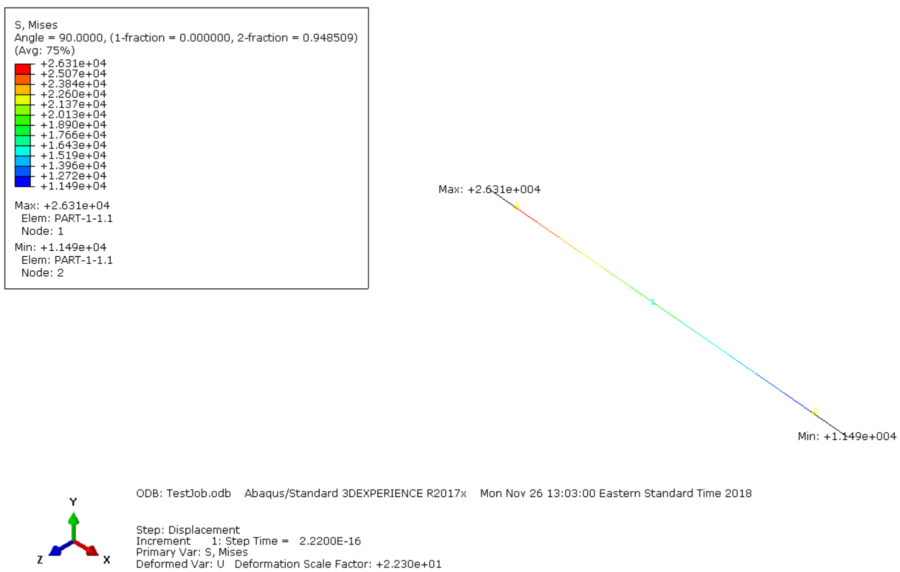


**Figure 4.3: von Mises Stress from the 3D Beam Finite Element Analysis**

The pipe section used in Abaqus is shown in Figure 4.4, the stresses extracted from the pipe section profile are located at the middle point at -90°, 0°, 90°, and 180° of the tube wall, which are marked as 1, 2, 3, and 4 in the pipe cross section of beam in space.

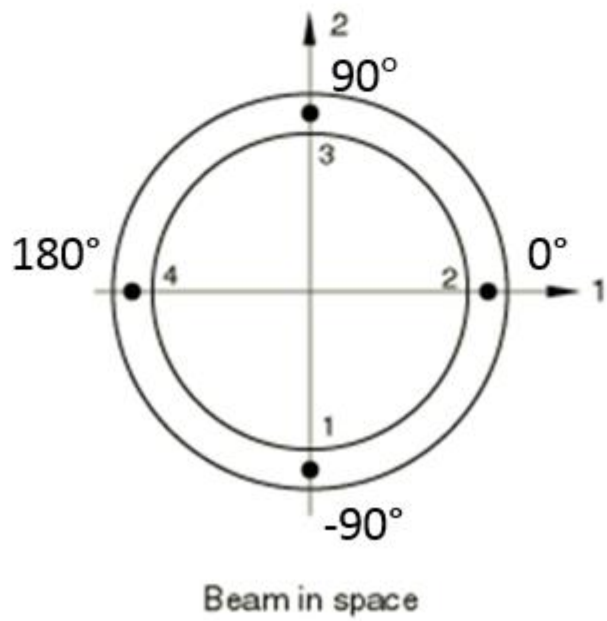


**Figure 4.4: Pipe Section used in Abaqus**

The component stress $\sigma_x$ is the combination of the bending stress $\sigma_{bend}$, and the axial stress $\sigma_{axial}$, which is $S11_{0°}$. Hence, the bending stress for the thick-walled pipe profile is calculated as

$$\sigma_{bend} = \frac{2r_o}{r_o+r_i}(S11_{90°} - S11_{0°}) \tag{4.24}$$

Therefore, $\sigma_x$ is given as

$$\sigma_x = \frac{2r_o}{r_o+r_i}(S11_{90°} - S11_{0°}) + S11_{0°} \tag{4.25}$$

In the beam finite element model, the shear stress value provided by Abaqus, $S12$, has the same value everywhere on the cross-section. Thus, $S12_{-90°}$=$S12_{0°}$=$S12_{90°}$=$S12_{180°}$ =

$S12$. The maximum shear stress due to torsion for the thick-walled pipe profile, $\tau_{xy}$, is given as

$$\tau_{xy} = \frac{2r_o}{r_o+r_i} S12 \tag{4.26}$$

## 4.3 Gradient-based Optimization Result

The initial values of the design variables were set at [ro, t] = [0.5, 0.2]. The optimization problem was solved by the SQP implementation in 27 iterations. *fmincon* stopped because the size of the current step was less than the default Step Tolerance = $1.0 \times 10^{-6}$, and the relative maximum constraint violation, 0, was less than constraint tolerance =$1.0 \times 10^{-6}$. The accuracy of the solution to 1/100th of a pound. The solution point found by the SQP algorithm was [$r_o$ = 0.6066, $t$ = 0.0625], the optimum weight is 0.1078 lbs. The optimum weight found by the graphical optimization method was 0.1057 lbs. The percent difference between the solution found by the SQP algorithm and the result solved by using graphical optimization method was 1.99%. Since the calculation of graphical optimization method was based on the Euler-Bernoulli beam theory equations, the result of graphical optimization method could be used to validate the structural optimization system for small-scale finite element models. The percent difference between the results of these two method was 1.99%, the structural optimization system for small-scale finite element models is valid.

# Chapter 5

# Response Surface Optimization combined with 3D Brick Element Model

This chapter represents a structural optimization approach combined with response surface optimization methods and the 3D continuum type finite element model. A general framework has been developed that enables the user to create solid finite element models through a Python script and manipulate the finite element results through the optimization toolbox embedded in MATLAB.

## 5.1 3D Solid Finite Element Model

### 5.1.1 Parametric Modeling

The brick finite element has more displacement degrees of freedom than the 3D beam finite element, and it is capable of resolving mechanics that the beam element is not capable of, such as stress concentrations and a 3D state of stress/strain. In order to develop the structural optimization program for the large-scale finite element models and demonstrate what the software can do, a 3D continuum type element model was created. Unlike the input file of the 3D beam finite element model, the input file of the 3D brick element model does not explicitly include the design variables used for the geometry and the section properties, since the model has to be recreated and remeshed when the section shape of the model is updated. Thus, the modeling approach used in the structural optimization program for 3D beam finite element models, which updates the model by changing the parameters in the input file, is useless in the structural optimization program for 3D continuum finite element models. Parametric finite element modeling enables the optimization program to update the finite element model automatically, and easily modifies the geometries shared by several parts in one assembly. In order to create and update the 3D brick finite element model in the structural optimization process, Python parametric expressions using numbers and numerical operations were used to define independent parameters [27]. In order to modify the parameters of the finite element model and manipulate the optimization process, users can setup the parameters in MATLAB. The parameters are passed through a Python script to run the analysis automatically during the optimization process.

### 5.1.2 Solid Finite Element Model

The material used in the 3D model is the same as that used in the hand calculations and in the 3D beam element model. The material properties were listed in Table 3-1. The applied loads were the same as the loads use in the 3D beam finite element model, and the information detailing the loads was listed in Table 3-2.

There are no elements at the center of the 3D brick tube model, since the tube is hollow. A reference point was created at the center of the free end surface. A kinematic coupling constraint between the reference point and the free end surface was defined. The loads were applied at the reference point, and the boundary conditions ENCASTRE, defined as fully build-in (all available degrees of freedom are constrained), was prescribed at the tube's fully built-in end at the wall.

After the assembly step, the model was partitioned into several regions. As shown in Figure 5.1, the assembly was divided into 4 parts to subdivide 3D regions into simpler regions. The Abaqus brick element C3D8I is an 8-noded linear continuum element with incompatible modes. C3D8I is fully integrated and it has 13 internal degrees of freedom, therefore, it has good performance when bending stresses occur in the element. Although C3D8I elements are more expensive than regular first-order displacement elements, C3D8, they are significantly more economical than the 20-noded second-order elements, C3D20. Since C3D8I accounts for the correct bending mechanics, and it balances computational expense, time, and the performance, C3D8I is the appropriate element for this model.

The meshed model with boundary conditions is shown in Figure 5.2, and the visualization of the von Mises stress contours for the 3D brick finite element model is shown in Figure 5.3. As shown in Figure 5.3, the maximum von Mises stress was developed at the top fiber of the cross section. In the 3D beam finite element model, the maximum value of the von Mises stress was developed at the 90° section point of the cross section, thus, the critical section of the 3D beam finite element model and the 3D brick finite element model were located at the same fiber.

