Can you find ~~that~~ whether the final pareto optimal designs satisfy the ASME safety standard??

DIFFERENTIALLY EVOLVED DESIGN OF AN EQUALIZER BEAM FOR USE IN GENERAL PURPOSE LIFTING AND HANDLING

prob statement

Aaron T. Moore
B.S. May 2013, Old Dominion University

A Project Report Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF ENGINEERING

MECHANICAL ENGINEERING

May 2019

Mechanical and Aerospace Engineering
Old Dominion University
Norfolk, Virginia, USA
2019-04-30

Approved by:
NOTE: This is a draft.
Approvals have not been given.
Dr. Gene Hou (Advisor)

Contents

List of Figures

List of Code Listings

CHAPTER 1

MOTIVATION, PROBLEM DESCRIPTION, OBJECTIVE AND EXPECTED OUTCOME

In industry, cranes are commonly used to lift heavy materials and transport them from location to location. In some cases, equipment or material may exceed the lifting capacity of a single crane or lifting fixture. In these cases, multiple cranes must be used to move the load. In order to ensure multi-crane lifts are performed safely, it must be ensured that each crane shares the load equally. Structural components known as equalizer beams are commonly employed to ensure that these loads are evenly distributed between the cranes.

Equalizer beams are typically single weldments with few, if any, moving parts. They are typically constructed from steel. The beam is typically built with 3 major attachment points. One for the load to be lifted, and 2 equidistant crane attachment points. The centers of the three attachment points are typically along the same axis to ensure equal load sharing between the two cranes, even in the event that the beam is out of level. A typical design for an equalizer beam is shown in Figure 1.

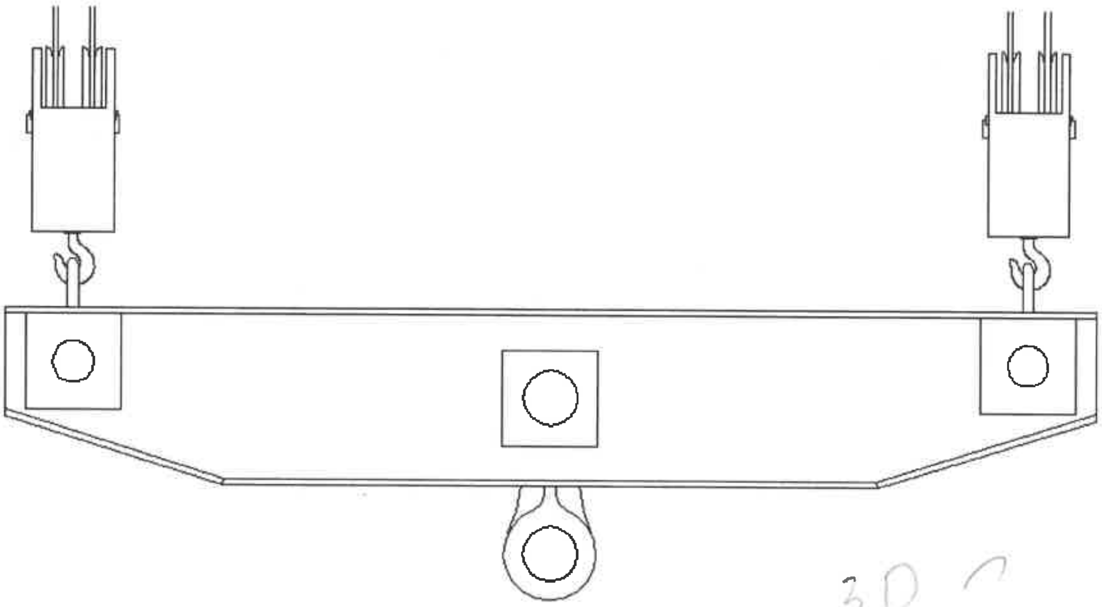
any reference

reference

Currently, the commonly accepted industry standard in use in the US is ASME BTH-1. This design standard lists recommended minimum design standards for these devices. This standard is widely accepted and is referenced in ASME's safety standard for below the hook devices, B30.20. The designs prepared and available for purchase in industry tend to be pre-engineered units utilizing standard components and beam sections. In some cases, the performance of these off-the-shelf components does not meet the needs of certain extremely stringent operating environments. Since the weight of rigging equipment is considered part of the lifted load, lifting heavy loads near the capacity of the cranes in use can require extremely weight-efficient rigging. At the same time, safety considerations require the stress in these components to be kept as low as achievable to maximize the margin of safety in the rigging system. Because these two objectives are often at odds with one another, design of these equalizer beams taking both objectives into account can be a difficult exercise for the designer.

The nature of the competing requirements imposed against the design of equalizer beams makes them suitable for multi-objective optimization. In general, multi-objective optimization is intended to find a series of optimal solutions across a range of values for the two (or more) different design objectives and present them to a designer. For example, in the case of the equalizer beam, multi-objective optimization can present a series of optimal designs that have the lowest stress at numerous different weight values. Using this technique, a designer can evaluate a number of designs at numerous different combinations of the design variables and select a design that best suits the situation.

→ should mention that two approaches are developed & empolyed to hand uncertainty in loadings



3D ?

Figure 1.1: A Basic Equalizer Beam

1.1 Objective

random load

This study will seek to develop a method to design an equalizer beam using multi-objective optimization. subject to It is the intent of this study to apply existing techniques for multi-objective optimization to this specific component and evaluate the effectiveness of such a method.

