SELECTED TOPICS IN DEVELOPMENT OF A CAD APPLICATION FOR SIZING LIGHT GAUGE COLD ROLLED STEEL ROOF TRUSSES

By

KEYUR SHARADKUMAR PANDYA

Bachelor of Engineering

National Institute of Technology

Surat, India

2005

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July, 2008

SELECTED TOPICS IN DEVELOPMENT OF A CAD

APPLICATION FOR SIZING LIGHT GUAGE COLD

ROLLED STEEL ROOF TRUSSES

Thesis Approved:

Dr. Ronald D. Delahoussaye

Thesis Adviser

Dr. Hongbing B. Lu

Dr. Jeremy A. Morton

Dr. Blayne E. Mayfield

Dr. A. Gordon Emslie

Dean of the Graduate College

ACKNOWLEDGMENTS

I would like to take this chance to thank my advisor, Dr. Ronald D. Delahoussaye. He has supported me and guided me not only as an advisor but like a family member. This project would not have been possible without him. Also thanks to my committee members Dr. Hongbing B. Lu, Dr. Jeremy A. Morton, and Dr. Blayne E. Mayfield, who offered their guidance and support. Thanks to NPDC and our corporate sponsors for providing with the financial means to complete this project. And finally, thanks to my parents, and numerous friends who endured this long process with me, always offering support and love.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Page

CHAPTER I

INTRODUCTION

Pre-fabricated roof trusses rapidly increased in popularity during the 1960s and they are widely used in many buildings even today. However, the design procedure for these roof trusses is very tedious and involves much calculation based on the codes. This complexity in the design procedure may leads to an improper designing and over loading. This may lead to a failure of the roof trusses. A computer program which can do all of these calculation will be very useful in increasing the accuracy in the design procedure.

The design procedure starts with an estimation of different loads on the whole roof structure. These loads are transferred into the trusses and forces on each truss member are calculated. Determination of these forces leads to material and cross section selection. However, there may be many combinations of materials and cross sections which satisfy the strength and safety factor requirement. Thus, it is an iterative process to reach an optimum solution. Sometimes an even better solution can be achieved by changing the truss layout in a roof structure. Therefore, the entire procedure is an iterative process.

In most cases, there are many combinations which satisfy the engineering requirement. The optimization is determined based on economic consideration. The Cross section of a truss member directly influences its weight, which is closely related to material cost. Perfect balance between load safety and cost of raw material needs to be achieved for the most economical solution. The procedure becomes even more complicated when other factors like the cost of labor, availability of machinery and transportation come into consideration. If a tool were available that would allow the user to enter various combinations and choose the best one based on a given set of parameters, it would dramatically increase the efficiency of the roof truss design.

The goal of the current project is to develop an application which will let users easily define building and truss geometries. Then, optimum truss member size will be calculated based on load inputs. The application will be user friendly with required features and is expected to facilitate a wide range of users. However, this project is a proprietary work and much information related to this project can not be disclosed. The purpose of this document is to discuss the selected topics involved in the development of this CAD application. The selected topics cover the use of an optimization technique to generate the desired roof slopes, the procedure to size the trusses, the data structure required for the computer program, the fonts to display text in 3D, the method to associate constraints on the geometry and the implementation of Undo/Redo functionality in the application.

CHAPTER II

OUTLINE OF THE PROJECT

The computer graphics enabled interactive tool, which understands a group of trusses, is very useful in analyzing the truss system in a better way. It assists in the overall and detailed visualization of trusses arranged under the roof envelop. The application understands the basic structure of a building, which influences the overall truss system. Also, the application contains a predefined library of trusses which can be used in the program. Moreover, enough functionality is provided by the application to achieve user-friendliness. The main focus of the application is to get data from the user about loading conditions and structural parameters including truss layout under the roof envelop. With these input parameters, the application will size all trusses using its engineering knowledge programmed underneath. Development of such an application required two major parts. First, a user-friendly CAD application which allows a user to input all parameters specified above was developed. Second, a method to size the trusses to achieve optimum results was developed.

Sizing of Trusses

A method was required to be developed or selected to analyze the whole roof-truss system. Different loads are applied on roofs which are transferred to the supports through the trusses. [1]First, a technique is required to distribute overall loads on the roofs to all trusses based on their position and orientation in the system. The next step is to calculate the forces in all the members using FEA analysis. Optimum member sizes are chosen based on these loads. The method is described in detail in chapter –3 and chapter – 4.

A CAD application

The goal of this part of the project is to develop general features required in any CAD application. Discussion about which operating system or which programming language will give the best result is not in the scope of this project. The application is a Microsoft Foundation Classes (MFC) based Visual C++ (VC++) application built with Visual Studio 2005. This selection was based on the learning curve involved in the programming language and the fact that Major CAD applications like AutoCAD and SolidWorks are MFC based VC++ applications [8].