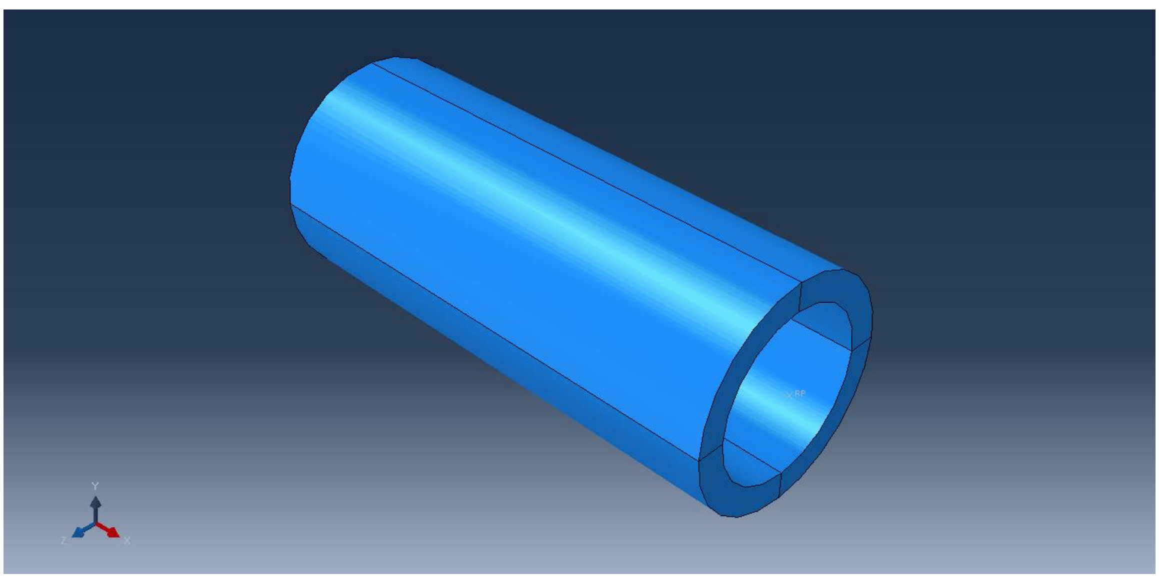

**Figure 5.1: The Assembled 3D Brick Finite Element Model with Partitions**

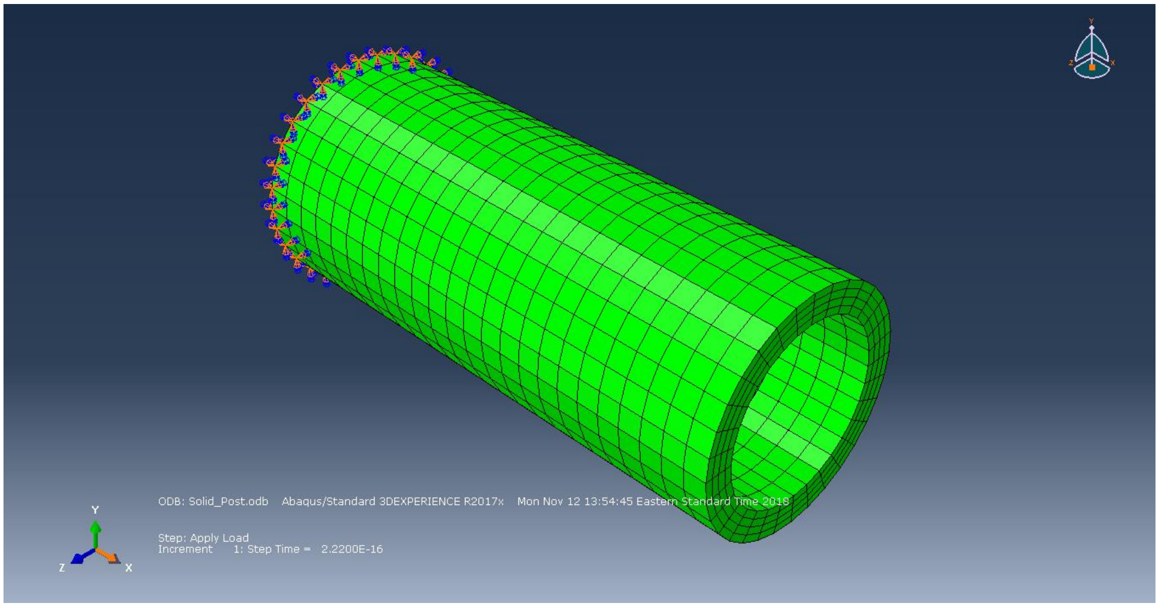


**Figure 5.2: The 3D Brick Finite Element Hollow Tube Model**

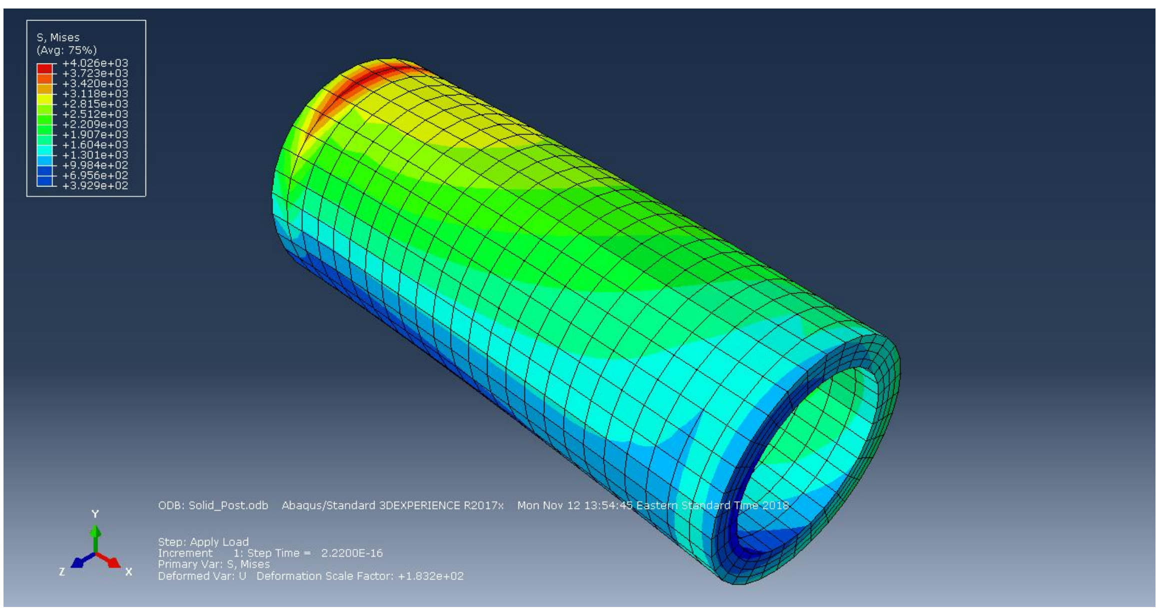


**Figure 5.3: von Mises Stress Contours for the 3D Brick Finite Element Model**

The finite element analysis results highly depend on the finite element mesh refinement. In order to refine the mesh by using an effective meshing scheme with the appropriate number of the elements, a mesh convergence study for the 3D brick model was performed. In order to get a high quality mesh, the convergence level was set within a tolerance of

0.1%. The model with design variables [$r_o$ = 0.6875, $t$ = 0.15625] was chosen to perform the mesh convergence test, since the sample point with this pair of design variables was at the center of the design space. The model was remeshed in 9 iterations with denser element distribution. The information for the convergence test is shown in Table 5-1. The Extrude/Sweep meshing method was used for meshing the hollow tube. Edge-mesh seeds were defined along the longitudinal edges, the radial edges, and the circumferential edges by entering the number of elements. In the original meshed model, 24 seeds were positioned along longitudinal edges, 4 seeds were positioned along radial edges, and 8 seeds were positioned along circumferential edges. After performing the convergence test of twist deflection, the percent difference narrowed to 0.2% at the 3$^{rd}$ iteration. The percent difference was defined as

$$\% \text{ Percent Difference} = \frac{New\ Value - Initial\ Value}{Initial\ Value} * 100$$

The minimum twist deflection at the reference point was converged to 0.06% after the 5$^{th}$ iteration. As shown in Figure 5.4, the twist deflection converged at the 7$^{th}$ iteration. Thus, the meshed assembly with 48 seeds positioned along longitudinal edges, 16 seeds positioned along radial edges, and 32 seeds positioned along circumferential edges was used to generate the response surface. It is worth mentioning that the aspect ratio, which was defined as the ratio of the longest edges to the shortest height of the element, should be limited to 5 in order to improve mesh quality.

**Table 5-1. Information for Mesh Convergence Test**

| Convergence Level (r = 0.6875, t = 0.15625) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Use the central point in the design space | | | | | | | | | |
| Case | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Longitudinal | 24 | 24 | 24 | 24 | 32 | 32 | 48 | 48 | 48 |
| Radial | 4 | 8 | 8 | 16 | 16 | 24 | 16 | 24 | 32 |
| Circumferential | 8 | 8 | 16 | 16 | 24 | 24 | 32 | 32 | 32 |
| Number of Elements | 3072 | 6144 | 12288 | 24576 | 49152 | 73728 | 98304 | 147456 | 196608 |
| Minimum tTwist Deflection UR1(radians) | -3.5206E-03 | -3.5206E-03 | -3.4868E-03 | -3.4868E-03 | -3.4806E-03 | -3.4806E-03 | -3.4785E-03 | -3.4785E-03 | -3.4785E-03 |
| Percent Difference | | 0.000% | -0.960% | 0.000% | -0.178% | 0.000% | -0.060% | -0.001% | 0.001% |

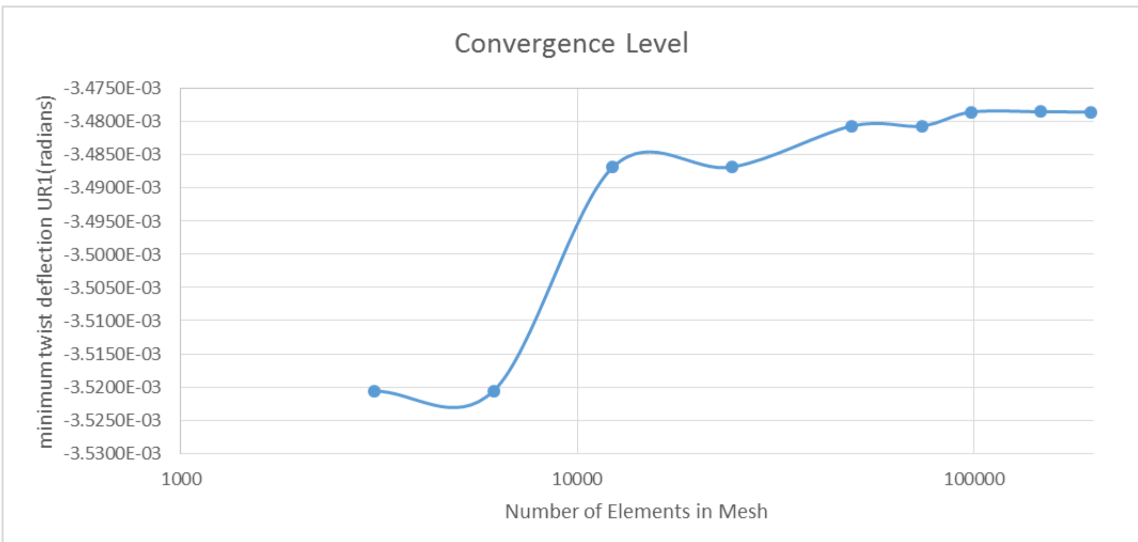


**Figure 5.4: Convergence Test for the Minimum Twist Deflection of Reference Point versus the Number of Elements**

### 5.1.3 Modeling

In the structural optimization software system for large-scale finite element models, the functional elements of finite element model creating and finite element analysis were executed by a Python script named "main.py". This Python script consists of modeling scripts, job submitting scripts, and result extraction scripts. To execute these functions in the "main.py" script, some modules, such as *abaqus*, *abaqusConstants*, *caeModulesstep*, were imported into the script. The details of the modules used in the scripts can be found in the documentation provided by *Dassault Systemes Simulia Corporation* [28]. The command "*print sys.argv*" was used to pass the parameters from MATLAB into Python. The design variables and the execute command were written in the same for-loop, so that the design variables can be passed into Python in each optimization iteration. Parametric expressions were written in the "main.py" script to convert the design variables to parameters used for modeling. The parametric expressions implement the parametric finite element modeling function for the system. In this research, a 3D deformable body model with homogeneous solid section was created to test the system. Abaqus CAE sets were created by using the *findAt* command to sort out the key regions. The Abaqus model partitions were created into cells as shown in Figure 5.1. The *Static Linear Perturbation* step was selected as the analysis step after the initial step. As mention in Section 5.1.2, a reference point was created to locate the loads. The reference point was coupled with the free end surface with a kinematic coupling type constraint. The selection of boundary conditions (BCs) and constrained degrees of freedom for coupling significantly influence on the finite element analysis. To achieve the best optimization solution, appropriate settings for BCs and coupling should be used. The mesh seeds were distributed on the partition edges by the *seedEdgeBynumber* command. Once the model was meshed and assembled, the job for FEA was generated and submitted by the job submitting module.

### 5.1.4 Data Processing

By using the extraction function in the script, the axial displacement *U1*, the vertical displacement *U2*, and the twist deflection *UR1* at the reference point were extracted from .odb files and written into .csv files. The node, which the maximum von Mises stress was located on, was considered as the critical section. Thus, the component stresses including *S11*, *S22*, *S33*, *S12*, *S13*, *S23* at this node were extracted and written to the .csv files.

To manipulate the results from Abaqus, MATLAB was used to store and form the results in a matrix named "AbaqusData". In the structural optimization program for small-scale finite element models, the programs, which evaluate the objective functions and constraint functions directly, read the results from this matrix during the optimization process. Different from the approach used in the structural optimization program for the small-scale finite element models, the results at each integration point were extracted and used for generating the response surfaces. In the structural optimization program for large-scale models, the response surfaces were used to approximate the objective and constraint functions. A MATLAB program, used to form interpolation functions, was called to generate the response surfaces. Then, the objective and constraint functions were evaluated in the MATLAB programs named "mycon" and "myfun". Considering the response surface is a continuous function, the SQP algorithm was appropriate to solve the problem. The constrained minimization solver, *fmincon*, provided the optimum solution when the size of the current step was less than the default step tolerance.