How to do the evaluation ?

1.2 Problem Description

As mentioned, the aim of this project is to evaluate multi-objective optimization as a design tool for use in the development of designs for equalizer beams. In order to perform this evaluation and provide a "benchmark case" for any comparisons between methods, an example system will be used as a subject for the design. The basic parameters for the solution are presented below.

Example System

The example system to be studied is based on a few basic fixed design parameters. The basic outline of the beam structure is shown in Figure 1.2.

Also fixed is the material, which is assumed to be ASTM A36 steel. The material has properties assumed to be:

- Yield Strength: Mean 250MPa, Std. Deviation 32.5 MPa
- Young's Modulus: 200 GPa

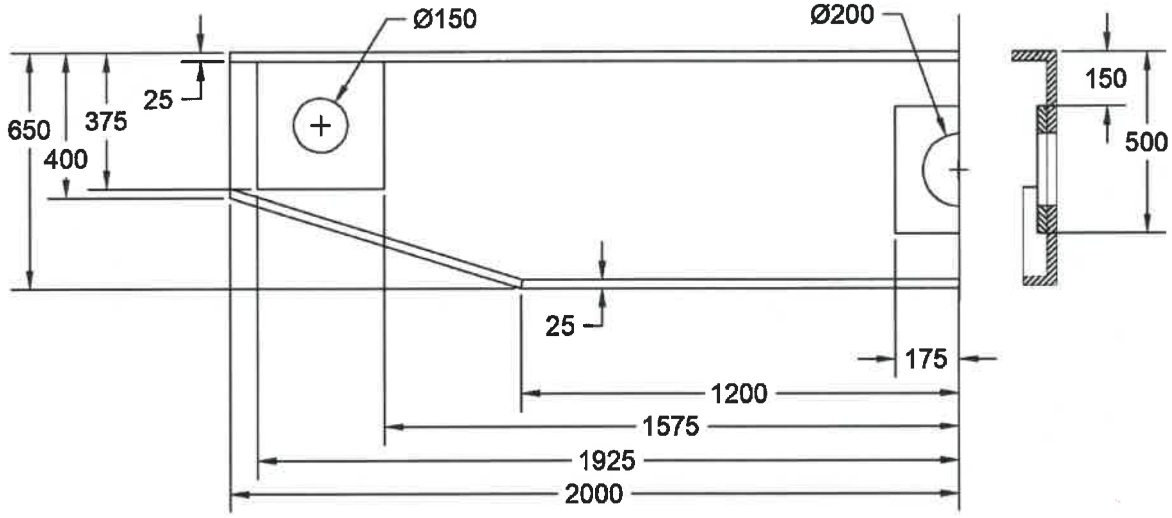


Figure 1.2: Fixed Dimensions for the Example System Beam

Performance requirements

In this case, performance requirements primarily relate to the lifting capacity and the allowable side pull on the beam. The requirements selected for this problem are:

1. Lifting capacity: 60 metric tons, 60,000kg, 132,000 lbm, 588 kN. For this problem, 600 kN was used.
2. Minimum capable side load: ±5° what does it mean ?

The beam to be analyzed is symmetric on 2 planes. This allows for the model to consist of only one quarter of the beam under study. Additionally, It will be later clarified that the selected solution methods utilize stochastic methods to model the loading. To enable the use of stochastic method of solution, the following statistically-defined loads have been used to approximate the above performance requirements:

- Load magnitude, vertical: Mean: 150 kN, Std. Deviation: 19.5 kN
- Load magnitude, horizontal: Mean: 0 kN, Std. Deviation: 13 kN

does it relate to the horizontal load? How ?

Design Objectives

For this design, we seek designs of minimum weight and maximum strength in the loading conditions the part is subject to. Formally stated, the objective functions for the optimized designs are:

1. Minimize component mass, expressed as the beam structure's mass in kilograms.
2. Maximize the critical value for the constant β, which is discussed further in section 2.4 on page 11.

Minimize the max stress in Approach 1 &

Dsgn variables ← Dsgn constraint? P.18 mooed here

weight < 10000 lbs

Mathematical Formulation ?

CHAPTER 2

SOLUTION PRINCIPALS

provide a brief outline of chapter 2

2.1 Numerical Optimization

Optimization is defined as “a mathematical technique for finding a maximum or minimum value of a function of several variables subject to a set of constraints.”[2] In the specific case of engineering design, one of several techniques is used to find local or global extrema of a function of one or multiple variables. These techniques use various criteria to traverse the independent variables and detect these extrema. The method the optimizer uses to traverse the solution space has a significant impact on the speed at which the operation converges to a solution or solutions.[1]

Is this section complete enough?

Introduce the concept of the objective function

indent

2.1.1 Differential Evolution

Differential evolution was selected as the optimizer for this study.