A graphic language is needed to render the three dimensional display on the computer screen. OpenGL 1.1 version is used as the graphics API in this application. OpenGL [9] is a very powerful open source graphics API with good document support. Other graphics API like DirectX [10] are also available; however selecting the best graphics API for the CAD application is not the focus of this project. A screen shot of the application is shown below.

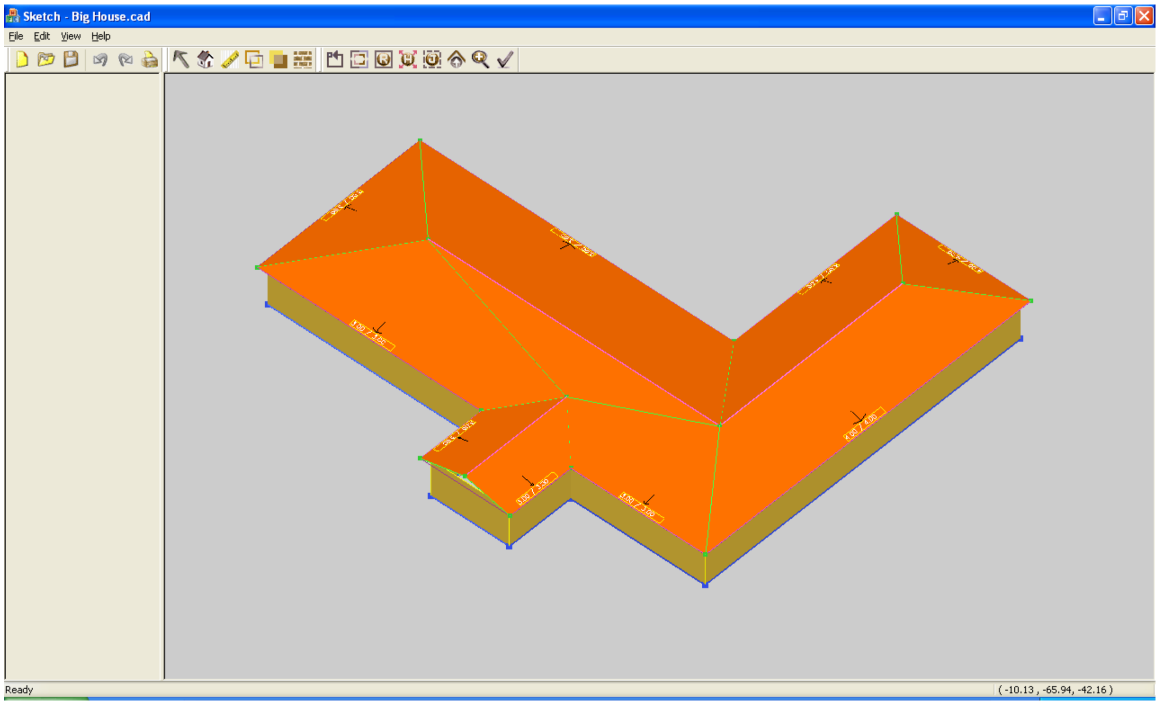

Figure 1: Screenshot of the application

There have been many problems encountered during the development of this CAD application. All major problems encountered are explained in chapter – 5 to chapter – 8 with detailed consideration of all influencing factors. Related discussion of all problems such as other people's approach, methods used and reasons to support those methods are included in chapter – 5 to chapter – 8.

Discussions in chapter – 3 to chapter – 8 are general and do not relate to any specific programming language or operating system. The issues discussed are general enough to be considered for CAD application development in general.

CHAPTER III

GENERATION OF DESIRED ROOF SLOPES USING OPTIMIZATION

When architect define a roof panel, they define it in terms of the slope of a panel. The profile of every roof panel is calculated from the intersection of adjoining panels. Likewise the profile of a whole roof envelope can be calculated. However, many times the slopes of these panels are not flawlessly defined and the intersection of panels results in discontinuity of the roof profile. This may cause many problems like non-flat panels, leakage in roof, etc. Moreover, if the parameters of the roof profile are incorrect, then complications related to the trusses are likely, as they are designed based on incorrect data. Erroneous design of roof a panel might lead to disastrous results. For this reason determining the correct slope of a roof panel is of high importance.

Consider a simple rectangular building covered with a four sided roof as shown in the figure. All combinations of slopes on these four panels are possible as long as the Ridge Line stays inside the building perimeter.