### 5.1.5 Calculations for Constraint Functions

In order to build the safety factors for fatigue yield and the for fatigue fracture constraint functions, Equation 5.1 [24] was used to calculate the octahedral normal stress, or the von Mises stress.

$$\sigma_{vonMises} = \frac{1}{3}\sqrt{(\sigma_x - \sigma_y)^2 + (\sigma_y - \sigma_z)^2 + (\sigma_z - \sigma_x)^2 + 6({\tau_{xy}}^2 + {\tau_{yz}}^2 + {\tau_{zx}}^2)} \quad (5.1)$$

Normal stresses, $\sigma_y$ and $\sigma_z$ were represented as *S22* and *S33*. Shear stresses, $\tau_{xy}$, $\tau_{yz}$, and $\tau_{zx}$ were represented as *S12*, *S23*, and *S13* respectively. The normal stress, $\sigma_x$, or *S11*, consisted of $\sigma_{axial}$ and $\sigma_{bend}$. In the 3D brick finite element model, the maximum value of *S11* was the sum of maximum $\sigma_{bend}$ and the stress value of $\sigma_{axial}$. The axial stress caused by axial load had the same values everywhere on the cross-section. The maximum value of the bending stress caused by the downward transverse force was located at the 90° section point of the cross section, and the minimum value was located at the -90° section point of the cross section. The maximum and minimum bending stresses had the same magnitudes but opposite signs. Thus, the axial stress and the bending stress are given as

$$\sigma_{axial} = \frac{S11_{90^\circ}+S11_{-90^\circ}}{2} \tag{5.2}$$

$$\sigma_{bend} = \frac{S11_{90^\circ}-S11_{-90^\circ}}{2} \tag{5.3}$$

Applying the scaling method mentioned in Section 4.2.2, the midrange and alternating component stresses were obtained.

## 5.2 Response Surface Optimization Method

### 5.2.1 Response Surface Optimization Method Overview

In order to check the structural optimization program for large-scale finite element models, a 3D brick finite element hollow tube model was created. Since the finite element analysis for a 3D continuum type finite element model is a time-consuming process, an appropriate optimization method should be selected to reduce the number of finite element analysis runs. In many practical cases, the Response Surface Methodology (RSM) can significantly reduce the number of actual analysis runs, because it requires the incremental sampling of points in the design space to obtain the approximated response surface instead of performing a massive number of experiments to build the actual function surface.

RSM was introduced and developed by Box and his colleagues from 1951 [16] [17]. It is a representative method for generating an explicit function using the information at a sequence of sample points. The response surface approximation is generated by several types of functions, such as linear, quadratic, cubic, and other functions [1]. Once the results at the sample points are obtained, the results of the objective function and constraint functions at any point in the design space can be evaluated using the functional representation of the response surfaces. The response surface are considered as continuous functions. Therefore, once the response surface composed by the approximated function values is generated, a gradient-based optimization can be performed on the response surface.

### 5.2.2 Biquadratic Interpolation Function

The biquadratic interpolation function was used to generate the response surfaces on the 2D continuous grid base composed of two design variables. The functions were developed from Lagrange interpolating polynomials, which is given as

$$P_j(x) = y_j \prod_{\substack{k=1 \\ k \neq j}}^{n} \frac{x - x_k}{x_j - x_k} \tag{5.4}$$

In this project, the sample points at the four corners of the grid base, the midpoints of grid base's four sides, and the center of the grid base were employed to build the response surface. The coordinates and the sequence numbers of the samples points are shown in Figure 5.5.

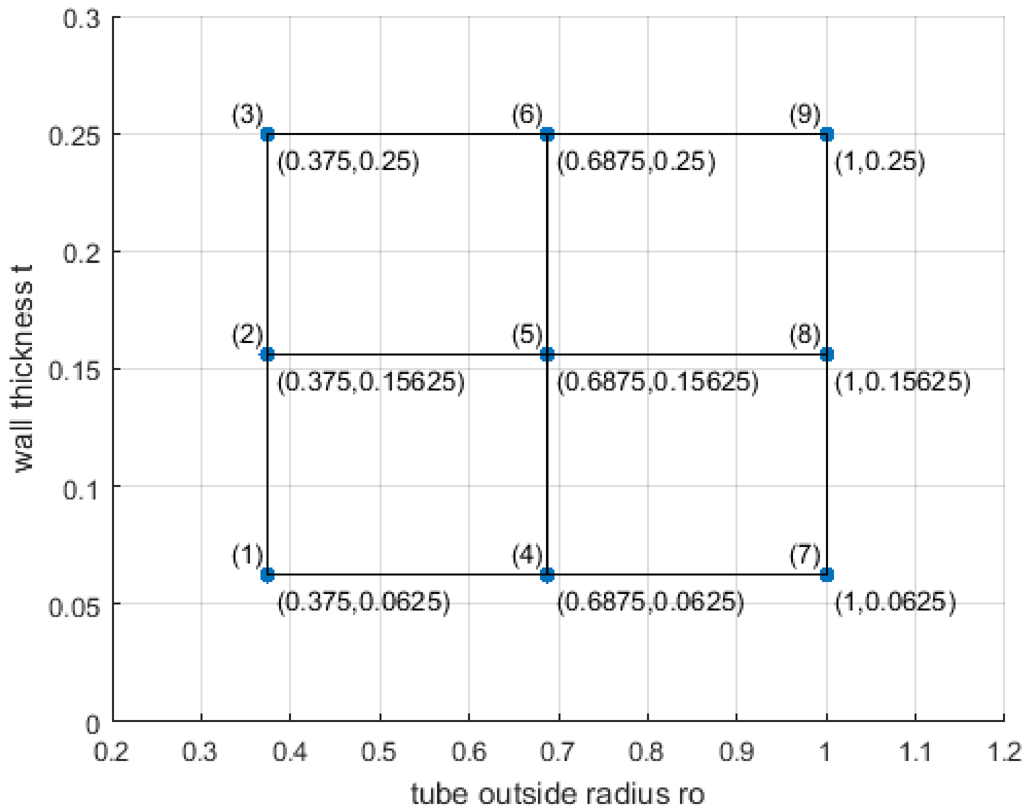


**Figure 5.5: The Coordinate and the Sequence Numbers of the Sample Points**

The biquadratic interpolation function for this problem $F(x)$ is written as

$$F(x) = [N_{11} N_{12} N_{13} N_{21} \dots N_{33}] \begin{Bmatrix} f_1 \\ f_2 \\ \vdots \\ f_9 \end{Bmatrix} \tag{5.5}$$

In Equation 5.5, the $f_k$ are the values of objection or constraint functions at the sample points, and $N_{ij}$ are the interpolation coefficients. The expression of the interpolation coefficients is shown as below:

$$N_{ij} = N_{roi} N_{tj} \tag{5.6}$$

The expressions of $N_{roi}$ and $N_{tj}$ are given in Equations 5.7 – 5.12. In the expressions 5.7 – 5.12, $r_{o1}$, $r_{o2}$, $r_{o3}$, $t_1$, $t_2$, and $t_3$ are the *x* and *y* coordinates of the selected 9 sample points.

$$Nro1 = \frac{(ro-ro2)(ro-ro3)}{(ro1-ro2)(ro1-ro3)} \tag{5.7}$$

$$Nro2 = \frac{(ro-ro1)(ro-ro3)}{(ro2-ro1)(ro2-ro3)} \tag{5.8}$$

$$Nro3 = \frac{(ro-ro1)(ro-ro2)}{(ro3-ro1)(ro3-ro2)} \tag{5.9}$$

$$Nt1 = \frac{(t-t2)(t-r3)}{(t1-t2)(t1-t3)} \tag{5.10}$$

$$Nt2 = \frac{(t-t1)(t-t3)}{(t2-t1)(t2-t3)} \tag{5.11}$$

$$Nt3 = \frac{(t-t1)(t-t2)}{(t3-t1)(t3-t2)} \tag{5.12}$$

The comparison between the analytical reference surfaces of each constraint based on the beam finite element model and the response surfaces of each constraint based on the brick element model are shown in Figure 5.6 – 5.10. The colored surfaces are response surfaces generated by using the results from the brick element models, the transparent surfaces are generated by using the calculations based on the Euler-Bernoulli beam theories. Plotting the actual surfaces of the functions and comparing them to the response surfaces will help understand the functions and give confidence in the predictive quality of response surfaces. $g_1(x)$, $g_2(x)$, $g_3(x)$, $g_4(x)$, and $g_5(x)$ were defined in Section 3.2.2.