Differential Evolution is a member of optimizers collectively known as *Evolutionary Algorithms*. These optimizers use various approaches to emulate the concept of biological evolution to optimize a given objective function. Differential optimization performs this task through the use of a 4-phased approach: Initialization, Mutation, Crossover, and Selection. [3]

Initialization

Prior to starting the optimization loop, an initial “population” of individuals has to be generated for the optimizer to work from. An individual consists of a vector $\vec{x}$ of values of the objective function’s independent variables. The Initialization process generates a vector $\vec{X} = \{\vec{x}_1, \vec{x}_2 \cdots \vec{x}_n\}$ of individuals by randomly selecting values for the independent variables. This vector represents the complete population that will be operated on in the first generation of the optimization loop. These individuals will be collectively be known as *parents*.[3]

In the case of the implementation presented here, Latin Hypercube Sampling (LHS) was employed to generate the random values to assemble $\vec{X}$. More detail on LHS can be found in section 2.2.

Mutation

The mutation operator is the first step for each cycle of the optimization loop. This algorithm emulates random mutations in genetic code commonly found in nature. Taking each individual from the vector of parents, a mutated vector of properties is generated. These vectors are known as *trial vectors.* To perform this action, the basic process flow shown below is employed [3]:

```
trial_vector = empty_2d_vector[num_individuals][num_design_vars]
for integer j in [0 ... num_individuals]:
    x_1 = individuals[j]
    x_2 = random_member_of_individuals
    x_3 = random_member_of_individuals
    beta = x.x // (arbitrary amplification factor selected by user)
    for integer i in [0 ... num_design_vars]:
            trial_vector[j][i] = x_1[i] + beta * (x_2[i] - x_3[i])
return trial_vector
```

Listing 2.1: Pseudo-code for the Mutation Operator [3]

This set of trial vectors is one of the distinguishing facets of Differential Evolution. If the equation on line 8 above is reviewed carefully, it can be seen that the trial vector is different than its associated parent vector by the distance between 2 other random individuals within the solution space. This has the interesting effect of causing the differences to between parent and trial vectors to change based on the condition of the solution. Early in the solution process when the individuals are sparsely spaced across the solution space, the trial individuals tend to spread apart similarly. In later cycles as minima start to become identified, the distance between individuals becomes smaller. This has the effect of making the trial vectors land more closely to their associated parents. This allows Differential evolution to converge relatively quickly once minima start to appear in the solution space [3].

Crossover

The crossover operator combines the parent individuals and their associated trial individuals to make a single set of *child individuals.* It does this using the following general procedure [3]:

```
child_vector = empty_2d_vector[num_individuals][num_design_vars]
for integer j in [0 ... num_individuals]:
    C = x.x // Constant that dictates how often the crossover picks from the
            // trial vector.
    for integer i in [0 ... num_design_vars]:
        rnd = make_random_number()
        if rnd > C
            child_vector[j][i] = trial_vector[j][i]
        else
            child_vector[j][i] = parent_vector[j][i]
return child_vector
```

Listing 2.2: Pseudo-code for the Crossover Operator [3]

Selection

The final major phase in the Differential Evolution loop is the Selection Operator. This operator takes the parent and child vectors and calculates the value of the objective function for each individual. The operator then compares each parent individual's fitness against the associated child individual. The one with a more desirable fitness value is retained and added to the *output vector* while the other is discarded. The general process is outlined below [3]:

```
output_vector = empty_2d_vector[num_individuals][num_design_vars]
for integer j in [0 ... num_individuals]:
    f1 = get_fitness(parent_vector[j])
    f2 = get_fitness(child_vector[j])
    if f1 < f2
        output_vector[j] = parent_vector[j]
    else
        output_vector[j] = child_vector[j]
return child_vector
```

Listing 2.3: Pseudocode for the Selection Operator (Assuming a minimization approach) [3]

2.12 Extending DE to Multi-Objective

Differential Evolution Optimization is originally a single-objective method. However, it can easily be extended to multi-objective operation by changing the method by which fitness is evaluated. Instead of a single fitness function, multiple independent fitness functions are evaluated using the concept of Pareto dominance.

Complete this section

Pareto Dominance

Pareto Dominance is a simple way to compare systems based on multiple different fitness criteria. Pareto dominance for a minimization problem can be described by the following formula [3]:

Let the vector of fitness values for two arbitrary solutions be defined as:

$$C_a = \{f_1, f_2, ...f_n\}$$
$$C_b = \{F_1, F_2, ...F_n\}$$

n = # of samples ? or n fitness values

Then, an operator $P_{a,b}$ can be defined which returns true if C_a dominates C_b. This would take the following form:

$$P_{a,b} = \begin{cases} \text{True when } C_a^i < C_b^i \forall i \in \{1..n\} \\ \text{False otherwise} \end{cases} \quad (2.1)$$

Note that the operator $P_{a,b}$ does not necessarily imply the value of $P_{b,a}$. While only C_b or C_a can be dominant, it is possible that neither is dominant. In this case, both $P_{a,b}$ and $P_{b,a}$ would be false[3].

Implementing Pareto Dominance in DE

In order to extend DE to be used with multi-objective problems, the code presented in this report simply made a slight modification to the selection operator shown in code listing 2.3:

```
output_vector = empty_2d_vector[num_individuals][num_design_vars]
for integer j in [0 ... num_individuals]:
    //These statements would return arrays of several fitness values.
    f1 = get_fitness(parent_vector[j])
    f2 = get_fitness(child_vector[j])

    if P(f2,f1)
        output_vector[j] = child_vector[j]
    else
        output_vector[j] = child_vector[j]
return child_vector
```

Listing 2.4: Modified Selection Operator

Note that the structure of the if condition "defaults" to selecting the parent. The child is only selected if it is dominant. The parent is selected if it is dominant or if neither dominates.