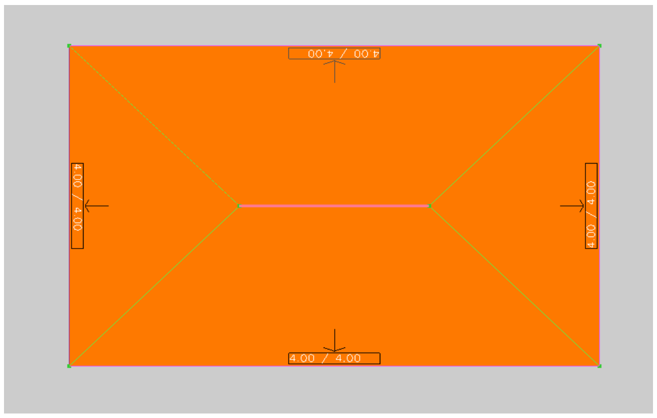

Figure 2: Simple four sided building

Now consider an ‘L’-shaped building shown in the figure with both sides having different widths. There cannot be a case where the slopes of each roof panel are the same. The only possible solution is to get the desired slope on some panels and calculate modified slope on other panels. The modified slope of the panels needs to be calculated to achieve the closest solution to a desired roof profile as mathematically / physically possible. This scenario leads to the conclusion that the problem can be solved by an optimization technique.

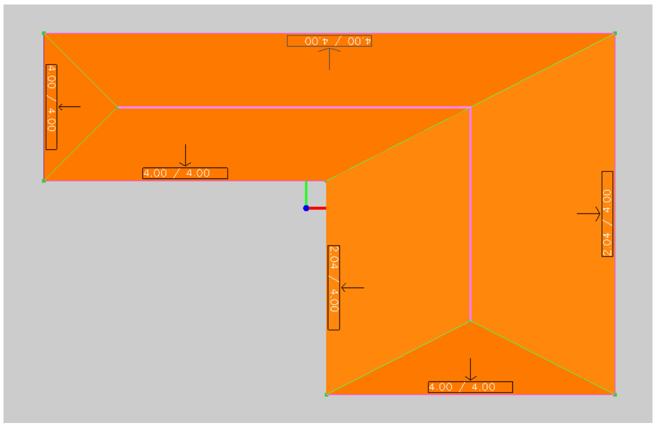

Figure 3: Simple ‘L’ shaped building

[11]Optimization is a formalized process of selecting alternatives and choosing between them to achieve the “best” design. In the above case, a design will be considered “best” if the modified roof profile is closest to desired roof profile. In other words, cumulative change in modified roof slope from user input should be as small as possible.

Optimization technique

There are many computational techniques to determine an optimal solution. Selection of the technique depends on the following parameters:

- Set of possible options
- Nature of an error function
- Number of degrees of freedom

As there is no constraint on the set of possible options, it is not possible to pick one option and compare with another. Therefore, comparing a set of possible options is not a feasible selection in this case.

Our objective function is a simple weighted sum of the difference between modified slope and desired slope. Methods which require derivatives of an objective function cannot be used. In such cases, simplex method would be a good option to choose. However, the number of the design variables is not fixed which restricts the use of Linear Programming. In such a case, downhill simplex method/ Nelder-Mead method serves as a good alternative.

There are other optimization methods also which may be faster than Nelder-Mead method [11]. However, comparing different optimization technique is not within the scope of the project. The Nelder-Mead method is described in detail in the next section.

Nelder-Mead method

The Nelder-Mead method is a numerical method for minimizing objective functions in multi-dimensional space. It uses an N+1 dimensional space for N number of vertices. Nelder-Mead generates new test positions by extrapolating the behavior of an objective function. The objective function is measured at each test point and arranged as a simplex. The algorithm then decides to replace one point in a simplex with the new test point. The progressive replacement takes place by reflecting the worst point about the centroid of remaining points.

Moreover, if reflection generates better result, then the radius of reflection is increased and if it generates a poor result, it is an indication that the minima has passed, implying that the radius of reflection should be reduced. Thus, the algorithm proceeds towards the minima by increasing or decreasing the radius of reflection. However, one of the disadvantages in Nelder-Mead is that it often finds the local minima. One solution to resolve this problem is to restart optimization with a large radius of reflection.

Optimization parameter

In any optimization, there are three factors that are taken into consideration: objective function, design variable, and performance variable. In the present scenario, objective function is a function of the difference between a modified slope and the desired slope. Next design variables must be selected.

Design variables are the set of variables which need to be optimized. At first glance, it looks like slopes are the final values which need to be optimized. However, when design variables are selected, optimization might generate a combination of slope which will result in discontinuity of the Ridge Line. But, if the end points of the Ridge Line are controlled in such a way that it will not result in discontinuity then the problem is nullified. Moreover, the slope of each panel can be controlled by controlling the end points of a Ridge Line. Therefore end points of a Ridge Line serve the purpose of the design variable in optimization. Some of the results from the application have been shown below.

Example 1: A four sided building with desired slope of 4 in 12 on all roof panels.