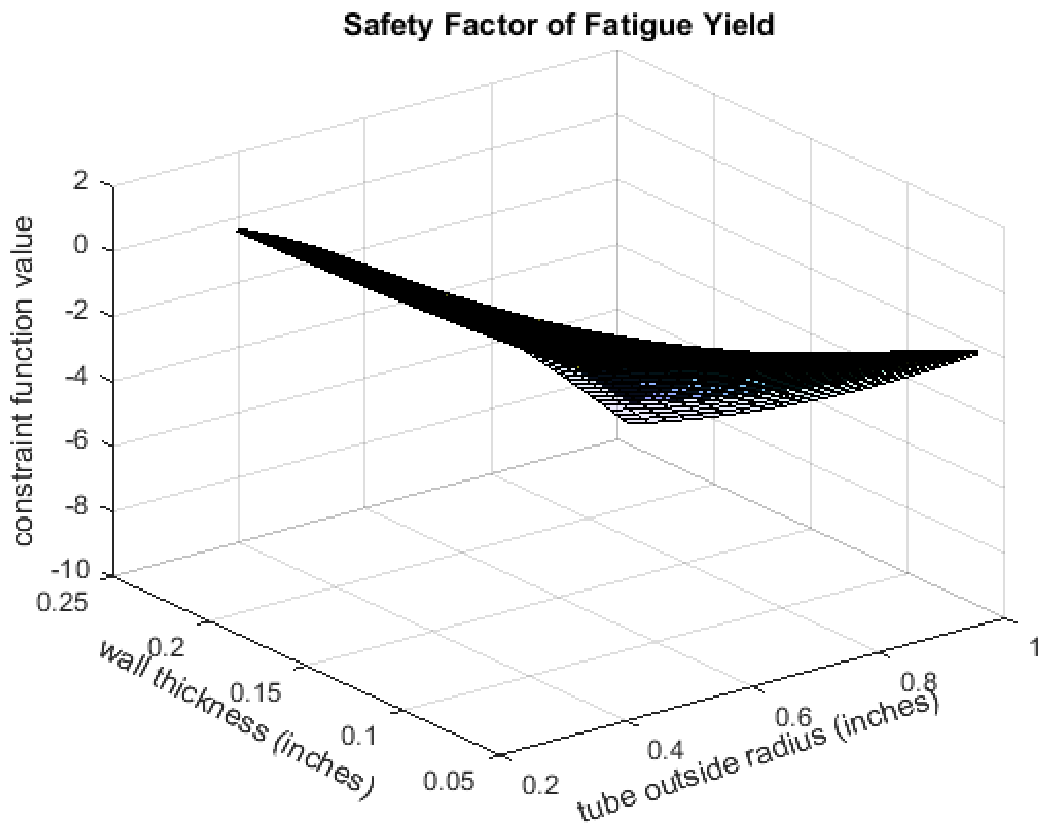


**Figure 5.6: The $g_1(x)$ Surfaces Comparison of Safety Factor for Fatigue Yield**

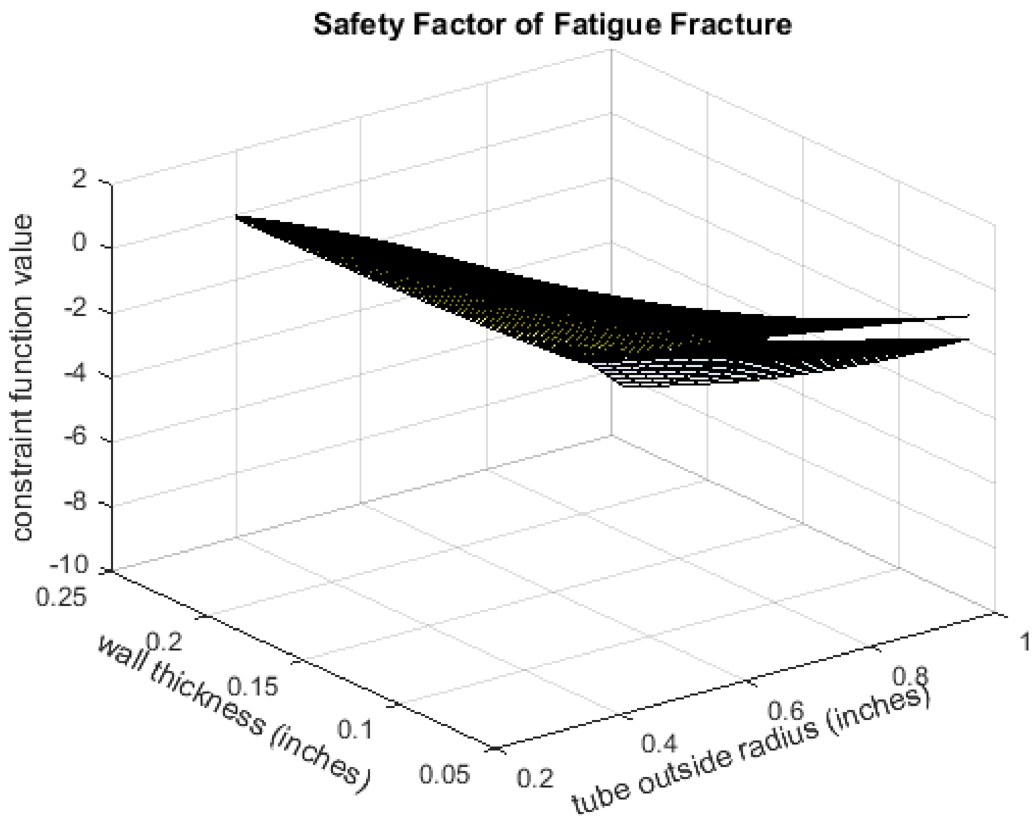


**Figure 5.7: The *$g_2(x)$* Surfaces Comparison of Safety Factor for Fatigue Fracture**

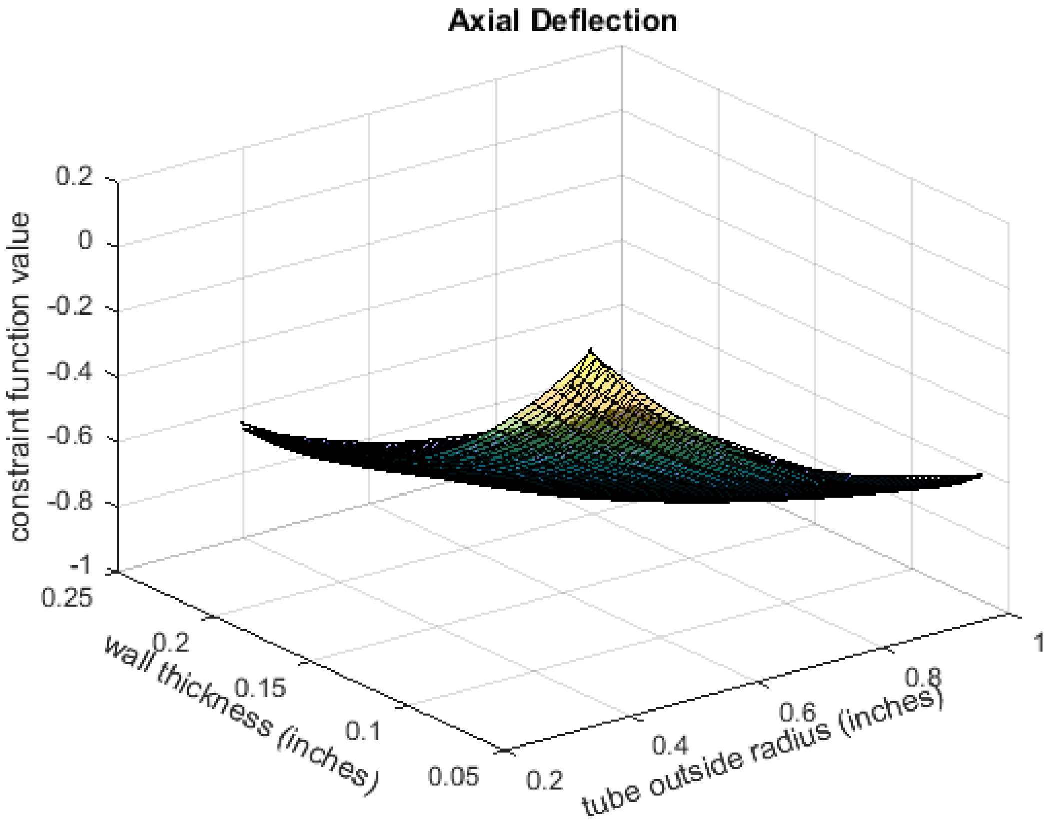


**Figure 5.8: The $g_3(x)$ Surfaces Comparison of Axial Deflection**

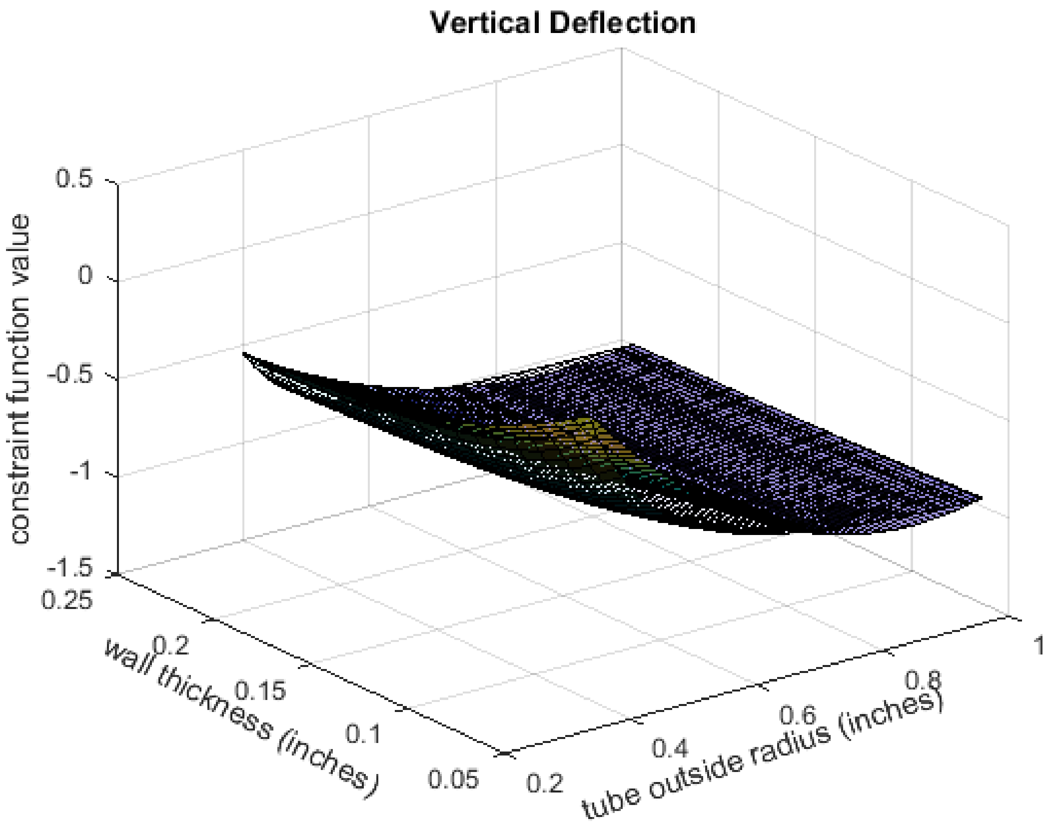


**Figure 5.9: The *$g_4(x)$* Surfaces Comparison of Vertical Deflection**

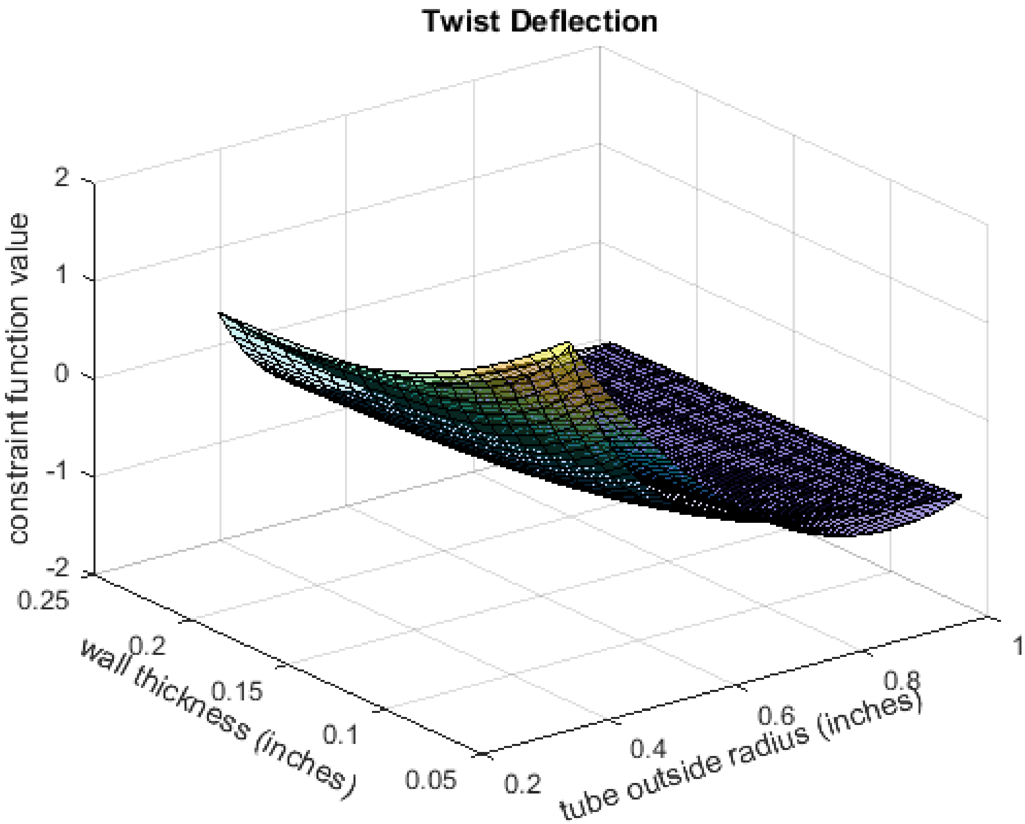


**Figure 5.10: The $g_5(x)$ Surfaces Comparison of Twist Deflection**

### 5.2.3 Analysis of DOE Data

The appropriate representation of response surfaces of the actual constraint and objective functions can provide more accurate results. The mathematical model found after fitting the function to the data may not describe the experimental domain correctly [17]. To evaluate the quality of the model fitted, the application of analysis of variance (ANOVA) was used.

