Interoperable Software for Parametric Structural Analysis and Optimization

by

Garrett P. Jones

B.S., University of Texas at Austin, 2012

Submitted to the Department of Civil and Environmental Engineering in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Civil and Environmental Engineering

at the

Massachusetts Institute of Technology

June 2013

Signature of Author: ____

Department of Civil and Environmental Engineering
May 10th, 2013

Certified by: ____

Jerome J. Connor
Professor of Civil and Environmental Engineering
Thesis Supervisor

Certified by: ____

Rory Clune
Massachusetts Institute of Technology
Thesis Reader

Accepted by: ____

Heidi Nepf
Chair, Departmental Committee for Graduate Students

Interoperable Software for Parametric Structural Analysis and Optimization

by

Garrett P. Jones

Submitted to the Department of Civil and Environmental Engineering on May 10, 2013 in Partial Fulfillment of the Requirements for the Degree of Master of Engineering in Civil and Environmental Engineering

Abstract

The advent of building information modeling in the structural engineering profession has brought forth new challenges to the traditional methods of design and analysis. The need for faster, more robust analyses to mitigate expenses and increase structural insight is a demand that stems from the implementation of BIM modeling. Current software interoperability now allows engineers limited opportunity to engage directly and immediately with the design process. The development of tools which can bring together the architectural and structural engineering professions are of paramount importance in the next phase of professional design.

In response to this professional demand, a software framework for Rhino3D modeling software was created which explores the various methods of searching a design space and finding solutions. Both parametric design generation and genetic optimizations were employed, allowing architects and engineers to explore the design space of a structure using metrics important to each field. A case study is performed using the developed software framework to quantify results and validate the effectiveness of such a new design tool in the current engineering profession. The outcome is an improved design experience that is feasible in time and scope, allowing architects and engineers an opportunity to truly explore the design space.

Keywords: Parametric modeling and analysis, Genetic optimization, Building information modeling

Thesis supervisor: Jerome J. Connor

Title: Professor of Civil and Environmental Engineering

Acknowledgements

First, I would like to thank the faculty at MIT for their tireless efforts and incredible contributions to my education and learning. I want to extend a deep, personal gratitude to Dr. Connor and Rory Clune for their excellence in teaching and mentoring.

Second, I would like to thank all of my peers in the CEE MEng class of 2013 for their support and friendship throughout the year. They are an incredible group of people, and each and every one will be undoubtedly successful in structural engineering. It has been an honor to learn and grow aside them.

Third, and most important, I would like to thank my family. To Randy, I extend my thanks for being an amazing friend and incredible family member. To my grandparents, I want to give praise for the constant support provided to our family. To my brother I give thanks being an outstanding source of inspiration and shoulder to lean on when I became mentally fatigued. To my mother, who has shown me the true meaning of sacrifice and unconditional love, I want to express my deepest recognition. Her solitary effort to raise my brother and I is the most impressive act of grace and passion I will ever see. I would not be writing this thesis today without her.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 History and Development of the Structural Engineering Practice

The practice of structural engineering as it is currently performed is changing. As the Boston Society of Civil Engineers has observed, the industry is moving towards a more complex practice with a greater use of and dependence on computational power. From the advent of advanced computer methods and modeling techniques such as Computer Aided Design (CAD) in the late 1950's and 1960's to the introduction of Building Information Modeling (BIM) in the last decade, the design practice has moved into a realm that is expanding in scope and interoperability with other disciplines. At a time where cost strains all aspects of an engineer's job, however, methodologies need to be developed that afford engineers the ability to expand their practice while remaining financially responsible. The strain of this design need can be seen in the interaction between engineers and architects.

For a considerable amount of time, it was common practice to bring the structural engineers into the overall design discussion at a stage in which the architecture was set to a preliminary degree. The client would hire an architecture firm, who would then bring on a team of engineers, contractors, and construction managers to breakdown the tasks of completing the client's initiatives. This method of practice has changed, and as a consequence the interaction between architects and structural engineers has changed as well. The advent of Building Information Modeling, or BIM, has pushed for less stratification of the two design practices and a coherency between architects and engineers. BIM seeks to mitigate issues of interdisciplinary work by creating object oriented models that can share information pertinent to all parties involved in design. The promises of increased speed of work, better coordination, higher productivity, and cost savings are all tenants of a BIM-oriented project (Autodesk, 2003). Since BIM's arrival in the structural engineering profession, the benefits of information modeling have been studied by various research groups such as The National Institute of Standards and Technology (NIST), with a focus largely on the savings a BIM-oriented project may have versus a more traditional design-build scheme. Specific studies show that the current building industry may create additional costs upwards of 6 dollars per square foot for a new building design from project inefficiencies (Eastman, 2008). As can be seen in the following figure, the use of BIM technologies with emphasis and promotion of initial design drivers will help maximize savings on a project.

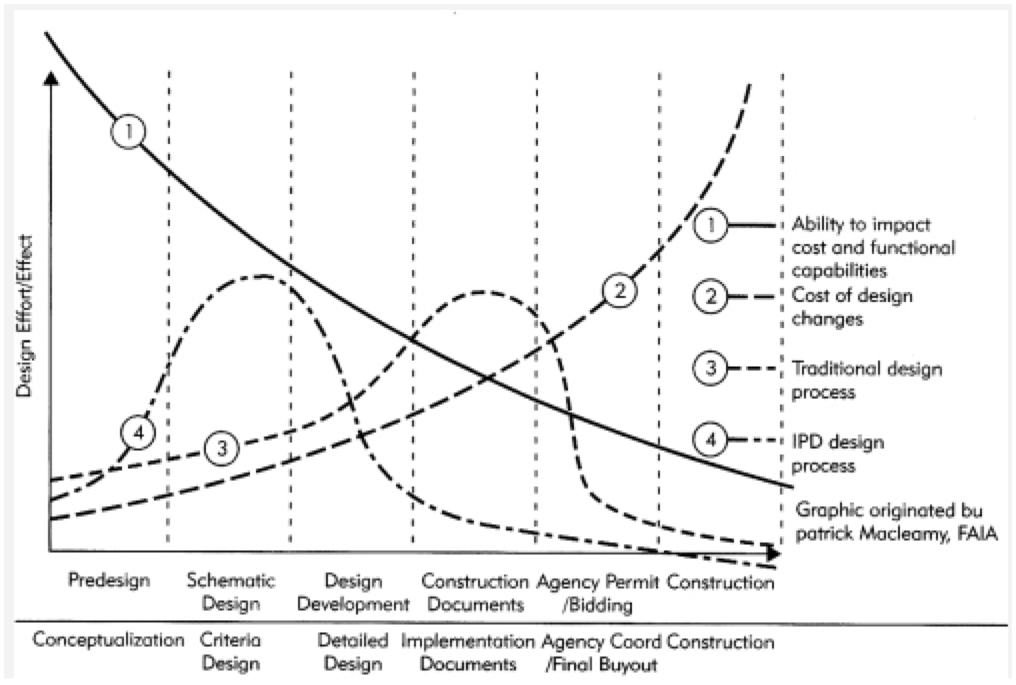


Figure 1: Typical Design Process and Integrated Project Delivery (IPD) or BIM Methods and Consequent Project Cost Impact (Smith, 2009)

If there is to be increased design in the early stages of a project, however, a new burden falls on the design team to increase production during initial interdisciplinary explorations. Current design practice methods do not allow for this increase in early production at an effective rate. While BIM has certainly played a role in the overall project lifecycle, the schematic design and design development stages still rely on a back-and-forth methodology to come to consensus on a design package.

1.2 Identification of Problem and Proposed Solution

The task that this thesis attempts to solve is one of integration of disciplines. In order to allow for increased design exploration and evaluation at the early stages of a project, a new or remodeled design process needed to be developed. Before a solution could be had, however, the tools used by architects and structural engineers needed to be mapped out in order to understand how a new design process could be developed to increase productivity and performance. Rhino3D, a three-dimensional design platform that enables a wide variety of modeling, has in recent years been developed through the use of the Grasshopper plugin (Grasshopper, 2011). Grasshopper is an open-source, module based designer that allows the user to script designs on many tiers of development. For structural engineers, SAP2000 is a common finite element modeling and analysis tool that has a robustness in terms of computational capabilities and data output. Currently, each of these tools is used separately. For schematic design, a typical process is as follows:

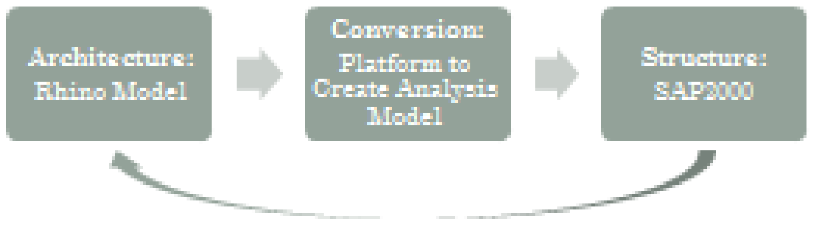

Architecture: Rhino Model
Conversion: Platform to Create Analysis Model
Structure: SAP2000

Figure 2: Typical Iterative Design Process Between Architect and Engineer

This process is linear, and will not typically allow for the amount of design needed to push the project into a more efficient process in the long term. Each architectural model must be transferred to the engineers, who then create an analytical model to critique the structure's performance. The engineers then pass their observations back to the architect for revision of the initial model, and the process repeats until a final form is found. This back-and-forth interaction takes time. Thus, the issue becomes one of combining this dual-discipline process into a collaborative effort that has the robustness to explore design and depth to provide insight to the furthering of the project structure and architecture. This thesis thus develops a software framework that combines both Rhino modeling software and SAP2000 analysis software to create a unified process to explore the design space at an early stage of development. With the implementation of this program, which will be discussed in the following pages, the new design process takes a different form:

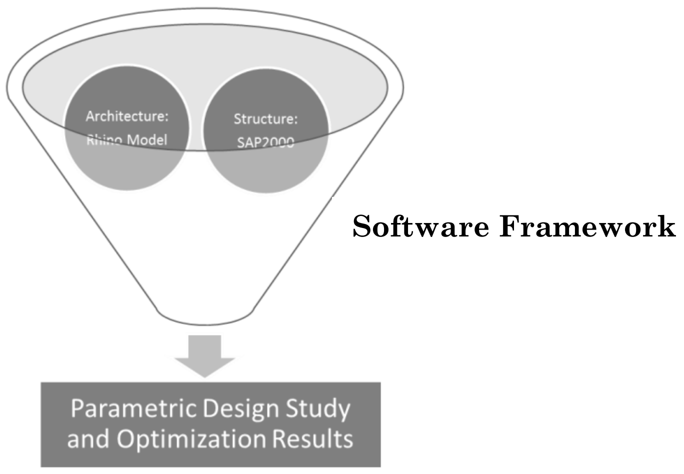

Architecture: Rhino Model
Structure: SAP2000
Software Framework
Parametric Design Study and Optimization Results

Figure 3: Proposed Integrated Design Process

This process, when complete, allows design teams to make decisions based on an array of data that is important to both the engineers and architects. In effect, it increases the robustness of the schematic design, allowing for a consequent increase in depth of exploration to meet new standards of practice.

1.3 Thesis Outline

The process for developing a parametric modeling software package with optimization capabilities relies on creating a wrapper to bring both Rhino and SAP2000 together with the ability to explore design options. Chapter 3 illustrates the methodology behind this software framework design, where the first step is creating a Grasshopper plugin that is easy to use and understand. Once this plugin is created, the necessary output for analysis must be gathered and stored. This data then passes to a routine which, when activated, creates parametric models based on initial user input. The results of the analyses are then represented in a meaningful way for understanding and interpretation. Additionally, an algorithm is developed to explore the design space and find an optimal solution based on various criteria. The results of the parametric modeling and optimization routine are then open for comparison. It can be seen from the results in Chapter 4 that the optimization scheme and parametric analyses complement each other. The optimization process, using the inputs defined for the parametric analysis, was observed to converge on the globally-optimal design as mapped by the parametric routine.

Chapter 2

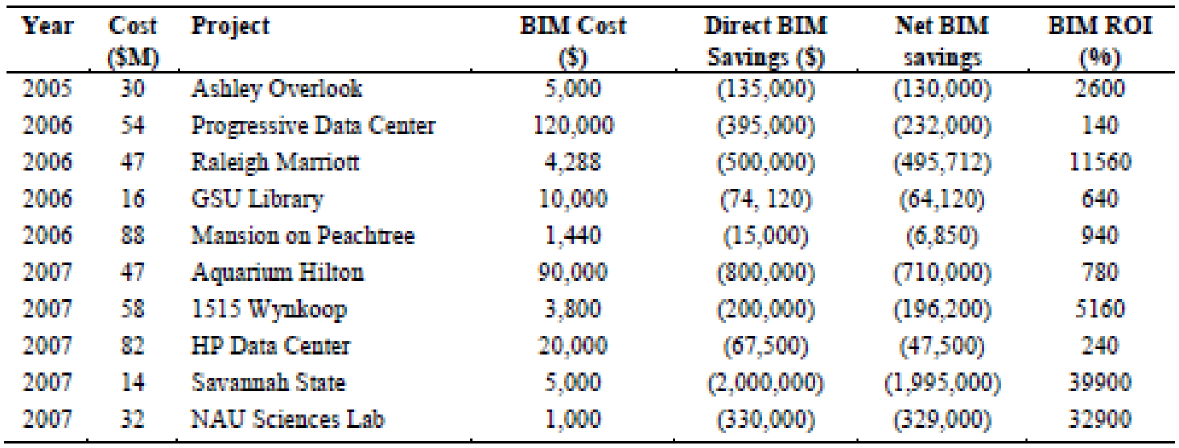
Literature Review

2.1 Chapter Overview

The intent of this chapter is to provide a background to the practice of structural engineering and the role it plays with Building Information Modeling. Literature on the development of parametric modelers and optimization routines are discussed to present the current state of such analytical methods in structural engineering practice. The strengths and limitations of existing tools are described in detail and the effects of these tools are analyzed through a number of case studies in both research and practice.

2.2 Building Information Modeling

The exploration of parametric modeling and genetic algorithm optimization has increased in recent years, with many research entities and companies alike developing tools and methods to create a more robust and intelligent practice. Seeking to push the design and development standards as they stand into new, more efficient methods adheres to the intent of BIM modeling practices. BIM, as mentioned briefly in Chapter 1, seeks to create multi-purpose models for interdisciplinary work. In doing so, transfer of information becomes much more efficient, creating multiple opportunities to save time and money. Many universities have studied the benefits of BIM-oriented projects, and have seen substantial results in terms of both time and monetary savings. Researchers at Auburn University dissected 10 United States projects from 2005 to 2007 that implemented BIM technologies and strategies. They noted that each project received a significant amount of net savings as well as rates of return on the investments put forth for said projects (Salman et. all, 2008). The reasons, the authors describe, for savings on a range of projects that included hotels, libraries, data centers, and laboratories, are the integration of the architecture, engineering, and construction disciplines. Such integration promotes "faster and more effective processes," and "better design." The designs allow for building proposals that "can be rigorously analyzed [where] simulations can be performed quickly and performance benchmarked, enabling improved and innovative solutions" (Salman et. all, 2008).



Year	Cost ($M)	Project	BIM Cost ($)	Direct BIM Savings ($)	Net BIM savings	BIM ROI (%)
2005	30	Ashley Overlook	5,000	(135,000)	(130,000)	2600
2006	54	Progressive Data Center	120,000	(395,000)	(232,000)	140
2006	47	Raleigh Marriott	4,288	(500,000)	(495,712)	11560
2006	16	GSU Library	10,000	(74, 120)	(64,120)	640
2006	88	Mansion on Peachtree	1,440	(15,000)	(6,850)	940
2007	47	Aquarium Hilton	90,000	(800,000)	(710,000)	780
2007	58	1515 Wynkoop	3,800	(200,000)	(196,200)	5160
2007	82	HP Data Center	20,000	(67,500)	(47,500)	240
2007	14	Savannah State	5,000	(2,000,000)	(1,995,000)	39900
2007	32	NAU Sciences Lab	1,000	(330,000)	(329,000)	32900

Figure 4: BIM Economics (Salman et. all, 2008)

A key component to the success of BIM has been the impact of parametric, three dimensional modeling. Issues of deliverable speed, coordination amongst design parties, and productivity, have become intrinsically woven into the interoperability of parametric modeling (Autodesk, 2003). The simple schematic below illustrates the shift in paradigm, from traditional linear design processes to compact and iterative solution networks. Objectives of multiple parties have the ability to influence the final solution, or building product, with the least compromise in cost and effort spent.

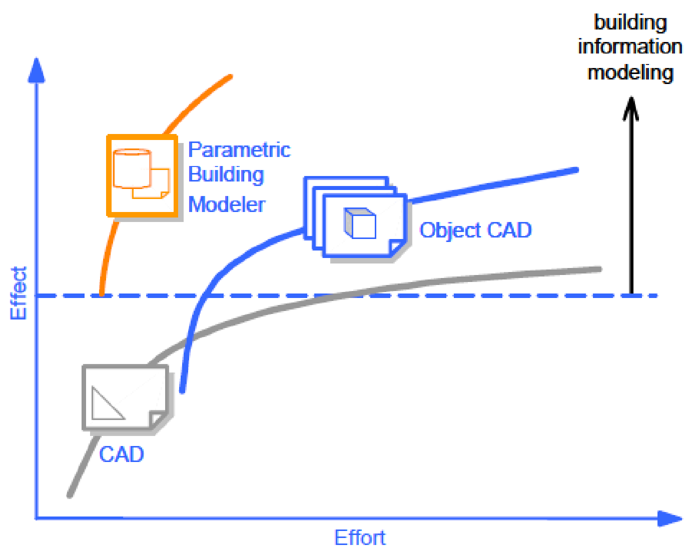


Figure 5: Standard CAD Practice vs. BIM Design Parametric and Object-Based Initiatives (Autodesk, 2003)

The philosophy of creating a system in which multiple sets of data from across the design spectrum are accrued and manipulated in a parametric study is a crucial to the effectiveness of BIM implementation.

Specific studies have gone as far as to measure the total impact of parametric modeling on the engineering profession, looking to validate not only the benefits associated with cost, but with time and productivity as well. Observations on the design and development of three concrete structures ranging from 5000 to 10000 cubic meters in volume give insight towards the total amount of time saved when implementing parametric modeling versus standard practice. The benefits vary depending on the size of the structure and the consequent amount of information a model must store, but regardless savings of time range from 21% to 61%. Additional research also shows a balance in percent of work done on a design project between the architect and structural engineer (Sacks, 2005). A smoother distribution of work allows for both parties to participate fully in the design decisions that are imperative to the success of a project. This research did have its limits, however, as the initial geometry of the various concrete structures was provided as a template. Thus, the design initiatives were already largely set, and the parametric study provided means for optimization. This thesis aims to take parametric modeling a step backwards, allowing it to search the design space to help define the key variables of a design that should be optimized. In doing so, greater influence is placed on finding the correct design initiatives for a successful design rather than attempting to simply make an existing design better.

2.3 Parametric Modeling

With BIM becoming a mainstay in the work environment, and the studied benefits of parametric modeling in such a process, it becomes a key objective to define the notion of parametric modeling and study its purpose. Parametric modeling entails studying "quantities of interest which can be varied over a domain specified by the user" (Sarkisian et. all, 2012). Specifically, parametric modeling allows a design team to explore a range of options or variations of an idea or concept. The intent is to determine what is important in the design and how different variables affect one another through manipulation in the design space. Parametric modeling alone allows for an increase in the scope of conceptual studies and the speed in which data is extracted from them. Such modeling is crucial to the creation of creative solutions. This process does however depend heavily on a thorough exploration early in the design process. When a problem is still in its infancy, the parameters that convey the concept of the designers are neither fixed nor variable. Once the design team begins to fix certain aspects of the concept, the design space shrinks, narrowing the variability of the remaining free parameters and consequently the scope of possible optimal solutions. Thus, the architect and engineer should explore as many possibilities of the concept in as many directions as they have time. If a proper search of the design space is done, the architect and engineer will have a much greater chance of determining what parameters are important and ultimately what solution is the best (Buelow, 2008). The benefits of breadth of exploration are a primary reason as the incorporation of a parametric analysis tool in this thesis' proposed software framework.

The German firm Knippers Helbig Advanced Engineering is an excellent example of the power permitted behind embracing parametric design. The firm is notorious for attempting complex, unique, and highly inter-disciplinary projects and making them feasible. One example of fixed form parametrization is the Bao'an International Airport. Knippers was brought on to remedy the issue of the creation of the structure's double curved façade. Implementing an unique combination of Rhino and Excel platforms, the firm was able to create parameters for observation and study of the form, allowing for the creation of the façade panels in an efficient and cost effective way. The figure below shows how the numerous variables controlling design were discretized into analytical boundaries (Scheible, 2011).

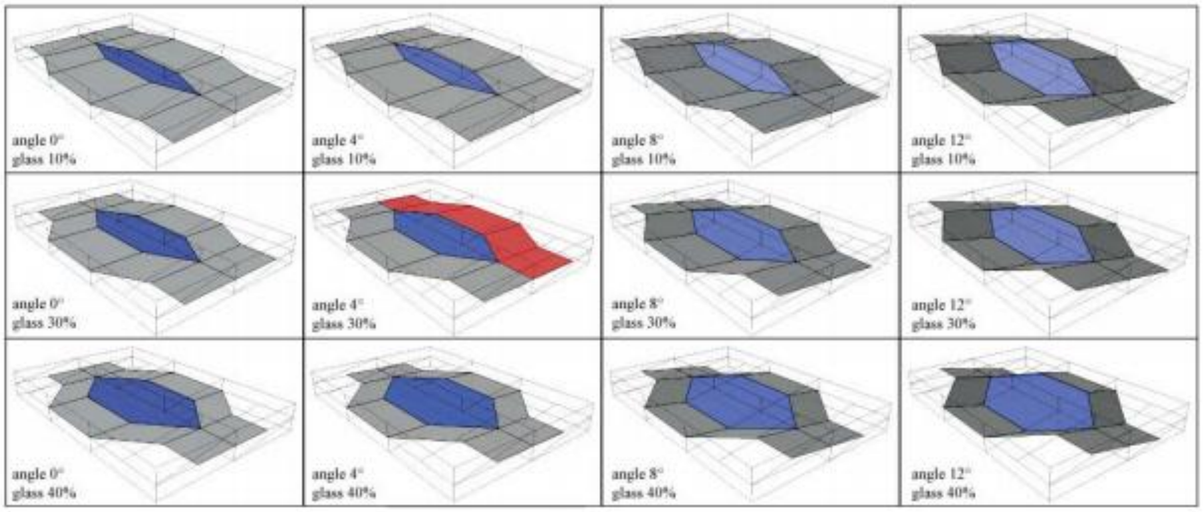


Figure 6: Parametric Modeling of Bao'an International Airport (Scheible, 2011)

Without using optimization techniques, the Knippers team was able to study the boundaries set by material use and cost. The firm was able to create a doubly curved façade that had planar elements of glass paneling. This allowed for both a decrease in production time and client costs. The firm also incorporated non-structural parameters to increase the overall performance of the structure. A daylighting tool was developed and implemented in Rhino to allow for a parametric study of the passive light, and thus heat, gained by the building across a typical day. Thus, the final design was a synthesis of controlled constructability, daylighting, and cost (Scheible, 2011).

Figure 7: Bao'an Facade Composed of Parametric Components (Scheible, 2011)

2.3 Structural Optimization

Genetic Algorithms are excellent methods of optimization in computer aided design due to their ability to solve complex optima problems in engineering. They provide a depth to compliment the breadth of parametric design exploration. By maintaining the laws of nature, a population of possible solutions is tested for their corresponding "fitness." These fitness scores are then used to filter the population until an optimal design is reached. This optimum is reached through the iteration of generations that utilize stochastic methods to produce test subjects. (Renner, 2003). Such iterations use methods of genetic crossover or mutation to vary and implement models for analysis. Figure 10 below displays two of the common methods for augmenting an existing population of structure options to create the next set of designs for comparison. The crossover operation incorporates components of two "parent" designs to create a "child" for analysis. The mutation operation modifies a single "parent" to create a new offspring structure. The ability of the genetic algorithm to create and test subjects is a key component to the depth of the design space explored for an optimal solution.

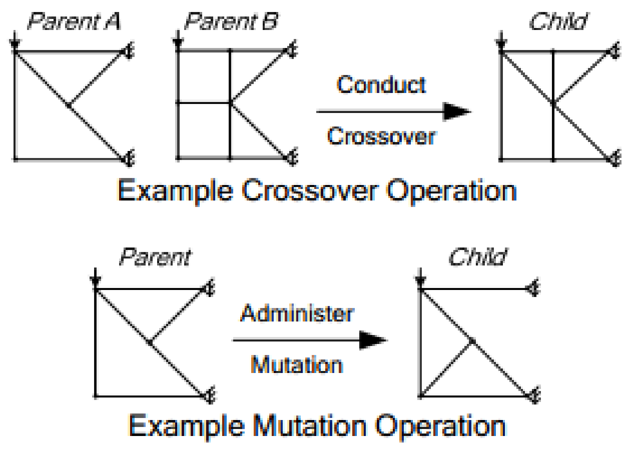


Figure 8: Genetic Algorithm Variations for Fitness Evaluation (Sarkisian et. all, 2009)

Within the realm of optimization, structural engineering problems search for the "best" design out of a complex system of engineering variables. The structural problem is represented as a mathematical model, which translates the geometry, loads, materials, and restraints into objectives and constraints. The mathematical model is tasked with finding variables that return extreme values of the objective without violating the constraints set on the optimization. Thus, the mathematical model is the method by which an engineer moves from a structural problem to an optimal structural design. The development of a real problem and its consequent real solution has been termed a "problem-seeks-design" methodology (Cohn, 1994). This, in essence, is one of the major tenants of engineering design practice. In his paper on the "Theory and Practice of Structural Optimization," Cohn states that

"[The] presentation of optimization applications following the 'problem-seeks-design' approach could expand the understanding and use of optimization by focusing designers' attention to both physical and mathematical aspects of their problems, with due priority to the former."

The exploration of a design space with optimization routines allow engineers to produce analyses in line with the discussion mentioned prior on BIM modeling. Genetic algorithms may produce preliminary designs that could offer additional information that a design teams' intuition and experience may not readily comprehend. In the formulation of such a "problem-seeks-design" model, the problem may be defined in a system of constraints and arrays:

$$\text{The Genetic Algorithm must find the variables of the design vector}$$

$$x = \{x1, x2, \ldots, xn\}$$

$$\text{Such that the Objective Function}$$

$$f(x)$$

$$\text{Is minimized while being subjected to } i \text{ inequality constraints}$$

$$C_i(x) \leq C_o$$

$$\text{While the design vector is bound by upper and lower limits}$$

$$\{x_{LB}1, x_{LB}2, \ldots, x_{LB}n\} \leq \{x1, x2, \ldots, xn\} \leq \{x_{UB}1, x_{UB}2, \ldots, x_{UB}n\}$$

The design vector may be whatever metrics are necessary to define the problem. The constraints are then the metrics used to determine whether or not a solution is valid, or “legal,” in the design space (Saunders, 2012). Thus, the stress limit of a member could define a constraint that, when breached, would eliminate the design solution from the optimization routine. The bounds for the design vector create the limits of the search space and are correlated to the variables within the design vector itself. The results of such an algorithm can be interpreted to express the performance of the optimization routine, whether it is efficient, reliable, accurate, or robust (Chen and Rajan, 1999). Such metrics dictate the speed, versatility, and consistency of a genetic algorithm, and are important in determining the best optimization routine for a given design problem. Issues common to genetic algorithms include the finding of local minima in the design space. The diversity of the population, as well as the probability of the variation size between design variables searchable by the algorithm, will greatly affect the results (Al-Shihri, 2010). Thus, results of a routine should be compared to the entire list of generation cycles for an understanding of the optimization process and its limitations (Sarkisian et. all, 2009).

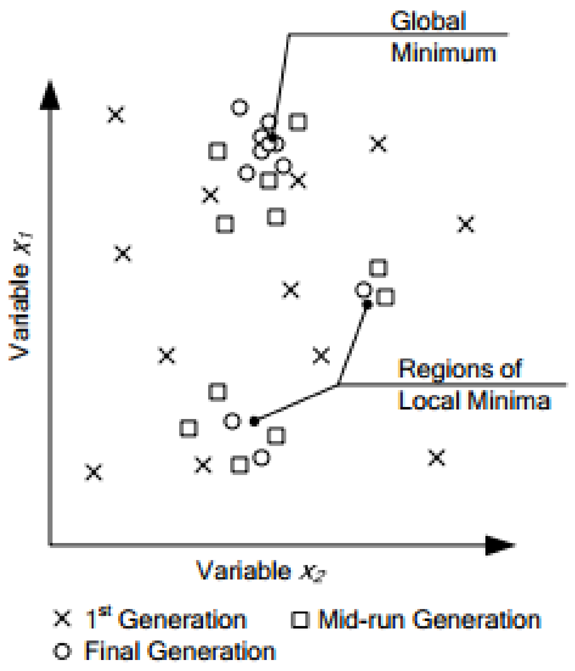


Figure 9: Example Search Space and Local Minima (Sarkisian et. all, 2009)

Complexities do arise, however, when multiple objectives are sought at the same time. In structural engineering, a common occurrence of such an issue is topology versus section size optimization. This issue can be avoided in structural engineering through the use of volume or weight as a means to control such diverse parameters, or develop fitness functions based on the stiffness of individual members so that deflection may be a unifying objective (Jakiela et. all, 1999).

The Knippers team has also pushed beyond sole parametric modeling into the realm of structural optimization. The MyZeil, a shopping mall in the center of Frankfurt, Germany, is perfect example of the application of custom optimization routines to work within a dialogue between architects and engineers. The intent, and ultimately the result, of the structure was to create a space with no interior columns. Working with surfaces from the architects, the Knippers team was able to generate a form that minimized material consumption to limit the self-weight of the structure (Dimcic, 2012). The results are telling, as they show how effective an optimized routine-based design process can be for the client, designers, and users.

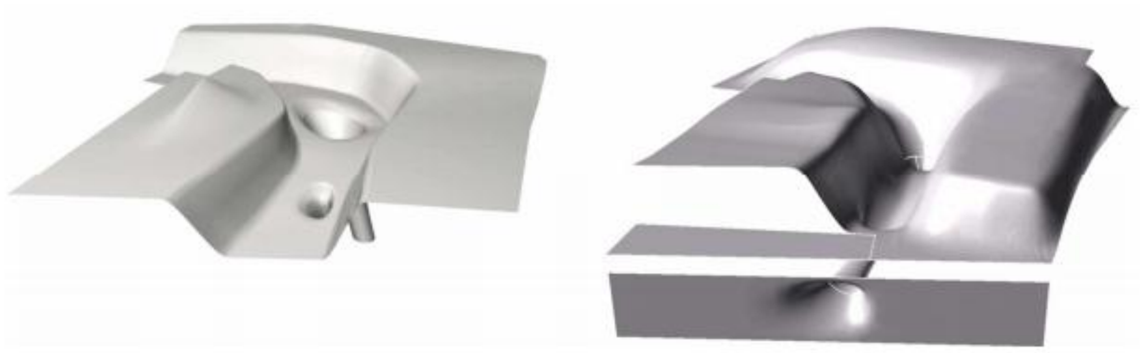


Figure 10: MyZeil First and Final, Optimized Model

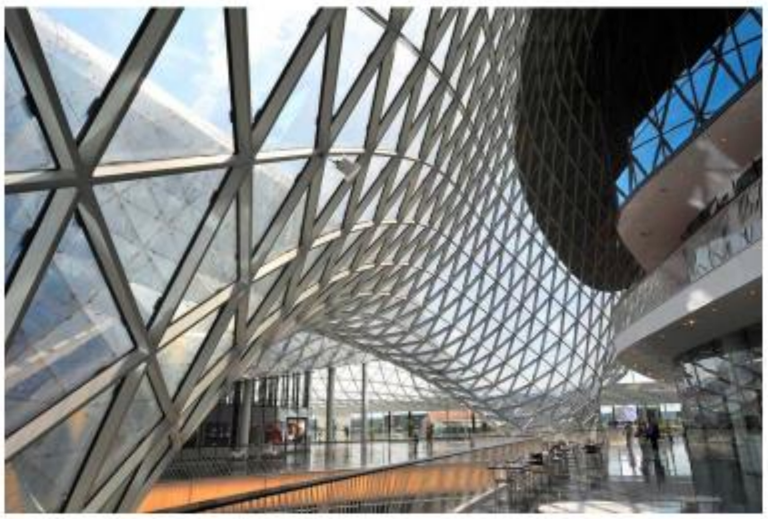


Figure 11: MyZeil Column-Free Interior

There are, however, issues with the use of genetic algorithms for design. If an optimization routine is used too early in the design process, the definition of parameters limiting the problem may be too narrow to render a wide array of adequate solutions. This, in essence, will reduce the potential for creativity in a design. The transparency of genetic algorithms is also a concern, as many currently available simply output the final "best" solution without exposing the individual designs that compose the population. This does not allow for true exploration of the design space. Limitations in optimization also exist when the criteria for the objective function are no longer bound to mathematics. Issues regarding aesthetics or context of a design are not easily integrated into equations for exploration (Buelow, 2008).