Starting Condition:

Panel	Slope (in 12)
A	1.92
B	1.65
C	1.65
D	1.92

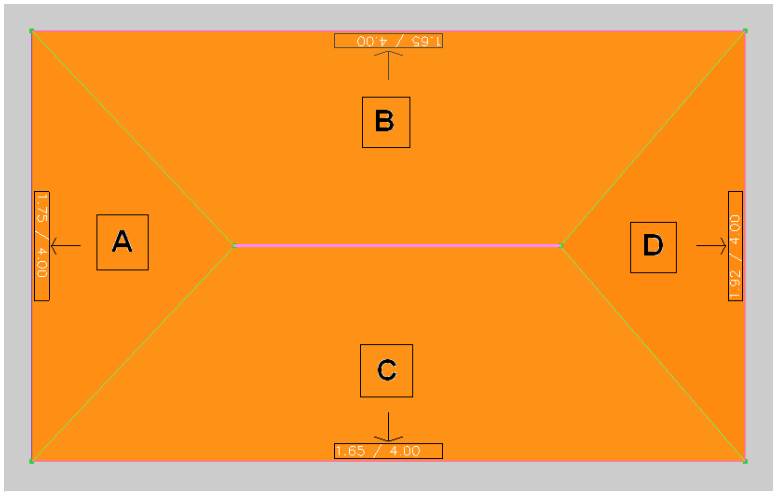

Figure 4: Starting condition (Example 1)

After Optimization:

Panel	Slope (in 12)
A	4.00
B	4.00
C	4.00
D	4.00

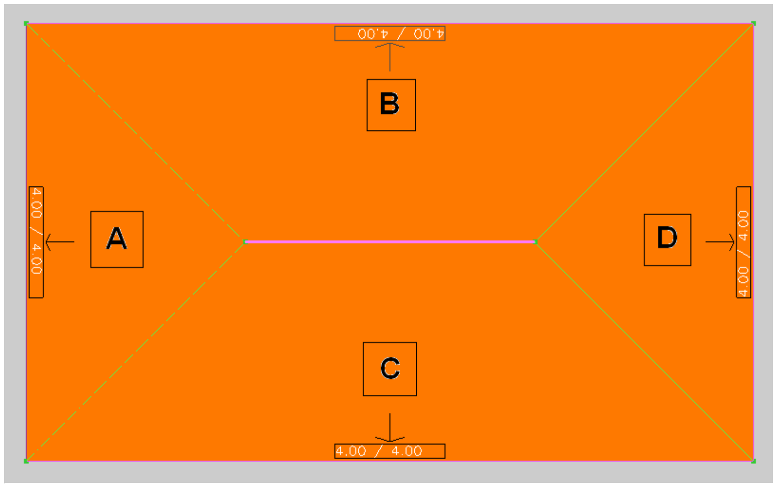

Figure 5: After optimization (Example 1)

As it is seen in the results, all panels have the desired slope after optimization. Required coordinates for ridge line are set to achieve the result.

Example 2: An ‘L’ shaped building with desired slope of 4 in 12 on all roof panels.

Starting Condition:

Panel	Slope (in 12)
A	2.60
B	3.04
C	3.04
D	1.89
E	1.89
F	3.18

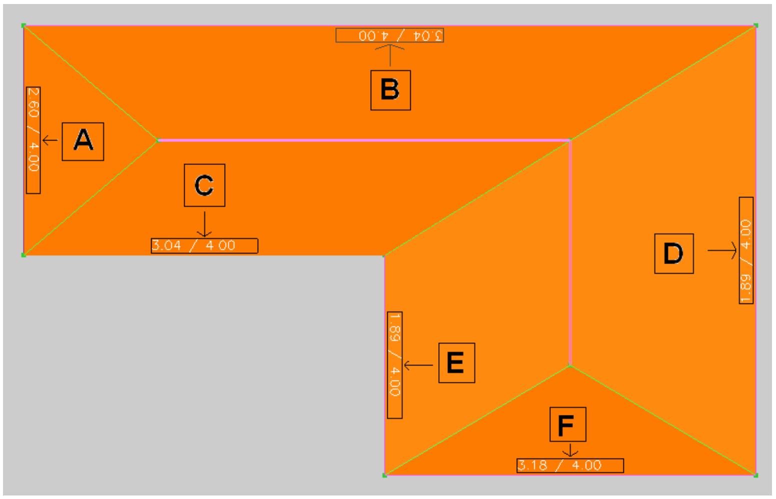

Figure 6: Starting condition (Example 2)

After Optimization:

Panel	Slope (in 12)
A	4.00
B	4.00
C	4.00
D	2.59
E	2.38
F	4.00

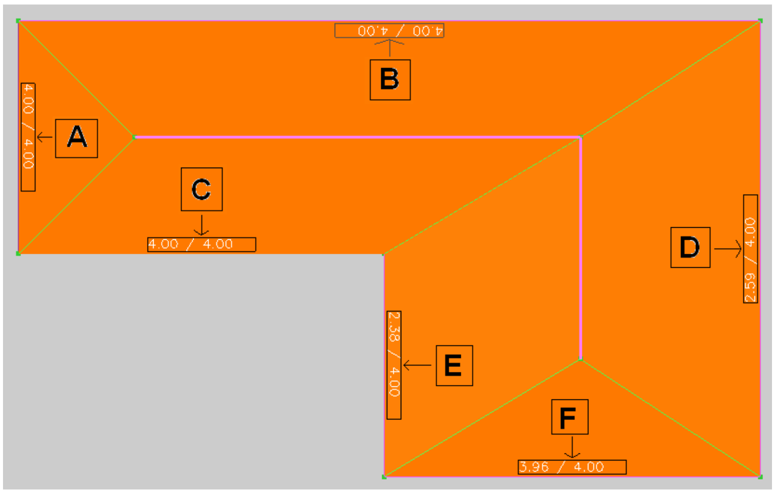

Figure 7: After optimization (Example 2)

Here, an ‘L’ shaped building with both sides having different width and equal desired slope on all panels is considered. However, there is no possible solution which can achieve the desired result without discontinuity in the Ridge line. Results from optimization shows that panels A, B, C and F have the desired slope of 4 in 12, while panel D and E are set to other slopes to achieve the closest configuration with the desired values.

Example 3: A ‘U’ shaped building with desired slope of 4 in 12 on all roof panels.

Starting Condition:

Panel	Slope (in 12)
A	3.31
B	2.94
C	3.31
D	3.06
E	3.14
F	2.11
G	2.11
H	2.94

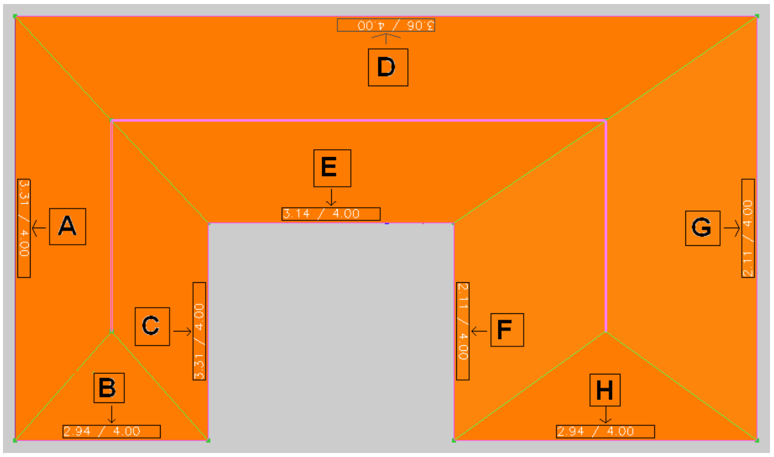

Figure 8: Starting condition (Example 3)

After optimization:

Panel	Slope (in 12)
A	4.28
B	4.00
C	4.28
D	4.00
E	4.00
F	2.72
G	2.72
H	4.00

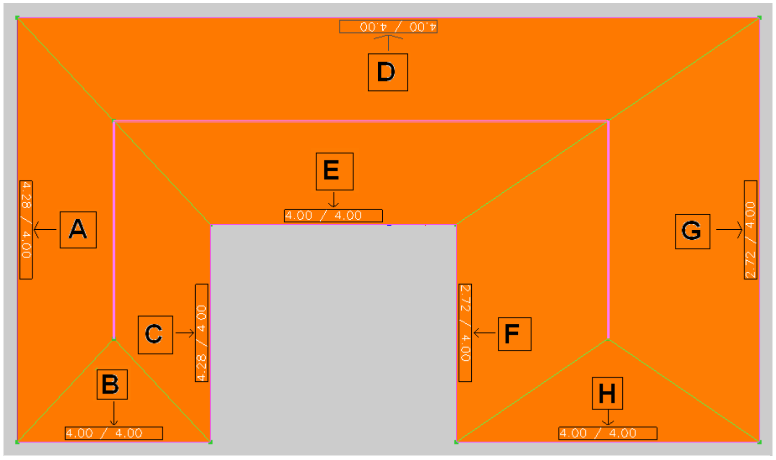

Figure 9: After optimization (Example 3)

It can be seen from the result that as the geometry becomes more complicated, the number of panels which achieve the desired slope decreases.

Performance Variables

Performance variables are functions of design variables that are used as a measure of performance and contribute to the objective function. Consider the case shown above, of an ‘L’-shaped building with both sides having different widths. Suppose that all the panels have the same desired slope. In such a case, the optimization will generate panels on one side with the desired slope while the others will have a modified slope. But, there is no control on which panel should get the desired slope. In the above case, panels B and C were had the desired slope and panels D and E had the modified slope. If performance of the result generated by optimization is rated, then the problem can be solved. In other words, if the difference between the modified slope and the desired slope of one panel has more importance than another in the objective function, then optimization will be driven towards one of the minima. This will reduce the change in slope of a panel with high importance.