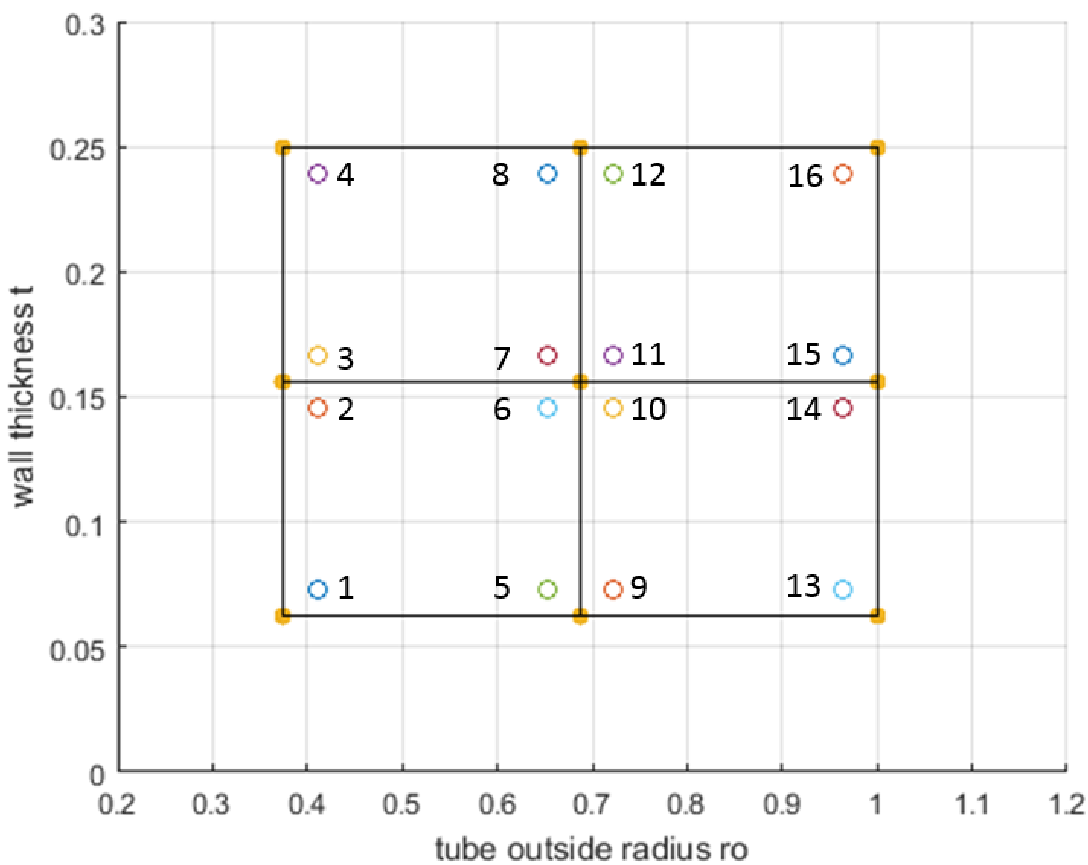


**Figure 5.11**: **The 16 Sample Points Selected to Generate the Response Surface for Regression Test**

After the response surface was generated, the regression analysis was performed to check if the response surface represented the objective function or constraint functions adequately. In this research, 16 samples points were selected to perform the test of linear regression, and the results of the objective and constraint functions at these 16 points were calculated by using the extracted data from Abaqus. As shown in Figure 5.11, the design space was divided into four parts, and the sequence numbers of the 16 samples points were marked. The 16 sample points were selected using the parameters

$$r_o = [0.4102; 0.6523; 0.7227; 0.9648],\ t = [0.0731; 0.1457; 0.1668; 0.2394]$$

The actual constraint function values at the 9 sample points to do regression analysis were defined as

$$\{f^9\} = \{f_1 f_2 \cdots f_9\}$$

The actual constraint function values at the 16 sample points to do regression analysis were defined as

$$\{f^{16}\} = \{f_1 f_2 \cdots f_{16}\}$$

To perform the regression analysis, the approximated values at the 16 origin sample points $\{\hat{f}\}$ were calculated by the equation

$$\{\hat{f}\} = N\{f^9\} \tag{5.13}$$

where $N = \left[N_{(x1,y1)} N_{(x2,y2)} N_{(x3,y3)} \cdots N_{(x16,y16)}\right]$ was the combination of interpolation coefficients for the 16 sample points.

The residual error associated with the regression is the vertical distance between the observed value and the predicted value from the regression line. The sum of squares of the residual errors of the function values at the selected sample points was solved using the least squares method. The safety factor for the fatigue fracture constraint function was chosen for linear regression test. The residuals $e$, the difference between the actual constraint function values and the approximated values at the 16 origin sample points, are given as

$$\{e\} = \left\{f^{16} - \hat{f}\right\} \tag{5.14}$$

The sum of squares of the errors *SSE* is given as represented as

$$SSE = \sum e^2 \tag{5.15}$$

If the response surface correctly represents the functions, *SSE* should be close to 0, and the estimated error variance should be small.

The estimated population variance of the residual error is given as

$$Variance = \frac{\sum e^2}{N-1} \tag{5.16}$$

The standard deviation of the residual error *S* is given as

$$S = \sqrt{Variance} \tag{5.17}$$

Total sum of squares $\bar{f}$ is the sum of squared errors of the actual values and the overall mean value of $\{f^{16}\}$, and it is given as

$$SST = \sum_{i=1}^{n}(f_i^{16} - \bar{f})^2 \tag{5.18}$$

R-squared is a statistical measure that's used to assess the goodness of fit of the response surface, R-squared is given by

$$R^2 = 1 - \frac{SSE}{SST} \tag{5.19}$$

A detailed explanation of Equation 5.14 to 5.19 can be found in [29]. The objective function, which was the mass of the tube, was used to perform the regression test. The statistical analysis results are listed in Table 5-2. The correlation coefficient R and the coefficient of determination $R^2$ is 0.9999. The SSE, mean of the residual error, and the standard deviation of the residual error are close to zero, therefore, the objective and constraint functions were represented by the response surfaces adequately.

**Table 5-2. Statistical Analysis Results**

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| **Predicted Value** | 8.1827E-02 | 1.4730E-01 | 1.6338E-01 | 2.0845E-01 | 1.3483E-01 | 2.5298E-01 | 2.8440E-01 | 3.8215E-01 |
| **Observed Value** | 8.1827E-02 | 1.4730E-01 | 1.6338E-01 | 2.0845E-01 | 1.3483E-01 | 2.5299E-01 | 2.8440E-01 | 3.8215E-01 |
| **Residual Error** $e$ | -5.3759E-08 | -1.2870E-07 | 1.2049E-07 | 4.3958E-08 | 5.2408E-07 | 2.1937E-07 | -1.547E-07 | -4.4692E-07 |
| | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** |
| **Predicted Value** | 1.5026E-01 | 2.8374E-01 | 3.1961E-01 | 4.3269E-01 | 2.0326E-01 | 3.8942E-01 | 4.4063E-01 | 6.0639E-01 |
| **Observed Value** | 1.5026E-01 | 2.8374E-01 | 3.1961E-01 | 4.3269E-01 | 2.0326E-01 | 3.8942E-01 | 4.4063E-01 | 6.0639E-01 |
| **Residual Error** $e$ | -4.9786E-07 | -4.2584E-07 | 3.6117E-07 | 4.2069E-07 | 7.8395E-08 | -6.5262E-08 | 7.3475E-08 | -6.8594E-08 |
| **SSE** | **SST** | **R^2** | **STD** | **VAR** | **Mean of** $e$ | **Mean of Observed Values** | | |
| 1.3395E-12 | 2.9978E-01 | 9.9900E-01 | 2.9883E-07 | 8.9299E-14 | -3.3827E-17 | 2.8010E-01 | | |

## 5.3 Results and Discussion

### 5.3.1 Results

The initial guess parameters were set at [$ro$, $t$] = [0.5, 0.2]. The optimization problem was solved by an SQP implementation in 15 iterations. First-order optimality is a measure of how close the point is to optimal, the SQP solver uses the relative first-order optimality as a stopping criterion [30]. *fmincon* completed because the relative first-order optimality measure, 1.2345e-07, was less than optimality tolerance = 1. $0x10^{-6}$, and the relative maximum constraint violation, 0, was less than constraint tolerance = 1. $0x10^{-6}$. The solution point found by the optimization toolbox in MATLAB was [$r_o$ = 0.6534, $t$ = 0.0625], the optimum weight of the 3D brick finite element model solved by SQP algorithm on response surfaces was 0.1166 lbs. The optimum weight of the 3D beam finite element model solved by SQP algorithm was 0.1078 lbs, the percent difference between these two solutions was 8.16%.

### 5.3.2 Discussion

As demonstrated in Figure 5.3, the von Mises stress distribution was not located at the edge of elements fixed along the wall. In this project, C3D8I, a linear element, was used for FE analysis for the 3D brick finite element model. The degrees of freedom of the element nodes, which attached to the wall, were constrained by the ENCASTRE boundary condition. Thus, the maximum von Mises stress was located on the intersection edges of the elements in the first row and second row. If the model is meshed by a quadratic brick element, such as C3D20, the maximum von Mises stresses will be developed at the edge of elements fixed on the wall, since C3D20 has more integration points on the edges of the body of the element than C3D8I. As shown in Figure 5.12, the maximum von Mises stress was located at the edge of elements fixed on the wall. Comparing with the model shown in Figure 5.3, the maximum von Mises stress of the model using C3D20 element is slightly higher, since C3D20 has more integration points than C3D8I.

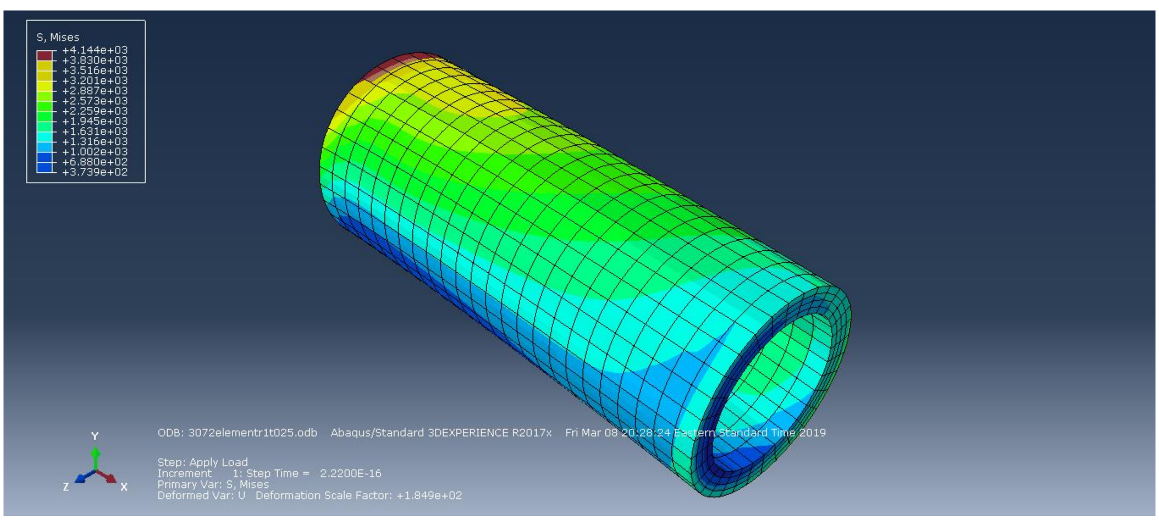


**Figure 5.12: von Mises Stress Contours for the 3D Brick Finite Element Model Using C3D20 Elements**

The accuracy of the result for FEA depends on many factors. If the FE model represents the behavior of the structure properly, the optimization system will provide a representative optimal solution based on the FEA results. The percent difference between the solution found by the SQP algorithm and the result solved by using graphical optimization method was 1.99%. The optimal solution of the 3D beam finite element model provided by the optimization system are close to the optimal solution obtained by using hand calculation and the graphical optimization method, since the equations used in calculation in these two approaches were based on the Euler-Bernoulli beam theory.

The differences between the results from the 3D beam finite element model and the 3D brick finite element model were caused by the differences of the capability of the beam element and the brick element and that the models are two different models. In particular, the boundary conditions at the wall are different between the 3D brick and beam models. The application of load at the free end of the 3D brick model is also slightly different, however the difference in load application has less impact on the critical section at the wall than the boundary conditions at the wall due to the proximity. The brick element has more displacement degrees of freedom than the 3D beam element and therefore is capable of resolving mechanics the beam element is not capable of. In this problem, the brick element can capture more detail and therefore get more detailed results, such as stress concentrations caused by the boundary conditions at the wall and 3D state of stress/strain caused by loads applied to the hollow tube.