The software framework established by this thesis mitigates these optimization issues. The incorporation of a parametric analysis to supplement the genetic algorithm is a key factor in the new framework's success. The parametric analysis allows the design team to explore the relationship between variables of a project. This exploration gives insight as to the importance of different factors and consequently allows the architect and engineer to select the correct design variables for optimization. Thus, the narrowing of scope too early in the design stage is avoided. The parametric analysis also yields transparency to the genetic algorithm by defining the design space. The design space is a map of all the variables deemed important by the architect and engineer, and thus may give insight as to where the genetic algorithm is exploring for a solution. The incorporation of the optimization and parametric analysis into the Grasshopper interface allows the architect and engineer to create the structure together. While not explicitly defining equations for issues of aesthetics or appropriateness, the interface promotes discussion about such issues which in turn will influence which design variables are important to both the architect and engineer.

2. 4 Implementing Object-Oriented Finite Element Modeling

Finite element analysis program design is driven by many variables. Issues such as development language, environment, and utilization are all important in defining how an analysis program will be constructed. To date there are many approaches to developing a finite element solver, each of which has its own benefits and constraints. Two of such methods are the Model-Analysis separation and Model-User Interface separation as presented in the paper "Using Design Patterns in Obect-Oriented Finite Element Programming" (Mackie, 2006).

Model-Analysis separation is a system that breaks a program into modeling and analysis processes. This method is simple in that it creates two stages: one to model the finite element geometry, and another to analyze the geometry. The analysis stage should be open to amendments and modifications based on the objects created in the geometry stage. This allows for ease of adaptation to various types of finite elements. The benefits of the Model-Analysis separation method are profound. This program architecture allows for a succinct division in labor between the geometry and analysis stages, which in turn makes maintenance and expansion of each stage easier. The openness of the analysis stage also allows for ease of reconstructing analysis without having to redefine geometry object definitions. For example, a line element may be used to create nodes, links, or plates for analysis.

Model-User Interface separation is another finite element system that isolates user input objects from the model geometry objects themselves. This simplifies the definitions of the geometry objects and allows for a more versatile user interface. A reference to the model is used for manipulation in the user interface, which can be developed to handle and manipulate the geometry as needed. This flexibility is the primary benefit of the disconnect between the user definitions and model space.

The software framework developed by this thesis seeks to combine these two methodologies in order to reap the benefits of both systems. The framework developed uses Rhino3D as a modeling tool, with data from the model sent to SAP2000 for analysis. With the geometry library of Rhino3D set, modifications can be made to the API of SAP2000 to interpret and analyze model geometry as the design team wishes. The use of Grasshopper to take the data from Rhino3D to SAP2000 separates the model from the user input, allowing for manipulation of the geometry to happen without attempting to physically discretize the Rhino3D object members. Thus, a three-part system is created in which each component can be modified independently to better suit the needs of the design team. This separation also allows for simplicity in representing the geometry as data for analysis, as well as simplifying the analysis process itself.

2. 5 Limitations of Parametric Modeling and Optimization in Practice

Since the advent of parametric modeling and genetic optimization in Civil Engineering, a few firms have attempted to emulate research regarding the matters in practice. Two firms in particular, SOM and Knippers Helbig Advanced Engineering, have implemented variations of the subject matter in a BIM format. Knippers Helbig developed structural optimization of a set form through a modification of Rhino with a C++ plugin. The plug-in then output data to GSA, a finite element package similar to SAP2000, for analysis. For the form, a surface was created in Rhino, which then used meshing tools in the software to create members. Both the surface form and member cross sections were fixed for analysis and optimization (Dimic, 2012). SOM took this approach a step further, introducing Grasshopper plugins for Rhino that allowed for parametric structural analysis of a model and consequent optimization through a genetic algorithm. The structural analysis was performed by the Grasshopper-based package Karamba, which was developed by the University of Applied Arts, Vienna. This package was coupled with a plugin which took surfaces in the same manner as the Knippers Helbig software and developed a structural scheme accordingly. Specifically, the routine focused on the optimization of diagrid systems across the set surface (Sarkisian et. all, 2009).

Both of these packages have their limits, however. The tool developed by Knippers Helbig does not push parametric mapping of a problem and instead focuses on optimization of a few variables. Thus, the solution space is narrow in scope and does not afford exploration for new ideas or solutions. The program implemented by SOM uses parametric modeling, but limits its capabilities by using a simple and specific analysis plugin rather than a robust finite element package. Thus, if the design were to switch from a diagrid structure to a shell object, a new analysis tool would have to be developed. Also, both optimization tools require some entity to be fixed, be it the geometry, cross sectional area, or both. This paper focuses on creating an optimization scheme that allows for parametric modeling as well as genetic optimization. It also focuses on incorporating design software and robust analysis packages to create the complete tool for a broad and deep conceptual analysis.

Chapter 3

Methodology

"The level and the way of cooperation between designers and engineers is slightly changing, creating a new area of expertise between those two fields – an area occupied by experts with the knowledge in design, programming and static analysis, that can solve the problems of structural design and fabrication arising from the complex geometry of the free form architectural design."

- Florian Scheible and Milos Dimcic, Knippers Helbig Advanced Engineering

3.1 Chapter Overview

This chapter describes the methodology used to create the software framework for parametric analysis and algorithmic optimization. Inasmuch, it discusses the three main portions of the developed framework: the Grasshopper interface as well as the parametric and optimization C# codes. While discussing the framework as a whole, specific detail is given to the development of the Grasshopper components, the SAP2000 API analysis code, and the C# implementation codes that comprise the final product created. The final portion of the chapter details the final output of the structural framework. It is here that the importance of the tool created is established in schematic design and design development.

3.2 Wrapper for Data Consolidation and Output

3.1.1 Grasshopper Development

In the development of the Grasshopper plugin, key considerations are taken to create a quick and effective interface that allows engineers and architects to communicate while designing. Knowing that this tool needs to be implemented at an early stage in design to effectively encourage BIM project results, considerations are made as to the scope of the wrapper and the tools it required. The goal of this wrapper is to create a robust parametric and genetically optimized design for study at an early stage in the design process. It is not, however, seen as a final analysis tool. Knowing this, the breadth of the wrapper was limited to allow seamless integration of architectural and structural disciplines without requiring time-consuming post production analyses such as connection design or construction sequencing. Specifically, the wrapper created may be used to design any type of frame-based structure, be it a truss, moment frame, or some hybrid of the two. For the context of this thesis, a truss structure was used for simplicity in analysis and ease of examination. The wrapper also focuses on linear analysis, sacrificing a degree of accuracy for the greater benefit of saving time. Time that would be spent on non-linear analysis may now be allocated to the amount of design iterations the program framework can perform, increasing the amount of possible solutions.

The Grasshopper element needed to be robust enough to accommodate different drawing and development styles so that only minor changes would be needed to run the analysis tools. Due to the large quantity of illustration methods available in Rhino3D, the Grasshopper element needed to be able to transform any structure created into the same data required for parametric study and optimization. Thus, the wrapper moves through design space in a similar way that a finite element package would, building a stiffness matrix and mapping members to the constructed design space. When using the wrapper, the first step is to define the model geometry. The primary reason Grasshopper was chosen for development is its ability to seamlessly integrate with Rhino3D. Thus, the architect and engineer have the opportunity to sit with one another and create a model that is both architectural and analytical in nature. As current software stands, there is no user interface that allows for both architectural design and engineering input to be used with independent packages such as SAP2000 and Rhino3D. There are small, Grasshopper-based analysis tools such as the SOM tool mentioned in Chapter 2 and Kangaroo (Kangaroo Physics, 2013) that allow a user to create structures for specific studies, but the depth and breadth of analysis is limited in comparison to a full finite element package such as SAP2000. The integration of major design platforms for both professions to use simultaneously will save precious time in the back-and-forth interplay of traditional design development.

Once the model is done, the designers then move into the Grasshopper element itself. The second step is defining the materials and section geometry. All material values required to perform a linear analysis are available for selection. The wrapper also allows the user to define the names of the materials and sections for ease of use in the SAP2000 program. An input for initial cross section is given, which for this study was a circular, solid member. The section can easily be changed to represent any arbitration of shape supported by the SAP2000 shape libraries and configures.

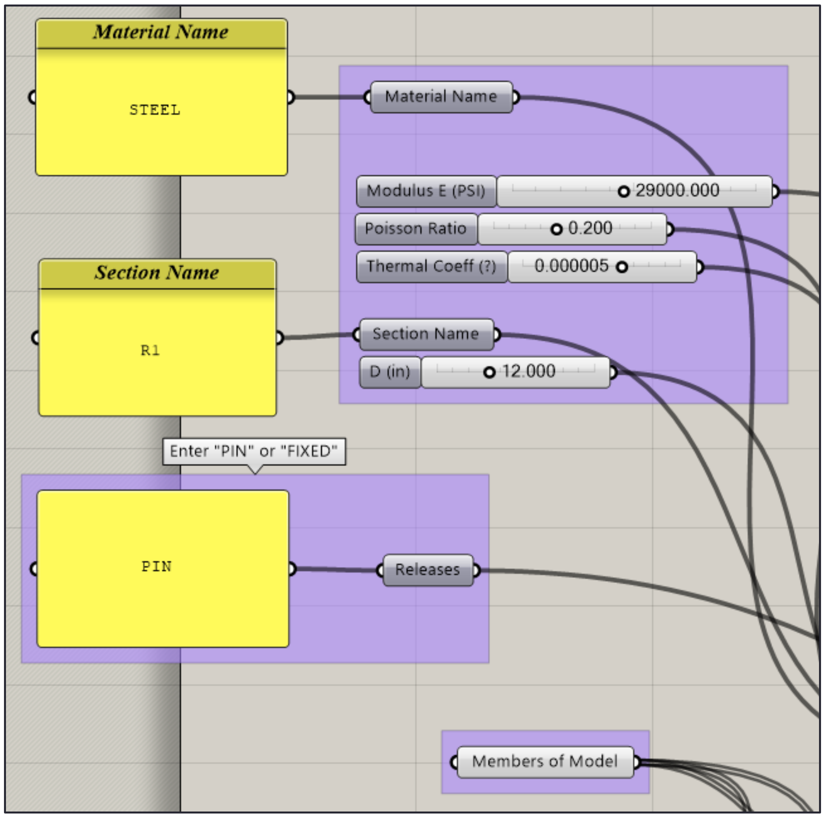


Figure 12: Grasshopper Wrapper Material and Model Definition

Once the material definitions are developed, the designer simply selects the model for analysis and inputs the frames into the tool in the "Members of Model" object. In the current development, there is an option to release members to create axially loaded objects or fix them for moment resistance across the structure. The designers then are free to select nodes on the structure for loading and restraining. The load input takes the form of a traditional global interface, calling out forces in the x, y and z directions, as well as moments about each of these axes. Once the corresponding load values are input, and corresponding load cases are defined, the model is ready to be output for its multiphase analysis. The users are free to simply copy and paste the existing framework for both the selection of loaded and restrained objects as well as load case definitions to create additional restraint and load types. The last step involves directing the output to a directory that is convenient for the users. The module creates a series of text files which it uses to send information to the parametric analysis and genetic algorithm C# programs. These text files are submitted when a Boolean toggle is switched to "True."

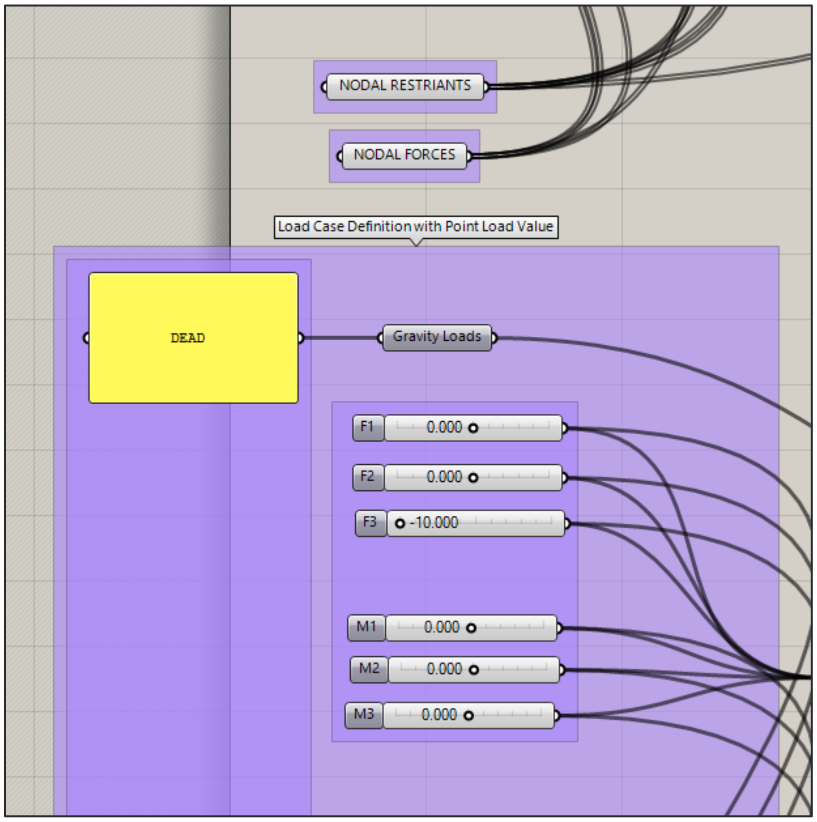


Figure 13: Nodal Restraints, Nodal Force Locations, and Load Case Definitions

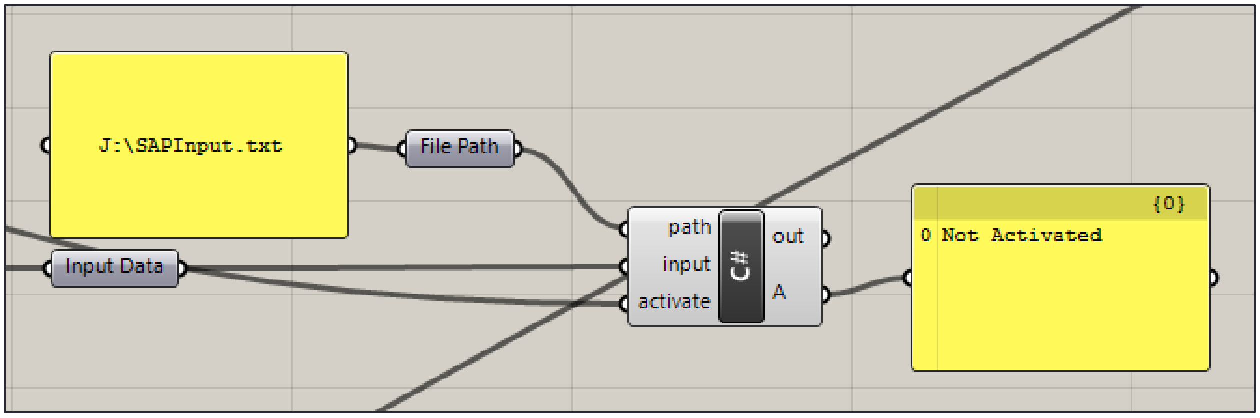


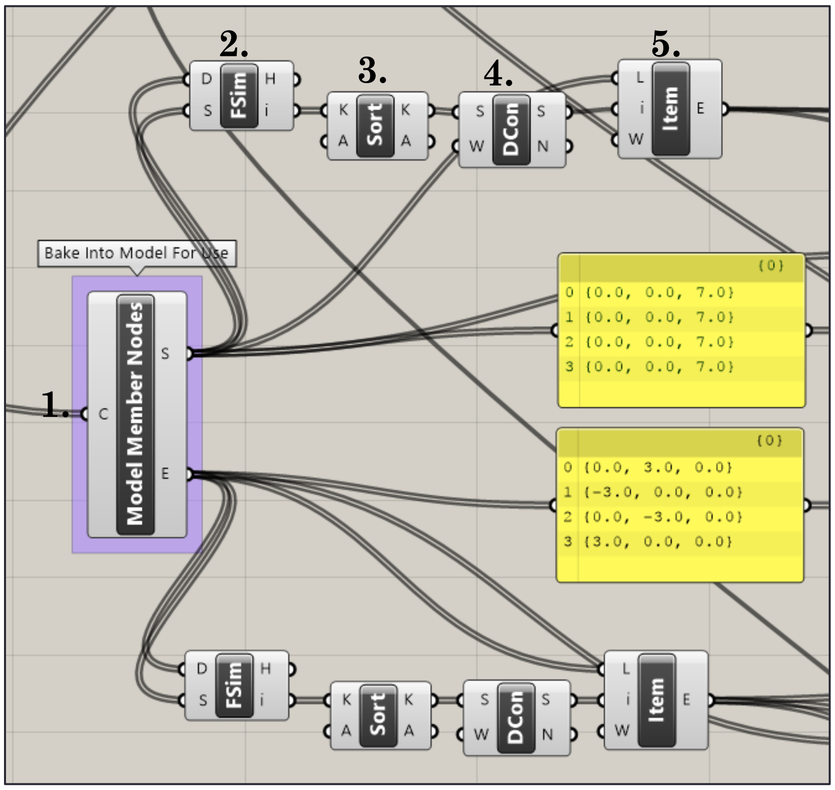
Figure 14: Output Station in Wrapper with Text File Location Input

It is important to note that the intent of the wrapper is to create a dialogue between the architect and the engineer. From the very beginning, the two practices must decide what an appropriate model entails, knowing how it will be analyzed through the wrapper. The two practices must then use their relative expertise to give insight and input to the constraints and restraints the wrapper requires.

3.1.2 Understanding and Implementing SAP2000 API

The wrapper needed to create an output that was meaningful for an analysis package to import and manipulate. The analysis package chosen for this research was SAP2000, the CSI finite element modeler (Computers and Structures Inc., 2013). SAP2000 was chosen for many reasons. The software has a robust solver that lends itself towards quick analysis of linear models which constitute most of schematic design and design development. This property is a common concern among authors on the subject, as custom made finite element packages are often computationally expensive (Chen and Rajan, 1999). The analysis package also has an intricate and open-source API library that is well documented. Another key driver for the use of SAP2000 is the program's widespread use. Many firms of various scales use the analysis software, and thus the wrapper developed would readily integrate into a typical structural engineering work environment.

Research was conducted that determined how the API of SAP2000 describes the software's use and abilities. The SAP2000 API is object oriented, and consequently lends itself to being developed in a variety of methods and with a variety of programs. Moving through the design space, methods had to be found for constructing the respective parts of an analytical model. It was crucial at this point to understand the limitations of input for the SAP2000 API, as well as those for the output of Grasshopper. Ultimately, a solution was found in the creation of a mapping scheme that utilized a series of grasshopper array manipulators to form the stiffness matrix and member-to-node map of the structural model. This matrix was used by the SAP2000 API to create the model nodes, as each row or column was tied to geometry in three-dimensional space, and the members that combine them. This mapping scheme was then reintroduced at multiple levels to allow for quick and efficient creation of constraints, restraints, and loads in SAP2000.



1. Breakdown of members into start and end nodes
2. Array routine to find duplicates of nodes resulting from member joining
3. Array routine to sort nodes by index
4. Array routine to eliminate consecutive duplicates
5. Map of final index to original start or end nodes

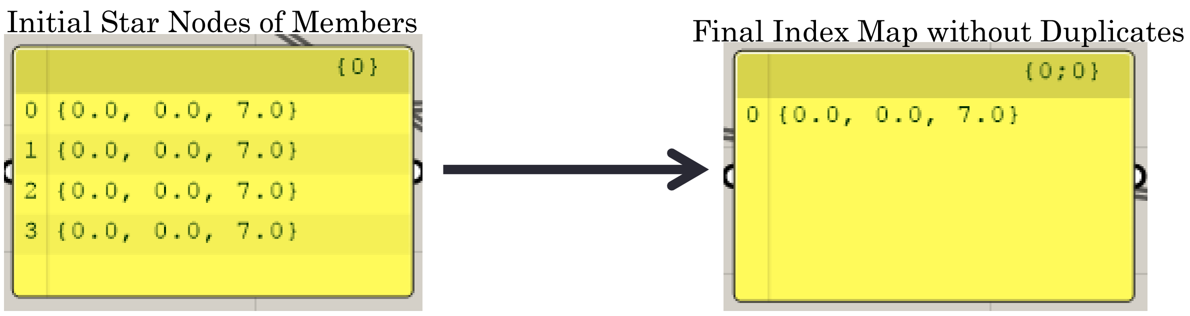


Figure 15: Model Discretization for Construction of Stiffness Matrix and Consequent Node Map

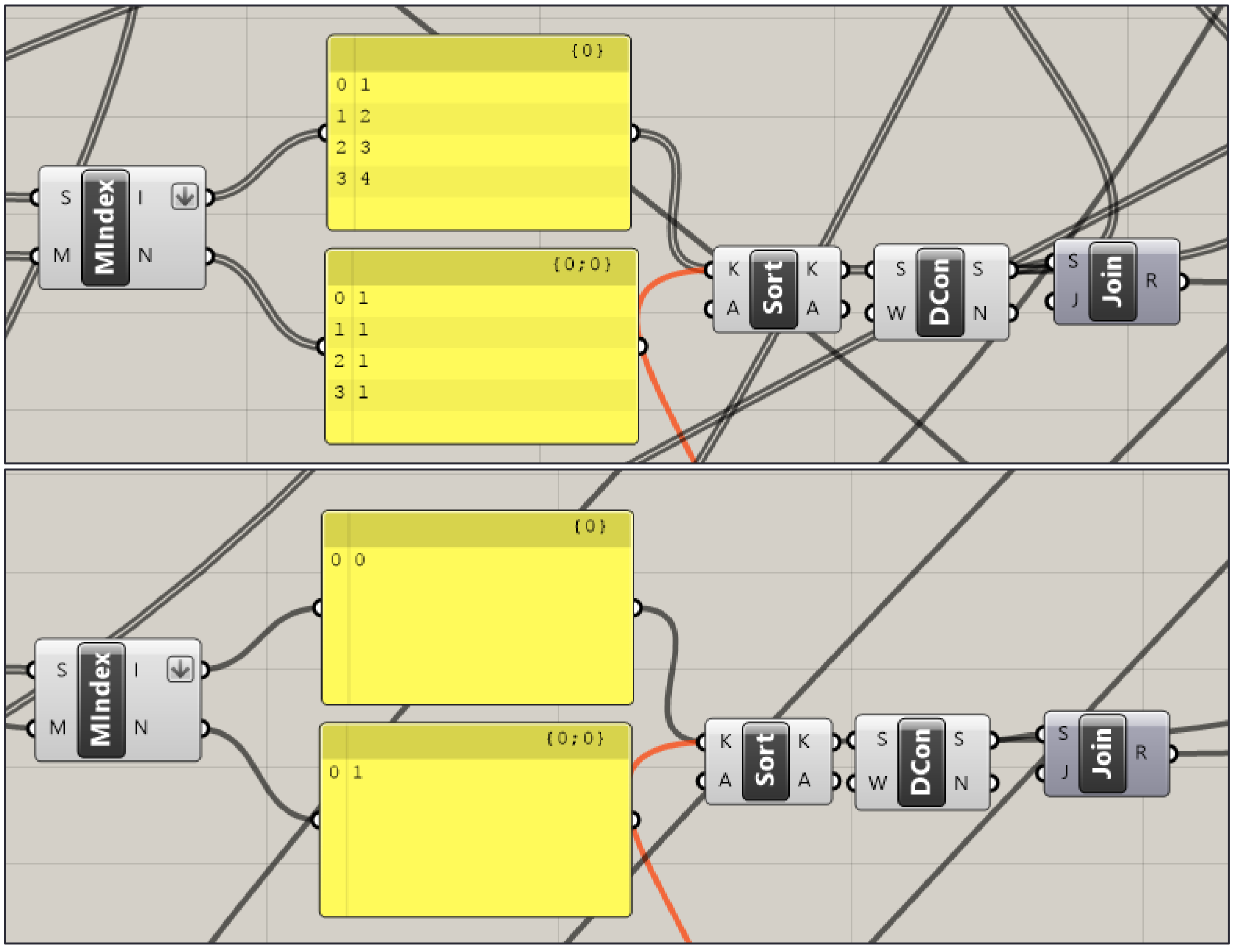


Figure 16: Use of Index Map to Allocate 5 Global Nodes to 4 Restraints (1,2,3,4) and 1 Load (0) for Data Output to SAP2000

3.1.3 Utilizing C# for Synthesis of SAP2000 API and Grasshopper Plugin

The C# platform was used as the messenger between Grasshopper and SAP2000, as it is object based and supports the libraries of each respective code. There has also been significant development of various optimization routines in C# that can be readily integrated into existing problem spaces. NLOpt, an open source library created by professors at MIT, contains numerous algorithms for robust optimization in varying programming languages (Johnson, 2013). A wrapper to C# was created for use and manipulation of this library database (Clune, 2012). Both the parametric and genetic algorithm routines used in this paper were produced in C#.

3.3 Parametric Analysis Program

Using the wrapper created for Grasshopper, a parametric modeling software component was created in C# for developing a user-based analysis tool to explore a design space. The parametric modeler was developed to manipulate geometry and cross sectional properties of a structure for this paper, but can be augmented to adjust any of the properties that SAP2000 imports from the Grasshopper user definitions. To enable the parametric modeler, the user simply inputs the type of property to be manipulated, selects said properties in the three dimensional model, and develops the array that will manipulate the given model property in the SAP2000 analysis. The parametric array consists of a starting value, step size, and total number of operations to be done by the parametric modeler.

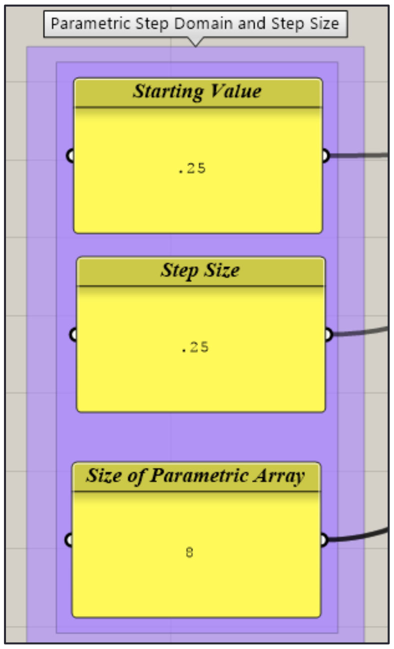


Figure 17: Parametric Modeler Design Input

Once this array is created, the Grasshopper wrapper can be toggled to run and send information to C# for compilation and analysis. The parametric modeler uses the array created by the user to loop through multiple variations of the initial design. For this paper, nodal geometry was chosen to be augmented for analysis. Thus, the C# program creates numerous versions of the nodal map that the Grasshopper wrapper outputs. For each version of the nodes and member map created, a new model is built and run for analysis. A final version of the code used may be found in the Appendix.

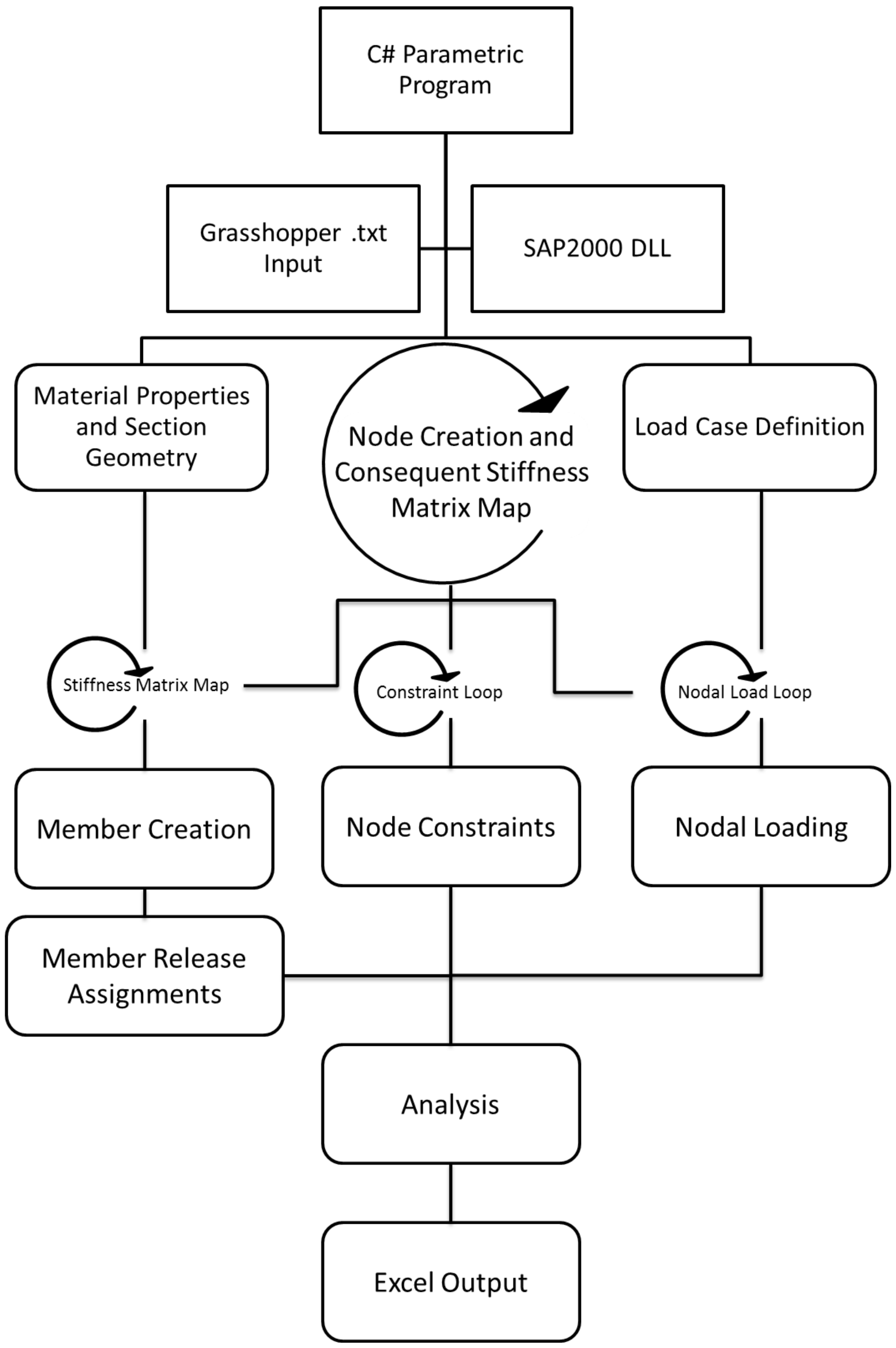


Figure 18: Parametric Modeling Process

3.4 Genetic Algorithm Program

To optimization program was designed to take user input from the Grasshopper wrapper and push SAP2000 analysis through a genetic algorithm. In doing as much, the Grasshopper wrapper had to be designed to work with the NLOpt C# wrapper developed for the non-linear optimization routines available to the NLOpt online library. Thus, research was performed to determine the best routines for a simple and robust optimization that would also be able to tie into the SAP2000 API implementation script that had been created for the parametric solver. The method chosen was the Controlled Random Search with local mutation, or CRS method. The CRS genetic algorithm is coupled with an additional algorithm known as the Augmented Lagrangian, which uses inequality constraints and simple bounds for the design vector as described in Chapter 2. The Augmented Lagrangian method combines the objective function and constraints into a single modified objective function. This modified objective function, or penalty function, is then passed to the CRS which does not have constraints. A simple representation of the penalty function is as follows:

$$Penalty\ \ Function\colon f(x) - \ \ C_i(x)$$

The CRS compares the penalty function to a set of data points created from variations in the design vector space. If the penalty function is better than the worst point in the data set, then it replaces said point. If the penalty function is not better, local mutation introduces a new trial point near the current best point in the data set. This mutation increases the efficiency and robustness of the CRS method (Kaelo and Ali, 2006). This new solution is subject to the Augmented Lagrangian constraints, however, which in return assign a penalty to modify the result. This process is repeated until convergence at an optimum or a running time threshold is reached (Johnson, 2013).