Performance can be rated by any number of discrete values. Three levels of performance i.e. Low, Medium, and High are being used in the application. This means, optimization will try to generate the closest match in slope of a panel with a high performance rating. This is achieved by changing the slope of the panels with Low performance ratings. An example is shown below.

Example 4: An ‘L’ shaped building with desired slope of 4 in 12 on all roof panels.

Starting Condition:

Panel	Slope (in 12)	Importance
A	3.06	Mid
B	3.20	High
C	3.20	High
D	1.87	Low
E	1.87	Low
F	3.35	Mid

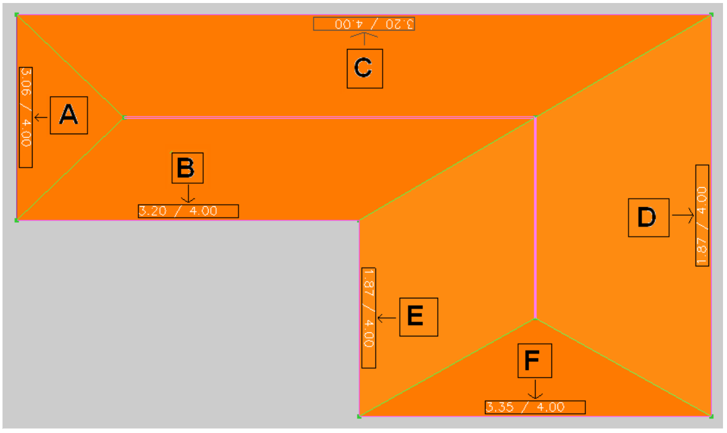

Figure 10: Starting condition (Example 4)

After optimization:

Panel	Slope (in 12)	Importance
A	4.00	Mid
B	4.00	High
C	4.00	High
D	2.46	Low
E	2.21	Low
F	4.00	Mid

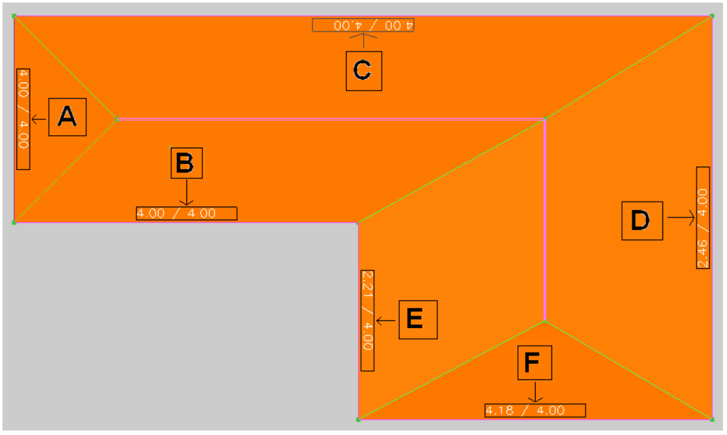

Figure 11: After optimization (Example 4)

Here it can be seen that optimization has generated the desired slope on panel B and C because of their high importance, by changing the slope of panel D & E. Running the optimization again by setting High importance on panel D & E and setting Low importance on panel B & C should generate different results.

Example 5: An ‘L’ shaped building with desired slope of 4 in 12 on all roof panels.

Starting Condition:

Panel	Slope (in 12)	Importance
A	3.06	Mid
B	3.20	Low
C	3.20	Low
D	1.87	High
E	1.87	High
F	3.35	Mid

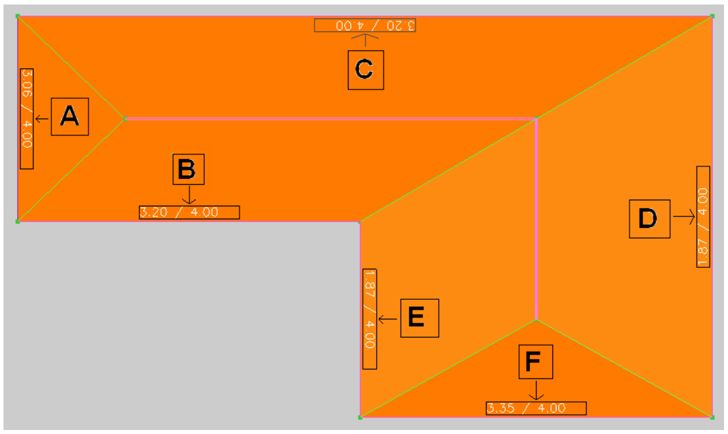

Figure 12: Starting condition (Example 5)

After optimization:

Panel	Slope (in 12)	Importance
A	4.00	Mid
B	7.02	Low
C	6.72	Low
D	4.00	High
E	4.00	High
F	4.00	Mid