2.2 Random Load

with contours

~~2.2~~ 2.2.1 Latin Hypercube Sampling for loads

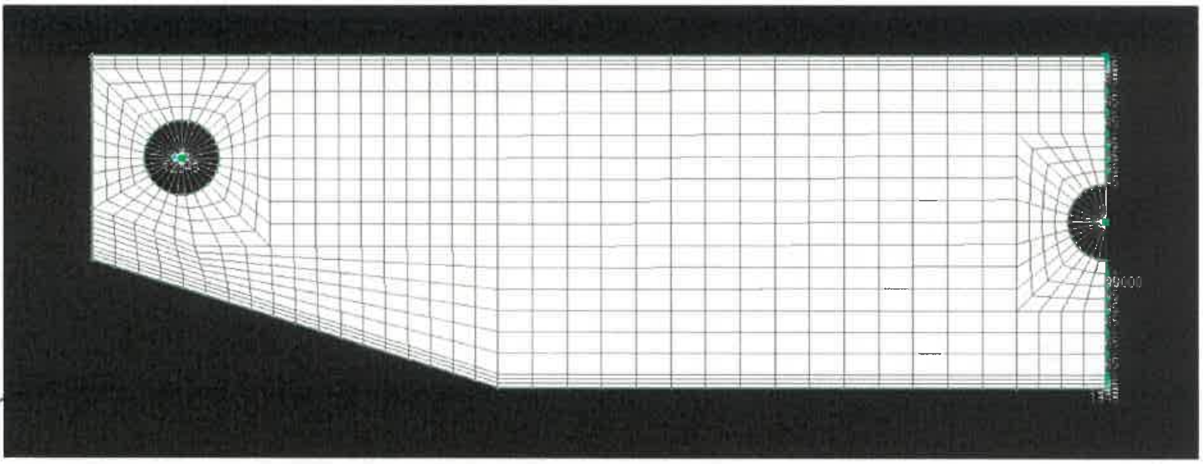
Latin hypercube sampling is used throughout the code presented here to generate evenly distributed random data. this method of sampling was selected due to its ability to evenly spread out the sampling points throughout the variable's domain. It does this by programmatically ensuring that each value of each input variable is only represented once. It does this by separating the domain of each input variable into a predetermined number of strata, each with an identical probability of containing an arbitrary sample point. Then, one sample is taken from each of the strata. In this implementation, the different variables in the space do not interact, and the given random values of each input variable are combined at random to form a vector of input variables, also known as an individual [5].

2.2.2 reliability index & stress

2.3 ~~2.3~~ Finite Element Modeling

When performing optimization to find designs, numerous designs must be analyzed and evaluated for fitness. In some cases, these designs and their fitness functions can be modeled using closed form equations. In the many cases, however, the designs are complex enough that finding closed form models of their behavior is infeasible. In the case that no closed form solutions can be found, numerical analysis can offer an alternative method of evaluating for fitness.

For this study, numerical analysis in the form of Finite Element Modeling was chosen as the method for evaluating the designs for fitness. Each design and load case each are represented as slight modifications



no black background,
Can it be just a mesh graph?

Figure 2.1: Mesh Geometry as shown in Siemens Femap

type of elements
# of nodes
# of elements

Yes

to a basic model file containing the fixed geometry and "starting" parameters for the design and loading variables.

Do I need to summarize the basics of finite element modeling?

This study makes exclusive use of quadrilateral shell elements, known to NASTRAN as CQUAD4 elements. These elements are 2-dimensional, but do model deformation in all thee dimensions. However, an important assumption made by using these elements is that the entire thickness of the plate distributes the stress applied to the element equally. This implies that deformations in the thickness direction cannot be modeled this way. For example, if a plate can buckle or bend through its thickness this type of element would not be suitable. This model makes use of these elements for the beam flanges, but the elements are oriented to be coplanar with the beam web. This means the flange width is modeled as plate thickness. This orientation will not detect local flange effects such as buckling or local bending. However, this study is primarily concerned with failure of the gross section. Due to the system geometry, these stresses are negligible and therefore the model's inability to properly model them does not appreciably affect the accuracy of this model. This modeling strategy also has the advantage of simplifying the alteration of the flange thickness programmatically.

Shell thickness ?

CQUAD can be bent

no middle

( ) ~~Are~~ all elements in 2D,

Selected Model

The model developed for the example problem considered in this work is shown in figure 2.1. This model was developed using the dimensional data shown in figure 1.2. As shown, the model consists of several regions, all of which are modeled using CQUAD4 elements. The regions that are of importance for this study are:

flange
Do you use CQUADs to model flange

1. The top flange
2. The bottom flange
3. The web
4. A thickened region surrounding the hoist mounting lug
5. A thickened region surrounding the load mounting lug.

buckling

CQUAD4 does have bending stress

buckling happens when the structure is subjected to compression.