The C# program created uses the CRS and Augmented Lagrangian methods to iterate through the design space to find an optimal structural configuration. Once the design team defines the objective function and inequality constraint, as well as creates the design vector with its bounds and initial guesses, the data can be output through the same toggle in Grasshopper as mentioned in Section 3.3. Them, using the SAP2000 and NLOptDotNet libraries, the optimization routine takes the user input from Grasshopper and begins searching for the best solution based on the initial start values. Analysis through SAP2000 is performed for each modification of the design variables, with the corresponding fitness value from the inequality constraint giving insight as to the next iteration. This is repeated until the result is output for comparison.

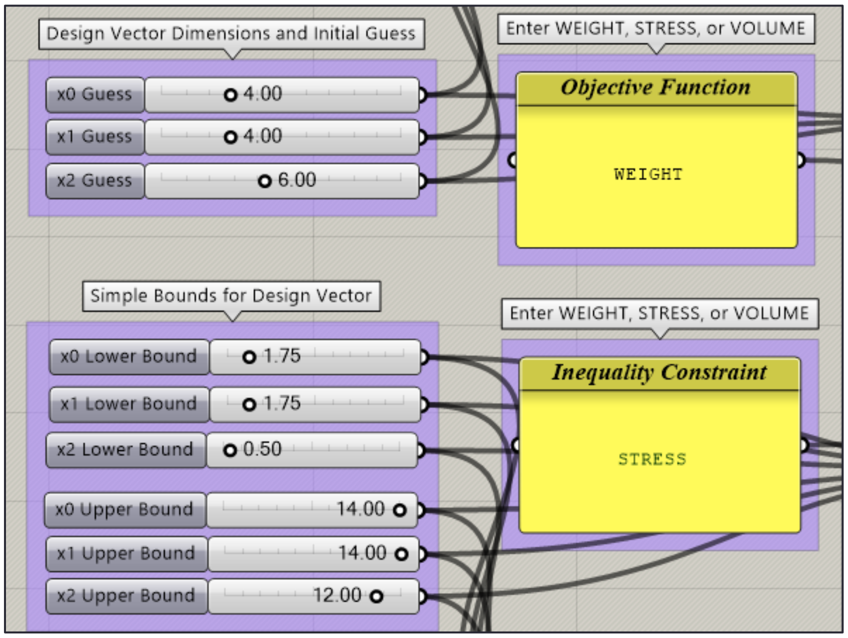


Figure 19: Grasshopper Genetic Algorithm User Input Showing the Considerations and Constraints a User Defines

The development of the optimization routine and Grasshopper user interface allows for an expedient extension or modification of the software framework to solve new problems. From the Grasshopper interface, the user may simply copy and paste additional design variable inputs to make the design vector larger or smaller. The same may be said for the bounds of the design vector. To augment the objective function, the user would simply have to define a new name and corresponding equation to optimize. In the figure shown above, the WEIGHT objective function triggers a calculation of the structures weight based on geometry defined by the design vector. The same process is applicable for the Inequality constraint. Now, if the user wanted to run multiple constraints simultaneously, some additional coding would be required for the C# program to pull the appropriate data from Grasshopper. Regardless, changing the optimization program could happen on the scale of minutes, limiting the amount of time used to change the software framework by a design team.

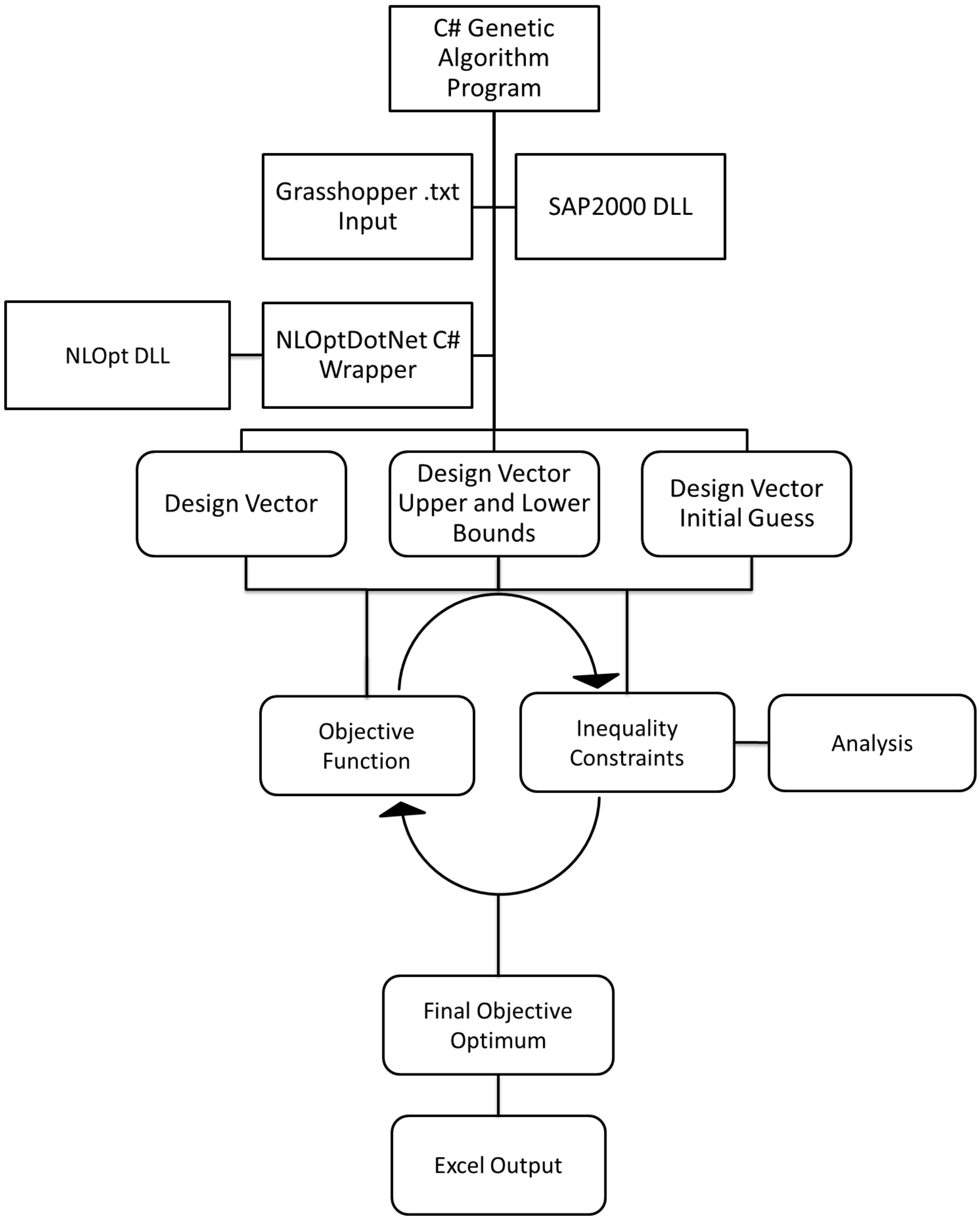


Figure 20: Optimization Process with Optimization Loop Demonstrating Augmented Lagrangian and Controlled Random Search iterations

3.5 Data Output

The final process in developing this wrapper is creating an effective way to view and interpret the data created. Excel was chosen as the collection depot for all analyses, and was used to create succinct visuals that would allow designers and engineers to create a discussion about the results. A three dimensional scatter plot graph was developed for use in interpreting the parametric analyses, as it allows for a surface to be created mapping the benefits and consequences of manipulating a variable in the design space. In each three-dimensional plot rotations allow for the design team to observe the trends between two of the three plot variables. The results of the optimization routine were also placed with the parametric data so that the optima could be seen relative to the design field. Additional charts compare results of the optimization routines and the time allotted for the algorithm to run, allowing the design team the opportunity to weight the cost of an analysis versus accuracy.

3.6 Software Framework Summary

The development of this software framework allows architects and engineers to collaborate and explore designs in s new and effective manner. Through the integration of many robust and independent programs, the tool created allows for both a breadth and depth in analysis that, as noted in Chapter 2, current packages are not able to deliver. The opportunity for the framework to be modified to match the scope of a new design problem adds flexibility to this system, which in turn gives more options for exploration to the design team. Rhino3D allows designers to create complex geometry, and Grasshopper has the capabilities necessary to understand and compile this geometry into data that is then used by the parametric analysis and optimization C# routines. Of key significance to this framework is the required interaction of the architect and engineer. It is this interaction that allows for the best design exploration to occur.

Chapter 4

Results

4.1 Chapter Overview

A case study was created to test the interoperability of the Grasshopper wrapper and its many components. Although simple, it is sufficiently rich to demonstrate the proposed software's capabilities. A rectangular pyramid was created in Rhino3D with four members combining five nodes. Each base node lay on the same global XY plane, and the apex node was centered at all times between the four base nodes, which allowed for translation only in the global Z direction. The four pyramid members were assigned cylindrical frame sections of a six inch initial cross sectional diameter. The material chosen for the sections was steel with a standard modulus of 29000 KSI. The apex of the pyramid, or first node in the model, was loaded with DEAD, WIND, and COMB load cases, each of which described a general load scheme typical of schematic design.

Table 1: Load Case Definitions with Load in Kips

Load Case	Load Direction		
	X	Y	Z
DEAD	0	0	-10
WIND	10	0	0
COMB	10	0	-10

The bottom four nodes in the XY plane were all restrained with translational fixities, thus making them pinned. Once the geometry and corresponding framework definitions were established, the data was pushed through for analysis.

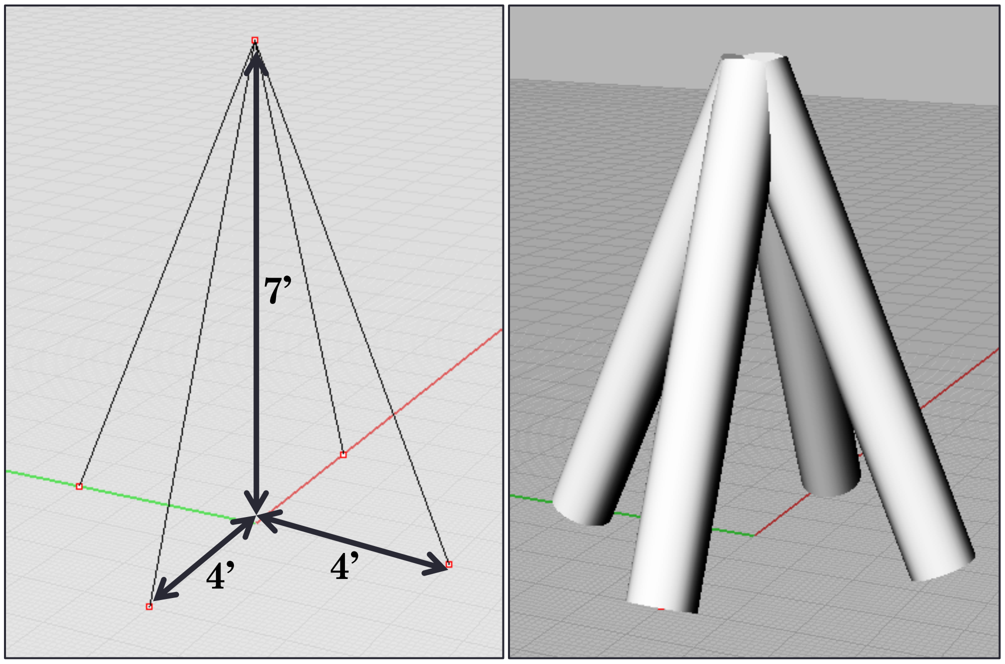


Figure 21: Rhino Design Model Shown in Both Analytical and Architectural Form

4.2 Parametric Study

For the parametric study, the nodal geometry was selected for manipulation by the parametric array. The goal of the study was to understand how the geometry of the structure affected the stress of the members, the deflection of the apex node, the total weight of the pyramid, and the useable volume contained within the design. These metrics were chosen so that if this were to be a design entering the schematic process, both the architect and engineer would have guidelines for discussion and evaluation.

The four base nodes constrained to the XY plane were allowed to translate along vectors away from the center of the pyramid. The apex node was allowed to move along the Z axis in a similar fashion. The parametric array was set to eight variations of .75 feet which were applied to both the apex and the base nodes. Thus, for each step in the height of the apex, the base nodes would translate across the XY plane eight times, allowing for a total of 64 variations of the same model to be analyzed. There is also a parametric constraint for the cross sectional area, which steps in diameter by .5 inches from a start of .5 inches to 12 inches. With each diameter varying according to 64 geometries, the final number of models analyzed became 1536. The base nodes began analysis at .75 feet from the center of the pyramid and moved to 6 feet by the eighth iteration. The apex node began at 1.75 feet from the XY plane, and ended at 14 feet in the Z axis direction.

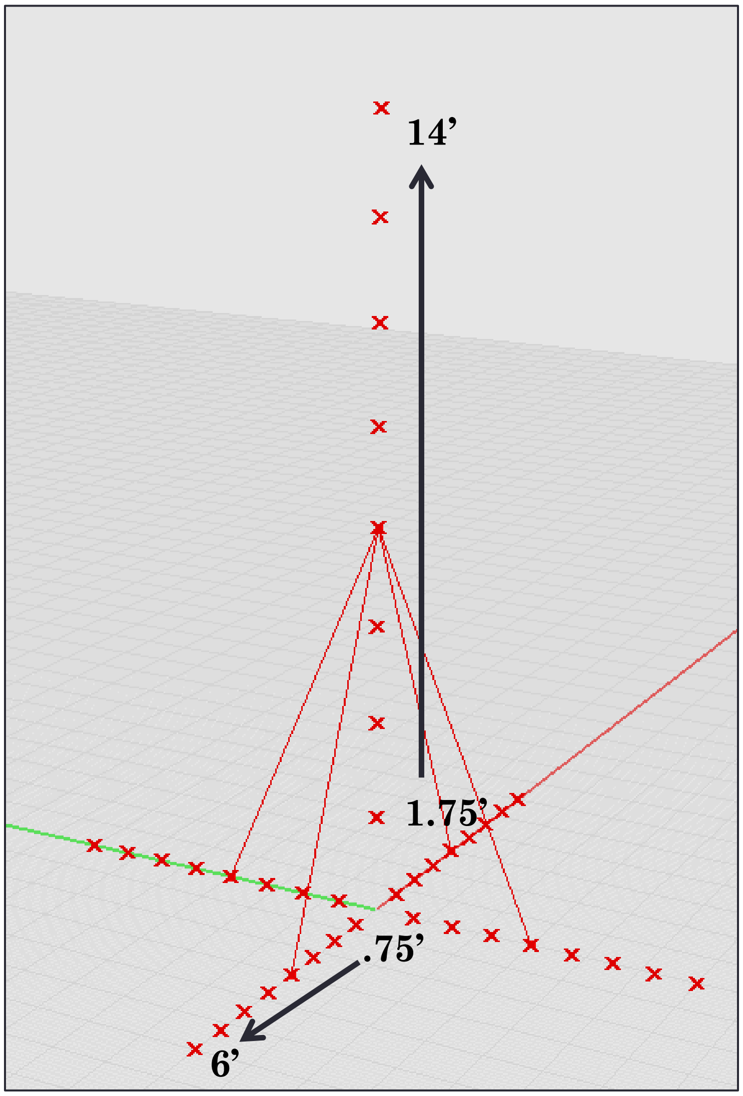


Figure 22: Parametric Model Points Developed by the Grasshopper Wrapper

Once this model was run, there existed 64 data sets of member stresses and weights, nodal deflections, and useable volume for each variation of cross sectional area. This data is exported into Excel from the C# program for three dimensional representation and analysis. For following discussion of results, plots of the parametric analysis with 6 inch diameter members were chosen for display. Due to the simplicity of the model, the results for different diameters analyzed were scaled versions of one another. Thus, any member size would be representative of the structure’s response as a whole.

The output of the parametric modeling gives insight as to how the structure behaves under a variety of constraints as developed by the architect and engineer in the Grasshopper wrapper. There are two tiers of graphics produced for the parametric analysis, the first of which shows how all of the primary metrics interact with one another. In the case of the pyramid, the primary graph shows the relationship between stress unity, nodal displacement, and weight of a given member diameter for a specific load case. The second tier graphics show the individual relationships between the primary metrics and the parametric quantities. In this case, each variable in the primary graphic is shown compared to the base spacing and height of the structure.

The first graph shown below, Figure 21, is the primary output for the parametric model under gravity loading. The vertical axis shows the unity of stress applied versus the available stress capacity of the members with respect to buckling. The horizontal axes show deflection and weight. The planes of the three dimensional space show the projections of the data set for corroboration of the trends observed between two of the three metrics considered.

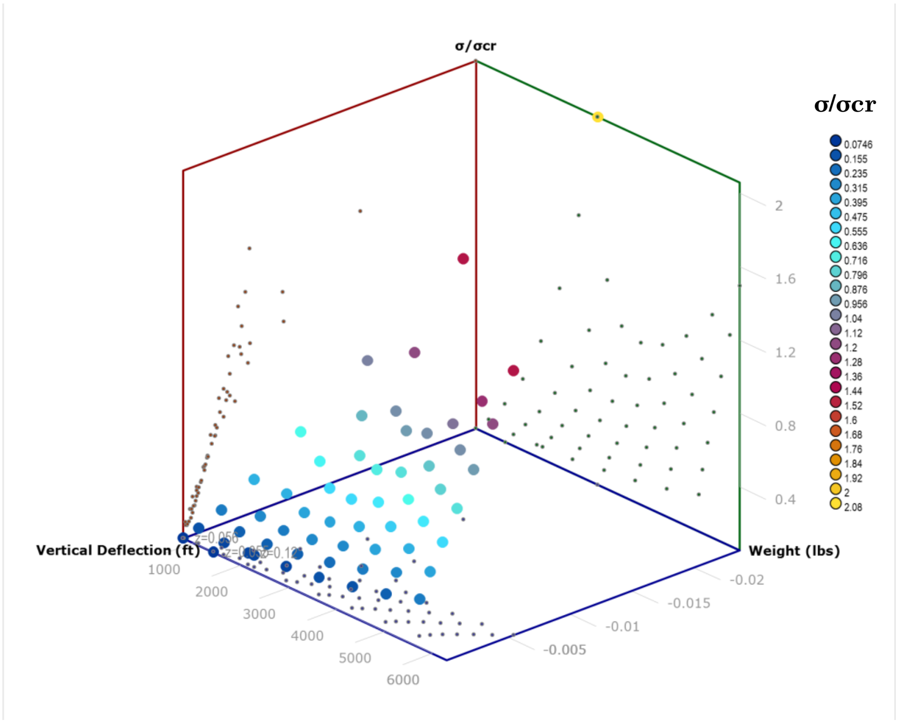


Figure 23: DEAD Load Case Parametric Surface – Deflection vs. Weight vs. Unity for 6 Inch Diameter Members

Figure 22 displays the same information, with the exception that the color gradient now reflects the volume of the structure in cubic feet. This allows the architect the opportunity to understand how useable space correlates to the stresses and key deflections of a structure. Using the secondary plots that correlate each design metric to the geometry of the structure, as is described in detail later in this section, the design team is able to link the performances shown in Figure 22 with variations of the structural model.

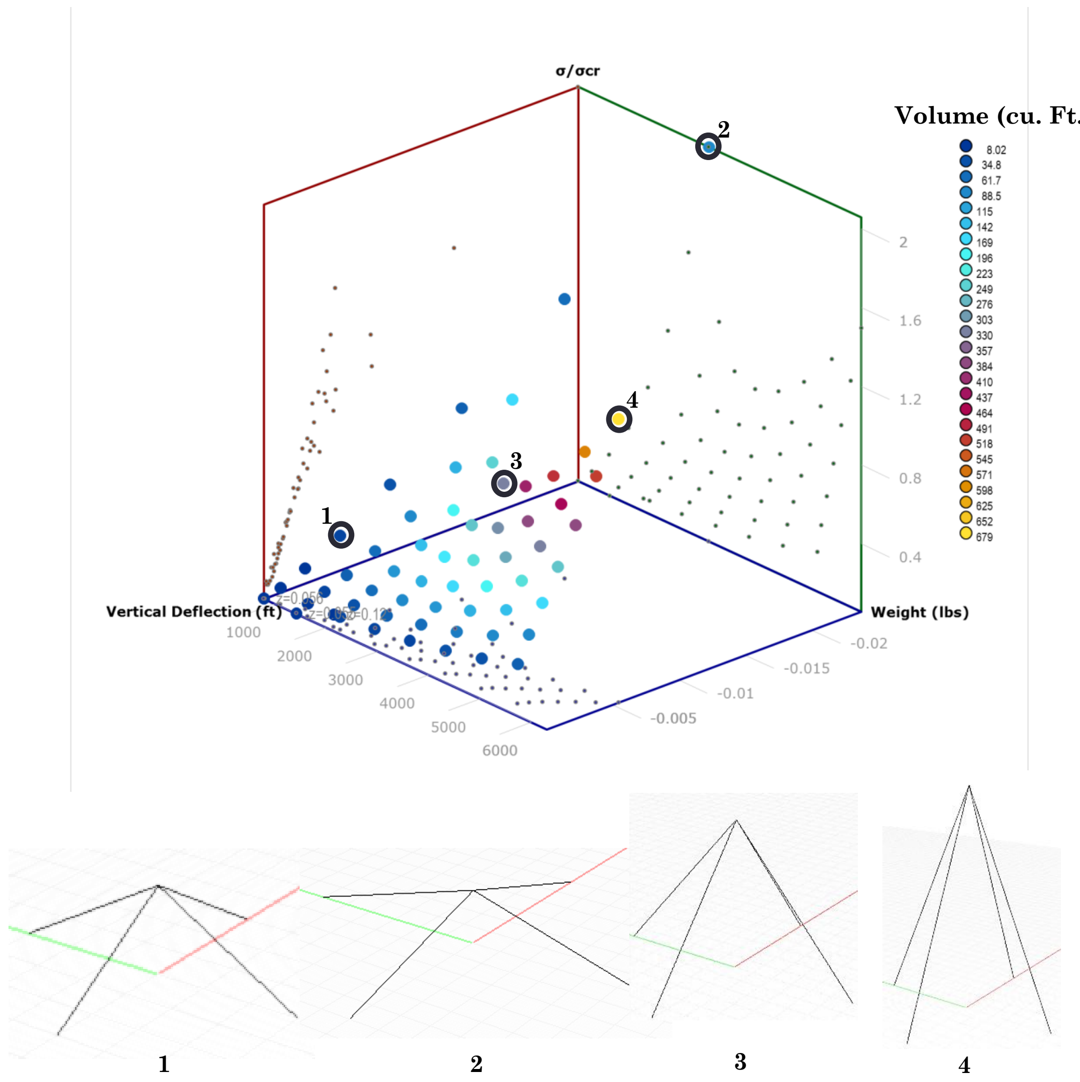


Figure 24: DEAD Load Case Parametric Surface – Deflection vs. Weight vs. Unity vs. Volume for 6 Inch Diameter Members

Table 2: Comparison of Parametric Analysis Results

Parametric Analysis Model DEAD Results Sample Model Population 6 in. Diamter Members						
Model	Height (Feet)	Base Spacing (Feet)	Unity -	Weight (Pounds)	Deflection (Inches)	Volume (Cu. Feet)
1	1.75	6	0.44	1766	0.002	21
2	1.75	12	2.1	3334	0.024	84
3	7	12	0.94	4233	0.004	336
4	14	12	1.5	6300	0.005	672

It can be observed from Figure 23 that stress unity will increase as the weight of the structure increases. This is to be expected, however, the pyramid reaches a design geometry in which a decrease in stress unity occurs with a continued increase in weight, after which the stress unity will begin to increase further than the initial state observed. Visuals such as these allow for designers to immediately set target values for performance of a structure and understand workable limits for optimization of correlated parameters.

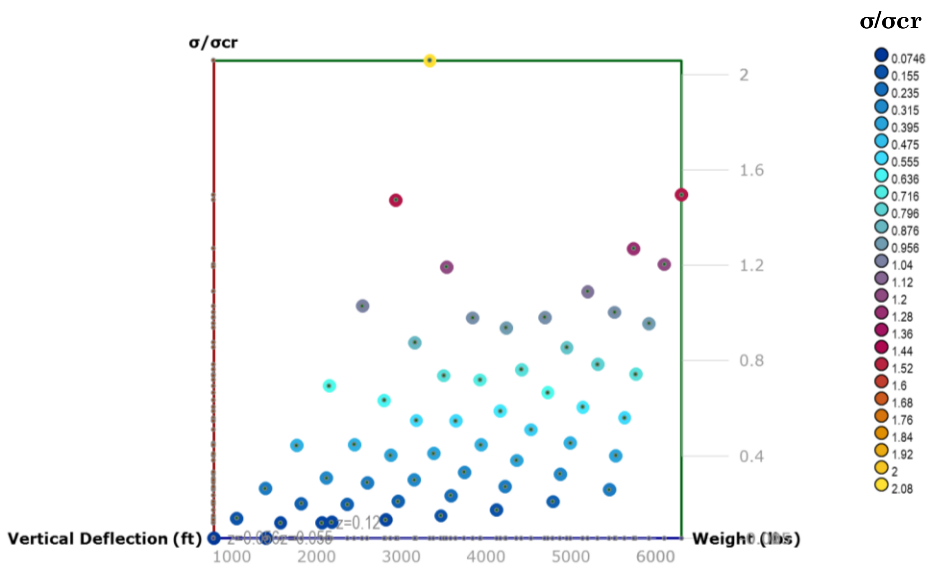


Figure 25: DEAD Load Case Parametric Surface – Global and Local Optima Between Weight vs. Unity for 6 Inch Diameter Members

From Figure 22, it may also be observed that there is approximately a linear relationship between the stress unity and deflection of the structure. For the pyramid model created, this is intuitive as the relationship between deflection and force is most nearly linear, with the length of the members contributing to the change in structure stiffness. When considering weight and deflection, Figure 24 shows that, regardless of total weight, approximately 94% of the structures fall under a deflection of .005 inches. Thus, the design team can relax concerns regarding weight and its impact on the structure's serviceability requirements as they move through the proposed design options.

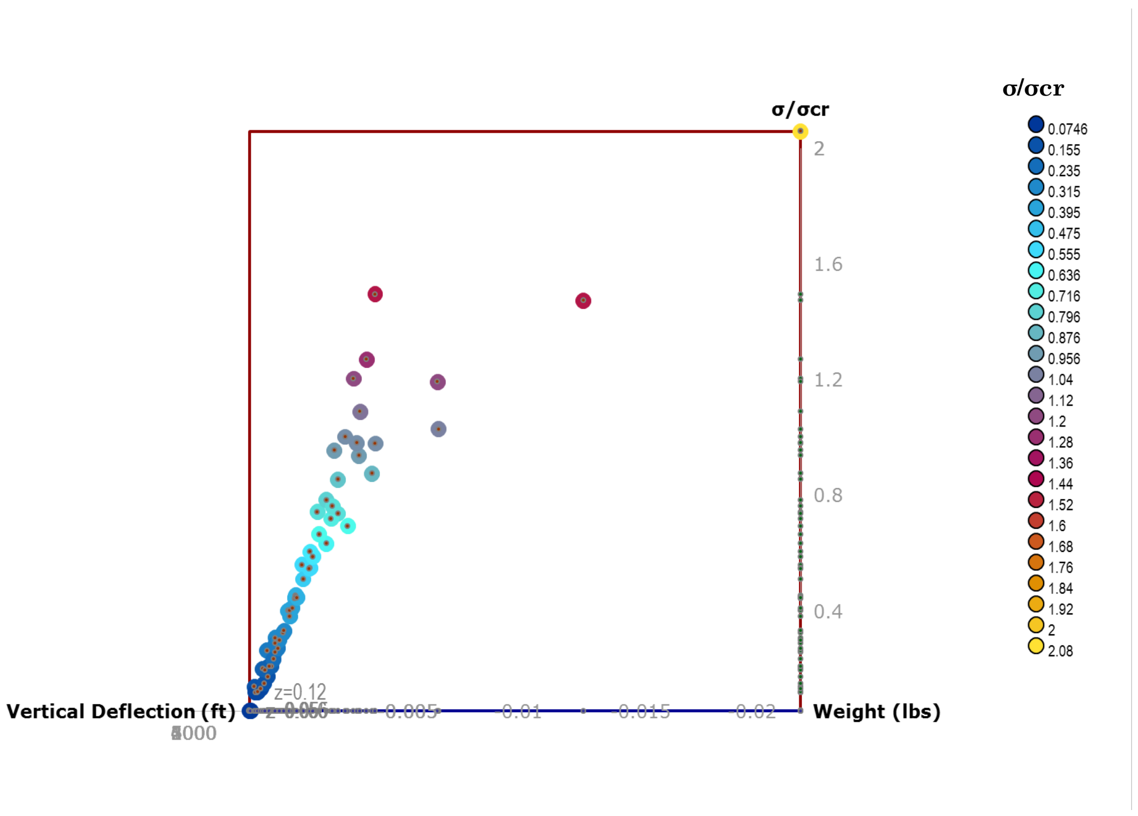


Figure 26: DEAD Load Case Parametric Surface – Linear Response Between Deflection vs. Unity for 6 Inch Diameter Members

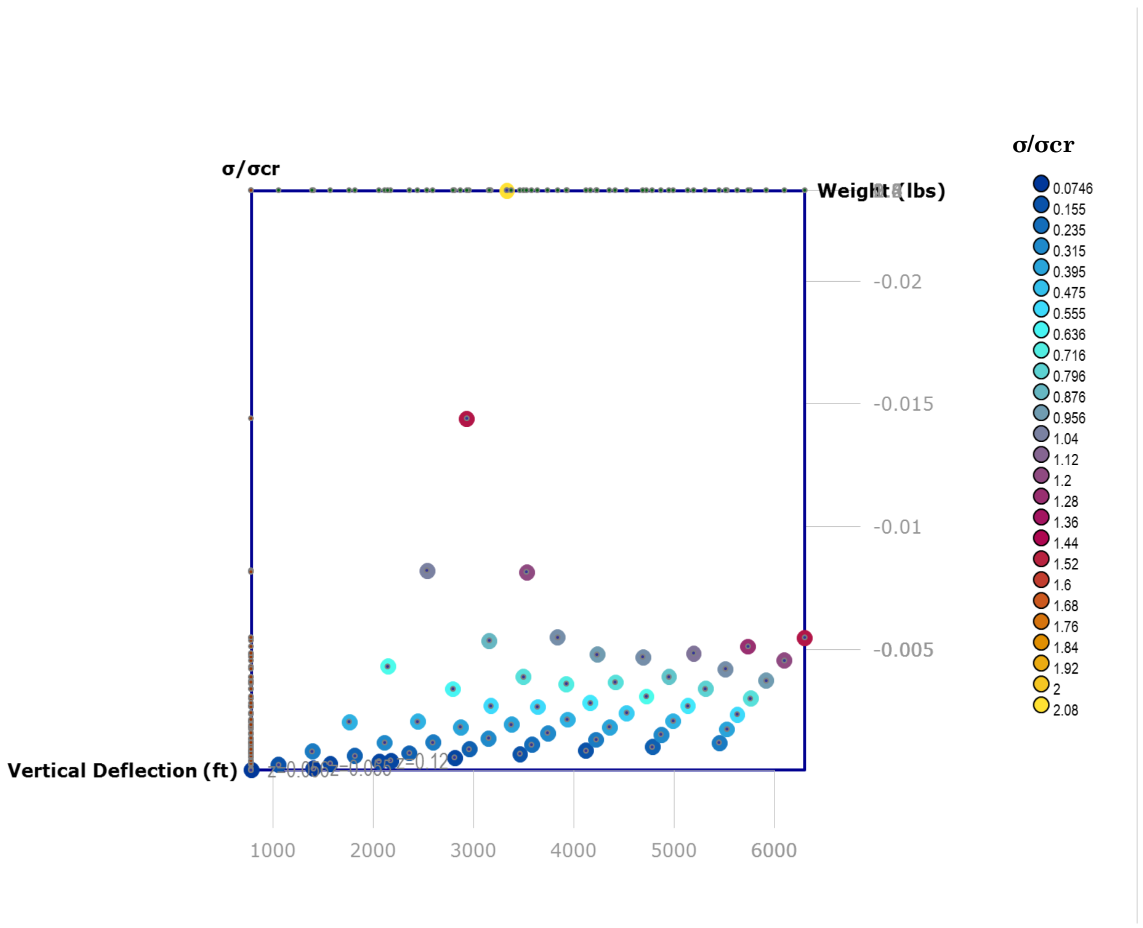


Figure 27: DEAD Load Case Parametric Surface – Banded Performance with Global Maxima Between Deflection vs. Weight for 6 Inch Diameter Members

These plots, which show the interplay of both engineering and architectural design criteria, allow the design team to push forward into schematic design with an understanding of how changes will be reflected in the structure as the development process is refined.