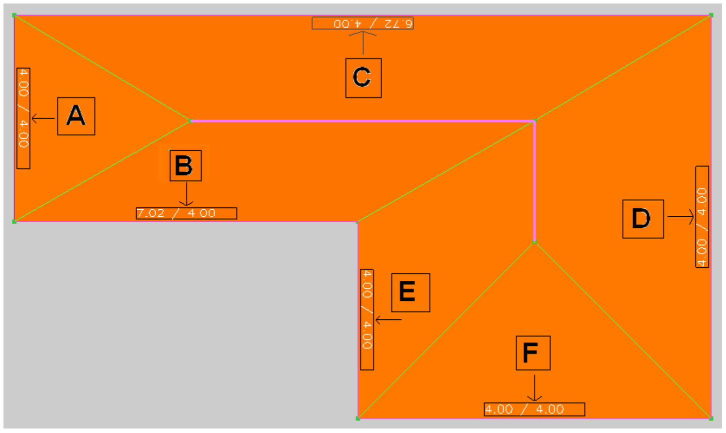

Figure 13: After optimization (Example 5)

The results show that optimization has generated panel D and E with the desired slope, by changing the slope of panels B & C. Thus optimization is trying to match the slope of the panels with high performance importance. The result of optimization can be controlled in a better way by using performance variables.

Using constraints

Example 4 and 5 explain the use of performance variables. In example 4, optimization generated the desired slopes on the panel B and C by changing the slopes of the panel D and E. But, these modified slopes of the panel D and E are not equal. One of the ways to get equal slope on both the panels is to associate a constraint to the Ridge line. Constraints will be discussed later in more detail in chapter 7.

As explained earlier, optimization changes the x, y, and z coordinates of the end points of a Ridge line to generate the desired roof profile. If x location of the Ridge line between the panel D and E is fixed then optimization will generate equal slopes on the panel D and E, which can be seen in example 6 below.

Example 6: An 'L' shaped building with desired slope of 4 in 12 on all roof panels.

Starting Condition:

Panel	Slope (in 12)	Importance
A	2.79	Mid
B	2.82	Low
C	2.82	Low
D	1.87	High
E	1.87	High
F	2.93	Mid

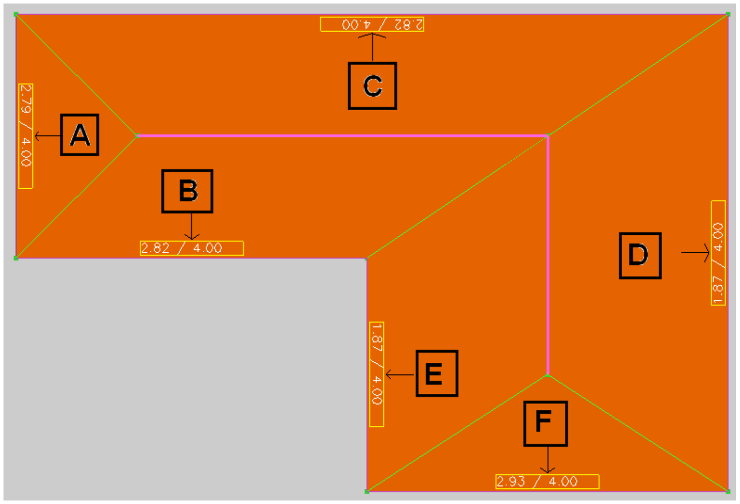

Figure 14: Starting condition (Example 6)

After optimization:

Panel	Slope (in 12)	Importance
A	4.00	Mid
B	6.01	Low
C	6.01	Low
D	4.00	High
E	4.00	High
F	4.00	Mid

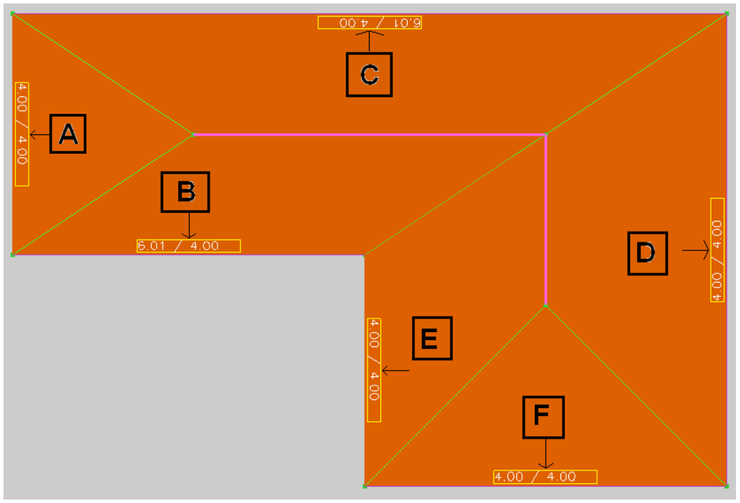

Figure 15: After optimization (Example 6)

The results show that optimization has generated the desired slopes on the panel D and E, by changing the slope of the panels B & C but keeping them equal.