The 3D beam finite element model can be used as a practical analytical tool for predicting the mechanical behaviors caused by the loads applied at the neutral axis of the hollow tube. The beam FE model represents the mechanics on the neutral axis of the beam, which means that all loads, BCs, section and material properties are assigned at the neutral axis. The brick element model represents the actual solid body. The material properties are assigned to the solid body. Thus, the optimum solutions found by the two different models may be different due to the capability of the elements and the similar but differences between the 1D FE model and the 3D FE model.

In order to compare the beam finite element model and the brick finite element model, a 3D beam FE model and a 3D brick FE model with geometries of [$r_o$ = 1, $t$ = 0. 25], were used to perform FEA. As described in the problem statement, one end of the tube was fixed on the wall, thus, ENCASTRE was used as the boundary conditions in the 3D beam FE model and the 3D brick FE model. The finite element analysis for the models with single axial load and with single transverse load were performed. The von Mises stress results of the finite element analysis for 3D beam finite element model and 3D brick finite element model with single axial loads are shown in Figure 5.13 and 5.14. The von Mises stress results of the finite element analysis for 3D beam finite element model and 3D brick finite element model with single transverse loads are shown in Figure 5.15 and 5.16. As shown in Figure 5.13 and 5.14, the von Mises stresses distributed among the hollow tube with a magnitude of 1222 psi, and the von Mises stresses in these models were the same. However, Figure 5.15 and 5.16 indicate that the von Mises stresses simulated from 3D beam FE model and 3D brick FE model were different. The percent difference between the maximum magnitudes of the von Mises stresses from these two models is 49.3%. As shown in Figure 5.15 and 5.16, the magnitude of the maximum von Mises stress 3D brick finite element model is larger than the magnitude of 3D beam finite element model. The brick finite element has more displacement degrees of freedom than the 3D beam finite element, thus, the 3D brick model can represent additional mechanics, such as stress concentrations, that the beam element cannot represent. And the mechanics of the 3D brick finite element model, such as the strain-displacement, stress-state, and the displacement filed, are different with the 3D beam finite element model. Additionally, the brick element considers stress concentration's influence on the result. Thus, the von Mises stress from 3D brick FE model is larger than the results from 3D beam FE model.

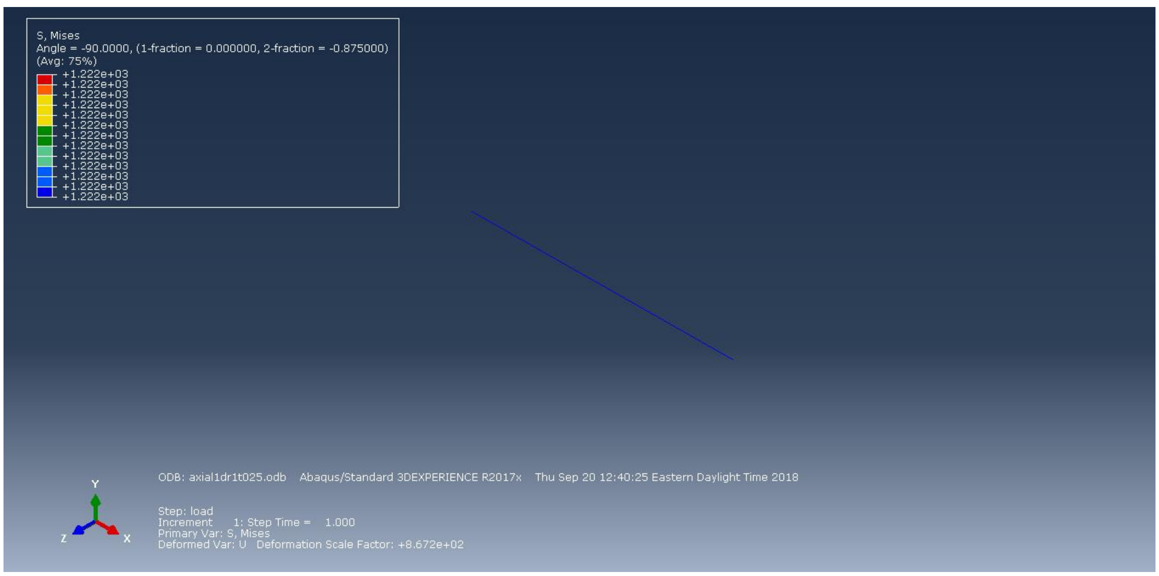


**Figure 5.13: von Mises Stress for Beam Element Model with Axial Load**

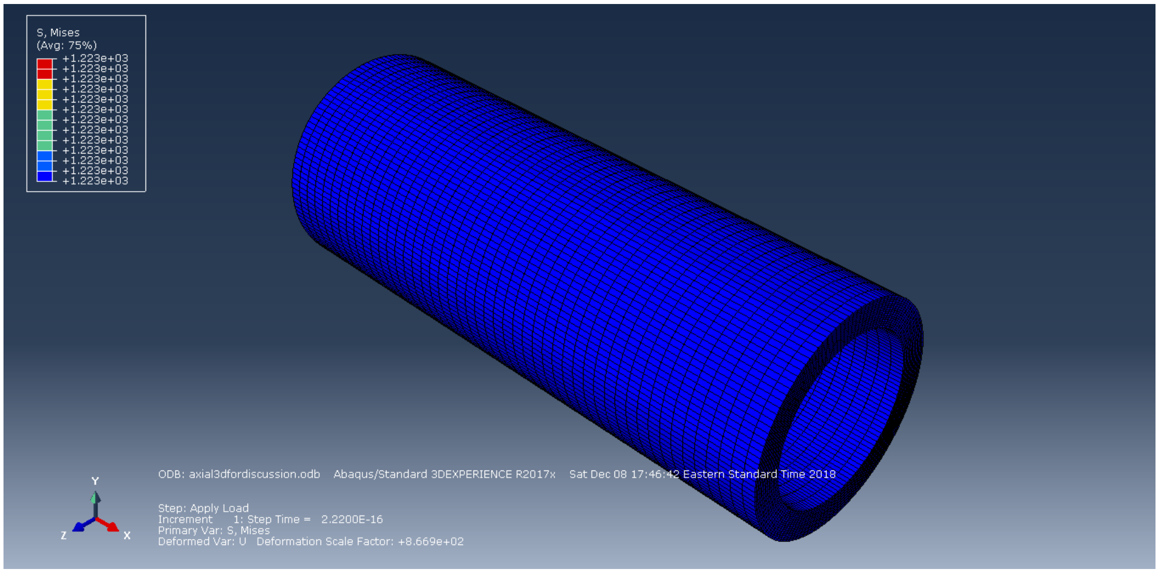


**Figure 5.14: von Mises Stress for Brick Element Model with Axial Load**

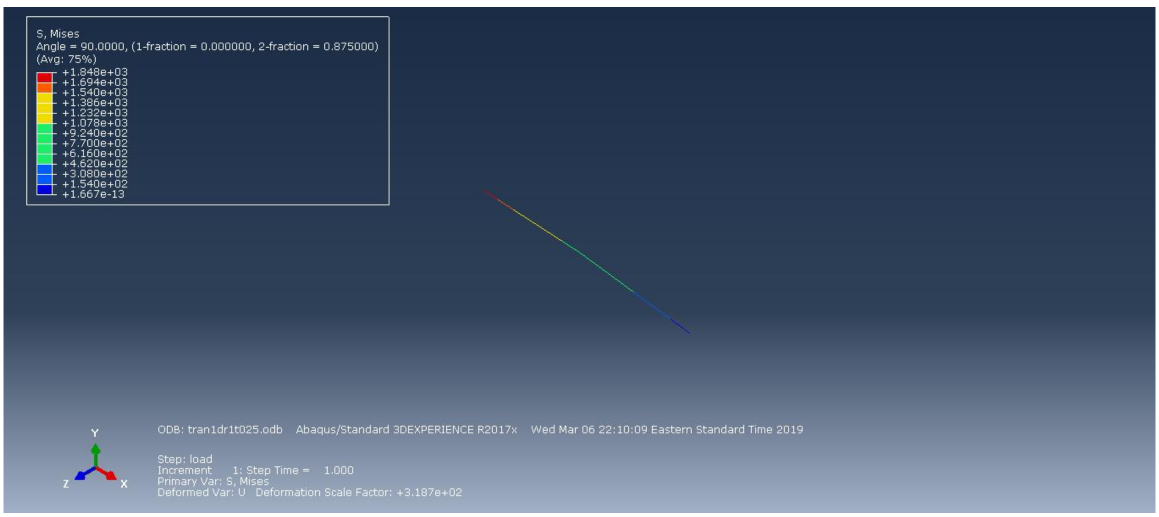


**Figure 5.15: von Mises Stress for Beam Element Model with Transverse Load**

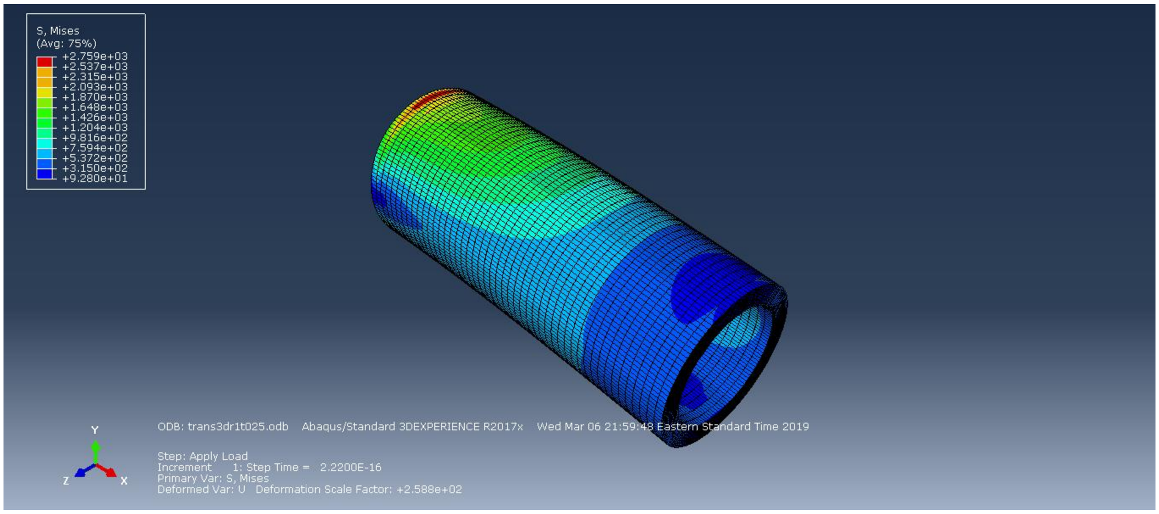


**Figure 5.16: von Mises Stress for Brick Element Model with Transverse Load**

According to the design space shown in Figure 3.10, the constraint function for the safety factor of fatigue fracture $g_2(x)$ was the active constraint. The optimal point's location on the lower bound of the wall thickness was t= 0.0625 inches. Thus, the models with outside radii varying from 0.375 inches to 1 inch and 0.0625 inches as wall thickness were created to figure out the differences between the results obtained by different methods. In Figure 5.17, the blue curve represents constraint function values calculated by the information extracted from the result files from 3D beam finite element models, and it is one edge of the reference surface. The black-colored dashed curve represents constraint function values calculated by the information extracted from the result files from 3D brick finite element models. The red-colored dashed curve represents constraint function values calculated by the response surface which was generated by 3D brick finite element models, and it is one edge of the response surface. The blue curve is beneath the two dashed curves. According to the Equation 3.49, which is $g_2(x) = 1 - \frac{1}{1.2(\frac{\sigma_a'}{S_f}+\frac{\sigma_m'}{S_{ut}})}$, the value of $g_2(x)$ becomes larger if the alternating and midrange von Mises stresses are getting larger. Due to the stress concentration's influence on the result, the von Mises stress from 3D brick FE model is larger than the results from 3D beam FE model. The curve of $g_2(x)$ constraint function for 3D beam finite element models reached 0 earlier than other curves. Since the capabilities of the 3D beam finite element and 3D brick finite element are different, the optimization problems based on 3D beam finite element model 3D brick finite element model are two different optimization problems.