Once design criteria are determined, the architect and engineer are able to move into the secondary plots created by the modeler. These plots relate each primary design metric to the geometry of the pyramid. Thus, the plots act as a map, allowing users to first pick limiting engineering and architectural performance criteria, such as member stress or deflection, and then trace those performances back to their respective parametric model to identify the geometry.

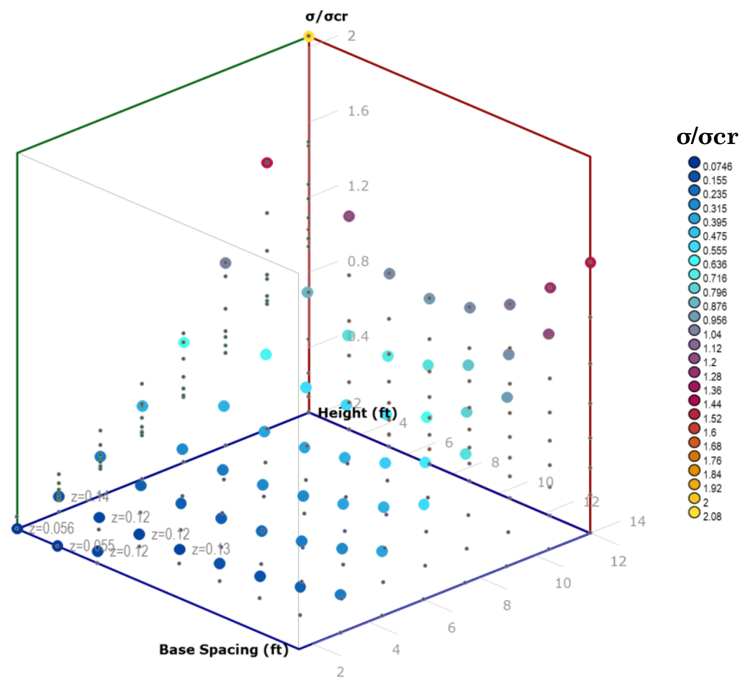


Figure 28: DEAD Load Case Parametric Surface – Parabolic Surface with Local Optima Between Stress Unity vs. Geometry for 6 Inch Diameter Members

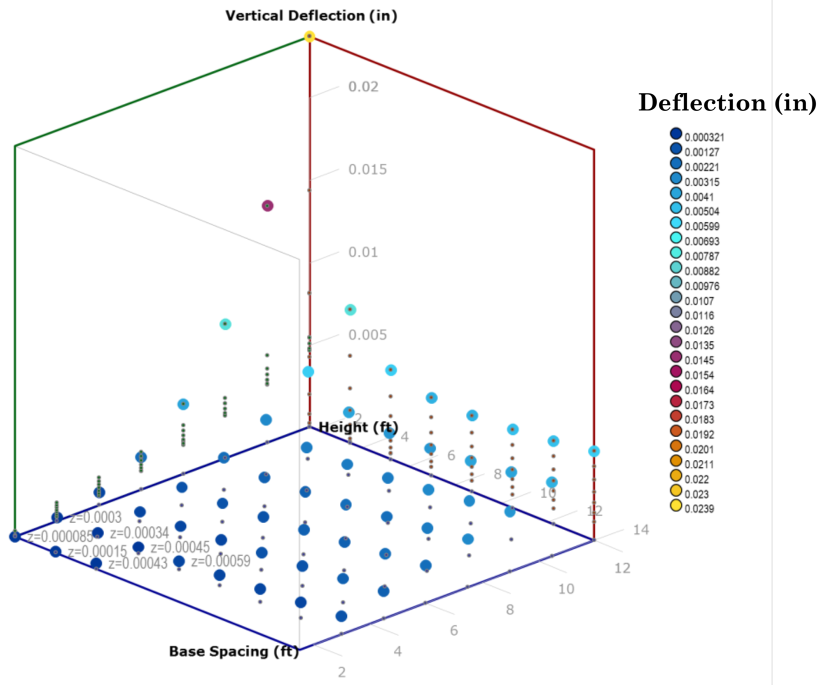


Figure 29: DEAD Load Case Parametric Surface – Definite Global Minima and Maxima Between Deflection vs. Geometry for 6 Inch Diameter Members

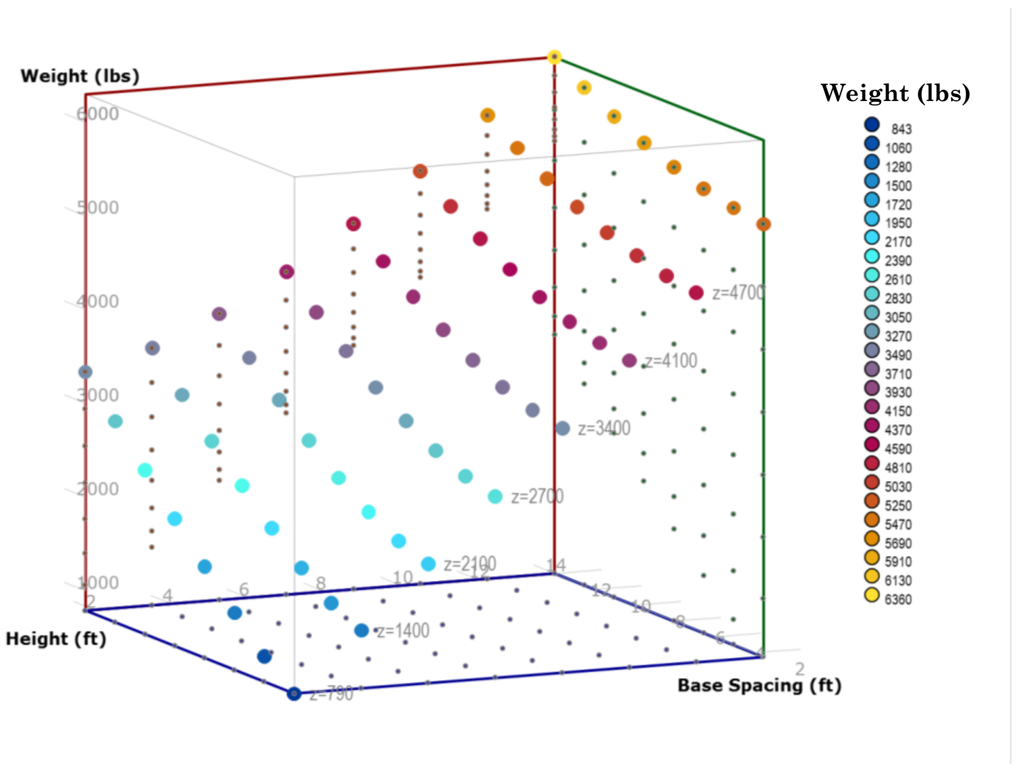


Figure 30: DEAD Load Case Parametric Surface – Banded Surface Between Weight vs. Geometry for 6 Inch Diameter Members

The same process of comparison and evaluation can be created for each load case specified by the design team, as well as for each member cross section. Thus, there is a plethora of information at the disposal of the designers. The time required to generate this data set for a single cross sectional area ran between 2.5 and 2.75 minutes. When there are more nodes added to the model, however, the time to run becomes larger as the stiffness matrix assembly and consequent mapping scheme for members is more intricate. When the nodal amount is doubled, the time required for parametric analysis becomes approximately 2 times that of the original run time. When the node amount is tripled, the time becomes approximately 3.75 times that of the original. This increase is due to the size of the stiffness matrix for analysis in SAP2000 as well as the amount of data the program is required to output to Excel for storage.

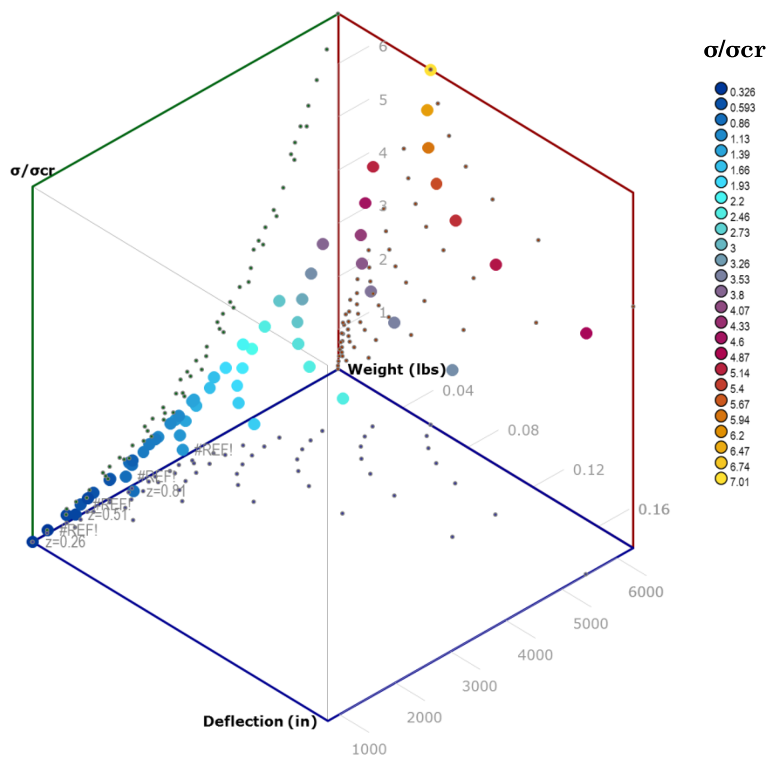


Figure 31: WIND Load Case Parametric Surface – Conical Performance with Definite Global Optimum Between Deflection vs. Weight vs. Unity for 6 Inch Diameter Members

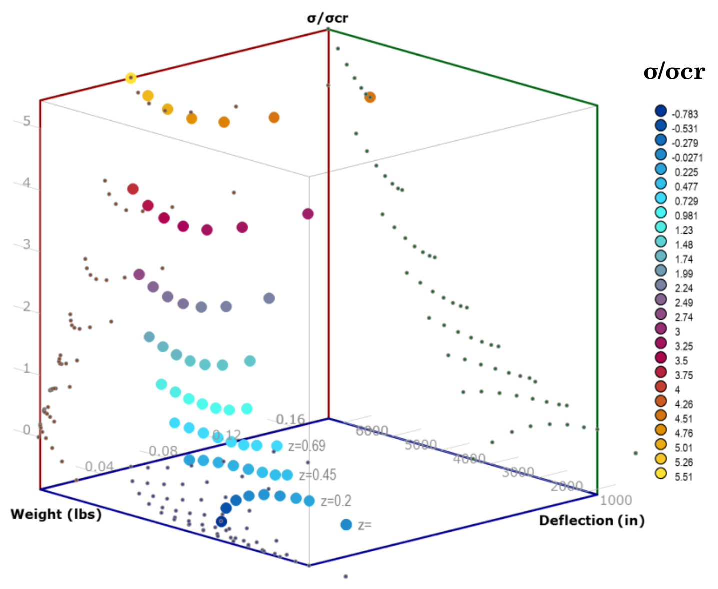


Figure 32: COMB Load Case Parametric Surface – Banded Performance Between Deflection vs. Weight vs. Unity for 6 Inch Diameter Members

4.3 Genetic Optimization Study

Just as in the parametric study, the architect and engineer are free to select what criteria they consider important for optimization. The parametric analysis gives the design team insight as to what key interactions of structure and architecture will constitute the primary issues of progressing towards the reality of the final product. For the case study of the pyramid, the issue of weight was chosen to be optimized. Steel structures are typically paid for by the ton, and thus weight is of concern to both the engineer and architect. This cost is based largely on the tonnage of steel required to complete the project. Thus, the best solution in the design space was sought that would give adequate performance with the lightest overall design.

The design vector for the optimization was set to three variables: base spacing, apex height, and member cross section diameter. These design components allow the algorithm to search the entirety of the structural space, linking the complexities of member performance in terms of stress to the global parameters of overall weight. Upper and lower bounds were set to the design vector that reflects the parametric values initially defined by the design team.

Design Vector $x = \{Base\ Spacing,\ Apex\ Height,\ Cross\ Section\ Diameter\}$

Design Vector Bounds $\{1.75\ ft, 1.75\ ft, .5\ in\} \leq\ x \leq\ \{14\ ft, 14\ ft, 12\ in\}$

Knowing that total weight of the structure was the criteria to minimize, the objective function used the properties of steel defined in the Grasshopper wrapper to determine the weight of each structure in the optimization process.

Objective Function $f(x) =\ \rho_{steel} * N * A * L$

Where ρ_{steel} is 490 PCF, N is the number of members, A is the cross-sectional area calculated from the design vector diameter variable, and L is the length of a member calculated from the design vector base spacing and apex height variables. The performance criteria for the structure were prescribed in the inequality constraints of the optimization. The first analysis used member performance for constraining the optimization.

Inequality Constraint $\frac{F_{member}}{A} -\ \sigma_{critical} \leq 0$

The force in each member, over its cross section area as defined by the design vector, was compared to the critical buckling stress $\sigma_{critical}$. The critical buckling stress was calculated using the standard procedure as defined in the AISC Manual.

$$\sigma_{critical} = .658^{\sqrt{\frac{Fy}{Fe}}} * Fy \quad if \quad \frac{KL}{r} \leq 4.71, \ \ else, \ \ .877 * Fe$$

Elastic Buckling Stress $\quad Fe = \frac{\pi^2 * E}{(\frac{KL}{r})^2}$

Where Fy is the yield stress of steel, K is the effective length factor, and r is the radius of gyration or the square root of the area moment of inertia divided by the member area. The effective length factor was set to 1 for pinned members. For this case study, the DEAD load case was analyzed, which due to symmetry experiences the same force in each member. For the WIND and COMB load cases described earlier, the total number of members exceeding critical stress were counted and used in the penalty function.

Once the parameters of the genetic algorithm were set, the program was run for different time increments. The augmented lagrangian non-linear optimization allows the user to define tolerances for comparison of results and time and evaluation limits. The maximum time allowed for the optimization to run was varied between 100, 300, and 900 seconds. The program was limited to 5000 evaluations, and a tolerance of 1e-4 was set for comparison of generated solutions. The results of this system were telling of the performance of the algorithm and, more importantly, gave insight to the appropriate time required for a thorough exploration of the design space.

When the optimization was run for 100 seconds, the program was unable to find a set minimum in the solution, and thus would exit with whatever iteration it was currently on. This is evident by the consistency of iterations performed for a single routine completion, which were 15 or 16 loops. The reduction in weight from the starting point, which was 1885 pounds, ranged anywhere from 60 % to 98 %. When the routine was allowed to run for 300 seconds, anywhere from 39 to 84 iterations would be completed. This entails that the program would occasionally find local minima that would be considered the answer to the optimization objective function. The reduction in weight was a much tighter band of performance, ranging from 92 % to 99%, with one in every ten runs giving a maximum reduction allowable in the design space, or the lower bound of each design variable vector, of 99.7%. Once the run time was increased to 900 seconds, all iterations developed the same solution to give a 99.7% reduction in weight. Interesting to this analysis are the number of solutions explored in the design space, which ranged from 130 to 180. This is approximately the same deviance as seen in the trial ran for 300 seconds. The following graphs show the optimized design vector results versus the resultant weight.

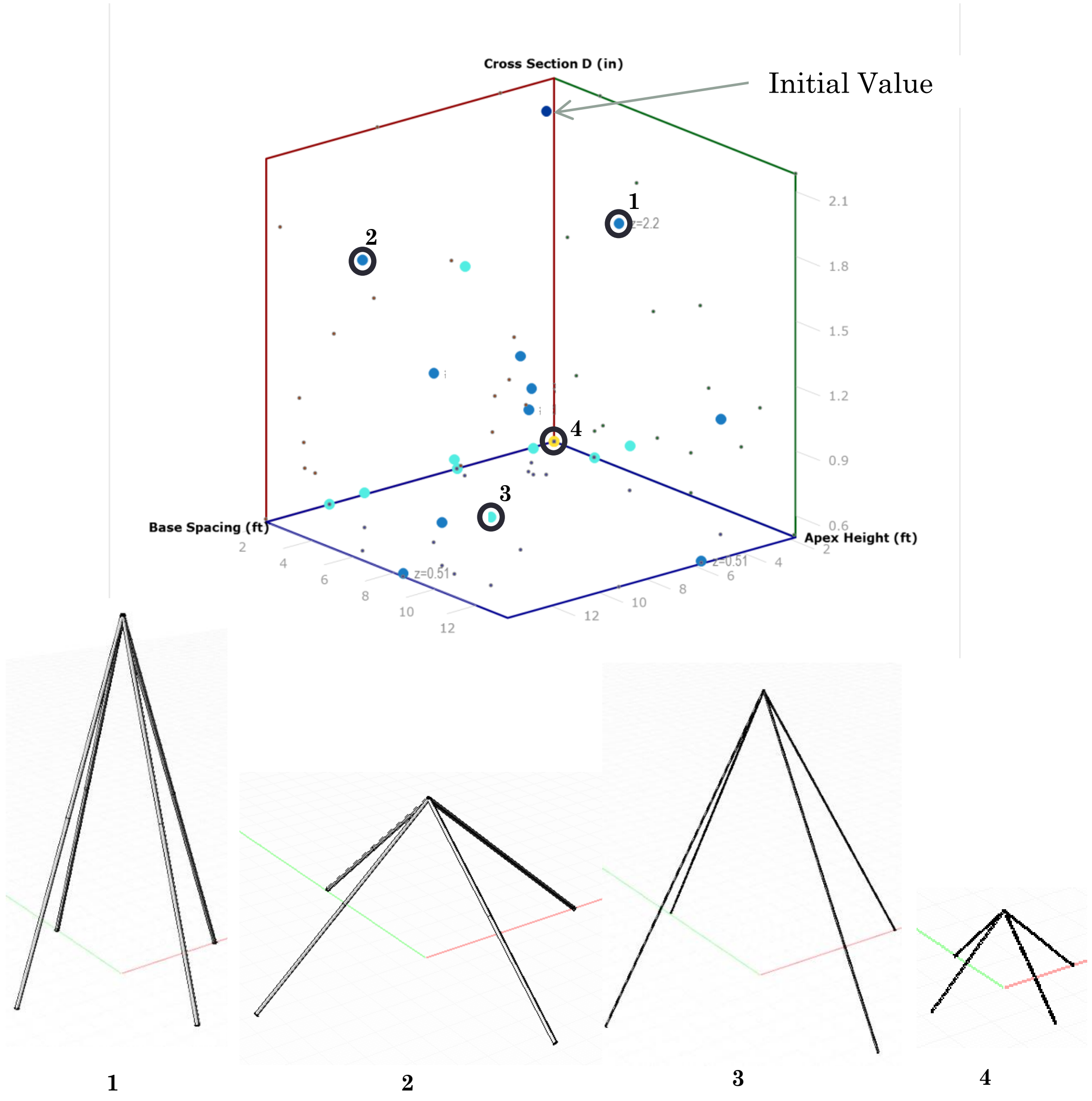


Figure 33: Results of Genetic Algorithm for 100 (Dark Blue), 300 (Light Blue), and 900 (Yellow) Time Limits Relative to Initial Start Point at Top Center

Table 3: Comparison of Genetic Optimization Results

Genetic Optimization Model Dead Results Sample Model Population					
Model	Base Distance ft	Height ft	Cross Section D in	WEIGHT lbs	% Reduction of Weight
1	9.147	13.546	2.181	763.1	59.52471911
2	13.214	5.779	1.846	400.14	78.77633483
3	12.214	10.884	0.818	99.37	94.7293557
4	1.75	1.75	0.5	5.728	99.69618345

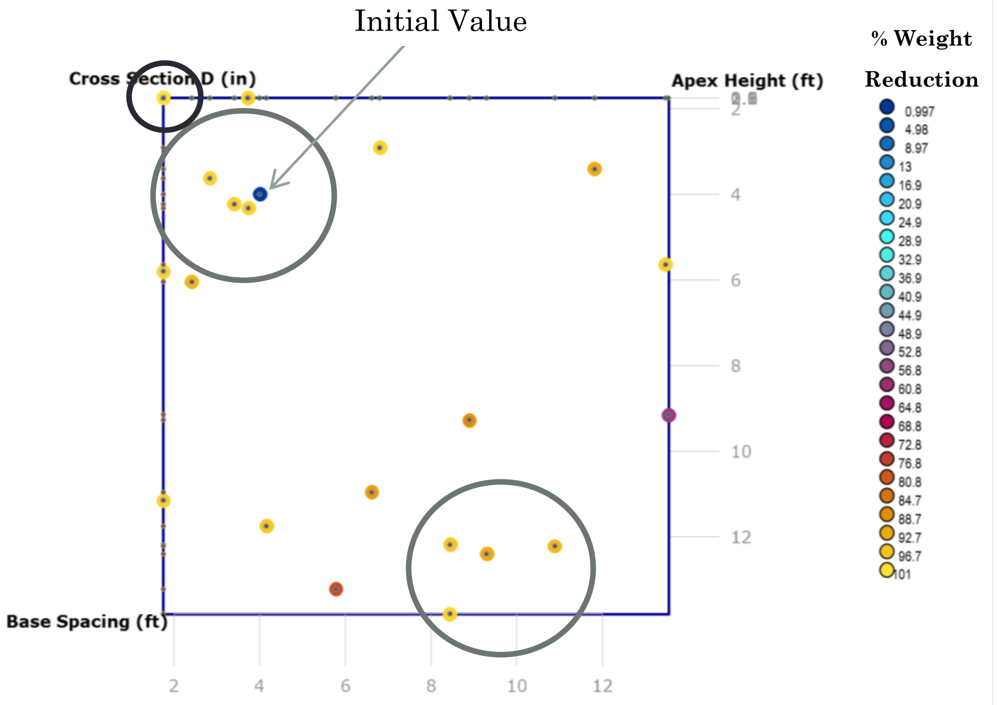


Figure 34: Results of Optimization Showing Local (Grey) and Global (Black) Minima as Percent of Weight Reduction

The optimal result became a pyramid that had a base spacing of 1.75 feet, an apex height of 1.75 feet, and a diameter of .5 inches. When these results are compared to the corresponding parametric results for a member of the same diameter, the results confirm that a pyramid with this geometry is the lightest option that satisfies the buckling constraints of the individual members.

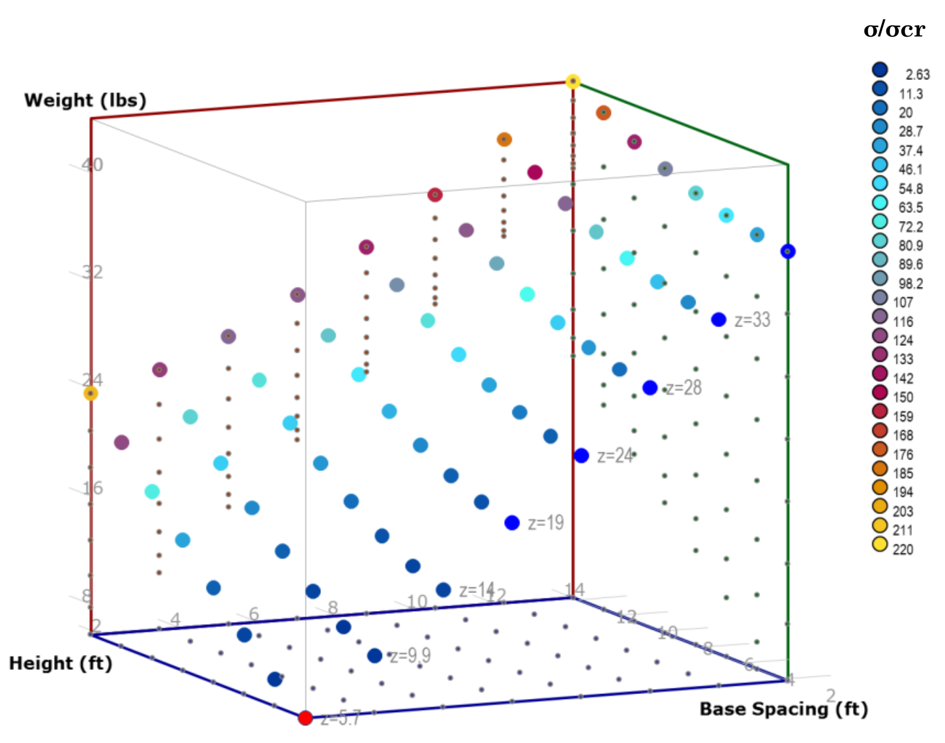


Figure 35: Parametric Surface for Weight vs. Geometry of .5 Diameter Members - Node Shown in Red, at Base Spacing of 1.75 and Apex Height of 1.75m is the Only Point Satisfying Buckling Critical Stress

While the geometry found for the pyramid is the lightest form that can withstand the DEAD load applied to it, it may not be the target of the design team in the larger scale. If there is a minimum volume that must be contained within the structure, or a limited range in height for the apex due to external criteria, then the design space would need to be reevaluated and explored. The benefit of having both the parametric design space and global optimum is that it allows the design team to quickly augment the analysis because the correlations between variables are clearly presented. Rather than begin anew, the architect and engineer can simply translate across the design space, redefining the scope and boundaries of the structure at will. This ability is an answer to the limitations of previous software developments. Rather than fixate only on a few isolated variables and implement parametric analysis or optimization, but not both, the design team may adequately explore the solution space for inspiration and solution.

Chapter 5

Conclusions

5.1 Progress of Structural Engineering Practice

The practice of structural engineering is changing. In an effort to limit costs while maintaining productivity and quality of results, firms have begun to integrate building information modeling, or BIM, technologies into their design process. The use of BIM in design has been shown through research to be very effective when specific focus is placed on increasing the breadth and depth of schematic design and design development. There is, however, limited software currently available that allows a design team the opportunity to adequately explore the design space so early in the project timeline as a collective unit. A few firms, as mentioned in Chapter 2, have developed in-house solutions to this problem. These solutions are limited, however, as they are either specific to some constant geometry, bound by a simple structural analysis tool, or limited to only parametric exploration of a problem. The solutions are also problem specific and thus cannot easily be adapted to a different design problem.

Thus, this thesis developed a software framework that will allow architects and engineers the opportunity to develop a design space and explore it together. The framework developed consists of three major components: a Grasshopper interface for model creation and variable designation, a parametric analysis C# code, and a genetic optimization C# code. In order to maximize analytical robustness while minimizing computational effort of the framework, the codes created use SAP2000 for analysis.

The Grasshopper interface allows the architect and engineer to develop a structure simultaneously, streamlining the conceptual design process. The interface also requires the design team discuss the important metrics in the scope of the project, as the parametric analysis and optimization tools require variable definitions for augmentation and exploration. The metrics defined, be they limitations in weight, deflection, volume, or some other performance criteria, give the architect and engineer common goals to work towards.

The combination of the parametric analysis and genetic optimization give the design team significant breadth and depth to the structural concept. The three-dimensional output of the analysis results allows for an understanding of the relationship between multiple design variables crucial to the structure. For the engineer, trends can be determined that give insight to performance which may drive design initiatives. For the architect, an understanding of the tradeoff between architectural values, such as volume or floor area, and engineering values, such as stress or deflection, may be garnered. With the design space mapped, a unified solution, or at least direction, can be obtained by both parties early in the project timeline.

The genetic optimization routine permits the design team to delve further into the solution of a specific set of variables. Using the parametric analysis as a map of the design space, the architect and engineer are able to make an educated optimization of the structural model. Rather than attempt to optimize every variable of a structure, the architect and engineer can use the results of the parametric analysis to target specific aspects of the design to run through the genetic algorithm. This simplifies the optimization problem and may consequently save time. As issues arise and the importance of different variables change during the development of the project, the design team may simply return to the parametric analysis and determine the consequences associated with said changes. The architect and engineer may then redefine the design vector of the genetic algorithm and determine the new optimal design.

The flexibility and depth of this software framework pushes the design practice in a new, versatile direction. The framework eliminates the limitations of previous work through the adaptation of multiple software platforms and the streamlining of a user interface. Issues such as limits of variables for consideration, simplicity of analysis tools, and specificity of design objects are mitigated. In their wake, a single tool is available for development of a design. Time and energy are saved as a consequence of the framework, and thus implementation during the schematic design and design development stages of a project is feasible.

5.2 Future Development

This thesis represents the first step towards a more unified design practice. With more time and further research, this software framework could become broader in its capabilities. Each component of the framework has the potential to become something more.

The Grasshopper user interface should be developed to discretize not only line elements, but surface, shell, and solid objects as well. The tools to create surfaces are already embedded within the Grasshopper design space, so a scheme similar to that discussed in Chapter 3 for the mapping of nodes may be created to extract the important mesh information for analysis in SAP2000. For parametric analysis and genetic optimization, the mesh size could be augmented along with the geometry.

In response to the development of the Grasshopper component of the software framework, the C# programs for parametric study and optimization should be broadened. A library of definitions should be produced so that the program can easily recognize the variety of data being imported from Grasshopper. In detail, this would include solutions for working with frame, shell, and solid objects with a full array of restraints and constraints. This would allow for ease of use of the C# program without manually changing pieces of the code to complete a task, ultimately saving time for the design team. For this to be done, additional time will be needed to compile the SAP2000 API into blocks that are called upon only when specific criteria are defined by the user in the Grasshopper interface.

The method for viewing the results of the parametric and optimization routines should be developed into a more fluid exploration space. As it currently stands, data is exported to excel spreadsheets from C#. This data is then organized with some user effort into three-dimensional plots. While sufficient to study the design space, an independent graphical interface for viewing the results would be ideal. Such an interface could store all data from the output and require simple instructions to display results, rather than have the users move the arrays of information across spreadsheets. The goal of such an interface would be to save time.

References

Al-Shihri, M. A. "Structural Optimization Using a Novel Genetic Algorithm for Rapid Convergence." (2010).

Autodesk (2003), “Building Information Modeling in Practice.”

Beghini, Alessandro, Lauren L. STROMBERG, William F. BAKER, Glaucio H. PAULINO, and Arkadiusz MAZUREK. "A New Frontier in Modern Architecture: Optimal Structural Topologies."

Birgin, E. G., and J. M. Martínez. "Improving ultimate convergence of an Augmented Lagrangian method." *Optimization Methods and Software* 23, no. 2 (2008): 177-195.

Chen, S. Y., and S. D. Rajan. "Using genetic algorithm as an automatic structural design tool." In *Short Paper Proceedings of 3rd World Congress of Structural and Multidisciplinary Optimization*, vol. 1, pp. 263-265. 1999.

Cohn, M. Z. "Theory and practice of structural optimization." *Structural optimization* 7, no. 1-2 (1994): 20-31.

Conn, A. R., Nick Gould, and Ph L. Toint. "A globally convergent Lagrangian barrier algorithm for optimization with general inequality constraints and simple bounds." *isi* 1 (1992): 1.

Clune, R.. (October 2012). In NLOptDotNet. Retrieved April 8, 2013, from https://github.com/roryclune/NLoptDotNet.

Clune, Rory, Jerome J. Connor, John A. Ochsendorf, and Denis Kelliher. "An object-oriented architecture for extensible structural design software."*Computers & Structures* 100 (2012): 1-17.