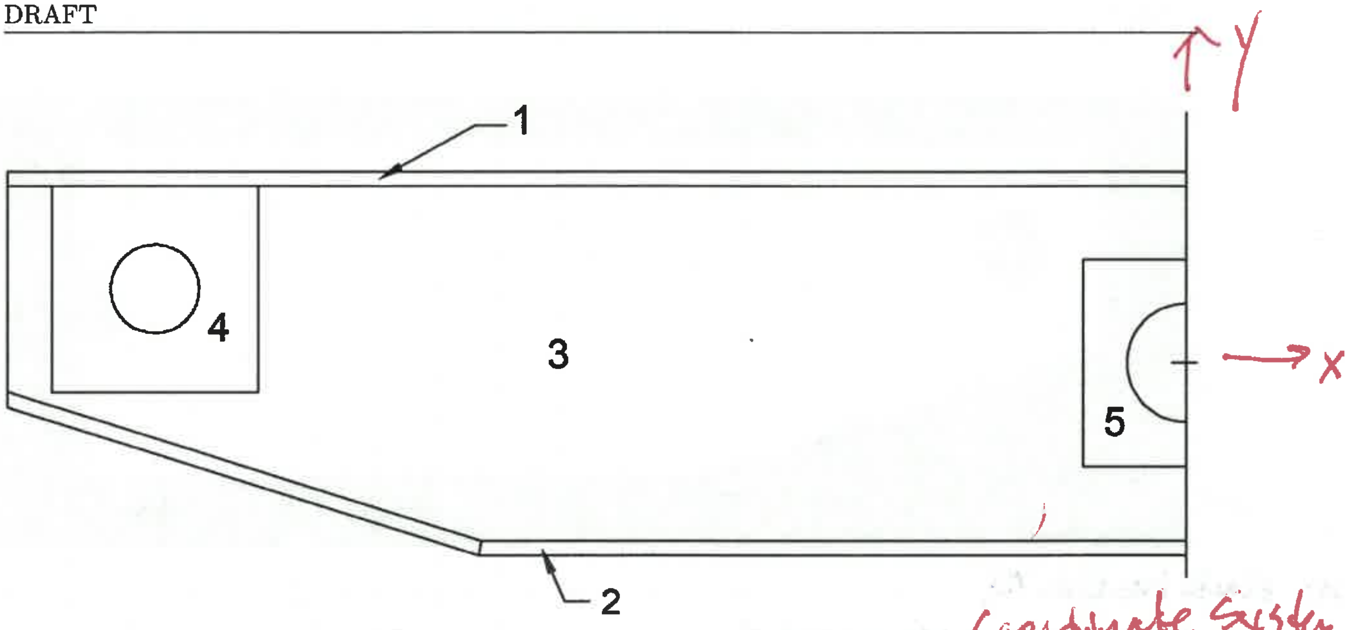


Figure 2.2: Diagram of modeled beam showing region numbers

Coordinate System

Figure 2.2 shows the named regions as they correspond to portions of the model/drawing.

Also worth noting is that this model is a partial representation of the whole beam. As the example beam in figure 1 shows, the beam features symmetry along the front view. Due to the typical construction of these beams, they also exhibit symmetry along the top view as well. Because of this, the model presented is only one quarter of the complete beam. To account for this, constraints are imposed along the symmetry cut that simulate the remainder of the beam. These constraints can be seen in figure 2.1 as symbols along the right edge of the diagram. These constraints are along model axes 1, 3, 4, and 5. This equates to constraining X and Z translation, as well as X and Y rotations. This simulates a moment bearing connection at the center of the beam. This is a commonly accepted way of simulating symmetry in NASTRAN.

Needs Citation

singularity along y translation ??

2.4 2 The Reliability Index of Van Mises stress

The reliability index is a unitless constant that can be used to infer the probability of a system or component to fail in service. Roughly, the Reliability index describes the safety factor of a system as the Z-score of a standard normal distribution, where higher numbers indicate higher probabilities of success. The Reliability Index of a loading condition can be written as:

$$\beta = \frac{\mu_{Sy} - \mu_\sigma}{\sqrt{\sigma_\sigma^2 + \sigma_{Sy}^2}} \tag{2.2}$$

where mean and standard deviation of stress are calculated by Eqs. 2.3, 2.4

The reasons that stress is random is because that the loading are random.

Where

μ_{Sy} = Mean of the material yielding stress

σ_{Sy} = Std. Deviation of the yielding stress

μ_σ = Mean of the Von Mises stress resulting from the applied load

σ_σ = Std. Deviation of the Von Mises stress resulting from the applied load

The first two variables in the list above relate to properties of the material the object is made from, and are therefore given or assumed. The second two, however, relate to the von Mises stress and must be calculated.

Finding μ_σ and σ_σ

If $g(x_1, x_2, ...x_n) = Y$ denotes a function of multiple random variates , x_i, and a single, dependent variable, let: cite

mean
$$\mu(Y) = g(x_1, x_2, ...x_n) + \frac{1}{2}\sum_{i=1}^{n}\left(\frac{\partial^2 g}{\partial x_i^2}\sigma_{xi}^2\right) \quad (2.3)$$

standard deviation
$$\sigma^2(Y) = \sum_{i=1}^{n}\left(\frac{\partial g}{\partial x_i}\sigma_{xi}\right)^2 + \frac{1}{4}\sum_{i=1}^{n}\left(\frac{\partial^2 g}{\partial x_i^2}\sigma_{xi}^2\right)^2 \quad (2.4)$$

evaluate at the mean of P_x P_y

and σ_{x_i} is σ_{P_x} and σ_{P_y}

The tensor definition of the von Mises stress at a given location can be written as:

$$\sigma = \sqrt{\frac{3}{2}\cdot(\sigma_{dev} : \sigma_{dev})} \quad (2.5)$$

Where σ_{dev} is the stress deviator tensor, which will be more completely addressed later. We can simplify derivation of equation 2.5 by hiding the tensor contraction in the equation with a placeholder variable:

$$\alpha = (\sigma_{dev} : \sigma_{dev}) \quad (2.6)$$

This turns equation 2.5 into:

$$\sigma' = \sqrt{\frac{3}{2}\cdot(\alpha)} \quad (2.7)$$

standard deviation of P_x and P_y

This simplified equation can be derived as shown below:

$$\frac{\partial\sigma'}{\partial P_x} = \frac{3}{4}\frac{\partial\alpha}{\partial P_x}\left(\frac{3}{2}\alpha\right)^{-\frac{1}{2}} \quad (2.8)$$

$$\frac{\partial^2\sigma'}{\partial P_x^2} = \frac{3}{4}\frac{\partial^2\alpha}{\partial P_x^2}\left(\frac{3}{2}\alpha\right)^{-\frac{1}{2}} - \frac{9}{16}\left(\frac{\partial\alpha}{\partial P_x}\right)^2\left(\frac{3}{2}\alpha\right)^{-\frac{3}{2}} \quad (2.9)$$

$$\frac{\partial\sigma'}{\partial P_y} = \frac{3}{4}\frac{\partial\alpha}{\partial P_y}\left(\frac{3}{2}\alpha\right)^{-\frac{1}{2}} \quad (2.10)$$

$$\frac{\partial^2\sigma'}{\partial P_y^2} = \frac{3}{4}\frac{\partial^2\alpha}{\partial P_y^2}\left(\frac{3}{2}\alpha\right)^{-\frac{1}{2}} - \frac{9}{16}\left(\frac{\partial\alpha}{\partial P_y}\right)^2\left(\frac{3}{2}\alpha\right)^{-\frac{3}{2}} \quad (2.11)$$

Of course, this introduces derivatives of α as values that must be calculated. In order to calculate these derivatives, the concept and application of the deviator tensor must be investigated further.

The Deviatoric Stress Tensor (or deviator tensor) describes the component of stress that tends to deform an element. It is given in terms of the overall stress tensor σ' as:

$$\sigma_{dev} = \sigma' - \frac{1}{3}\mathrm{tr}(\sigma')\,[\mathbf{I}] \tag{2.12}$$

For purposes that will become clear later, we can call this operation a matrix operator γ:

$$\begin{aligned} \gamma(x) &= x - \frac{1}{3}\mathrm{tr}(x)\,[\mathbf{I}] \\ \sigma_{dev} &= \gamma(\sigma') \end{aligned} \tag{2.13}$$

In this study, σ' is constructed from the component response tensors, σ_{Px} and σ_{Py}:

Explain earlier in the paper how these values come to be.

$$\sigma' = \sigma_{Px} \cdot P_x + \sigma_{Py} \cdot P_y$$

Applying equation 2.13 to the above definition of σ' yields:

$$\gamma(\sigma') = (\sigma_{Px} \cdot P_x + \sigma_{Py} \cdot P_y) - \frac{1}{3}\mathrm{tr}\left(\sigma_{Px} \cdot P_x + \sigma_{Py} \cdot P_y\right)[\mathbf{I}] \tag{2.14}$$

From here, it is important to remember that taking the trace of a matrix is a distributive operation, as is multiplying a scalar and a matrix. Therefore, the above equation can be rewritten as:

$$\begin{aligned} \gamma(\sigma') &= (\sigma_{Px} \cdot P_x + \sigma_{Py} \cdot P_y) - \frac{1}{3}\left(\mathrm{tr}\left(\sigma_{Px} \cdot P_x\right) + \mathrm{tr}\left(\sigma_{Py} \cdot P_y\right)\right)[\mathbf{I}] \\ &= (\sigma_{Px} \cdot P_x + \sigma_{Py} \cdot P_y) - \frac{1}{3}\left(\mathrm{tr}\left(\sigma_{Px} \cdot P_x\right)[\mathbf{I}] + \mathrm{tr}\left(\sigma_{Py} \cdot P_y\right)[\mathbf{I}]\right) \\ &= (\sigma_{Px} \cdot P_x + \sigma_{Py} \cdot P_y) - \frac{1}{3}\left(P_x \cdot \mathrm{tr}\left(\sigma_{Px}\right)[\mathbf{I}] + P_y \cdot \mathrm{tr}\left(\sigma_{Py}\right)[\mathbf{I}]\right) \\ &= P_x \cdot \left(\sigma_{Px} - \frac{1}{3}\mathrm{tr}(\sigma_{Px})\,[\mathbf{I}]\right) + P_y \cdot \left(\sigma_{Py} - \frac{1}{3}\mathrm{tr}(\sigma_{Py})\,[\mathbf{I}]\right) \\ &= P_x \cdot \gamma(\sigma_{Px}) + P_y \cdot \gamma(\sigma_{Py}) \end{aligned} \tag{2.15}$$

Note that the terms $\gamma(\sigma_{Px})$ and $\gamma(\sigma_{Py})$ do not contain any terms related to P_x or P_y. This implies that they are constant when deriving with respect to these variables. From this point forward, we will refer to these deviatoric unit tensors as:

$$\begin{aligned} \gamma(\sigma_{Px}) &= \sigma_{devx} \\ \gamma(\sigma_{Py}) &= \sigma_{devy} \end{aligned}$$