The black-colored dashed curve represents the behavior of $g_2(x)$ constraint function values for 3D brick finite element model. The red-colored dashed curve represents the behavior of $g_2(x)$ constraint function values for response surface generated by 3D brick finite element models. As shown in Figure 5.17, the dashed curves highly overlap. The response surface generated by the biquadratic interpolation function represents the constraint functions appropriately. The $g_2(x)$ constraint function values calculated by different methods are listed in Table 5-3.

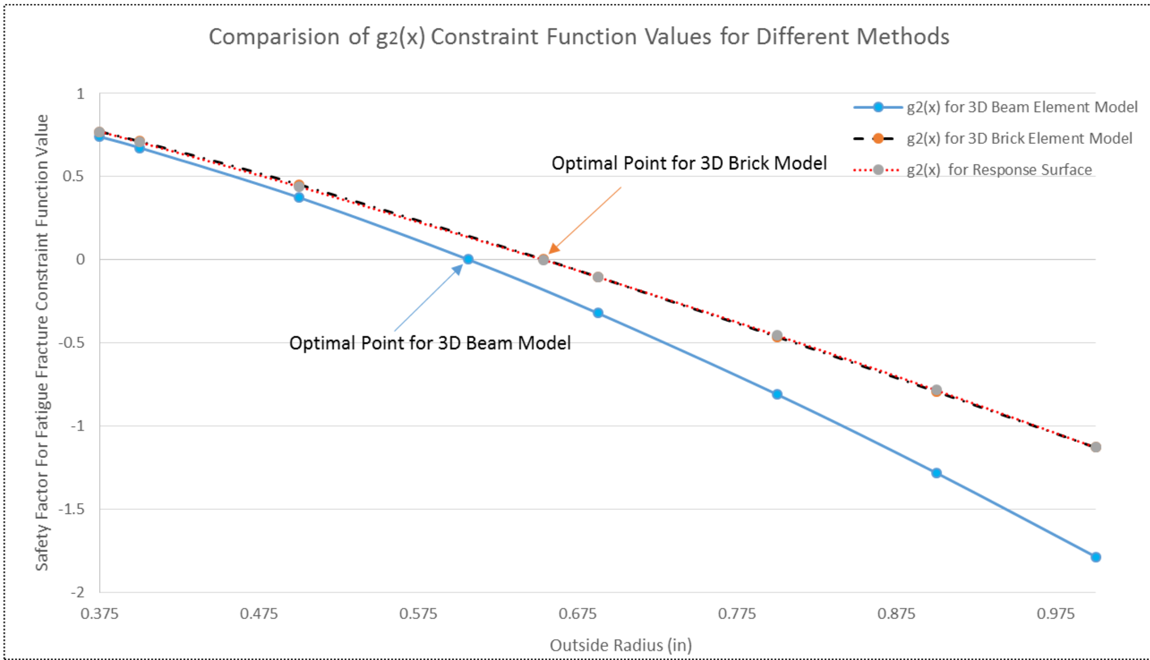


**Figure 5.17: Comparison of $g_2(x)$ Constraint Functions Values for Different Methods**

**Table 5-3. $g_2(x)$ Constraint Function Values Calculated by Different Methods**

| Outside Radius (in) | g(2) for 1D Beam Element Model | g(2) for 3D Brick Element Model | g(2)for Response Surface | Outside Radius (in) |
|---|---|---|---|---|
| 0.375 | 0.7401 | 0.771 | 0.771 | 0.375 |
| 0.4 | 0.6741 | 0.7119 | 0.7067 | 0.4 |
| 0.5 | 0.3748 | 0.4526 | 0.4398 | 0.5 |
| 0.6066 | 0 | 0.0031 | 0 | 0.6534 |
| 0.6875 | -0.3213 | -0.1027 | -0.1027 | 0.6875 |
| 0.8 | -0.8101 | -0.4626 | -0.4545 | 0.8 |
| 0.9 | -1.2826 | -0.7925 | -0.7838 | 0.9 |
| 1 | -1.787 | -1.1287 | -1.1287 | 1 |

# Chapter 6

# Summary and Conclusion

## 6.1 Summary

Commercial optimization packages seek to integrate a suite of tools, including finite element analysis software, CAD/CAE software and internally developed optimization programs, to execute simulation-based design processes. The commercial packages provide visual interfaces to set up and manage these tools. Unfortunately, commercial optimization packages do not contain a wide range of analysis options to handle real-world industrial models. Based on business considerations, commercial optimization software systems are designed for general-purpose use and robust solution. However, the default optimization formulations of some software systems may not be suitable in many cases. For instance, Isight will only extract extreme values at integration points unless custom scripts are written and embedded into the extraction program by users. Nevertheless, in this case study, this method does not work due to lack of information required for the extrapolation of the element stresses and strains to the nodes. If the commercial codes do not provide a framework for users to properly formulate and solve required optimization problems, users have to write custom codes to assist the software systems to formulate optimization problems, including the utility software needed to integrate the software systems, such as retrieving data from the .odb files generated by FEA software.

The aim of this research is to develop a structural optimization software system that consists of the Abaqus Finite Element solver and the MATLAB optimization toolbox. The software system for this research was developed in Python and MATLAB. This optimization program must have the following functionality: (1) allow users to define the variables as well as the objective and constraint functions; (2) allow users to control the data extraction and operate on the result data to properly evaluate the objective and constraint functions; (3) automatically execute the optimization process by evaluating the objective and constraint functions repetitively until the optimal solution is obtained; and (4) apply the design variables to the model for solution.

To accomplish the research goals, approaches for interfacing the MATLAB optimization toolbox with Abaqus, the finite element analysis (FEA) solver, were explored. The modeling, finite element analysis and extraction of results were executed in Abaqus through Python scripts. MATLAB programs were developed to read the results and evaluate the objective and constraint functions. The optimization process was driven by the optimization toolbox in MATLAB. As the design parameters were determined within the optimization code, the MATLAB programs evaluated the objective and constraint functions as well as called the Python scripts associated with the utility MATLAB programs. The parametric finite element modeling approach was developed to apply the design variables to the finite element model. The response surfaces were generated by using the results of sample points and the MATLAB interpolation functions. Due to the use of the response surface method, the time-consuming optimal solution stage in the large-scale model structural optimization were significantly reduced.

In the large-scale optimization program, the finite element model needs to be rebuilt or meshed repetitively upon applying the design variable updates. Instead of creating the model manually, a modeling script based on parametric finite element modeling methods are required for building and remeshing the model in the 3D optimization process. The parametric finite element modeling methods enable users to change the design variables in MATLAB without opening the Abaqus CAE GUI. The geometries used for building and meshing the model were written as expressions in terms of design variables. The objective function of the optimization problem for this research was to determine the optimum outside radius and wall thickness of the tube. The main structure of the hollow tube and the load conditions did not change during the optimization process. Thus, the Python scripts for building, meshing, and extracting results were not changed as the finite element model was being updated.

The response surface method was used in the optimization system for large-scale finite element models. In general, the objective and constraint functions of real-world optimization problems cannot be written in terms of the design variables in closed-form. The gradients of the objective functions and constraints to determine the search direction for design update and the step size cannot be written in terms of the design variables in closed form. In order to form smooth surfaces for the objective and constraint functions, a large number of models are required to be created and analyzed. Response surfaces can estimate the function values by using interpolation or approximation functions, and can be developed to a “representative level” that requires fewer computational samples that can be incrementally developed offline. This approach leads to a decrease of time-consuming and expensive finite element analyses. Additionally, the response surface can be checked and updated easily based on the previous response surface approximation without making additional FE analysis runs, and the number of FE analysis runs can be controlled. In this study case, 9 samples of design variables in the design space were used to generate the response surface, where upon, the optimization toolbox was able to find the solution on the response surface using an SQP algorithm. The results from regression tests verified that the response surface was generated correctly by using appropriate interpolation functions.

In a structural optimization problem, the designers should consider the representative loading conditions and perform FEA for these loading conditions to evaluate the constraints. In this case study, fluctuating loads were applied to the hollow tube. Since the safety factors for fatigue yield and fatigue fracture were inequality constraints, the midrange and alternating stresses have to be solved. A load scaling method provides a way to represent the midrange and alternating stresses with a single FEA run. The details of scaling method were introduced in Section 4.4.2.

In the structural optimization program for small-scale models, the finite element analysis (FEA) is executed based on the information from the .inp file. The MATLAB program will read the .inp file and catch the text line that includes the variables. Then, Abaqus is driven by this MATLAB program to perform FEA. Python scripts are written to extract data from the .odb file and write the required result data to a .csv file. In order to perform the optimization processes through the optimization toolbox, the required Abaqus results in the .csv files are read and formed into a matrix in MATLAB. The optimization toolbox is responsible for checking convergence of the optimization solution, and will determine if the process is to continue or stop. If the optimization does not converge, MATLAB will start the next iteration optimization loop. Then, the new design variables created by optimization toolbox will be written to the new .inp file and subsequently the process will restart from a new pair of geometry parameters by a MATLAB program. Once the optimization converges at the optimum parameters, MATLAB will stop the process.

A benchmark hollow tube weight minimization problem was used to test the optimization software system. First, the hand calculations based on Euler-Bernoulli beam theory were made to form the objective and constraint functions. The graphical optimization and the subsequent NLP formulation associated with hand calculations provided a reference optimum solution for the 3D beam element model. In the next stage, a structural optimization process for a small-scale finite element model based on a gradient-based optimization algorithm was developed and tested on an Abaqus 3D beam finite element model. Finally, a structural optimization process for large-scale models based on parametric finite element modeling and response surface methodologies was developed on a 3D brick finite element model. The developed optimization software system was able to solve the small=l-scale and large-scale structural optimization problems. The optimum results found by these approaches are shown in Table 6-1.

**Table 6-1: Optimum Solutions Found by Different Methods**

| | Optimum Outer Radius (in) | Optimum Thickness (in) | Optimum Weight (lbs.) | Different Percentage (%) |
|---|---|---|---|---|
| Graphical Optimization Method | 0.5953 | 0.0625 | 0.1057 | 0 |
| 3D Beam Element Model with SQP Method | 0.6066 | 0.0625 | 0.1078 | 1.97 |
| 3D Brick Element Model with Response Surface Method | 0.6534 | 0.0625 | 0.1166 | 9.81 |

The optimum solution gained from the hand calculation was [$r_o$ = 0.6, $t$ = 0.0625]. The graphical optimization solution was obtained by visual inspection, thus the solution was estimated with limited accuracy. Using the *fzero* function in MATLAB, the optimum parameters solved by graphical optimization was [$r_o$ = 0.5953, $t$ = 0.0625].

In the optimization system for small-scale finite element models, the optimization processes were performed by the MATLAB optimization toolbox *fmincon* function using the SQP solver. As the optimization was processing, the models were generated by rewriting the design parameters in the .inp file through MATLAB. The optimum solution obtained from this optimization system was [$r_o$ = 0.6066, $t$ = 0.0625]. As shown in Table 6-1, the percent difference between the results from the 3D beam finite element model with SQP solver and the graphical optimization method was 1.99%.

In the optimization system for large-scale finite element models, the optimization toolbox obtained the optimum solution by using SQP algorithm on response surfaces. The solution provided by the optimization system for the large-scale model was [$r_o$ = 0.6534, $t$ = 0.0625]. As shown in Table 6-1, the percent difference between the results from the 3D brick finite element model with RSM and the result from the graphical optimization method was 10.31%. The percent difference between the results from the 3D beam finite element model with SQP solver and the result from 3D brick finite element model with RSM was 8.16%.