Dimcic, Milos, and Jan Knippers. "Integration of FEM, NURBS and Genetic Algorithms in Free-Form Grid Shell Design." In *Computational Design Modelling*, pp. 97-103. Springer Berlin Heidelberg, 2012.

Eastman, Chuck, Paul Teicholz, Rafael Sacks, and Kathleen Liston. *BIM handbook: A guide to building information modeling for owners, managers, designers, engineers and contractors*. Wiley, 2011.

Heng, B. C. P., and R. I. Mackie. "Using design patterns in object-oriented finite element programming." *Computers & Structures* 87, no. 15 (2009): 952-961.

Jakiela, Mark J., Colin Chapman, James Duda, Adenike Adewuya, and Kazuhiro Saitou. "Continuum structural topology design with genetic algorithms." *Computer Methods in Applied Mechanics and Engineering* 186, no. 2 (2000): 339-356.

Johnson, S. (March, 2013). NLOpt. In NLOpt Algorithms. Retrieved April 10, 2013, from http://abinitio.mit.edu/wiki/index.php/NLopt_Algorithms#Augmented_Lagrangian_algorithm.

Student, P. Kaelo PhD, and M. M. Ali. "Some variants of the controlled random search algorithm for global optimization." *Journal of optimization theory and applications* 130, no. 2 (2006): 253-264

Piker, Daniel. "Kangaroo: Form Finding with Computational Physics."*Architectural Design* 83, no. 2 (2013): 136-137.Miller, G. R. "An object-oriented approach to structural analysis and design."*Computers & Structures* 40, no. 1 (1991): 75-82.

Pezeshk, Shahram, and Charles V. Camp. "State of the art on the use of genetic algorithms in design of steel structures." *Recent advances in optimal structural design* (2002): 55-80.

Renner, Gábor, and Anikó Ekárta. "Genetic algorithms in computer aided design." *Computer-Aided Design* 35 (2003): 709-726.

Sacre, C. (n.d.). Boston Society of Civil Engineers Section. In Structural Engineering Challenges from the 20th to the 21st Century: Responsibilities in Shaping the Structural Engineer. Retrieved March 28, 2013, from http://www.bsces.org/index.cfm/page/Structural-Engineering-Challenges-from-the-20th-to-the-21st-Century:-Responsibilities-in-Shaping-the/cdid/11464/pid/10371.

SAP2000 v.14 (2013), Computer and Structures Inc. http://www.csiberkeley.com/sap2000

Sarkisian, M., E. Long, C. S. Doo, and D. Shook. "Optimization Tools for the Design of Structures." In *Structural Engineers Association of California Convention Proceedings*. 2009.

Saunders, Michael. "Augmented Lagrangian Methods." *Stanford University, Managemetn Science and Engineering*, no. 9 (2012).

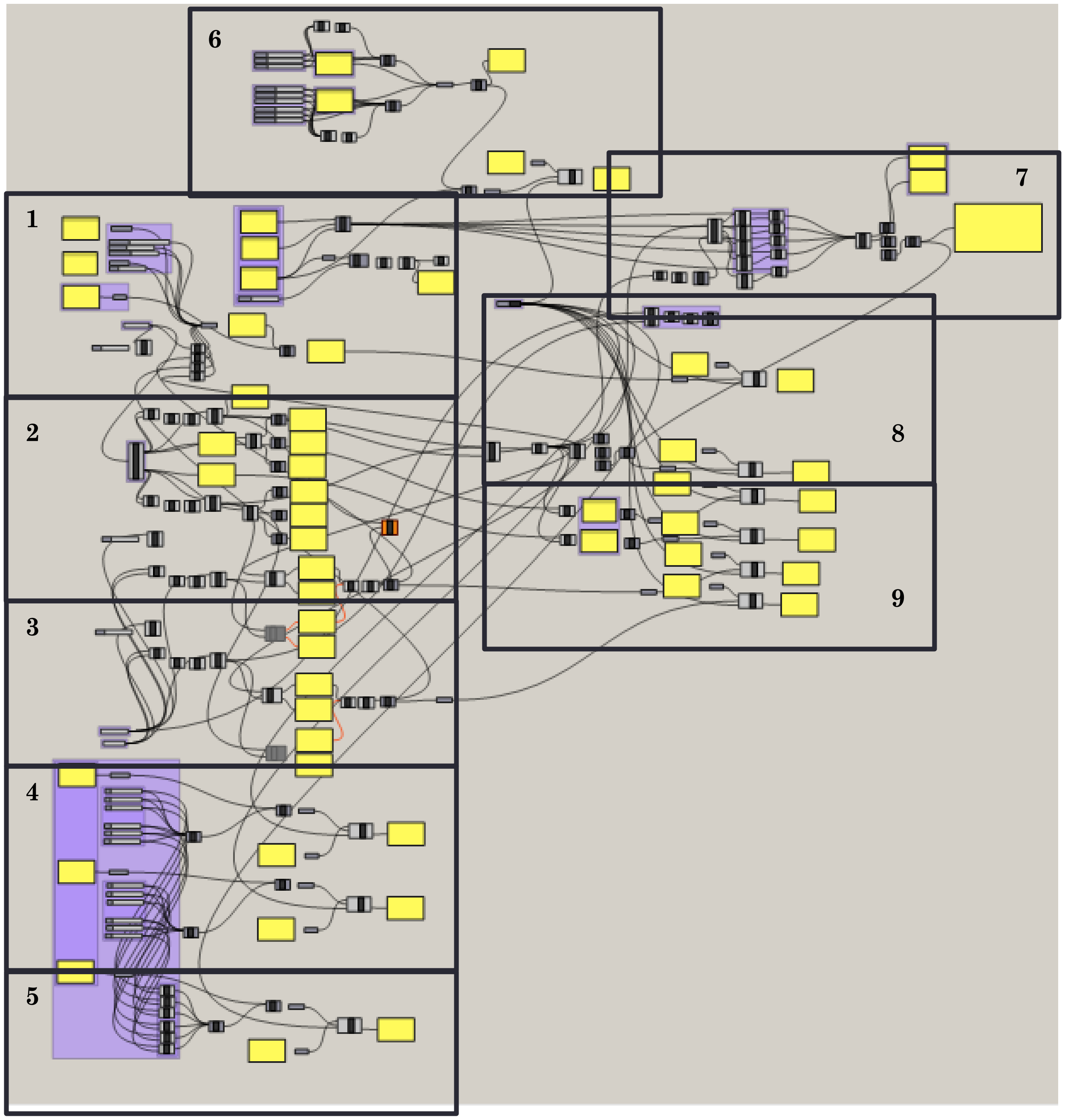
Scheible, Florian. "Parametric Engineering Everything is Possible." In *IABSE-IASS Symposium*. 2011.

Scheible, Florian and Milos, Dimcic. "Controlled Parametrical Design Over Double Curved Surfaces."

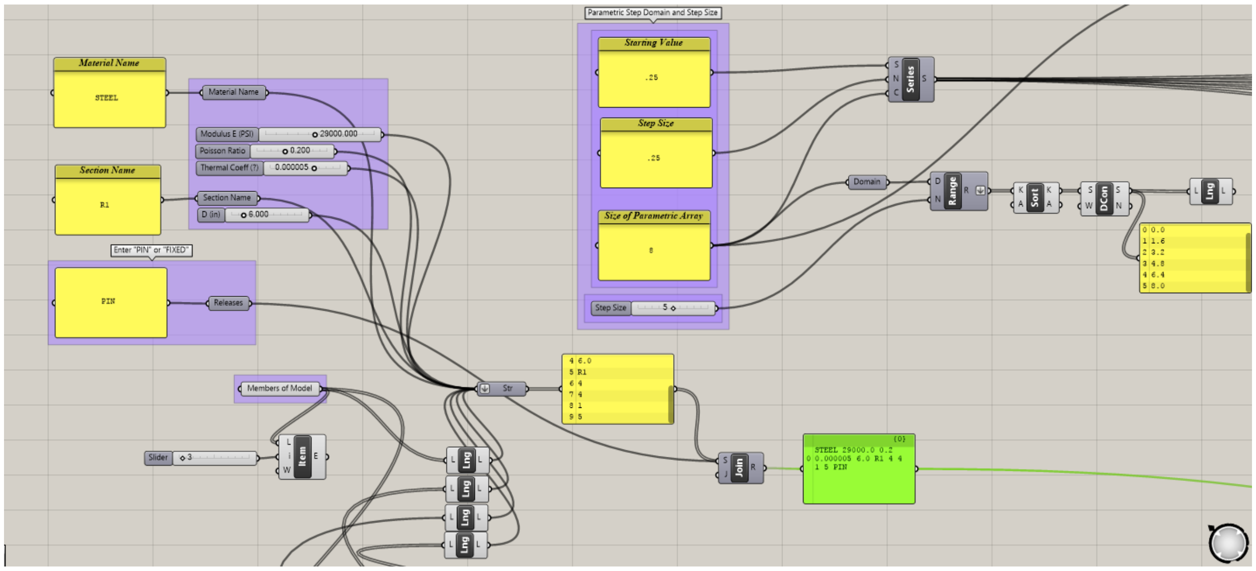
Smith, Dana K., and Michael Tardif. *Building information modeling: a strategic implementation guide for architects, engineers, constructors, and real estate asset managers*. Wiley, 2012.

Von Buelow, Peter. "Suitability of genetic based exploration in the creative design process." *Digital Creativity* 19, no. 1 (2008): 51-61.

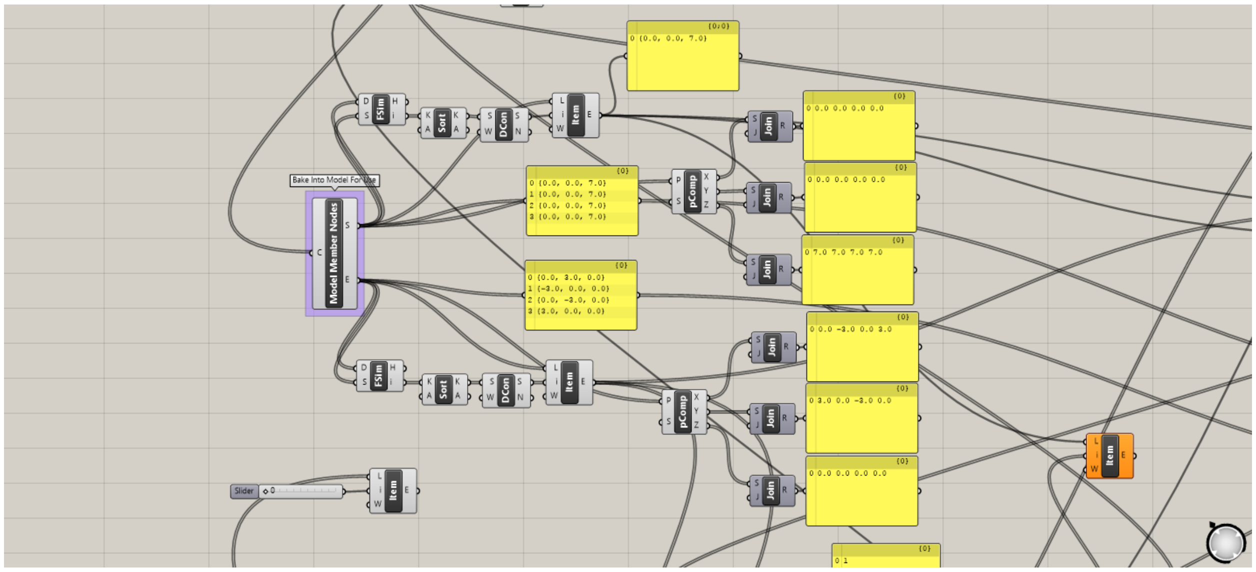
Appendix



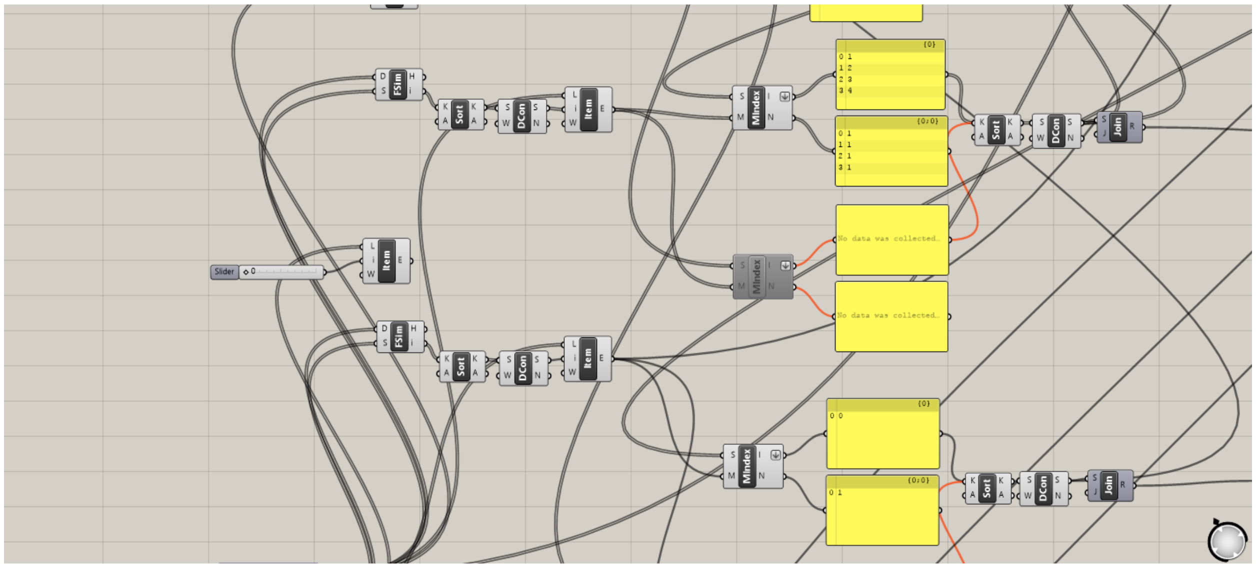
Grasshopper User Interface with Definitions


Material Name
STEEL
Material Name
Modulus E (PSI) 29000.000
Poisson Ratio 0.200
Thermal Coeff (?) 0.000005
Section Name
R1
Section Name
D (in) 6.000
Enter "PIN" or "FIXED"
PIN
Releases
Parametric Step Domain and Step Size
Starting Value
.25
Step Size
.25
Size of Parametric Array
8
Step Size 5
Series
Domain
Range
Sort
DCon
Lng
0 0.0
1 1.6
2 3.2
3 4.8
4 6.4
5 8.0
Str
4 6.0
5 R1
6 4
7 4
8 1
9 5
Join
{0}
STEEL 29000.0 0.2
0 0.000005 6.0 R1 4 4
1 5 PIN
Members of Model
Slider 3
Item
Lng

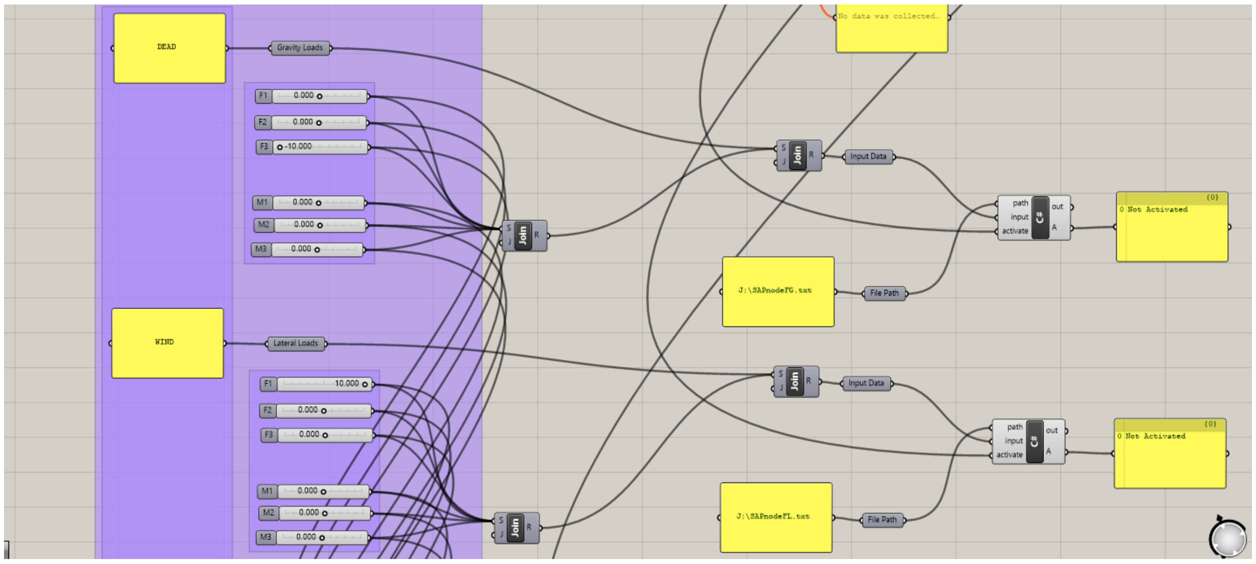
Tile 1


{0;0}
0 {0.0, 0.0, 7.0}
FSim
Sort
DCon
Item
Join
{0}
0 0.0 0.0 0.0 0.0
Bake Into Model For Use
Model Member Nodes
0 {0.0, 0.0, 7.0}
1 {0.0, 0.0, 7.0}
2 {0.0, 0.0, 7.0}
3 {0.0, 0.0, 7.0}
pComp
0 0.0 0.0 0.0 0.0
0 7.0 7.0 7.0 7.0
0 {0.0, 3.0, 0.0}
1 {-3.0, 0.0, 0.0}
2 {0.0, -3.0, 0.0}
3 {3.0, 0.0, 0.0}
0 0.0 -3.0 0.0 3.0
0 3.0 0.0 -3.0 0.0
0 0.0 0.0 0.0 0.0
Slider 0
Item
0 1

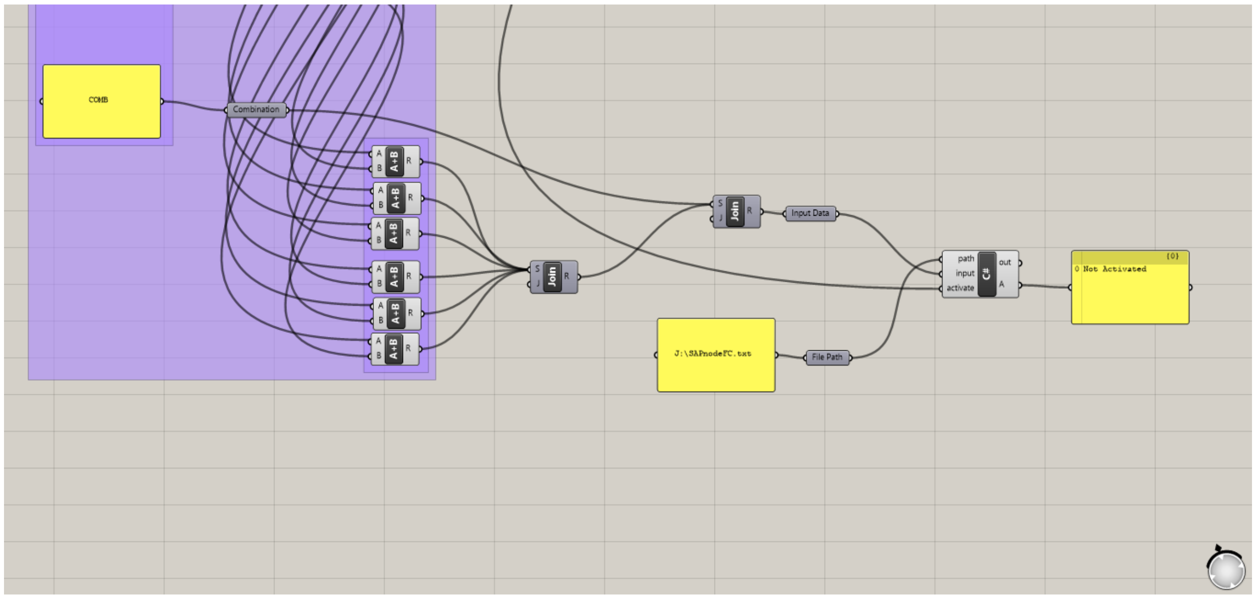
Tile 2


FSim
Sort
DCon
Item
MIndex
Slider
No data was collected.

Tile 3

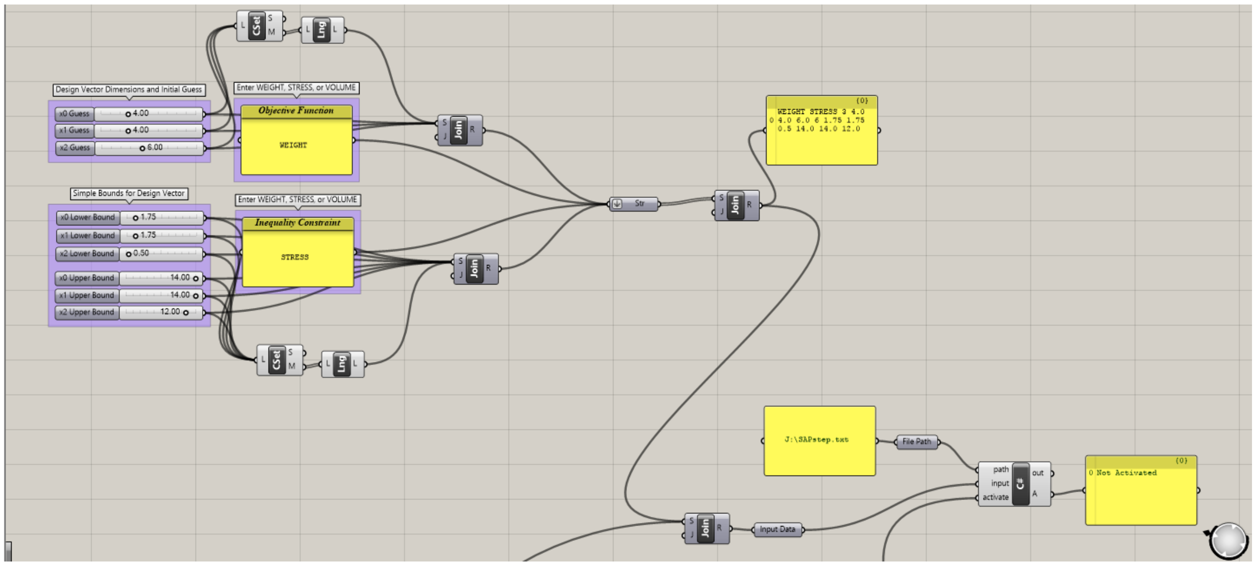

DEAD
Gravity Loads
WIND
Lateral Loads
Join
Input Data
C#
J:\SAPnodeFG.txt
J:\SAPnodeFL.txt
File Path
0 Not Activated

Tile 4



```
0 Not Activated
```

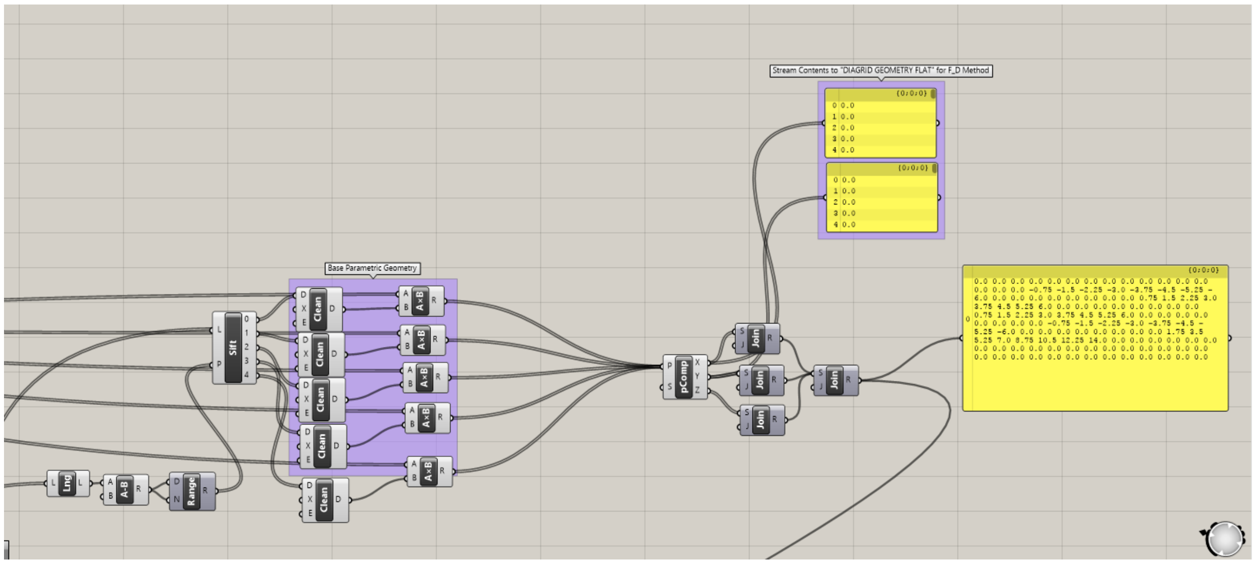
Tile 5



```
  WEIGHT STRESS 3 4.0
0 4.0 6.0 6 1.75 1.75
  0.5 14.0 14.0 12.0
```

```
0 Not Activated
```

Tile 6

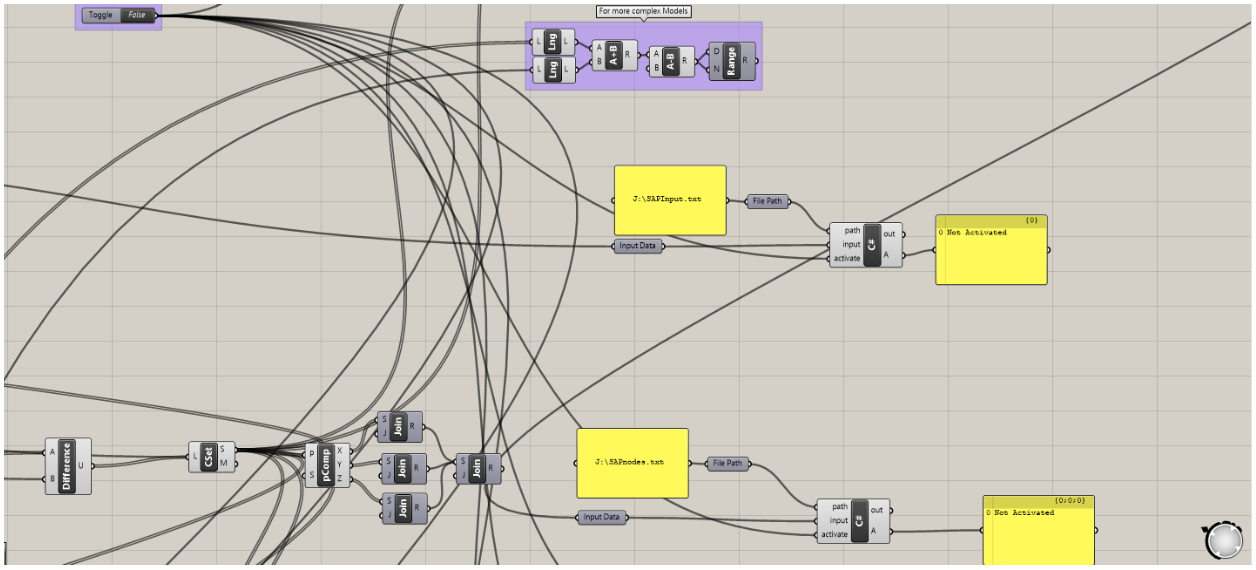


```
{0;0;0}
0 0.0
1 0.0
2 0.0
3 0.0
4 0.0
```

```
{0;0;0}
0 0.0
1 0.0
2 0.0
3 0.0
4 0.0
```

```
{0;0;0}
0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
  0.0 0.0 0.0 -0.75 -1.5 -2.25 -3.0 -3.75 -4.5 -5.25 -
  6.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.75 1.5 2.25 3.0
  3.75 4.5 5.25 6.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
  0.75 1.5 2.25 3.0 3.75 4.5 5.25 6.0 0.0 0.0 0.0 0.0
  0.0 0.0 0.0 0.0 -0.75 -1.5 -2.25 -3.0 -3.75 -4.5 -
  5.25 -6.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.75 3.5
  5.25 7.0 8.75 10.5 12.25 14.0 0.0 0.0 0.0 0.0 0.0 0.0
  0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
  0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
```

Tile 7



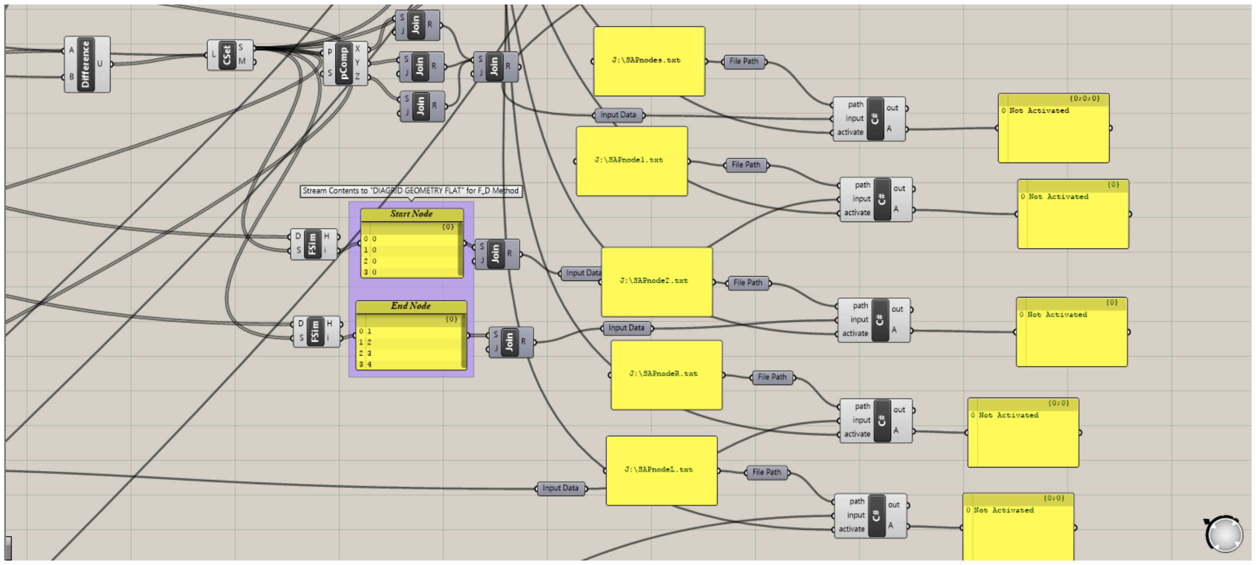
```
J:\SAPInput.txt
```

```
{0}
0 Not Activated
```

```
J:\SAPnodes.txt
```

```
{0;0;0}
0 Not Activated
```

Tile 8


Difference
CSet
pComp
Join
FSIm
Stream Contents to "DIAGRID GEOMETRY FLAT" for F_D Method
Start Node
End Node
J:\SAPnodes.txt
J:\SAPnode1.txt
J:\SAPnode2.txt
J:\SAPnodeR.txt
J:\SAPnodeL.txt
File Path
Input Data
path
input
activate
out
C#
0 Not Activated

Tile 9

Parametric Analysis Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Sap2000;
using System.IO;
using Microsoft.Office.Interop.Excel;
using System.Runtime.InteropServices;



namespace ConsoleApplication2
{
    class Program
    {
        static void Main(string[] args)
        {
            //Dimension Variables for Script
            Sap2000.SapObject SapObject;

            Sap2000.cSapModel SapModel;
            int ret;
            int i;
            double[] ModValue;
            double[] PointLoadValue;
            bool[] Restraint;
            string[] Point_Name;
            string[] FrameName;
            string[] PointName;
            int NumberResults;
            string[] Obj;
            string[] Elm;
            string[] LoadCase;
            string[] StepType;
            double[] StepNum;
            double[] U1;
            double[] U2;
            double[] U3;
            double[] R1;
            double[] R2;
            double[] R3;
            double[,] SapResult;
            string temp_string1;
            string temp_string2;
            bool temp_bool;