This means:

$$\sigma_{dev} = \sigma_{devx} \cdot P_x + \sigma_{devy} \cdot P_y \tag{2.16}$$

Returning to equation 2.6, we can begin to define α in terms of these new tensors. To start with, we need to breakdown the ":" operator in the equation. This is the tensor contraction operator, and is defined as:

$$[A] : [B] = \text{tr}([A]\,[B]) \tag{2.17}$$

Because both matrix multiplication and the trace of a matrix are both distributive, the tensor contraction operator is also distributive. Using the definition in 2.17 and applying it to 2.6 yields:

$$\begin{aligned}\alpha &= \text{tr}\left[(\sigma_{devx} \cdot P_x + \sigma_{devy} \cdot P_y) \cdot (\sigma_{devx} \cdot P_x + \sigma_{devy} \cdot P_y)\right] \\ &= \text{tr}\left[\sigma_{devx} \cdot \sigma_{devx} \cdot P_x^2 + (\sigma_{devx} \cdot \sigma_{devy} + \sigma_{devy} \cdot \sigma_{devx}) \cdot P_x P_y + \sigma_{devy} \cdot \sigma_{devy} \cdot P_y^2\right] \end{aligned} \tag{2.18}$$

Note that the center term is arranged as shown due to the non-commutative nature of matrix multiplication. Now, we can take this definition of α and easily find the derivatives we need.

$$\frac{\partial \alpha}{\partial P_x} = \text{tr}\left[\sigma_{devx} \cdot \sigma_{devx} \cdot 2P_x + (\sigma_{devx} \cdot \sigma_{devy} + \sigma_{devy} \cdot \sigma_{devx}) \cdot P_y\right] \tag{2.19}$$

$$\frac{\partial^2 \alpha}{\partial P_x^2} = \text{tr}\left[2 \cdot \sigma_{devx} \cdot \sigma_{devx}\right] \tag{2.20}$$

$$\frac{\partial \alpha}{\partial P_y} = \text{tr}\left[\sigma_{devy} \cdot \sigma_{devy} \cdot 2P_y + (\sigma_{devx} \cdot \sigma_{devy} + \sigma_{devy} \cdot \sigma_{devx}) \cdot P_x\right] \tag{2.21}$$

$$\frac{\partial^2 \alpha}{\partial P_x^2} = \text{tr}\left[2 \cdot \sigma_{devy} \cdot \sigma_{devy}\right] \tag{2.22}$$

With equations 2.19 through 2.22 coupled with equations 2.8 through 2.11, we have all of the terms needed to construct and solve equations 2.3 and 2.4 for σ. It is then easy to substitute the results of equations 2.3 and 2.4 into 2.2 and determine β. In the code presented with this report, all of these terms are assembled separately and combined. Therefore, the final equation with all substitutions performed will not be shown here.

CHAPTER 3

TOOLS AND SOFTWARE USED

For this study several externally developed or commercially purchased tools, hardware, and software were used. Each major tool or piece of hardware will be briefly introduced and given a brief description of its source, purpose and method of procurement.

3.1 Computing Hardware

The analyses presented in this report were obtained using commercially available computing hardware. The computer's relevant specifications are given below:

Cite the spec sheet for These parts!

- CPU: AMD Ryzen 2400G. 4 Processing Cores with 8 execution threads. Frequency during tests: 3.8GHz
- Memory: 16GB DDR4 Memory at a frequency of 2666 MHz
- Storage: Intel 540 Series SSD. Capacity:240GB, up to 540 MBps read speed, 490 MBps write

While this computer is a general purpose unit and sees daily use outside the scope of this report, no other tasks were performed simultaneously with the workloads presented herein. Execution times presented represent near-maximum performance for these workloads.

3.2 Software

NASTRAN

The name NASTRAN is short for **NASA Structural Analysis**. It is a finite element solver originally developed by NASA. It enjoys wide popularity throughout industry and sees use for performing linear and nonlinear structural analyses on a variety of materials.

The version of NASTRAN employed in the code presented herein is a recently open-sourced copy of the NASTRAN95 solver. As the name implies, it is a version originally developed in 1995. While newer versions of this software exist, this is the newest copy that is freely available. For simple solutions such as those needed for this work, NASTRAN95 is functionally very similar to the more modern commercial utilities available. This version of NASTRAN is freely available on GitHub.

Whole section needs citations.

msslhs

msslhs is a Python utility that provides latin hypercube sampling in several locations throughout the code presented. This utility is created and maintained by Justin Hughes. It is available from the Mississippi State University CyberDesign Wiki [4].

CHAPTER 4

SOLUTION METHODS

→ outline of chapter 4

4.1 Modeling the base beam

For both solution methods, a common basic design of the beam to be studied was constructed. For this study, a basic design using two "C"-style sections to form the two main moment-bearing sections was selected. This style was selected because the cross section lends itself to parameter-based optimization, whereas more complex designs such as box beam spreaders are more well suited to full geometry optimization. Section 1.2 has details on the design.

This basic design was modeled in Siemens's Finite Element Pre-processor, FEMAP. The modeled area takes advantage of the two-plane symmetry in the beam to only model one quarter of the beam. It is worth noting that the loading on the beam is affected by the symmetry, and the loads presented for analysis are one fourth of the actual beam loadings.