The mechanics of the 3D brick finite element model, such as the strain-displacement, stress-state, and the displacement filed, are different with the 3D beam finite element model. The brick finite element has more displacement degrees of freedom than the 3D beam finite element, thus, the 3D brick model can represent additional mechanics, such as stress concentrations, that the beam element cannot represent. The mesh refinement have a significant influence on the analysis results for the 3D brick model. In order to get a high quality mesh, the convergence level was set within a tolerance of 0.1%. The convergence level associated with mesh refinement for the 3D model was 0.06%. The regression test for the response surface demonstrates that the response surfaces were generated by representative interpolation functions. The optimal thickness is 0.0625, which was the same as the result from the graphical optimization and optimization system for the small-scale finite element model. The result the from optimization system for the large-scale model was 0.6534, thus, the solution is reasonable with the consideration of the additional modeling capability inherent in the 3D finite element models.

## 6.2 Conclusion

The main focus of this thesis was to develop a structural optimization framework with the following functionalities:

1. Allow users to define the design variables, the objective function, and constraint functions.

2. As the required parameters are input to the system, the finite element analysis model is created as a parametric finite element model in Abaqus.

3. Allow users to control the data extraction and manipulate the data to evaluate the objective and constraint functions.

4. Automatically executes the optimization process by evaluating the objective and constraint functions repetitively until the solution is obtained.

One of the main contributions of this work was to develop a structural optimization approach which allows users to define the optimization problem in NLP standard form, and extract and manipulate data for optimization without limitations that users cannot do in the commercial structural optimization software systems. This approach could be useful for solving optimization problems in other areas, such as frequency response or heat transfer optimization problems if the optimization problem can be defined in NLP standard form. The approach is efficient since the modeling stage in the large-scale model structural optimization were significantly reduced by using the parametric finite element modeling method and the response surface optimization method.

The structural optimization software system developed in this research is able to solve structural optimization problems of small-scale and large-scale finite element models. The structural optimization program for small-scale finite element models consists of the gradient-based optimization method and input file regeneration program. In this system, users only need to put in the parameters of the initial design, and the system will then run the optimization process and generate new models automatically until the solutions are obtained.

Another contribution was to develop an approach for solving the structural optimization problems of large-scale finite element models combines parametric finite element modeling methods executed by Python scripts with response surface optimization methods. This approach reduces the number of finite element analyses as well as reduces the optimization process solution time. The optimization module of the system is performed by the MATLAB optimization toolbox. The modeling and finite element analysis are executed by the Abaqus with scripts implemented in Python.

The building, meshing, creating jobs, and extracting functions were combined into one Python script named “main”. Users are able to fully control the CAD and FEA processes including building and meshing through the main Python script. The script is easy to edit, since the function modules were written as Python functions and were called in the main script during the optimization process. For an experienced user, the structural optimization software system can be used to solve the structural optimization problems of different finite element models by modifying the script.

Response surface methodology enables users to control the number of FE analysis runs. The number of simulations necessary to perform the optimization is much lower than in a traditional optimization method, which leads to a decrease of time-consuming and expensive finite element analyses, and the response surface can be checked and updated easily based on the previous response surface approximation without making addition FE analysis runs. In this thesis, the biquadratic interpolation function was used to approximate the response surfaces of constraint functions and objective function based on the order of the functions. The linear regression test proved that the biquadratic interpolation function approximate the functions appropriately.

In this thesis, an optimization problem was designed for the project. The objective of the problem was to minimize the weight of a hollow tube shown in Figure 3.1. This optimization problem was formulated in the nonlinear programming standard form. The Euler-Bernoulli beam equations were used to formulate constraint functions. The graphical optimization method was used to obtain a reference answer for the problem. The calculations in graphical optimization method was based on the Euler-Bernoulli beam theory. The 3D beam finite element model was optimized by the structural optimization program for small-scale finite element models. Thus, the reference answer obtained by the graphical optimization method could be used to validate the approach for solving the structural optimization problems of small-scale finite element model. The percent difference between the results of these two method was 1.99%, the structural optimization system for small-scale finite element models is valid.

To indicate the differences between the 3D beam finite element and the 3D brick finite element, a comparison of the results obtained from the programs for solving the structural optimization problems of small-scale finite element models and large-scale finite element models was performed. The percent difference between these two solutions was 8.16%. The 3D beam finite element model can be used as a practical analytical tool for predicting the mechanical behaviors. The differences between the results from the 3D beam finite element model and the 3D brick finite element model were caused by the differences of the capability of the beam element and the brick element. The mechanics of the 3D brick finite element model, such as the strain-displacement, stress-state, and the displacement filed, are different with the 3D beam finite element model. The brick finite element has more displacement degrees of freedom than the 3D beam finite element, thus, the 3D brick model can represent additional mechanics, such as stress concentrations, that the beam element cannot represent.

## 6.3 Recommendation and Future Work

The structural optimization software system is still in the early stage of development and exploration. In order to optimize the models with complex geometries, such as industrial components, the codes used to extract Abaqus results from the target regions from .odb files requires additional exploration. Recently, the stresses used for creating constraint functions are extracted from the specific node or element with the known index number. A new “main” Python script, based on Abaqus “sets”, should be able exploit the associativity of the Python scripted objects, and refer to those relevant quantitates within all functions such as modeling, meshing, and extracting functions in the main script. Using the new “main” Python script, users will have less workloads for changing the partitions, meshing, seeding the mesh elements, and extracting sets when the structure of the model requires some changes. For example, the new extraction function in the Python script will be able to automatically identify the target node or element, and then extracts the results from the relevant model sets instead of extracting the stress from the node by manually entering the node index in the command line. Recently, the stresses at the target node were calculated by averaging the element nodal stresses from the elements attached to the node. After the results were extracted from the .odb files and written in .csv files, users have to determine the location of the element nodal stresses at the target node in the .csv file, then pass the results into a MATLAB matrix named “AbaqusData” and average them manually. In the future, the script should be able to post-process the Abaqus results properly from elemental values to nodal values.

Future works for this case study should focus on designing the optimization software framework to include tools capable of solving all kinds of optimization problems, such as linear statics, heat transfer, and frequency response problems. In order to accomplish this goal, the modeling, meshing, and extracting functions in the main script should be generalized and extended to solve additional types of models. The main script could be separated into several functional modules depending on the type of problem. Each of functional modules can be developed for an independent optimization system with more details and operational choices. Users must be able to use their domain specific finite element knowledge and expertise to formulate the problems and post-process the results properly within the scripting in order to evaluate the objective functions and constraints seamlessly.

# Bibliography


[1] Arora, J. S. (2011). "Introduction to optimum design." 3rd.

[2] Yang, X. S. and S. Koziel (2011). Computational Optimization: An Overview. Computational Optimization, Methods and Algorithms. **356:** 1-11.

[3] Schmit LA Jr (1971) Structural synthesis 1959–1969: A decade of progress, Recent Advances in Matrix Methods of Structural Analysis and Design. Univ. Alabama Press, Tuscaloosa, pp 565–634

[4] Kim, Y. I. and B. M. Kwak (2002). Design space optimization using a numerical design continuation method. International Journal for Numerical Methods in Engineering. **53:** 1979-2002.

[5] Choi, W. H., et al. (2016). Comparison study of some commercial structural optimization software systems. Structural and Multidisciplinary Optimization. **54:** 685-699.

[6] Thomas, H., et al. (2002). "Issues of commercial optimization software development." Structural and Multidisciplinary Optimization **23**(2): 97-110.

[7] Abaqus, V., 2017, "Beam Cross-section Library," Dassault Systemes Simulia Corporation. Retrieved from https://www.sharcnet.ca/Software/Abaqus610/Documentation/docs/v6.10/books/usb/default.htm?startat=pt06ch26s03abm01.html

[8] Vanderplaats, G. N. (1997). "Structural optimization: where we've been and where we're going." Computer Aided Optimum Design of Structures V: 45-54.

[9] Haftka R.T., Sobieszczanski-Sobieski J. (2008) Structural Optimization: History. In: Floudas C., Pardalos P. (eds) Encyclopedia of Optimization. Springer, Boston, MA

[10] Vanderplaats, G. N. (1982). "Structural Optimization - Past, Present, and Future." Aiaa Journal 20(7): 992-1000.

[11] "Computing." Encyclopedia of Communication and Information. . Retrieved December 20, 2018 from Encyclopedia.com: https://www.encyclopedia.com/media/encyclopedias-almanacs-transcripts-and-maps/computing

[12] Bendsøe, M. P. (1995). Optimization of structural topology, shape, and material. Berlin; New York, Springer.

[13] Isight, V., 2017, "Simulia Isight Brochure," Dassault Systemes Simulia Corporation. Retrieved from https://www.3ds.com/fileadmin/PRODUCTS-SERVICES/SIMULIA/RESOURCES/simulia-isight-brochure.pdf

[14] Andia, A. and T. Spiegelhalter (2015). "Post-Parametric Automation in Design and Construction Preface." Post-Parametric Automation in Design and Construction: 13-15.

[15] C. X. Yue et al., "High-Speed Cutting Finite Element Model Parametric Modeling Based on ABAQUS", Materials Science Forum, Vols. 800-801, pp. 353-357, 2014

[16] Box, G. E. P. and K. B. Wilson (1951). "On the Experimental Attainment of Optimum Conditions." Journal of the Royal Statistical Society Series B-Statistical Methodology **13**(1): 1-45.

[17] Gilmour, S. G. (2006). Response surface designs for experiments in bioprocessing. Biometrics. 62: 323-331.

[18] Wu, You-Sheng Cui, Wei-Cheng Zhou, Guo-Jun. (2001). Practical Design of Ships and Other Floating Structures, Volume 1 - 42.1 Introduction. Elsevier. Retrieved from https://app.knovel.com/hotlink/pdf/id:kt00C6FKV1/practical-design-ships/practical--introduction-4

[19] Bezerra, M. A., et al. (2008). "Response surface methodology (RSM) as a tool for optimization in analytical chemistry." Talanta 76(5): 965-977.

[20] Gustafsson, E. and N. Stromberg (2008). "Shape optimization of castings by using successive response surface methodology." Structural and Multidisciplinary Optimization 35(1): 11-28.

[21] Gustafsson, E., & Strömberg, N. (2005). Structural optimization of castings by using ABAQUS and Matlab. Competent Design by Castings : Improvements in a Nordic Project (Vol. 237, pp. 39–52). Espoo, Finland: VTT. Retrieved from http://urn.kb.se/resolve?urn=urn:nbn:se:oru:diva-48254

[22] Overvelde, J.T.B. Learn Abaqus Script in One Hour. (2010). Retreieved from bertoldi.seas.harvard.edu/files/bertoldi/files/learnabaqusscriptinonehour.pdf.

[23] "Aluminum 2014-T4 material properties." (n.d.). ASM Aerospace Specification Metals Inc. Retrieved from http://asm.matweb.com/search/SpecificMaterial.asp?bassnum=MA2014T4

[24] Dowling, N. E. (2013). Mechanical behavior of materials : engineering methods for deformation, fracture, and fatigue. Boston, Pearson.

[25] Budynas, R. G., et al. (2011). Shigley's mechanical engineering design. New York, McGraw-Hill.

[26] MATLAB, R2018b, “Optimization Toolbox,” Retrieved from https://www.mathworks.com/help/optim/

[27] Abaqus, V., 2017, "Abaqus Scripting Reference Guide" Retrieved from http://docs.software.vt.edu:8080/abaqus/v2017/English/DSSIMULIA _Established.htm

[28] Abaqus, V., 2017, "Parametic Modeling," Retrieved from https://abaqus-docs.mit.edu/2017/English/SIMACAEMODRefMap/simamod-c- parinput.htm

[29] Wynne, J. D. (1982). Learning statistics : a common-sense approach. New York London, Macmillan ; Collier Macmillan Publishers.

[30] MATLAB, R2018b, “First-Order Optimality Measure,” Retrieved from https://www.mathworks.com/help/optim/ug/first-order-optimality-measure.html