            //Import values from GH
            string file_name = "J:\\SAPInput.txt";
            string file_node_n = "J:\\SAPnodes.txt";
            string file_node_1 = "J:\\SAPnode1.txt";
            string file_node_2 = "J:\\SAPnode2.txt";
            string file_node_r = "J:\\SAPnodeR.txt";
            string file_node_l = "J:\\SAPnodeL.txt";
            string file_node_g = "J:\\SAPnodeFG.txt";
            string file_node_w = "J:\\SAPnodeFL.txt";
            string file_node_c = "J:\\SAPnodeFC.txt";
            string file_node_x = "J:\\SAPstep.txt";
            string[] MatProp;
            string[] Nodes;
            string[] Node1;
```

```
string[] Node2;
string[] NodeR;
string[] NodeL;
string[] NodeG;
string[] NodeW;
string[] NodeC;
string[] NodeX;
string line;
string line0;
string line1;
string line2;
string line3;
string line4;
string line5;
string line6;
string line7;
string line8;


//Material Values
System.IO.StreamReader sr = new System.IO.StreamReader(file_name);
line = sr.ReadLine();
MatProp = line.Split(' ');

//Node Locations
System.IO.StreamReader sr0 = new System.IO.StreamReader(file_node_n);
line0 = sr0.ReadLine();
Nodes = line0.Split(' ');

//Member Geometry
System.IO.StreamReader sr1 = new System.IO.StreamReader(file_node_1);
line1 = sr1.ReadLine();
Node1 = line1.Split(' ');

System.IO.StreamReader sr2 = new System.IO.StreamReader(file_node_2);
line2 = sr2.ReadLine();
Node2 = line2.Split(' ');

//Restrained Nodes
System.IO.StreamReader sr3 = new System.IO.StreamReader(file_node_r);
line3 = sr3.ReadLine();
NodeR = line3.Split(' ');

//Loaded Nodes
System.IO.StreamReader sr4 = new System.IO.StreamReader(file_node_l);
line4 = sr4.ReadLine();
NodeL = line4.Split(' ');

//Nodal Loads
System.IO.StreamReader sr5 = new System.IO.StreamReader(file_node_g);
line5 = sr5.ReadLine();
NodeG = line5.Split(' ');

System.IO.StreamReader sr6 = new System.IO.StreamReader(file_node_w);
line6 = sr6.ReadLine();
NodeW = line6.Split(' ');

System.IO.StreamReader sr7 = new System.IO.StreamReader(file_node_c);
line7 = sr7.ReadLine();
NodeC = line7.Split(' ');

//Parametric Step Definitions
System.IO.StreamReader sr8 = new System.IO.StreamReader(file_node_x);
```

```
            line8 = sr8.ReadLine();
            NodeX = line8.Split(' ');

            //Create Excel Worksheet for analysis input
            Microsoft.Office.Interop.Excel.Application xlApp = new
Microsoft.Office.Interop.Excel.Application();
            Workbook wb = xlApp.Workbooks.Add(XlWBATemplate.xlWBATWorksheet);
            Workbook wb2 = xlApp.Workbooks.Add(XlWBATemplate.xlWBATWorksheet);
            Worksheet ws = wb.Worksheets[1];
            Worksheet ws2 = wb2.Worksheets[1];

            //Create SAP object
            SapObject = new Sap2000.SapObject();

            //Start Sap2000 application
            temp_bool = true;
            SapObject.ApplicationStart(Sap2000.eUnits.kip_in_F, temp_bool, "");

            //Create SapModel Object
            SapModel = SapObject.SapModel;

            //Initialize Model
            ret = SapModel.InitializeNewModel((Sap2000.eUnits.kip_in_F));

            //New blank model
            ret = SapModel.File.NewBlank();

            //Define Material Property
            ret = SapModel.PropMaterial.SetMaterial(MatProp[0],
Sap2000.eMatType.MATERIAL_STEEL, -1, "", "");

            //Assign mehcanical properities to material defined
            ret = SapModel.PropMaterial.SetMPIsotropic(MatProp[0],
Convert.ToDouble(MatProp[1]), Convert.ToDouble(MatProp[2]),
Convert.ToDouble(MatProp[3]), 0);

            //Define Section Geometry
            ret = SapModel.PropFrame.SetCircle(MatProp[5], MatProp[0],
Convert.ToDouble(MatProp[4]), -1, "", "");

            //Define Section property modifiers
            ModValue = new double[8];
            for (i = 0; i <= 7; i++)
            {
                ModValue[i] = 1;
            }

            System.Array temp_SystemArray = ModValue;
            ret = SapModel.PropFrame.SetModifiers("R1", ref temp_SystemArray);

            //Switch Units
            ret = SapModel.SetPresentUnits(Sap2000.eUnits.kip_ft_F);

            //Add Joints
            Point_Name = new string[Convert.ToInt32(MatProp[9])];
            int numnodes = Convert.ToInt32(MatProp[9]);
            double[,] pointmatrix = new double[numnodes * Convert.ToInt32(NodeX[0]),
3];
            var horiz2 = 0;
            for (var z = 0; z <= (Convert.ToDouble(MatProp[9]) *
Convert.ToDouble(NodeX[0])) - 1; z++)
            {
                pointmatrix[z, 0] = Convert.ToDouble(Nodes[(z)]);
                pointmatrix[z, 1] = Convert.ToDouble(Nodes[z +
(Convert.ToInt32(MatProp[9]) * Convert.ToInt32(NodeX[0]))]);
                pointmatrix[z, 2] = Convert.ToDouble(Nodes[z + (2 *
Convert.ToInt32(NodeX[0]) * (Convert.ToInt32(MatProp[9])))]);
            }

            string fileContent = File.ReadAllText(file_node_r);
            string[] integerStrings = fileContent.Split(new char[] { ' ', '\t', '\r',
'\n' }, StringSplitOptions.RemoveEmptyEntries);
            double[] matrix = new double[integerStrings.Length];
            for (int n = 0; n < integerStrings.Length; n++)
            {
                matrix[n] = double.Parse(integerStrings[n]);
            }
            //Create Iteration Vectors
            for (var iteration = 0; iteration <= Convert.ToInt32(NodeX[0]) - 1;
iteration++)
            {

                foreach (string t in NodeR)
                {

                    double Point1;
                    Point1 = double.Parse(t);
                    int horiz = Convert.ToInt32(Point1);
                    ret = SapModel.PointObj.AddCartesian(pointmatrix[iteration +
(Convert.ToInt32(NodeX[0]) * (horiz)), 0], pointmatrix[iteration +
(Convert.ToInt32(NodeX[0]) * (horiz)), 1], pointmatrix[iteration +
(Convert.ToInt32(NodeX[0]) * (horiz)), 2], ref Point_Name[horiz],
Convert.ToString(horiz));
                }
                //FOR MULTIPLE PARAMETERS ONLY double Point2;
                //foreach (string u in NodeL)
                //{
                //    Point2 = double.Parse(u);
                //    int horiz2 = Convert.ToInt32(Point2);
                    for (var iter2 = 0; iter2 <= 7; iter2++)
                    {

                        ret = SapModel.PointObj.AddCartesian(pointmatrix[iter2 +
(Convert.ToInt32(NodeX[0]) * (horiz2)), 0], pointmatrix[iter2 +
(Convert.ToInt32(NodeX[0]) * (horiz2)), 1], pointmatrix[iter2 +
(Convert.ToInt32(NodeX[0]) * (horiz2)), 2], ref Point_Name[horiz2],
Convert.ToString(horiz2));


                        //Add Frame Objects by Points
                        FrameName = new string[Convert.ToInt32(MatProp[6])];
                        temp_string1 = FrameName[0];
                        temp_string2 = FrameName[0];
                        for (i = 0; i <= (Convert.ToDouble(MatProp[6]) - 1); i++)
                        {
                            ret =
SapModel.FrameObj.AddByPoint((Point_Name[Convert.ToInt32(Node1[(i)])]),
Point_Name[Convert.ToInt32(Node2[i])], ref FrameName[i], MatProp[5],
Convert.ToString(i));
                        }

                        //Add Frame Releases
                        bool[] ii = new bool[6];
                        bool[] jj = new bool[6];
                        double[] StartValue = new double[6];
                        double[] Endvalue = new double[6];
                        if (MatProp[10] == "PIN")
                        {
                            for (i = 0; i <= 3; i++)
                            {
                                ii[i] = false;
                                jj[i] = false;
                            }
                            for (i = 4; i <= 5; i++)
                            {
                                ii[i] = true;
                                jj[i] = true;
                            }
                        }
                        else
                        //if (MatProp[11] == "FIXED")
                        {
                            for (i = 0; i <= 5; i++)
                            {
                                ii[i] = false;
                                jj[i] = false;
                            }
                        }
                        for (i = 0; i <= 5; i++)
                        {
                            StartValue[i] = 0;
                            Endvalue[i] = 0;
                        }
                        for (i = 0; i <= (Convert.ToDouble(MatProp[6]) - 1); i++)
                        {
                            ret = SapModel.FrameObj.SetReleases(Convert.ToString(i),
ii, jj, StartValue, Endvalue, eItemType.Object);
                        }
                        //Add Frame/Truss Objects by COORDINATES
                        //for (i = 0; i <= (Convert.ToDouble(MatProp[6])-1); i++)
                        //{
                        //    ret =
SapModel.FrameObj.AddByCoord(Convert.ToDouble(Node1[(i)]),
Convert.ToDouble(Node1[i+Convert.ToInt32(MatProp[6])]),
Convert.ToDouble(Node1[i+(2*(Convert.ToInt32(MatProp[6])))]),
Convert.ToDouble(Node2[i]), Convert.ToDouble(Node2[i+(Convert.ToInt32(MatProp[6]))]),
Convert.ToDouble(Node2[i+(2*(Convert.ToInt32(MatProp[6])))]), ref temp_string1,
MatProp[5], Convert.ToString(i), "Global");
                        //    FrameName[i] = temp_string1;
                        //}
                        //ret = SapModel.FrameObj.AddByCoord(0, 0, 10, 8, 0, 16, ref
temp_string1, "R1", "2","Global");
                        //     FrameName[1] = temp_string1;
                        //ret = SapModel.FrameObj.AddByCoord(-1, 0, 10, 0, 0, 10, ref
temp_string1, "R1", "3","Global");
                        //     FrameName[2] = temp_string1;

                        //Assign Point Object Restraint at Base
                        PointName = new string[2];
                        Restraint = new bool[6];
                        double Point;
                        for (i = 0; i <= 3; i++)
                        {
                            Restraint[i] = true;
                        }
                        for (i = 3; i <= 5; i++)
                        {
                            Restraint[i] = false;
                        }
                        foreach (string s in NodeR)
                        {
                            Point = double.Parse(s);
                            int point = Convert.ToInt32(Point);
                            System.Array temp_SystemArray1 = Restraint;
                            ret = SapModel.PointObj.SetRestraint(Point_Name[point],
ref temp_SystemArray1, eItemType.Object);
                        }



                        //Refresh View and update or initialize zoom
                        temp_bool = false;
                        ret = SapModel.View.RefreshView(0, temp_bool);

                        //Load Patterns
                        temp_bool = true;
                        ret = SapModel.LoadPatterns.Add(NodeG[0],
Sap2000.eLoadPatternType.LTYPE_OTHER, 1, temp_bool);
                        ret = SapModel.LoadPatterns.Add(NodeW[0],
Sap2000.eLoadPatternType.LTYPE_OTHER, 0, temp_bool);
                        ret = SapModel.LoadPatterns.Add(NodeC[0],
Sap2000.eLoadPatternType.LTYPE_OTHER, 0, temp_bool);

                        //Assign Nodal Loads
                        PointLoadValue = new double[6];

                        //Gravity Loads
                        for (i = 1; i <= Convert.ToDouble(NodeG.Length) - 1; i++)
                        {
                            PointLoadValue[i - 1] = Double.Parse(NodeG[i]);
                        }
                        foreach (string s in NodeL)
                        {
                            Point = double.Parse(s);
                            int point = Convert.ToInt32(Point);
                            System.Array temp_SystemArraya = PointLoadValue;
                            ret = SapModel.PointObj.SetLoadForce(Point_Name[point],
NodeG[0], ref temp_SystemArraya, false, "Global", 0);
                        }

                        //Lateral Loads
                        for (i = 1; i <= Convert.ToDouble(NodeW.Length) - 1; i++)
                        {
                            PointLoadValue[i - 1] = Double.Parse(NodeW[i]);
                        }
                        foreach (string s in NodeL)
                        {
                            Point = double.Parse(s);
                            int point = Convert.ToInt32(Point);
                            System.Array temp_SystemArraya = PointLoadValue;
                            ret = SapModel.PointObj.SetLoadForce(Point_Name[point],
NodeW[0], ref temp_SystemArraya, false, "Global", 0);
                        }
                        //Combination (Linear ADD)
                        for (i = 1; i <= Convert.ToDouble(NodeC.Length) - 1; i++)
                        {
                            PointLoadValue[i - 1] = Double.Parse(NodeC[i]);
                        }
                        foreach (string s in NodeL)
                        {
                            Point = double.Parse(s);
                            int point = Convert.ToInt32(Point);
                            System.Array temp_SystemArraya = PointLoadValue;
                            ret = SapModel.PointObj.SetLoadForce(Point_Name[point],
NodeC[0], ref temp_SystemArraya, false, "Global", 0);
                        }

                        //Example of Point and Distributed Loading
                        //ret = SapModel.FrameObj.GetPoints(FrameName[2], ref
temp_string1, ref temp_string2);
                        //PointName[0] = temp_string1;
                        //PointName[1] = temp_string2;
                        //PointLoadValue = new double[6];
                        //PointLoadValue[2] = -10;
                        //System.Array temp_SystemArraya = PointLoadValue;
                        //ret = SapModel.PointObj.SetLoadForce(PointName[0], "2", ref
temp_SystemArraya,false,"Global",0);
                        //ret = SapModel.FrameObj.SetLoadDistributed(FrameName[2],
"2", 1, 10, 0, 1, 1.8, 1.8,"Global",System.Convert.ToBoolean(-
1),System.Convert.ToBoolean(-1),0);

                        //Example of Point Load on Frame
                        //ret = SapModel.FrameObj.SetLoadPoint(FrameName[1], "7", 1,
2, 0.5, -15, "Local",System.Convert.ToBoolean(-1),System.Convert.ToBoolean(-1),0);

                        //Switch Units
                        ret = SapModel.SetPresentUnits(Sap2000.eUnits.kip_in_F);

                        //SAVE MODEL
                        ret = SapModel.File.Save(@"C:\API\API_1-003.sdb");

                        //Create ANALYSIS model (run)
                        ret = SapModel.Analyze.RunAnalysis();

                        //Initialize Sap2000 results
                        SapResult = new double[Convert.ToInt32(MatProp[9]), 9];
                        ret = SapModel.FrameObj.GetPoints(FrameName[1], ref
temp_string1, ref temp_string2);
                        PointName[0] = temp_string1;
                        PointName[1] = temp_string2;

                        //Get Sap2000 results for ALL Load patterns
                        NumberResults = 0;
                        Obj = new string[1];
                        double[] ObjSta = new double[1];
                        Elm = new string[1];
                        double[] ElmSta = new double[1];
                        LoadCase = new string[1];
                        StepType = new string[1];
                        StepNum = new double[1];
                        U1 = new double[1];
                        U2 = new double[1];
                        U3 = new double[1];
                        R1 = new double[1];
                        R2 = new double[1];
                        R3 = new double[1];
                        double[] P = new double[1];
                        double[] V2 = new double[1];
                        double[] V3 = new double[1];
                        double[] T = new double[1];
                        double[] M2 = new double[1];
                        double[] M3 = new double[1];

                        //ObjSta[0] = 0;
                        //ElmSta[0] = 0;
                        //Axial Loads for all Load Patterns

                        //DEAD Load
                        double[] Axial = new double[1];
                        double[,] AxialResults = new
double[Convert.ToInt32(MatProp[6]), 3];
                        ret =
SapModel.Results.Setup.DeselectAllCasesAndCombosForOutput();
                        ret =
SapModel.Results.Setup.SetCaseSelectedForOutput(NodeG[0], System.Convert.ToBoolean(-
1));
                        for (i = 0; i <= Convert.ToDouble(MatProp[6]) - 1; i++)
                        {
                            System.Array temp_SystemArraya = Obj;
                            System.Array temp_SystemArrayb = Elm;
                            System.Array temp_SystemArrayc = LoadCase;
                            System.Array temp_SystemArrayd = StepType;
                            System.Array temp_SystemArraye = StepNum;
                            System.Array temp_SystemArrayf = P;
                            System.Array temp_SystemArrayg = V2;
                            System.Array temp_SystemArrayh = V3;
                            System.Array temp_SystemArrayi = T;
                            System.Array temp_SystemArrayj = M2;
                            System.Array temp_SystemArrayk = M3;
                            System.Array temp_SystemArrayl = ObjSta;
                            System.Array temp_SystemArraym = ElmSta;
                            ret = SapModel.Results.FrameForce(Convert.ToString(i),
Sap2000.eItemTypeElm.ObjectElm, ref NumberResults, ref temp_SystemArraya, ref
temp_SystemArrayl, ref temp_SystemArrayb, ref temp_SystemArraym, ref
temp_SystemArrayc, ref temp_SystemArrayd, ref temp_SystemArraye, ref
temp_SystemArrayf, ref temp_SystemArrayg, ref temp_SystemArrayh, ref
temp_SystemArrayi, ref temp_SystemArrayj, ref temp_SystemArrayk);
                            Axial[0] =
Convert.ToDouble(temp_SystemArrayf.GetValue(0));
                            AxialResults[i, 0] = Axial[0];
                        }
                        //WIND Load
                        ret =
SapModel.Results.Setup.DeselectAllCasesAndCombosForOutput();
                        ret =
SapModel.Results.Setup.SetCaseSelectedForOutput(NodeW[0], System.Convert.ToBoolean(-
1));
                        for (i = 0; i <= Convert.ToDouble(MatProp[6]) - 1; i++)
                        {
                            System.Array temp_SystemArrayaa = Obj;
                            System.Array temp_SystemArraybb = Elm;
                            System.Array temp_SystemArraycc = LoadCase;
                            System.Array temp_SystemArraydd = StepType;
                            System.Array temp_SystemArrayee = StepNum;
                            System.Array temp_SystemArrayff = P;
                            System.Array temp_SystemArraygg = V2;
                            System.Array temp_SystemArrayhh = V3;
                            System.Array temp_SystemArrayii = T;
                            System.Array temp_SystemArrayjj = M2;
                            System.Array temp_SystemArraykk = M3;
                            System.Array temp_SystemArrayll = ObjSta;
                            System.Array temp_SystemArraymm = ElmSta;
                            ret = SapModel.Results.FrameForce(Convert.ToString(i),
Sap2000.eItemTypeElm.ObjectElm, ref NumberResults, ref temp_SystemArrayaa, ref
temp_SystemArrayll, ref temp_SystemArraybb, ref temp_SystemArraymm, ref
temp_SystemArraycc, ref temp_SystemArraydd, ref temp_SystemArrayee, ref
```

```
temp_SystemArrayff, ref temp_SystemArraygg, ref temp_SystemArrayhh, ref
temp_SystemArrayii, ref temp_SystemArrayjj, ref temp_SystemArraykk);
                            Axial[0] =
Convert.ToDouble(temp_SystemArrayff.GetValue(0));
                            AxialResults[i, 1] = Axial[0];
                        }
                        //COMB Load
                        ret =
SapModel.Results.Setup.DeselectAllCasesAndCombosForOutput();
                        ret =
SapModel.Results.Setup.SetCaseSelectedForOutput(NodeC[0], System.Convert.ToBoolean(-
1));
                        for (i = 0; i <= Convert.ToDouble(MatProp[6]) - 1; i++)
                        {
                            System.Array temp_SystemArrayaaa = Obj;
                            System.Array temp_SystemArraybbb = Elm;
                            System.Array temp_SystemArrayccc = LoadCase;
                            System.Array temp_SystemArrayddd = StepType;
                            System.Array temp_SystemArrayeee = StepNum;
                            System.Array temp_SystemArrayfff = P;
                            System.Array temp_SystemArrayggg = V2;
                            System.Array temp_SystemArrayhhh = V3;
                            System.Array temp_SystemArrayiii = T;
                            System.Array temp_SystemArrayjjj = M2;
                            System.Array temp_SystemArraykkk = M3;
                            System.Array temp_SystemArraylll = ObjSta;
                            System.Array temp_SystemArraymmm = ElmSta;
                            ret = SapModel.Results.FrameForce(Convert.ToString(i),
Sap2000.eItemTypeElm.ObjectElm, ref NumberResults, ref temp_SystemArrayaaa, ref
temp_SystemArraylll, ref temp_SystemArraybbb, ref temp_SystemArraymmm, ref
temp_SystemArrayccc, ref temp_SystemArrayddd, ref temp_SystemArrayeee, ref
temp_SystemArrayfff, ref temp_SystemArrayggg, ref temp_SystemArrayhhh, ref
temp_SystemArrayiii, ref temp_SystemArrayjjj, ref temp_SystemArraykkk);
                            Axial[0] =
Convert.ToDouble(temp_SystemArrayfff.GetValue(0));
                            AxialResults[i, 2] = Axial[0];
                        }
                        //Nodal Disp. for all Load Patterns

                        //Dead Load
                        ret =
SapModel.Results.Setup.DeselectAllCasesAndCombosForOutput();
                        ret =
SapModel.Results.Setup.SetCaseSelectedForOutput(NodeG[0], System.Convert.ToBoolean(-
1));
                        for (i = 0; i <= Convert.ToDouble(MatProp[9]) - 1; i++)
                        {
                            System.Array temp_SystemArray2 = Obj;
                            System.Array temp_SystemArray3 = Elm;
                            System.Array temp_SystemArray4 = LoadCase;
                            System.Array temp_SystemArray5 = StepType;
                            System.Array temp_SystemArray6 = StepNum;
                            System.Array temp_SystemArray7 = U1;
                            System.Array temp_SystemArray8 = U2;
                            System.Array temp_SystemArray9 = U3;
                            System.Array temp_SystemArray10 = R1;
                            System.Array temp_SystemArray11 = R2;
                            System.Array temp_SystemArray12 = R3;
                            ret = SapModel.Results.JointDispl(Point_Name[i],
Sap2000.eItemTypeElm.ObjectElm, ref NumberResults, ref temp_SystemArray2, ref
temp_SystemArray3, ref temp_SystemArray4, ref temp_SystemArray5, ref
temp_SystemArray6, ref temp_SystemArray7, ref temp_SystemArray8, ref
```

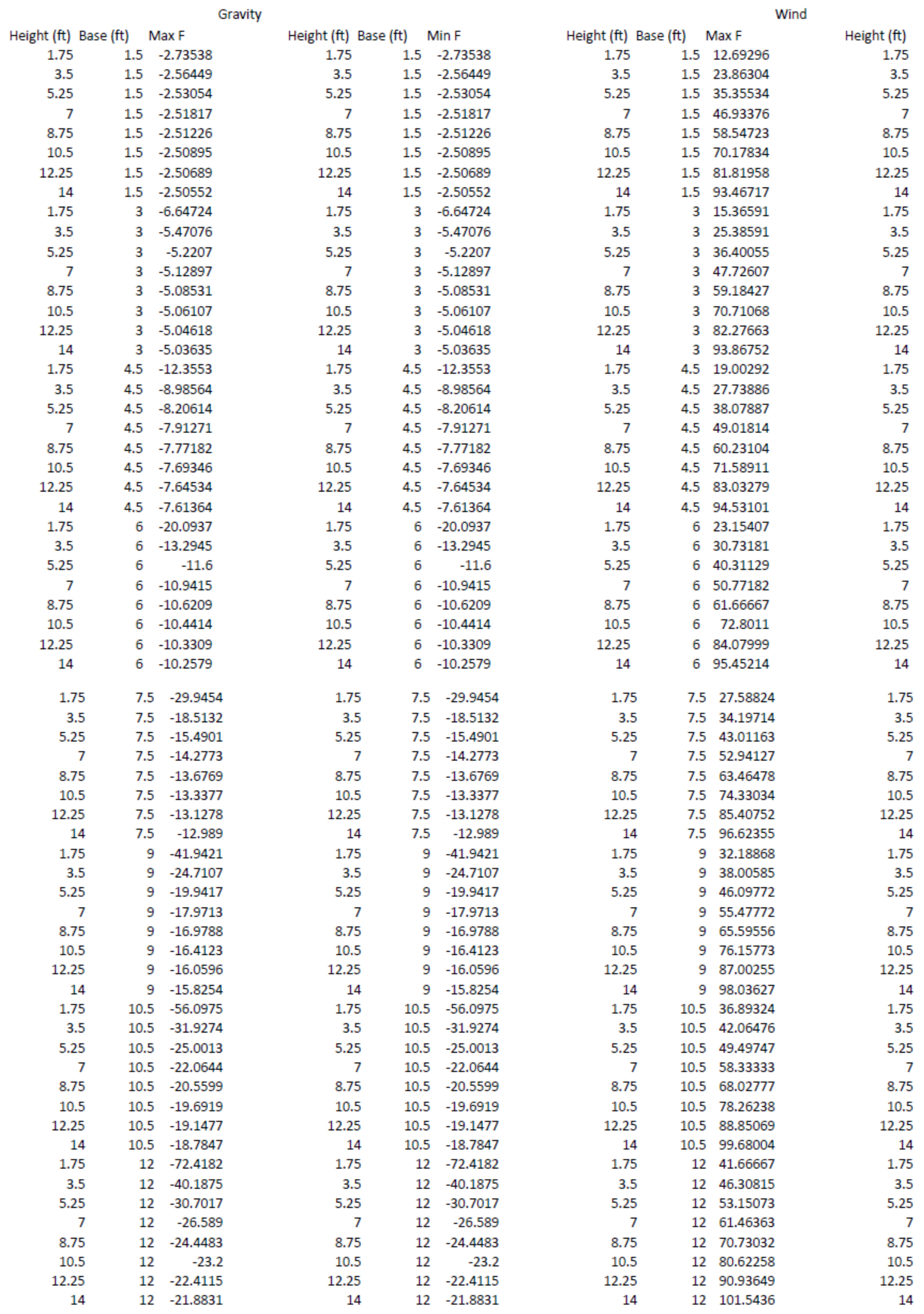
```
temp_SystemArray9, ref temp_SystemArray10, ref temp_SystemArray11, ref
temp_SystemArray12);
                            U1[0] = Convert.ToDouble(temp_SystemArray7.GetValue(0));
                            U2[0] = Convert.ToDouble(temp_SystemArray8.GetValue(0));
                            U3[0] = Convert.ToDouble(temp_SystemArray9.GetValue(0));
                            SapResult[i, 0] = U1[0];
                            SapResult[i, 1] = U2[0];
                            SapResult[i, 2] = U3[0];
                        }

                        //Wind Load
                        ret =
SapModel.Results.Setup.DeselectAllCasesAndCombosForOutput();
                        ret =
SapModel.Results.Setup.SetCaseSelectedForOutput(NodeW[0], System.Convert.ToBoolean(-
1));
                        for (i = 0; i <= Convert.ToDouble(MatProp[9]) - 1; i++)
                        {
                            System.Array temp_SystemArray2 = Obj;
                            System.Array temp_SystemArray3 = Elm;
                            System.Array temp_SystemArray4 = LoadCase;
                            System.Array temp_SystemArray5 = StepType;
                            System.Array temp_SystemArray6 = StepNum;
                            System.Array temp_SystemArray7 = U1;
                            System.Array temp_SystemArray8 = U2;
                            System.Array temp_SystemArray9 = U3;
                            System.Array temp_SystemArray10 = R1;
                            System.Array temp_SystemArray11 = R2;
                            System.Array temp_SystemArray12 = R3;
                            ret = SapModel.Results.JointDispl(Point_Name[i],
Sap2000.eItemTypeElm.ObjectElm, ref NumberResults, ref temp_SystemArray2, ref
temp_SystemArray3, ref temp_SystemArray4, ref temp_SystemArray5, ref
temp_SystemArray6, ref temp_SystemArray7, ref temp_SystemArray8, ref
temp_SystemArray9, ref temp_SystemArray10, ref temp_SystemArray11, ref
temp_SystemArray12);
                            U1[0] = Convert.ToDouble(temp_SystemArray7.GetValue(0));
                            U2[0] = Convert.ToDouble(temp_SystemArray8.GetValue(0));
                            U3[0] = Convert.ToDouble(temp_SystemArray9.GetValue(0));
                            SapResult[i, 3] = U1[0];
                            SapResult[i, 4] = U2[0];
                            SapResult[i, 5] = U3[0];
                        }

                        //COMB Load
                        ret =
SapModel.Results.Setup.DeselectAllCasesAndCombosForOutput();
                        ret =
SapModel.Results.Setup.SetCaseSelectedForOutput(NodeC[0], System.Convert.ToBoolean(-
1));
                        for (i = 0; i <= Convert.ToDouble(MatProp[9]) - 1; i++)
                        {
                            System.Array temp_SystemArray2 = Obj;
                            System.Array temp_SystemArray3 = Elm;
                            System.Array temp_SystemArray4 = LoadCase;
                            System.Array temp_SystemArray5 = StepType;
                            System.Array temp_SystemArray6 = StepNum;
                            System.Array temp_SystemArray7 = U1;
                            System.Array temp_SystemArray8 = U2;
                            System.Array temp_SystemArray9 = U3;
                            System.Array temp_SystemArray10 = R1;
                            System.Array temp_SystemArray11 = R2;
                            System.Array temp_SystemArray12 = R3;
                            ret = SapModel.Results.JointDispl(Point_Name[i], 
Sap2000.eItemTypeElm.ObjectElm, ref NumberResults, ref temp_SystemArray2, ref 
temp_SystemArray3, ref temp_SystemArray4, ref temp_SystemArray5, ref 
temp_SystemArray6, ref temp_SystemArray7, ref temp_SystemArray8, ref 
temp_SystemArray9, ref temp_SystemArray10, ref temp_SystemArray11, ref 
temp_SystemArray12);
                            U1[0] = Convert.ToDouble(temp_SystemArray7.GetValue(0));
                            U2[0] = Convert.ToDouble(temp_SystemArray8.GetValue(0));
                            U3[0] = Convert.ToDouble(temp_SystemArray9.GetValue(0));
                            SapResult[i, 6] = U1[0];
                            SapResult[i, 7] = U2[0];
                            SapResult[i, 8] = U3[0];
                        }

                        //Member Lengths
                        //int[] length = new int[Convert.ToInt32(MatProp[6])];
                        // for (var l = 0; l <= Convert.ToDouble(MatProp[6]) - 1; l++)
                        // {
                        //     length[l] = 
Math.Sqrt(Math.Sqrt(Math.Sqrt(Math.Pow(Convert.ToDouble(pointmatrix[iteration + (8 * 
(horiz)), 0]),2) + Math.Pow(Convert.ToDouble(pointmatrix[iteration + (8 * (horiz)), 
1]),2))) + Math.Pow(Convert.ToDouble(pointmatrix[iter2 + (8 * (horiz2)), 2]),2))

                        //Input Results to Excel Files Created Above
                        Range c1 = (Range)ws.Cells[1 + (Convert.ToInt32(MatProp[6]) * 
iter2) + (Convert.ToInt32(MatProp[6]) * Convert.ToInt32(NodeX[0]) * iteration), 1];
                        Range c2 = (Range)ws.Cells[1 + (Convert.ToInt32(MatProp[6]) * 
iter2) + (Convert.ToInt32(MatProp[6]) * Convert.ToInt32(NodeX[0]) * iteration) + 
(Convert.ToInt32(MatProp[6]) - 1), 3];
                        Range range = ws.get_Range(c1, c2);
                        range.Value = AxialResults;
                        Range c3 = (Range)ws2.Cells[1 + (Convert.ToInt32(MatProp[9]) * 
iter2) + (Convert.ToInt32(MatProp[9]) * Convert.ToInt32(NodeX[0]) * iteration), 1];
                        Range c4 = (Range)ws2.Cells[1 + (Convert.ToInt32(MatProp[9]) * 
iter2) + (Convert.ToInt32(MatProp[9]) * Convert.ToInt32(NodeX[0]) * iteration) + 
(Convert.ToInt32(MatProp[9]) - 1), 9];
                        Range range2 = ws2.get_Range(c3, c4);
                        range2.Value = SapResult;