4.2 Solution Strategies

Two solution strategies were attempted for the example problem proposed. One was termed as the *Aggregated Latin Hypercube Sampling Approach*, while the other was named the *Stochastic Loads Approach*. The following sections describe each solution method in detail and outline similarities and differences between them.

(or worst load case: cover 97.5% of all possible load cases)

4.2.1 Aggregated Latin Hypercube Approach

In a nutshell, the Aggregated Latin Hypercube (ALHS) Approach analyzes a variety of discrete load cases and collects Pareto Front data for each load case. It does this by:

[illegible] by ALHS

1. Identifying Load Cases to consider

2. Performing MODE Optimization for each identified load cases. This generates a unique Pareto Front for each load case.

3. Aggregating the Pareto Fronts from all load cases to find the overall best designs across all load cases.

Each individual step of this process is outlined in detail below.

Identify Load Cases

In order to find load cases for use, a Latin Hypercube Sampling algorithm is used to select 1000 independent loading conditions within the set of possible load conditions, which is assumed to be a set of normally distributed sample spaces. In this case, the most conservative loadings are those significantly above the mean for the magnitude of the applied force. In order to ensure reliability, the set of load cases to be analyzed is restricted to those load cases more than 1.96 standard deviations above the mean value of the applied load. This ensures that all of the load cases considered are in the top 2.5% of the sample space. This will result in a set of approximately 25 load cases to consider.

May need to describe this with more mathematical vigor

Analyze load Cases

For each load case selected, a set of 50 randomly selected candidate designs are generated. These designs are analyzed using Finite Element Analysis to find the maximum stress and design weight. These values are used as fitness functions to rank the designs. This process is repeated using a Multi Objective Differential Evolution algorithm to selectively refine the designs for a set number of generations.

Section needs expanding

moved to page 5

Constraints

During the MODE optimization process, 1 constraint is applied. The constraint requires the mass of the modeled beam portion to remain under 1 metric ton (1000kg). In order to accomplish this, a fitness penalty method is used which triggers if the inequality:

$$W_{beam} \leq 1000kg$$

is violated.

The above constraint is applied by generating a multiplier based on the degree to which the constraint has been violated. This multiplier is used to multiply the fitness results, ensuring the solutions that violate constraints are the least dominant and are therefore much less likely to be selected to move on to the next generation in favor of more dominant designs.

Aggregation

At this point, the solution algorithm has 25 separate load cases with individual Pareto fronts. The designs that make up each Pareto front are added to a single unified set of designs. From there, they are once again ranked according to dominance and a "final" Pareto front is generated. The designs that comprise this "final" Pareto front are considered to be the most desirable designs and are presented to the designer.

4.2.2 Stochastic Loads Approach

The stochastic loads approach is functionally similar to the Aggregated Latin Hypercube Sampling Approach, with a few key differences:

1. Only a single load case is considered. ?
2. Stress is not directly considered a fitness variable., but the reliability of failure due to stress ~~[illegible]~~ it is considered

The overall process is detailed below.

Load Case

Instead of using multiple load cases, this approach uses a single load case. However, it is specified as a stochastic set of loads instead of discrete loads applied. For example, the load case considered for the Example System was:

Parameter	Mean Value	Standard Deviation
Hoist Load X Direction	0N	5kN
Hoist Load Y Direction	150kN	19.5kN

Figure 4.1: Stochastic Parameters for Example System

$\mu_{P_X} = 0$ $\sigma_{P_X} = 5$

$\mu_{P_Y} = 150$ kN $\sigma_{P_Y} = 19.5$

in Eq. 2.3

Specifying the load case this way effectively covers the entire performance envelope and makes the single load case valid for all states of load on the beam.

Uniform

Analyze Load Case

The actual analysis of the load case selected is nearly identical to the ALHS Approach. The key difference is the use of the unitless parameter β to represent the stress on the beam. This allows the stress, and by inference the safety factor of the device, to be represented in stochastic terms. For a complete discussion and derivation of the parameter β, see section 2.4. Using β and the component weight as fitness measurements, MODE is performed on the load case.

what is the value of β ?

reliability value ?

Reporting

MODE optimization on the load case defined above generates a single Pareto Front that is reported to the user as recommended designs. Also reported is the Pareto Front presented graphically.

CHAPTER 5

SELECTED RESULTS

CHAPTER 6

CONCLUDING REMARKS

CHAPTER 7

REFERENCES

[1] Steven C. Chapra. *Applied Numerical Methods with MATLAB for Engineers and Scientists.* second. McGraw-Hill, 2008.

[2] Dictionary.com. *Optimization — Define Optimization at Dictionary.com.* 2019. URL: `https://www.dictionary.com/browse/optimization`.

[3] Andries P. Engelbrecht. *Computational Intelligence: An Introduction.* second. John Wiley and Sons, Ltd, 2007.

[4] Justin M. Hughes. *Latin Hypercube Sampling (LHS).* 2017. URL: `https://icme.hpc.msstate.edu/mediawiki/index.php?title=Latin_Hypercube_Sampling_(LHS)`.

[5] M. D. McKay, R. J. Beckman, and W. J. Conover. “A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code”. In: *Technometrics* 21.2 (1979), pp. 239–245. ISSN: 00401706. URL: `http://www.jstor.org/stable/1268522`.

CHAPTER 8

APPENDIX A: COMPLETE CODE LISTING