                        //Unlock Model
                        ret = SapModel.SetModelIsLocked(false);


                        //Delete Existing Frame Objects
                        for (var delete = 0; delete <= (Convert.ToDouble(MatProp[6]) - 
1); delete++)
                        {
                            ret = SapModel.FrameObj.Delete(Convert.ToString(delete), 
eItemType.Object);
                        }

                        //Rename Point Objects in Inner Loop
                        for (var point = horiz2; point == horiz2; point++)
                        {
                            ret = 
SapModel.PointObj.ChangeName(Convert.ToString(point), Convert.ToString(point + 100));
                        }


                        //Switch Units
                        ret = SapModel.SetPresentUnits(Sap2000.eUnits.kip_ft_F);

                        //var applicationClass = new Application();
                        //var workbook =
applicationClass.Workbooks.Open("J:\\Forces.xls", Type.Missing, Type.Missing,
Type.Missing, Type.Missing, Type.Missing, Type.Missing, Type.Missing, Type.Missing,
Type.Missing, Type.Missing, Type.Missing, Type.Missing, Type.Missing, Type.Missing);
                        //Worksheet worksheet = workbook.Worksheets.get_Item(1);
                        //Range c1 = (Range)ws.Cells[1, 1];
                        //Range c2 = (Range)ws.Cells[Convert.ToInt32(MatProp[6]), 3];


                    }
                    //Rename Point Objects in Outer Loop ONLY IF REQUIRED
                    //    for (var point = 1; point <= 4; point++)
                    //    {
                    //        ret =
SapModel.PointObj.ChangeName(Convert.ToString(point), Convert.ToString(point + 100));
                    //    }
                    }
                    //Save Excel Files
                    wb.SaveCopyAs("J:\\Forces.xls");
                    wb2.SaveCopyAs("J:\\Disp.xls");

                    //Close Sap2000
                    SapObject.ApplicationExit(false);
                    SapModel = null;
                    SapObject = null;
                }
            }
        }
    //}
//}
```

NLOpt Optimization Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using NLOptDotNet;
using Sap2000;
using Microsoft.Office.Interop.Excel;

namespace NLOPT_TEST
{
    class Program
    {

        // verbose console output trigger
        static bool VERBOSE_OUTPUT = false;

        [STAThread]
        static void Main(string[] args)
        {
            string file_node_x = "J:\\SAPstep.txt";
            string line8;
            string[] NodeX;
            //Parametric Step Definitions
            System.IO.StreamReader sr8 = new System.IO.StreamReader(file_node_x);
            line8 = sr8.ReadLine();
            NodeX = line8.Split(' ');
            //int var;
            //var = Convert.ToInt32(NodeX[4]);
            //int var2;
            //var2 = Convert.ToInt32(NodeX[5]);
            //int var3;
            //var3 = Convert.ToInt32(NodeX[6]);
            //int var4;
            //var4 = Convert.ToInt32(NodeX[8]);
            //int var5;
            //var5 = Convert.ToInt32(NodeX[9]);
            //int var6;
            //var6 = Convert.ToInt32(NodeX[10]);
            //int var7;
            //var7 = Convert.ToInt32(NodeX[11]);
            //int var8;
            //var8 = Convert.ToInt32(NodeX[12]);
            //int var9;
            //var9 = Convert.ToInt32(NodeX[13]);

            // The optimization algorithm
            Algorithm main_alg = Algorithm.AUGLAG;
            // The local/subsidiary optimization algorithm, which may or may not be
used (see NLOpt wiki)
            Algorithm secondary_alg = Algorithm.GN_CRS2_LM;

            // Create optimization object, setting algorithm type and number of
variables
            NLOptWrapper wrapper = new NLOptWrapper(main_alg, 3);

            // Turn verbose (console) output on or off
            wrapper.VERBOSE_OUTPUT = VERBOSE_OUTPUT;

            // Set some stopping criteria
            wrapper.MaxTime = 900; //in seconds
            wrapper.XTolRelative = 1e-4;
            wrapper.FTolAbsolute = 1e-2;
            wrapper.MaxEval = 5000;
            wrapper.StopVal = 0.55;

            // Create design vector and set initial guess
            double[] x = new double[3] {4, 4, 6 };

            // Apply lower bounds on the design vector
            wrapper.SetUpperBounds(new double[3] { 14, 14, 12 });
            wrapper.SetLowerBounds(new double[3] { 1.75, 1.75, .5 });

            // Create local optimization object, if appropriate (all AUGLAG
formulations and MLSL require this)
            // ...Note that this has to be done before the main wrapper's objectives
or constraints are set
            // to prevent crashing. Don't know exactly why yet. Part of the joys of
mixing unmanaged and managed code!
            //Creation and Implementation of SAP
            //Variables

            if (main_alg.ToString().Contains("AUGLAG") ||
main_alg.ToString().Contains("MLSL"))
            {
                NLOptWrapper local_wrapper = new NLOptWrapper(secondary_alg,
wrapper.Dimension);
```

```
                /* add stopping criteria */
                local_wrapper.XTolRelative = 1e-4; local_wrapper.FTolRelative = 1e-2;

                // Haven't figured out whether the local or main algorithm stopping
criteria dominate, and when.
                // Need to inspect nlopt source code to be sure of specifics, and
differences between AUGLAG and MLSL
                local_wrapper.VERBOSE_OUTPUT = VERBOSE_OUTPUT;
                wrapper.SetLocalOptimizer(local_wrapper);
            }

            // Delegate the objective function. Data can be passed in as type
System.Object and cast in the objective function
            wrapper.SetMinObjective(new FunctionDelegate(Objective), null);

            // Add inequality constraints. Data can be passed in as type
System.Object. Feasibility tolerance passed as an argument
            my_constraint_data[] data = new my_constraint_data[4] {new
my_constraint_data(0, 0),new my_constraint_data(1, 0),new my_constraint_data(2, 0),new
my_constraint_data(3, 0)};
            wrapper.AddInequalityConstraint(new FunctionDelegate(Constraint), data[0],
1e-8);
            wrapper.AddInequalityConstraint(new FunctionDelegate(Constraint), data[1],
1e-8);
            wrapper.AddInequalityConstraint(new FunctionDelegate(Constraint), data[2],
1e-8);
            wrapper.AddInequalityConstraint(new FunctionDelegate(Constraint), data[3],
1e-8);
            //wrapper.AddInequalityConstraint(new FunctionDelegate(Constraint),
data[4], 1e-8);

            // create variable to store min objective
            double minf = 0;

            //Run the optimization, passing minf by reference. NLOptDotNet.Result
returned
            Result r = wrapper.Optimize(x, ref minf);

            Console.WriteLine("\nFound minimum after " + neval + " objective function
evaluations");
            Console.WriteLine("Found minimum at f(" + x[0] + ", " + x[1] + ", " + x[2]
+") = " + minf);
            Console.WriteLine("\nNLOpt Result: " + r.ToString());
        }

        // Function signature for objective and constraint function enforced by the
delegate 'NLOptDotNet.FunctionDelegate'
        static int neval = 0; // counts how many times the objective fnction is called
        int[] rho = new int[1] { 490 };
        static double Objective(double[] x, ref double[] grad, Object data)
        {
            Console.WriteLine(neval++);
            if (VERBOSE_OUTPUT) { Console.WriteLine(".NET Objective #" + neval + ":
objective function called with point (" + x[0] + ", " + x[1] + ")"); }
            //if (grad != null)
            //{// NLOptDotNet.NLOptWrapper will send a gradient vector by ref when the
algorithm used is gradient-based
            //    grad[0] = 0.1;
            //    grad[1] = 50;
            //    if (VERBOSE_OUTPUT) { Console.WriteLine(".NET Objective: Gradient on
objective function is (" + grad[0] + ", " + grad[1] + ")"); }
            //}
            return (Math.Sqrt((Math.Pow(x[0], 2) / 2) + (Math.Pow(x[1], 2)))) * (4) *
(490) * ((Math.PI * (Math.Pow(x[2], 2)) / 4) / 144);
        }

        static double Constraint(double[] x, ref double[] grad, Object data)
        {
            //Dimension Variables for Script
            Sap2000.SapObject SapObject;

            Sap2000.cSapModel SapModel;
            int ret;
            int i;
            double[] ModValue;
            double[] PointLoadValue;
            bool[] Restraint;
            string[] Point_Name;
            string[] FrameName;
            string[] PointName;
            int NumberResults;
            string[] Obj;
            string[] Elm;
            string[] LoadCase;
            string[] StepType;
            double[] StepNum;
            double[] U1;
            double[] U2;
            double[] U3;
            double[] R1;
            double[] R2;
            double[] R3;
            double[,] SapResult;
            string temp_string1;
            string temp_string2;
            bool temp_bool;

            //Import values from GH
            string file_name = "J:\\SAPInput.txt";
            string file_node_n = "J:\\SAPnodes.txt";
            string file_node_1 = "J:\\SAPnode1.txt";
            string file_node_2 = "J:\\SAPnode2.txt";
            string file_node_r = "J:\\SAPnodeR.txt";
            string file_node_l = "J:\\SAPnodeL.txt";
            string file_node_g = "J:\\SAPnodeFG.txt";
            string file_node_w = "J:\\SAPnodeFL.txt";
            string file_node_c = "J:\\SAPnodeFC.txt";

            string[] MatProp;
            string[] Nodes;
            string[] Node1;
            string[] Node2;
            string[] NodeR;
            string[] NodeL;
            string[] NodeG;
            string[] NodeW;
            string[] NodeC;

            string line;
            string line0;
            string line1;
            string line2;
            string line3;
            string line4;
            string line5;
            string line6;
            string line7;



            //Material Values
            System.IO.StreamReader sr = new System.IO.StreamReader(file_name);
            line = sr.ReadLine();
            MatProp = line.Split(' ');

            //Node Locations
            System.IO.StreamReader sr0 = new System.IO.StreamReader(file_node_n);
            line0 = sr0.ReadLine();
            Nodes = line0.Split(' ');

            //Member Geometry
            System.IO.StreamReader sr1 = new System.IO.StreamReader(file_node_1);
            line1 = sr1.ReadLine();
            Node1 = line1.Split(' ');

            System.IO.StreamReader sr2 = new System.IO.StreamReader(file_node_2);
            line2 = sr2.ReadLine();
            Node2 = line2.Split(' ');

            //Restrained Nodes
            System.IO.StreamReader sr3 = new System.IO.StreamReader(file_node_r);
            line3 = sr3.ReadLine();
            NodeR = line3.Split(' ');

            //Loaded Nodes
            System.IO.StreamReader sr4 = new System.IO.StreamReader(file_node_l);
            line4 = sr4.ReadLine();
            NodeL = line4.Split(' ');

            //Nodal Loads
            System.IO.StreamReader sr5 = new System.IO.StreamReader(file_node_g);
            line5 = sr5.ReadLine();
            NodeG = line5.Split(' ');

            System.IO.StreamReader sr6 = new System.IO.StreamReader(file_node_w);
            line6 = sr6.ReadLine();
            NodeW = line6.Split(' ');

            System.IO.StreamReader sr7 = new System.IO.StreamReader(file_node_c);
            line7 = sr7.ReadLine();
            NodeC = line7.Split(' ');



            //Create Excel Worksheet for analysis input
            Microsoft.Office.Interop.Excel.Application xlApp = new
Microsoft.Office.Interop.Excel.Application();
            Workbook wb = xlApp.Workbooks.Add(XlWBATemplate.xlWBATWorksheet);
            Workbook wb2 = xlApp.Workbooks.Add(XlWBATemplate.xlWBATWorksheet);
            Worksheet ws = wb.Worksheets[1];
            Worksheet ws2 = wb2.Worksheets[1];

            //Create SAP object
            SapObject = new Sap2000.SapObject();

            //Start Sap2000 application
            temp_bool = true;
            SapObject.ApplicationStart(Sap2000.eUnits.kip_in_F, temp_bool, "");
            //Create SapModel Object
            SapModel = SapObject.SapModel;

            //Initialize Model
            ret = SapModel.InitializeNewModel((Sap2000.eUnits.kip_in_F));

            //New blank model
            ret = SapModel.File.NewBlank();

            //Define Material Property
            ret = SapModel.PropMaterial.SetMaterial(MatProp[0],
Sap2000.eMatType.MATERIAL_STEEL, -1, "", "");

            //Assign mehcanical properities to material defined
            ret = SapModel.PropMaterial.SetMPIsotropic(MatProp[0],
Convert.ToDouble(MatProp[1]), Convert.ToDouble(MatProp[2]),
Convert.ToDouble(MatProp[3]), 0);

            //Define Section Geometry
            ret = SapModel.PropFrame.SetCircle(MatProp[5], MatProp[0], x[2], -1, "",
"");

            //Define Section property modifiers
            ModValue = new double[8];
            for (i = 0; i <= 7; i++)
            {
                ModValue[i] = 1;
            }
            System.Array temp_SystemArray = ModValue;
            ret = SapModel.PropFrame.SetModifiers("R1", ref temp_SystemArray);

            //Switch Units
            ret = SapModel.SetPresentUnits(Sap2000.eUnits.kip_ft_F);

            //Create Nodes
            Point_Name = new string[5] { Convert.ToString(0), Convert.ToString(1),
Convert.ToString(2), Convert.ToString(3), Convert.ToString(4) };
                    ret = SapModel.PointObj.AddCartesian(x[0], 0, 0, ref
Point_Name[4], Convert.ToString(4));
                    ret = SapModel.PointObj.AddCartesian(0, -x[0], 0, ref
Point_Name[1], Convert.ToString(1));
                    ret = SapModel.PointObj.AddCartesian(-x[0], 0, 0, ref
Point_Name[2], Convert.ToString(2));
                    ret = SapModel.PointObj.AddCartesian(0, x[0],0, ref Point_Name[3],
Convert.ToString(3));
                    ret = SapModel.PointObj.AddCartesian(0, 0, x[1], ref
Point_Name[0], Convert.ToString(0));

            //Add Frame Objects by Points
                    FrameName = new string[Convert.ToInt32(MatProp[6])];
                    temp_string1 = FrameName[0];
                    temp_string2 = FrameName[0];
                    for (i = 0; i <= (Convert.ToDouble(MatProp[6]) - 1); i++)
                    {
                        ret =
SapModel.FrameObj.AddByPoint((Point_Name[Convert.ToInt32(Node1[(i)])]),
Point_Name[Convert.ToInt32(Node2[i])], ref FrameName[i], MatProp[5],
Convert.ToString(i));
                    }

                    //Add Frame Releases
                    bool[] ii = new bool[6];
                    bool[] jj = new bool[6];
                    double[] StartValue = new double[6];
                    double[] Endvalue = new double[6];
                    if (MatProp[10] == "PIN")
                    {
                        for (i = 0; i <= 3; i++)
                        {
                            ii[i] = false;
                            jj[i] = false;
                        }
                        for (i = 4; i <= 5; i++)
                        {
                            ii[i] = true;
                            jj[i] = true;
                        }
                    }
                    else
                    //if (MatProp[11] == "FIXED")
                    {
                        for (i = 0; i <= 5; i++)
                        {
                            ii[i] = false;
                            jj[i] = false;
                        }
                    }
                    for (i = 0; i <= 5; i++)
                    {
                        StartValue[i] = 0;
                        Endvalue[i] = 0;
                    }
                    for (i = 0; i <= (Convert.ToDouble(MatProp[6]) - 1); i++)
                    {
                        ret = SapModel.FrameObj.SetReleases(Convert.ToString(i), ii,
jj, StartValue, Endvalue, eItemType.Object);
                    }

                    //Assign Point Object Restraint at Base
                    PointName = new string[2];
                    Restraint = new bool[6];
                    double Point;
                    for (i = 0; i <= 3; i++)
                    {
                        Restraint[i] = true;
                    }
                    for (i = 3; i <= 5; i++)
                    {
                        Restraint[i] = false;
                    }
                    foreach (string s in NodeR)
                    {
                        Point = double.Parse(s);
                        int point = Convert.ToInt32(Point);
                        System.Array temp_SystemArray1 = Restraint;
                        ret = SapModel.PointObj.SetRestraint(Point_Name[point], ref
temp_SystemArray1, eItemType.Object);
                    }


                    //Refresh View and update or initialize zoom
                    temp_bool = false;
                    ret = SapModel.View.RefreshView(0, temp_bool);

                    //Load Patterns
                    temp_bool = true;
                    ret = SapModel.LoadPatterns.Add(NodeG[0],
Sap2000.eLoadPatternType.LTYPE_OTHER, 1, temp_bool);
                    ret = SapModel.LoadPatterns.Add(NodeW[0],
Sap2000.eLoadPatternType.LTYPE_OTHER, 0, temp_bool);
                    ret = SapModel.LoadPatterns.Add(NodeC[0],
Sap2000.eLoadPatternType.LTYPE_OTHER, 0, temp_bool);

                    //Assign Nodal Loads
                    PointLoadValue = new double[6];

                    //Gravity Loads
                    for (i = 1; i <= Convert.ToDouble(NodeG.Length) - 1; i++)
                    {
                        PointLoadValue[i - 1] = Double.Parse(NodeG[i]);
                    }
                    foreach (string s in NodeL)
                    {
                        Point = double.Parse(s);
                        int point = Convert.ToInt32(Point);
                        System.Array temp_SystemArraya = PointLoadValue;
                        ret = SapModel.PointObj.SetLoadForce(Point_Name[point],
NodeG[0], ref temp_SystemArraya, false, "Global", 0);
                    }

                    //Lateral Loads
                    for (i = 1; i <= Convert.ToDouble(NodeW.Length) - 1; i++)
                    {
                        PointLoadValue[i - 1] = Double.Parse(NodeW[i]);
                    }
                    foreach (string s in NodeL)
                    {
                        Point = double.Parse(s);
                        int point = Convert.ToInt32(Point);
                        System.Array temp_SystemArraya = PointLoadValue;
                        ret = SapModel.PointObj.SetLoadForce(Point_Name[point],
NodeW[0], ref temp_SystemArraya, false, "Global", 0);
                    }
                    //Combination (Linear ADD)
                    for (i = 1; i <= Convert.ToDouble(NodeC.Length) - 1; i++)
                    {
                        PointLoadValue[i - 1] = Double.Parse(NodeC[i]);
                    }
                    foreach (string s in NodeL)
                    {
                        Point = double.Parse(s);
                        int point = Convert.ToInt32(Point);
                        System.Array temp_SystemArraya = PointLoadValue;
                        ret = SapModel.PointObj.SetLoadForce(Point_Name[point],
NodeC[0], ref temp_SystemArraya, false, "Global", 0);
                    }

                    //Switch Units
                    ret = SapModel.SetPresentUnits(Sap2000.eUnits.kip_in_F);

                    //SAVE MODEL
                    ret = SapModel.File.Save(@"C:\API\API_1-003.sdb");

                    //Create ANALYSIS model (run)
                    ret = SapModel.Analyze.RunAnalysis();

                    //Initialize Sap2000 results
                    SapResult = new double[Convert.ToInt32(MatProp[9]), 9];
                    ret = SapModel.FrameObj.GetPoints(FrameName[1], ref temp_string1,
ref temp_string2);
                    PointName[0] = temp_string1;
                    PointName[1] = temp_string2;

                    //Get Sap2000 results for ALL Load patterns
                    NumberResults = 0;
                    Obj = new string[1];
                    double[] ObjSta = new double[1];
                    Elm = new string[1];
                    double[] ElmSta = new double[1];
                    LoadCase = new string[1];
                    StepType = new string[1];
                    StepNum = new double[1];
                    U1 = new double[1];
                    U2 = new double[1];
                    U3 = new double[1];
                    R1 = new double[1];
                    R2 = new double[1];
                    R3 = new double[1];
                    double[] P = new double[1];
                    double[] V2 = new double[1];
                    double[] V3 = new double[1];
                    double[] T = new double[1];
                    double[] M2 = new double[1];
                    double[] M3 = new double[1];

                    //Axial Loads for all Load Patterns

                    //DEAD Load
                    double[] Axial = new double[1];
                    double[,] AxialResults = new double[Convert.ToInt32(MatProp[6]),
3];
                    ret = SapModel.Results.Setup.DeselectAllCasesAndCombosForOutput();
                    ret = SapModel.Results.Setup.SetCaseSelectedForOutput(NodeG[0],
System.Convert.ToBoolean(-1));
                    for (i = 0; i <= Convert.ToDouble(MatProp[6]) - 1; i++)
                    {
                        System.Array temp_SystemArraya = Obj;
                        System.Array temp_SystemArrayb = Elm;
                        System.Array temp_SystemArrayc = LoadCase;
                        System.Array temp_SystemArrayd = StepType;
                        System.Array temp_SystemArraye = StepNum;
                        System.Array temp_SystemArrayf = P;
                        System.Array temp_SystemArrayg = V2;
                        System.Array temp_SystemArrayh = V3;
                        System.Array temp_SystemArrayi = T;
                        System.Array temp_SystemArrayj = M2;
                        System.Array temp_SystemArrayk = M3;
                        System.Array temp_SystemArrayl = ObjSta;
                        System.Array temp_SystemArraym = ElmSta;
                        ret = SapModel.Results.FrameForce(Convert.ToString(i),
Sap2000.eItemTypeElm.ObjectElm, ref NumberResults, ref temp_SystemArraya, ref
temp_SystemArrayl, ref temp_SystemArrayb, ref temp_SystemArraym, ref
temp_SystemArrayc, ref temp_SystemArrayd, ref temp_SystemArraye, ref
temp_SystemArrayf, ref temp_SystemArrayg, ref temp_SystemArrayh, ref
temp_SystemArrayi, ref temp_SystemArrayj, ref temp_SystemArrayk);
                        Axial[0] = Convert.ToDouble(temp_SystemArrayf.GetValue(0));
                        AxialResults[i, 0] = Axial[0];
                    }
                    //WIND Load
                    ret = SapModel.Results.Setup.DeselectAllCasesAndCombosForOutput();
                    ret = SapModel.Results.Setup.SetCaseSelectedForOutput(NodeW[0],
System.Convert.ToBoolean(-1));
                    for (i = 0; i <= Convert.ToDouble(MatProp[6]) - 1; i++)
                    {
                        System.Array temp_SystemArrayaa = Obj;
                        System.Array temp_SystemArraybb = Elm;
                        System.Array temp_SystemArraycc = LoadCase;
                        System.Array temp_SystemArraydd = StepType;
                        System.Array temp_SystemArrayee = StepNum;
                        System.Array temp_SystemArrayff = P;
                        System.Array temp_SystemArraygg = V2;
                        System.Array temp_SystemArrayhh = V3;
                        System.Array temp_SystemArrayii = T;
                        System.Array temp_SystemArrayjj = M2;
                        System.Array temp_SystemArraykk = M3;
                        System.Array temp_SystemArrayll = ObjSta;
                        System.Array temp_SystemArraymm = ElmSta;
                        ret = SapModel.Results.FrameForce(Convert.ToString(i),
Sap2000.eItemTypeElm.ObjectElm, ref NumberResults, ref temp_SystemArrayaa, ref
temp_SystemArrayll, ref temp_SystemArraybb, ref temp_SystemArraymm, ref
temp_SystemArraycc, ref temp_SystemArraydd, ref temp_SystemArrayee, ref
temp_SystemArrayff, ref temp_SystemArraygg, ref temp_SystemArrayhh, ref
temp_SystemArrayii, ref temp_SystemArrayjj, ref temp_SystemArraykk);
                        Axial[0] = Convert.ToDouble(temp_SystemArrayff.GetValue(0));
                        AxialResults[i, 1] = Axial[0];
                    }
                    //COMB Load
                    ret = SapModel.Results.Setup.DeselectAllCasesAndCombosForOutput();
                    ret = SapModel.Results.Setup.SetCaseSelectedForOutput(NodeC[0],
System.Convert.ToBoolean(-1));
                    for (i = 0; i <= Convert.ToDouble(MatProp[6]) - 1; i++)
                    {
                        System.Array temp_SystemArrayaaa = Obj;
                        System.Array temp_SystemArraybbb = Elm;
                        System.Array temp_SystemArrayccc = LoadCase;
                        System.Array temp_SystemArrayddd = StepType;
                        System.Array temp_SystemArrayeee = StepNum;
                        System.Array temp_SystemArrayfff = P;
                        System.Array temp_SystemArrayggg = V2;
                        System.Array temp_SystemArrayhhh = V3;
                        System.Array temp_SystemArrayiii = T;
                        System.Array temp_SystemArrayjjj = M2;
                        System.Array temp_SystemArraykkk = M3;
                        System.Array temp_SystemArraylll = ObjSta;
                        System.Array temp_SystemArraymmm = ElmSta;
                        ret = SapModel.Results.FrameForce(Convert.ToString(i),
Sap2000.eItemTypeElm.ObjectElm, ref NumberResults, ref temp_SystemArrayaaa, ref
temp_SystemArraylll, ref temp_SystemArraybbb, ref temp_SystemArraymmm, ref
temp_SystemArrayccc, ref temp_SystemArrayddd, ref temp_SystemArrayeee, ref
temp_SystemArrayfff, ref temp_SystemArrayggg, ref temp_SystemArrayhhh, ref
temp_SystemArrayiii, ref temp_SystemArrayjjj, ref temp_SystemArraykkk);
                        Axial[0] = Convert.ToDouble(temp_SystemArrayfff.GetValue(0));
                        AxialResults[i, 2] = Axial[0];
                    }
                    //Nodal Disp. for all Load Patterns

                    //Dead Load
                    ret = SapModel.Results.Setup.DeselectAllCasesAndCombosForOutput();
                    ret = SapModel.Results.Setup.SetCaseSelectedForOutput(NodeG[0],
System.Convert.ToBoolean(-1));
                    for (i = 0; i <= Convert.ToDouble(MatProp[9]) - 1; i++)
                    {
                        System.Array temp_SystemArray2 = Obj;
                        System.Array temp_SystemArray3 = Elm;
                        System.Array temp_SystemArray4 = LoadCase;
                        System.Array temp_SystemArray5 = StepType;
                        System.Array temp_SystemArray6 = StepNum;
                        System.Array temp_SystemArray7 = U1;
                        System.Array temp_SystemArray8 = U2;
                        System.Array temp_SystemArray9 = U3;
                        System.Array temp_SystemArray10 = R1;
                        System.Array temp_SystemArray11 = R2;
                        System.Array temp_SystemArray12 = R3;
                        ret = SapModel.Results.JointDispl(Point_Name[i],
Sap2000.eItemTypeElm.ObjectElm, ref NumberResults, ref temp_SystemArray2, ref
temp_SystemArray3, ref temp_SystemArray4, ref temp_SystemArray5, ref
temp_SystemArray6, ref temp_SystemArray7, ref temp_SystemArray8, ref
temp_SystemArray9, ref temp_SystemArray10, ref temp_SystemArray11, ref
temp_SystemArray12);
                        U1[0] = Convert.ToDouble(temp_SystemArray7.GetValue(0));
                        U2[0] = Convert.ToDouble(temp_SystemArray8.GetValue(0));
                        U3[0] = Convert.ToDouble(temp_SystemArray9.GetValue(0));
                        SapResult[i, 0] = U1[0];
                        SapResult[i, 1] = U2[0];
                        SapResult[i, 2] = U3[0];
                    }

                    //Wind Load
                    ret = SapModel.Results.Setup.DeselectAllCasesAndCombosForOutput();
                    ret = SapModel.Results.Setup.SetCaseSelectedForOutput(NodeW[0],
System.Convert.ToBoolean(-1));
                    for (i = 0; i <= Convert.ToDouble(MatProp[9]) - 1; i++)
                    {
                        System.Array temp_SystemArray2 = Obj;
                        System.Array temp_SystemArray3 = Elm;
                        System.Array temp_SystemArray4 = LoadCase;
                        System.Array temp_SystemArray5 = StepType;
                        System.Array temp_SystemArray6 = StepNum;
                        System.Array temp_SystemArray7 = U1;
                        System.Array temp_SystemArray8 = U2;
                        System.Array temp_SystemArray9 = U3;
                        System.Array temp_SystemArray10 = R1;
                        System.Array temp_SystemArray11 = R2;
                        System.Array temp_SystemArray12 = R3;
                        ret = SapModel.Results.JointDispl(Point_Name[i],
Sap2000.eItemTypeElm.ObjectElm, ref NumberResults, ref temp_SystemArray2, ref
temp_SystemArray3, ref temp_SystemArray4, ref temp_SystemArray5, ref
temp_SystemArray6, ref temp_SystemArray7, ref temp_SystemArray8, ref
temp_SystemArray9, ref temp_SystemArray10, ref temp_SystemArray11, ref
temp_SystemArray12);
                        U1[0] = Convert.ToDouble(temp_SystemArray7.GetValue(0));
                        U2[0] = Convert.ToDouble(temp_SystemArray8.GetValue(0));
                        U3[0] = Convert.ToDouble(temp_SystemArray9.GetValue(0));
                        SapResult[i, 3] = U1[0];
                        SapResult[i, 4] = U2[0];
                        SapResult[i, 5] = U3[0];
                    }

                    //COMB Load
                    ret = SapModel.Results.Setup.DeselectAllCasesAndCombosForOutput();
                    ret = SapModel.Results.Setup.SetCaseSelectedForOutput(NodeC[0],
System.Convert.ToBoolean(-1));
                    for (i = 0; i <= Convert.ToDouble(MatProp[9]) - 1; i++)
                    {
                        System.Array temp_SystemArray2 = Obj;
                        System.Array temp_SystemArray3 = Elm;
                        System.Array temp_SystemArray4 = LoadCase;
                        System.Array temp_SystemArray5 = StepType;
                        System.Array temp_SystemArray6 = StepNum;
                        System.Array temp_SystemArray7 = U1;
                        System.Array temp_SystemArray8 = U2;
                        System.Array temp_SystemArray9 = U3;
                        System.Array temp_SystemArray10 = R1;
                        System.Array temp_SystemArray11 = R2;
                        System.Array temp_SystemArray12 = R3;
                        ret = SapModel.Results.JointDispl(Point_Name[i], 
Sap2000.eItemTypeElm.ObjectElm, ref NumberResults, ref temp_SystemArray2, ref 
temp_SystemArray3, ref temp_SystemArray4, ref temp_SystemArray5, ref 
temp_SystemArray6, ref temp_SystemArray7, ref temp_SystemArray8, ref 
temp_SystemArray9, ref temp_SystemArray10, ref temp_SystemArray11, ref 
temp_SystemArray12);
                        U1[0] = Convert.ToDouble(temp_SystemArray7.GetValue(0));
                        U2[0] = Convert.ToDouble(temp_SystemArray8.GetValue(0));
                        U3[0] = Convert.ToDouble(temp_SystemArray9.GetValue(0));
                        SapResult[i, 6] = U1[0];
                        SapResult[i, 7] = U2[0];
                        SapResult[i, 8] = U3[0];
                    }

                    //Input Results to Excel Files Created Above
                    Range c1 = (Range)ws.Cells[1 , 1];
                    Range c2 = (Range)ws.Cells[Convert.ToInt32(MatProp[6]) , 3];
                    Range range = ws.get_Range(c1, c2);
                    range.Value = AxialResults;
                    Range c3 = (Range)ws2.Cells[1, 1];
                    Range c4 = (Range)ws2.Cells[(Convert.ToInt32(MatProp[9])) , 9];
                    Range range2 = ws2.get_Range(c3, c4);
                    range2.Value = SapResult;

                    //Unlock Model
                    ret = SapModel.SetModelIsLocked(false);

                    //Delete Existing Frame Objects
                    for (var delete = 0; delete <= (Convert.ToDouble(MatProp[6]) - 1); 
delete++)
                    {
                        ret = SapModel.FrameObj.Delete(Convert.ToString(delete), 
eItemType.Object);
                    }

                    //Initialize Model in Same File Path
                    ret = SapModel.InitializeNewModel((Sap2000.eUnits.kip_in_F));

                    //Switch Units
                    ret = SapModel.SetPresentUnits(Sap2000.eUnits.kip_ft_F);

            //Save Excel Files
            wb.SaveCopyAs("J:\\ForcesOpt.xls");
            wb2.SaveCopyAs("J:\\DispOpt.xls");

            //Close Sap2000
            SapObject.ApplicationExit(false);
            SapModel = null;
            SapObject = null;

            //Buckling Considerations
            double fcr;
            double Fe;
            double k;
            double E;
            double I;
            double A;
            double Fy;
            double L;
            double r;
            k = 1;
            E = 29000;
            I = Convert.ToDouble(Math.PI*(Math.Pow(x[2]/2,4))/4);
            A = Convert.ToDouble(Math.PI * (Math.Pow(x[2] / 2, 2)));
            Fy = Convert.ToDouble(50);
            L = (Math.Sqrt((Math.Pow(x[0], 2) / 2) + (Math.Pow(x[1], 2))));
            r = Math.Sqrt(I / A);
            Fe = Math.Pow(Math.PI, 2) * E / (Math.Pow(k * (L*12) / r, 2));

            if (k * (L*12) / r <= 4.71 * Math.Sqrt(E / Fy))
            {
                fcr = (Math.Pow(.658, (Fy / Fe)) * Fy);
            }
            else
            {
                fcr = .877 * Fe;
            }

            if (VERBOSE_OUTPUT) { Console.WriteLine(".NET Constraint: constraint
function called with point (" + x[0] + ", " + x[1] + ")"); }
            // Data passed as System.Object to parameterize the constraint functions.
Convert to my_constraint_data struct
            my_constraint_data d = (my_constraint_data)data;
            double a = d.a;
            //if (grad != null)
            //{
            //    grad[0] = 3 * a * (a * x[0] + b) * (a * x[0] + b);
            //    grad[1] = -1.0;
            //    if (VERBOSE_OUTPUT) { Console.WriteLine(".NET Constraint: Gradient
on constraint function is (" + grad[0] + ", " + grad[1] + ")"); }
            //}
                double answer =
Math.Abs(AxialResults[Convert.ToInt32(a),0])/(Math.PI*(Math.Pow(x[2],2))/4) - fcr;
                return answer;

        }
        // A struct to pass data to constraint functions
        public struct my_constraint_data
        {
            public double a;
            public double b;
            public my_constraint_data(double a, double b)
            {
                this.a = a; this.b = b;
            }
        }
    }
}
```

Parametric Analysis Load Results (Forces in Kips)



		Gravity						Wind	
Height (ft)	Base (ft)	Max F	Height (ft)	Base (ft)	Min F	Height (ft)	Base (ft)	Max F	Height (ft)
1.75	1.5	-2.73538	1.75	1.5	-2.73538	1.75	1.5	12.69296	1.75
3.5	1.5	-2.56449	3.5	1.5	-2.56449	3.5	1.5	23.86304	3.5
5.25	1.5	-2.53054	5.25	1.5	-2.53054	5.25	1.5	35.35534	5.25
7	1.5	-2.51817	7	1.5	-2.51817	7	1.5	46.93376	7
8.75	1.5	-2.51226	8.75	1.5	-2.51226	8.75	1.5	58.54723	8.75
10.5	1.5	-2.50895	10.5	1.5	-2.50895	10.5	1.5	70.17834	10.5
12.25	1.5	-2.50689	12.25	1.5	-2.50689	12.25	1.5	81.81958	12.25
14	1.5	-2.50552	14	1.5	-2.50552	14	1.5	93.46717	14
1.75	3	-6.64724	1.75	3	-6.64724	1.75	3	15.36591	1.75
3.5	3	-5.47076	3.5	3	-5.47076	3.5	3	25.38591	3.5
5.25	3	-5.2207	5.25	3	-5.2207	5.25	3	36.40055	5.25
7	3	-5.12897	7	3	-5.12897	7	3	47.72607	7
8.75	3	-5.08531	8.75	3	-5.08531	8.75	3	59.18427	8.75
10.5	3	-5.06107	10.5	3	-5.06107	10.5	3	70.71068	10.5
12.25	3	-5.04618	12.25	3	-5.04618	12.25	3	82.27663	12.25
14	3	-5.03635	14	3	-5.03635	14	3	93.86752	14
1.75	4.5	-12.3553	1.75	4.5	-12.3553	1.75	4.5	19.00292	1.75
3.5	4.5	-8.98564	3.5	4.5	-8.98564	3.5	4.5	27.73886	3.5
5.25	4.5	-8.20614	5.25	4.5	-8.20614	5.25	4.5	38.07887	5.25
7	4.5	-7.91271	7	4.5	-7.91271	7	4.5	49.01814	7
8.75	4.5	-7.77182	8.75	4.5	-7.77182	8.75	4.5	60.23104	8.75
10.5	4.5	-7.69346	10.5	4.5	-7.69346	10.5	4.5	71.58911	10.5
12.25	4.5	-7.64534	12.25	4.5	-7.64534	12.25	4.5	83.03279	12.25
14	4.5	-7.61364	14	4.5	-7.61364	14	4.5	94.53101	14
1.75	6	-20.0937	1.75	6	-20.0937	1.75	6	23.15407	1.75
3.5	6	-13.2945	3.5	6	-13.2945	3.5	6	30.73181	3.5
5.25	6	-11.6	5.25	6	-11.6	5.25	6	40.31129	5.25
7	6	-10.9415	7	6	-10.9415	7	6	50.77182	7
8.75	6	-10.6209	8.75	6	-10.6209	8.75	6	61.66667	8.75
10.5	6	-10.4414	10.5	6	-10.4414	10.5	6	72.8011	10.5
12.25	6	-10.3309	12.25	6	-10.3309	12.25	6	84.07999	12.25
14	6	-10.2579	14	6	-10.2579	14	6	95.45214	14
1.75	7.5	-29.9454	1.75	7.5	-29.9454	1.75	7.5	27.58824	1.75
3.5	7.5	-18.5132	3.5	7.5	-18.5132	3.5	7.5	34.19714	3.5
5.25	7.5	-15.4901	5.25	7.5	-15.4901	5.25	7.5	43.01163	5.25
7	7.5	-14.2773	7	7.5	-14.2773	7	7.5	52.94127	7
8.75	7.5	-13.6769	8.75	7.5	-13.6769	8.75	7.5	63.46478	8.75
10.5	7.5	-13.3377	10.5	7.5	-13.3377	10.5	7.5	74.33034	10.5
12.25	7.5	-13.1278	12.25	7.5	-13.1278	12.25	7.5	85.40752	12.25
14	7.5	-12.989	14	7.5	-12.989	14	7.5	96.62355	14
1.75	9	-41.9421	1.75	9	-41.9421	1.75	9	32.18868	1.75
3.5	9	-24.7107	3.5	9	-24.7107	3.5	9	38.00585	3.5
5.25	9	-19.9417	5.25	9	-19.9417	5.25	9	46.09772	5.25
7	9	-17.9713	7	9	-17.9713	7	9	55.47772	7
8.75	9	-16.9788	8.75	9	-16.9788	8.75	9	65.59556	8.75
10.5	9	-16.4123	10.5	9	-16.4123	10.5	9	76.15773	10.5
12.25	9	-16.0596	12.25	9	-16.0596	12.25	9	87.00255	12.25
14	9	-15.8254	14	9	-15.8254	14	9	98.03627	14
1.75	10.5	-56.0975	1.75	10.5	-56.0975	1.75	10.5	36.89324	1.75
3.5	10.5	-31.9274	3.5	10.5	-31.9274	3.5	10.5	42.06476	3.5
5.25	10.5	-25.0013	5.25	10.5	-25.0013	5.25	10.5	49.49747	5.25
7	10.5	-22.0644	7	10.5	-22.0644	7	10.5	58.33333	7
8.75	10.5	-20.5599	8.75	10.5	-20.5599	8.75	10.5	68.02777	8.75
10.5	10.5	-19.6919	10.5	10.5	-19.6919	10.5	10.5	78.26238	10.5
12.25	10.5	-19.1477	12.25	10.5	-19.1477	12.25	10.5	88.85069	12.25
14	10.5	-18.7847	14	10.5	-18.7847	14	10.5	99.68004	14
1.75	12	-72.4182	1.75	12	-72.4182	1.75	12	41.66667	1.75
3.5	12	-40.1875	3.5	12	-40.1875	3.5	12	46.30815	3.5
5.25	12	-30.7017	5.25	12	-30.7017	5.25	12	53.15073	5.25
7	12	-26.589	7	12	-26.589	7	12	61.46363	7
8.75	12	-24.4483	8.75	12	-24.4483	8.75	12	70.73032	8.75
10.5	12	-23.2	10.5	12	-23.2	10.5	12	80.62258	10.5
12.25	12	-22.4115	12.25	12	-22.4115	12.25	12	90.93649	12.25
14	12	-21.8831	14	12	-21.8831	14	12	101.5436	14

Comb

Base (ft)	Min F	Height (ft)	Base (ft)	Max F	Height (ft)	Base (ft)	Min F
1.5	-12.693	1.75	1.5	9.973036	1.75	1.5	-15.4129
1.5	-23.863	3.5	1.5	21.30628	3.5	1.5	-26.4198
1.5	-35.3553	5.25	1.5	32.82996	5.25	1.5	-37.8807
1.5	-46.9338	7	1.5	44.41945	7	1.5	-49.4481
1.5	-5.03635	8.75	1.5	-5.03635	8.75	1.5	-61.0564
1.5	-7.69346	10.5	1.5	-7.69346	10.5	1.5	-72.6847
1.5	-7.61364	12.25	1.5	-7.61364	12.25	1.5	-84.3243
1.5	-10.4414	14	1.5	-10.4414	14	1.5	-95.9708
3	-15.3659	1.75	3	8.780519	1.75	3	-21.9513
3	-25.3859	3.5	3	19.94607	3.5	3	-30.8257
3	-36.4005	5.25	3	31.20047	5.25	3	-41.6006
3	-47.7261	7	3	42.61256	7	3	-52.8396
3	-59.1843	8.75	3	54.11133	8.75	3	-64.2572
3	-70.7107	10.5	3	65.65992	10.5	3	-75.7614
3	-82.2766	12.25	3	77.23929	12.25	3	-87.314
3	-93.8675	14	3	88.8389	14	3	-98.8961
4.5	-19.0029	1.75	4.5	6.786758	1.75	4.5	-31.2191
4.5	-27.7389	3.5	4.5	18.8228	3.5	4.5	-36.6549
4.5	-38.0789	5.25	4.5	29.91911	5.25	4.5	-46.2386
4.5	-49.0181	7	4.5	41.14022	7	4.5	-56.8961
4.5	-60.231	8.75	4.5	52.48705	8.75	4.5	-67.975
4.5	-71.5891	10.5	4.5	63.91884	10.5	4.5	-79.2594
4.5	-83.0328	12.25	4.5	75.40733	12.25	4.5	-90.6583
4.5	-94.531	14	4.5	86.93476	14	4.5	-102.127
6	-23.1541	1.75	6	3.307725	1.75	6	-43.0004
6	-30.7318	3.5	6	17.56104	3.5	6	-43.9026
6	-40.3113	5.25	6	28.79378	5.25	6	-51.8288
6	-50.7718	7	6	39.89214	7	6	-61.6515
6	-61.6667	8.75	6	51.09524	8.75	6	-72.2381
6	-72.8011	10.5	6	62.40094	10.5	6	-83.2013
6	-84.08	12.25	6	73.78448	12.25	6	-94.3755
6	-95.4521	14	6	85.22513	14	6	-105.679
7.5	-27.5882	1.75	7.5	-1.97059	1.75	7.5	-57.1471
7.5	-34.1971	3.5	7.5	15.87724	3.5	7.5	-52.517
7.5	-43.0116	5.25	7.5	27.65033	5.25	7.5	-58.3729
7.5	-52.9413	7	7.5	38.76057	7	7.5	-67.122
7.5	-63.4648	8.75	7.5	49.86518	8.75	7.5	-77.0644
7.5	-74.3303	10.5	7.5	61.05707	10.5	7.5	-87.6036
7.5	-85.4075	12.25	7.5	72.33494	12.25	7.5	-98.4801
7.5	-96.6236	14	7.5	83.6829	14	7.5	-109.564
9	-32.1887	1.75	9	-9.19677	1.75	9	-73.5741
9	-38.0058	3.5	9	13.57352	3.5	9	-62.4382
9	-46.0977	5.25	9	26.34156	5.25	9	-65.8539
9	-55.4777	7	9	37.6456	7	9	-73.3098
9	-65.5956	8.75	9	48.72813	8.75	9	-82.463
9	-76.1577	10.5	9	59.83822	10.5	9	-92.4772
9	-87.0026	12.25	9	71.02249	12.25	9	-102.983
9	-98.0363	14	9	82.28044	14	9	-113.792
10.5	-36.8932	1.75	10.5	-18.4466	1.75	10.5	-92.2331
10.5	-42.0648	3.5	10.5	10.51619	3.5	10.5	-73.6133
10.5	-49.4975	5.25	10.5	24.74874	5.25	10.5	-74.2462
10.5	-58.3333	7	10.5	36.45833	7	10.5	-80.2083
10.5	-68.0278	8.75	10.5	47.61944	8.75	10.5	-88.4361
10.5	-78.2624	10.5	10.5	58.69678	10.5	10.5	-97.828
10.5	-88.8507	12.25	10.5	69.81125	12.25	10.5	-107.89
10.5	-99.68	14	10.5	80.99004	14	10.5	-118.37
12	-41.6667	1.75	12	-29.7619	1.75	12	-113.095
12	-46.3081	3.5	12	6.61545	3.5	12	-86.0008
12	-53.1507	5.25	12	22.77888	5.25	12	-83.5226
12	-61.4636	7	12	35.12207	7	12	-87.8052
12	-70.7303	8.75	12	46.47992	8.75	12	-94.9807
12	-80.6226	10.5	12	57.58756	10.5	12	-103.658
12	-90.9365	12.25	12	68.66633	12.25	12	-113.207
12	-101.544	14	12	79.78429	14	12	-123.303

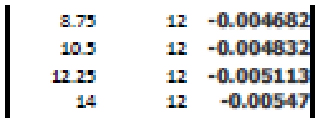
Diameter	6	inches
Area	28.27433	in^2
I	63.61725	in^4
r	1.5	in
E	29000	ksi
K	1	
Fy	50	ksi
rho	490	pcf
	0.283565	pci
Members	4	

Length					Weight			Utilization
ft	in	Fe	KL/r	Fcr	lbs	Total lbs	Total K	Gravity
2.046338	24.55606	1067.98	16.37071	49.02977	196.8808	787.5232	0.787523	0.05579
3.657185	43.88622	334.3674	29.25748	46.9665	351.8624	1407.45	1.40745	0.054602
5.356071	64.27286	155.8924	42.84857	43.71886	515.3145	2061.258	2.061258	0.057882
7.079901	84.95881	89.22024	56.63921	39.54593	681.1663	2724.665	2.724665	0.063677
8.814051	105.7686	57.56608	70.51241	34.76058	848.0111	3392.044	3.392044	0.072273
10.55344	126.6412	40.15411	84.42748	29.69097	1015.359	4061.438	4.061438	0.084502
12.29583	147.55	29.58025	98.36666	24.64417	1182.998	4731.991	4.731991	0.101723
14.04012	168.4815	22.68695	112.321	19.87724	1350.818	5403.272	5.403272	0.12605
2.75	33	591.3606	22	48.26151	264.581	1058.324	1.058324	0.137734
4.092676	49.11212	266.9949	32.74141	46.23057	393.7616	1575.046	1.575046	0.118336
5.662376	67.94851	139.4827	45.29901	43.03382	544.7844	2179.138	2.179138	0.121316
7.314369	87.77243	83.59186	58.51496	38.92628	703.7248	2814.899	2.814899	0.131761
9.003472	108.0417	55.16934	72.02777	34.21591	866.2355	3464.942	3.464942	0.148624
10.71214	128.5457	38.97311	85.69714	29.22574	1030.629	4122.516	4.122516	0.173172
12.43232	149.1878	28.93434	99.45853	24.25801	1196.129	4784.516	4.784516	0.208021
14.1598	169.9176	22.30506	113.2784	19.56578	1362.333	5449.331	5.449331	0.257406
3.63146	43.57752	339.1215	29.05168	47.00773	349.3874	1397.549	1.397549	0.262836
4.730222	56.76266	199.8733	37.84178	45.02956	455.1007	1820.403	1.820403	0.19955
6.139015	73.66817	118.6644	49.11212	41.91585	590.6424	2362.57	2.36257	0.195777
7.689278	92.27134	75.63915	61.51423	37.91502	739.7953	2959.181	2.959181	0.208696
9.310612	111.7273	51.5895	74.4849	33.32702	895.7859	3583.143	3.583143	0.233199
10.97155	131.6586	37.15194	87.77243	28.46649	1055.587	4222.349	4.222349	0.270264
12.65652	151.8782	27.91831	101.2522	23.62782	1217.7	4870.8	4.8708	0.323574
14.35705	172.2846	21.69637	114.8564	19.02772	1381.31	5525.242	5.525242	0.400134
4.58939	55.07268	212.3283	36.71512	45.30697	441.5511	1766.204	1.766204	0.443502
5.5	66	147.8401	44	43.40037	529.162	2116.648	2.116648	0.306322
6.75	81	98.1545	54	40.39932	649.4261	2597.704	2.597704	0.287133
8.185353	98.22423	66.74872	65.48282	36.54324	787.5232	3150.093	3.150093	0.299413
9.724325	116.6919	47.29321	77.7946	32.12123	935.5897	3742.359	3.742359	0.330651
11.32475	135.897	34.87068	90.59801	27.43656	1089.569	4358.275	4.358275	0.380565
12.96389	155.5667	26.61013	103.7111	22.77296	1247.273	4989.09	4.98909	0.453646
14.62874	175.5449	20.89796	117.0299	18.32752	1407.45	5629.798	5.629798	0.559702
5.584577	67.01492	143.3961	44.67662	43.21039	537.2993	2149.197	2.149197	0.693014
6.354133	76.24959	110.7657	50.83306	41.39201	611.3392	2445.357	2.445357	0.447264
7.462406	89.54887	80.30823	59.69925	38.52984	717.9676	2871.87	2.87187	0.40203
8.782084	105.385	57.98593	70.25667	34.8522	844.9355	3379.742	3.379742	0.409654
10.23169	122.7803	42.71918	81.85353	30.63482	984.404	3937.616	3.937616	0.44645
11.76329	141.1595	32.31917	94.10632	26.16693	1131.761	4527.045	4.527045	0.509716
13.34869	160.1843	25.09808	106.7895	21.71913	1284.294	5137.178	5.137178	0.604435
14.9708	179.6497	19.95389	119.7664	17.49956	1440.36	5761.441	5.761441	0.742246
6.600189	79.20227	102.6609	52.80152	40.7792	635.0126	2540.051	2.540051	1.028517
7.26292	87.15503	84.78037	58.10336	39.06313	698.7747	2795.099	2.795099	0.632583
8.25	99	65.70673	66	36.362	793.743	3174.972	3.174972	0.548422
9.460444	113.5253	49.96832	75.68355	32.89127	910.2014	3640.806	3.640806	0.546385
10.81954	129.8345	38.20322	86.55634	28.91118	1040.962	4163.848	4.163848	0.587273
12.27803	147.3363	29.6661	98.22423	24.69467	1181.285	4725.139	4.725139	0.664608
13.80444	165.6533	23.46823	110.4355	20.49713	1328.143	5312.57	5.31257	0.783504
15.37856	184.5427	18.90979	123.0285	16.58388	1479.591	5918.362	5.918362	0.954265
7.628073	91.53688	76.85782	61.02459	38.08172	733.9066	2935.627	2.935627	1.473083
8.208228	98.49873	66.37721	65.66582	36.47917	789.724	3158.896	3.158896	0.875223
9.093267	109.1192	54.08513	72.74613	33.95671	874.8748	3499.499	3.499499	0.73627
10.20417	122.45	42.94996	81.63333	30.71556	981.7558	3927.023	3.927023	0.718346
11.47552	137.7062	33.96043	91.80414	26.99875	1104.074	4416.297	4.416297	0.761512
12.85982	154.3179	27.04257	102.8786	23.06116	1237.26	4949.039	4.949039	0.853898
14.32437	171.8924	21.7955	114.5949	19.11465	1378.166	5512.663	5.512663	1.001727
15.84692	190.1631	17.80852	126.7754	15.61807	1524.653	6098.611	6.098611	1.202755
8.663862	103.9663	59.57921	69.31089	35.1902	833.5612	3334.245	3.334245	2.057907
9.17878	110.1454	53.08207	73.43024	33.70933	883.1021	3532.408	3.532408	1.192177
9.978101	119.7372	44.91816	79.82481	31.3784	960.0058	3840.023	3.840023	0.978435
11	132	36.96004	88	28.38335	1058.324	4233.296	4.233296	0.93678
12.18862	146.2635	30.10292	97.50897	24.94876	1172.683	4690.731	4.690731	0.979941
13.5	162	24.53863	108	21.31014	1298.852	5195.409	5.195409	1.088681
14.90176	178.8211	20.13922	119.2141	17.66209	1433.717	5734.87	5.73487	1.268906
16.37071	196.4485	16.68718	130.9656	14.63466	1575.046	6300.186	6.300186	1.49529

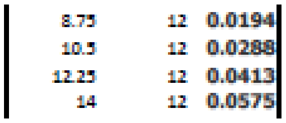
Wind	COMB MA	COMB Min	Volume cubic feet
0.258883	0.203408	0.314357	1.3125
0.508086	0.453648	0.562524	2.625
0.808698	0.750933	0.866462	3.9375
1.186816	1.123237	1.250396	5.25
1.6843	-0.14489	1.756484	6.5625
2.363626	-0.25912	2.448041	7.875
3.320038	-0.30894	3.421672	9.1875
4.70222	-0.52529	4.828172	10.5
0.318388	0.181936	0.454841	5.25
0.549115	0.431448	0.666783	10.5
0.845859	0.725022	0.966696	15.75
1.226063	1.094699	1.357427	21
1.72973	1.581467	1.877992	26.25
2.419466	2.246647	2.592285	31.5
3.39173	3.184073	3.599387	36.75
4.797535	4.540524	5.054546	42
0.404251	0.144375	0.664127	11.8125
0.616015	0.41801	0.814019	23.625
0.90846	0.71379	1.10313	35.4375
1.292842	1.085064	1.500621	47.25
1.807274	1.57491	2.039637	59.0625
2.514855	2.245407	2.784304	70.875
3.514196	3.191464	3.836929	82.6875
4.968069	4.568849	5.367288	94.5
0.511049	0.073007	0.949091	21
0.7081	0.404629	1.011572	42
0.997821	0.712729	1.282913	63
1.389363	1.091642	1.687084	84
1.91981	1.5907	2.24892	105
2.653434	2.274372	3.032496	126
3.692098	3.240005	4.144192	147
5.208133	4.650119	5.766147	168
0.638463	-0.0456	1.322531	32.8125
0.826177	0.383582	1.268772	65.625
1.11632	0.717634	1.515005	98.4375
1.519022	1.112141	1.925903	131.25
2.071655	1.627729	2.515581	164.0625
2.840621	2.333368	3.347875	196.875
3.932363	3.330471	4.534255	229.6875
5.521486	4.782001	6.26097	262.5
0.789341	-0.22553	1.804207	47.25
0.972934	0.347476	1.598392	94.5
1.267745	0.724425	1.811064	141.75
1.686701	1.144547	2.228854	189
2.268865	1.685442	2.852287	236.25
3.083974	2.423122	3.744826	283.5
4.244622	3.464997	5.024246	330.75
5.911539	4.96147	6.861608	378
0.968791	-0.4844	2.421978	64.3125
1.153117	0.288279	2.017956	128.625
1.457664	0.728832	2.186496	192.9375
1.899146	1.186966	2.611325	257.25
2.519664	1.763764	3.275563	321.5625
3.393688	2.545266	4.24211	385.875
4.648302	3.652237	5.644367	450.1875
6.382353	5.185662	7.579044	514.5
1.184042	-0.84574	3.213828	84
1.373749	0.19625	2.551247	168
1.693864	0.725942	2.661786	252
2.165482	1.237418	3.093545	336
2.835024	1.863016	3.807032	420
3.783296	2.702354	4.864238	504
5.148682	3.88778	6.409583	588
6.938573	5.451736	8.42541	672

Parametric Analysis Deflection Results

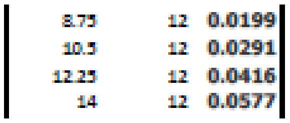
Gravity			Wind			Comb		
Height (ft)	Base (ft)	Disp (in)	Height (ft)	Base (ft)	Disp (in)	Height (ft)	Base (ft)	Disp (in)
1.75	1.5	-8.55E-05	1.75	1.5	0.0009	1.75	1.5	0.0009
3.5	1.5	-0.000146	3.5	1.5	0.006	3.5	1.5	0.006
5.25	1.5	-0.000218	5.25	1.5	0.0194	5.25	1.5	0.0194
7	1.5	-0.000296	7	1.5	0.0454	7	1.5	0.0454
8.75	1.5	-0.000378	8.75	1.5	0.0881	8.75	1.5	0.0881
10.5	1.5	-0.000466	10.5	1.5	0.1517	10.5	1.5	0.1517
12.25	1.5	-0.000557	12.25	1.5	0.2405	12.25	1.5	0.2405
14	1.5	-0.000653	14	1.5	0.3585	14	1.5	0.3585
1.75	3	-0.000299	1.75	3	0.0008	1.75	3	0.0008
3.5	3	-0.000342	3.5	3	0.0036	3.5	3	0.0036
5.25	3	-0.000455	5.25	3	0.0106	5.25	3	0.0106
7	3	-0.000586	7	3	0.0239	7	3	0.0239
8.75	3	-0.000726	8.75	3	0.0455	8.75	3	0.0455
10.5	3	-0.000873	10.5	3	0.0776	10.5	3	0.0776
12.25	3	-0.001025	12.25	3	0.1223	12.25	3	0.1223
14	3	-0.001183	14	3	0.1816	14	3	0.1816
1.75	4.5	-0.000845	1.75	4.5	0.001	1.75	4.5	0.0013
3.5	4.5	-0.000663	3.5	4.5	0.0031	3.5	4.5	0.0032
5.25	4.5	-0.000769	5.25	4.5	0.0081	5.25	4.5	0.0081
7	4.5	-0.000932	7	4.5	0.0172	7	4.5	0.0173
8.75	4.5	-0.001118	8.75	4.5	0.032	8.75	4.5	0.032
10.5	4.5	-0.001318	10.5	4.5	0.0537	10.5	4.5	0.0537
12.25	4.5	-0.001526	12.25	4.5	0.0838	12.25	4.5	0.0838
14	4.5	-0.001742	14	4.5	0.1236	14	4.5	0.1236
1.75	6	-0.002035	1.75	6	0.0014	1.75	6	0.0024
3.5	6	-0.001196	3.5	6	0.0032	3.5	6	0.0034
5.25	6	-0.001208	5.25	6	0.0072	5.25	6	0.0073
7	6	-0.001368	7	6	0.0144	7	6	0.0144
8.75	6	-0.00158	8.75	6	0.0257	8.75	6	0.0258
10.5	6	-0.001819	10.5	6	0.0424	10.5	6	0.0424
12.25	6	-0.002075	12.25	6	0.0652	12.25	6	0.0653
14	6	-0.002343	14	6	0.0955	14	6	0.0955
1.75	7.5	-0.004301	1.75	7.5	0.0018	1.75	7.5	0.0046
3.5	7.5	-0.002055	3.5	7.5	0.0035	3.5	7.5	0.004
5.25	7.5	-0.001827	5.25	7.5	0.007	5.25	7.5	0.0072
7	7.5	-0.001927	7	7.5	0.013	7	7.5	0.0132
8.75	7.5	-0.002137	8.75	7.5	0.0224	8.75	7.5	0.0225
10.5	7.5	-0.002399	10.5	7.5	0.0361	10.5	7.5	0.0361
12.25	7.5	-0.00269	12.25	7.5	0.0547	12.25	7.5	0.0548
14	7.5	-0.003	14	7.5	0.0792	14	7.5	0.0793
1.75	9	-0.008193	1.75	9	0.0024	1.75	9	0.0084
3.5	9	-0.003381	3.5	9	0.004	3.5	9	0.0052
5.25	9	-0.002692	5.25	9	0.0072	5.25	9	0.0076
7	9	-0.002651	7	9	0.0125	7	9	0.0128
8.75	9	-0.002817	8.75	9	0.0207	8.75	9	0.0208
10.5	9	-0.003077	10.5	9	0.0323	10.5	9	0.0325
12.25	9	-0.003388	12.25	9	0.0482	12.25	9	0.0483
14	9	-0.00373	14	9	0.0689	14	9	0.069
1.75	10.5	-0.014389	1.75	10.5	0.0031	1.75	10.5	0.0145
3.5	10.5	-0.005343	3.5	10.5	0.0047	3.5	10.5	0.007
5.25	10.5	-0.003881	5.25	10.5	0.0076	5.25	10.5	0.0085
7	10.5	-0.003586	7	10.5	0.0124	7	10.5	0.0129
8.75	10.5	-0.003654	8.75	10.5	0.0197	8.75	10.5	0.0201
10.5	10.5	-0.00388	10.5	10.5	0.0301	10.5	10.5	0.0303
12.25	10.5	-0.004188	12.25	10.5	0.044	12.25	10.5	0.0442
14	10.5	-0.004547	14	10.5	0.0621	14	10.5	0.0623
1.75	12	-0.023685	1.75	12	0.004	1.75	12	0.0237
3.5	12	-0.008142	3.5	12	0.0054	3.5	12	0.0097
5.25	12	-0.005485	5.25	12	0.0082	5.25	12	0.0098
7	12	-0.004785	7	12	0.0127	7	12	0.0136



8.75	12	-0.004682
10.5	12	-0.004832
12.25	12	-0.005113
14	12	-0.00547



8.75	12	0.0194
10.5	12	0.0288
12.25	12	0.0413
14	12	0.0575



8.75	12	0.0199
10.5	12	0.0291
12.25	12	0.0416
14	12	0.0577

Optimization Results for DEAD load case

Optimization Routine for 4 Element Truss				Initial Value	(4,4,6)	
Variables	b,h,D	(ft,ft,in)		UB	(14,14,12)	
				LB	(1.75,1.75,.5)	
Runtime (sec)	100	(15-16 Iterations)		Initial Weight	1885.34825	lbs
	Base Distance	Height	Cross Section D	WEIGHT	% Reduction	
Trial	ft	ft	in	lbs		
1	13.8	8.441	0.514	38.39	97.96377142	
2	5.637	13.472	0.508	38.8	97.94202477	
3	9.147	13.546	2.181	763.1	59.52471911	
4	3.623	2.836	0.8438	29.091	98.45699594	
5	12.175	8.448	0.699	62.29	96.69610111	
6	4.23	3.404	0.786	29.9	98.4140861	
9	10.954	6.615	1.283	179.2	90.49512471	
8	13.214	5.779	1.846	400.14	78.77633483	
9	3.406	11.817	1.034	137.93	92.68411022	
10	9.273	8.893	1.396	230.19	87.79058458	
Runtime (sec)	300	(39-84 Iterations)				
	Base Distance	Height	Cross Section D	WEIGHT	% Reduction	
Trial	ft	ft	in	lbs		
1	11.741	4.158	0.663	43.6	97.6874299	
2	6.041	2.413	1.468	112.96	94.00853397	
3	12.214	10.884	0.818	99.37	94.7293557	
4	1.75	3.723	0.5	10.485	99.44386932	
5	1.75	1.75	0.5	5.728	99.69618345	
6	12.394	9.303	1.03	145.5	92.28259289	
7	5.806	1.75	0.5	11.929	99.3672787	
8	11.146	1.75	0.5	21.577	98.855543	
9	4.32	3.741	0.623	20.036	98.93727856	
10	2.911	6.805	0.7056	37.8	97.99506537	
Runtime (sec)	900	(130-180 Iterations)				
	Base Distance	Height	Cross Section D	WEIGHT	% Reduction	
Trial	ft	ft	in	lbs		
1	1.75	1.75	0.5	5.728	99.69618345	
2	1.75	1.75	0.5	5.728	99.69618345	
3	1.75	1.75	0.5	5.728	99.69618345	
4	1.75	1.75	0.5	5.728	99.69618345	
5	1.75	1.75	0.5	5.728	99.69618345	
6	1.75	1.75	0.5	5.728	99.69618345	
7	1.75	1.75	0.5	5.728	99.69618345	
8	1.75	1.75	0.5	5.728	99.69618345	
9	1.75	1.75	0.5	5.728	99.69618345	
10	1.75	1.75	0.5	5.728	99.69618345	