Joint lines

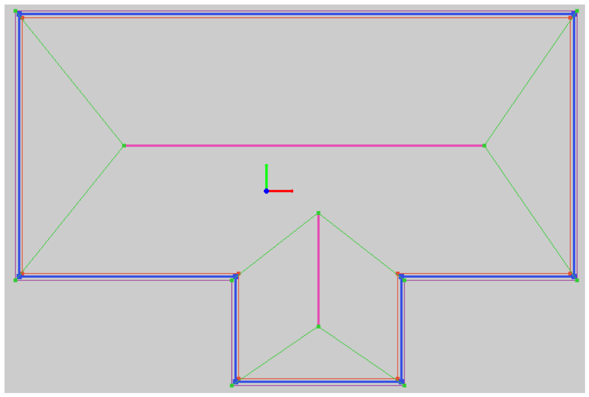
Figure 16: Plan view of 'T' shape building

Fig 16 shows a plan view of a T – shape building. If we see the same building in side view in fig. 17, then we can notice that the Ridge line between the panel E and F is not connected to the panel D. The same condition may result from the optimization also. A condition is required which will place one of the end points of Ridge line on the Panel D by maintaining the flatness of panels.

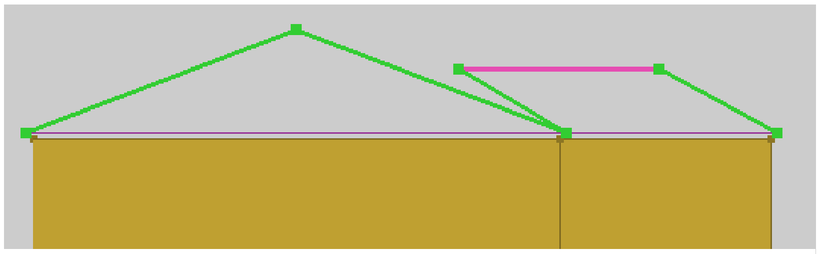
Figure 17: Side view of 'T' shape building (Without Joint line)

The problem is solved by adding the concept of Joint lines. If any two panels are sharing a Joint line as one their boundary then optimization will generate equal slope on those panels. Thus, Joint line will assure the continuity in panels and also keeping them flat. The result of the optimization after adding Joint lines is shown in Fig 18 below.

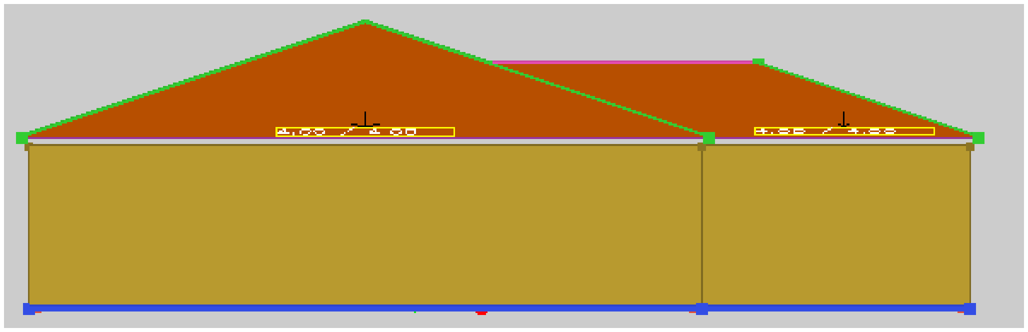
Figure 18: Side view of 'T' shape building (With Joint line)

CHAPTER IV

SIZING OF TRUSSES

Sizing of the trusses is done to determine the optimum cross section that will survive under all different load conditions. There are various types of loads which are applied on the roof panels and there are many ways to transfer these loads from the panel to the trusses. There are three main phases in the process of sizing trusses. They are:

1. Calculate different loads on a panel
2. Transfer loads from roof panel to trusses
3. Size truss members which can carry these loads

Each phase is discussed below.

Calculation of the loads on the panel

There are many types of loads which are applied on roof panels. However, only the three most significant loads; dead load, live load, and wind load are considered in this project. All other loads will be considered as special loads which explicitly need to be provided by the user. Calculation of all three types of loads is described in "The Analysis of Cold-Formed Steel Roof Trusses" by Matt C. Ritter[1]. Thus, all three types of loads on any panel can be calculated, with all other loads considered as special load. This adds an ability to perform calculations of different combinations of these four types of loads.

However, one important issue to be considered is in the calculation of wind load. When the wind is blowing on roof panels, it applies pushing pressure on some part of the panels and applies pulling pressure on the other parts. Thus, we need to consider two different values for wind loads, pulling and pushing. So theoretically, two different loads need to be considered for wind loads.

Transferring the load from panel to the trusses:

The method to calculate different loads on a panel was described in the previous section.

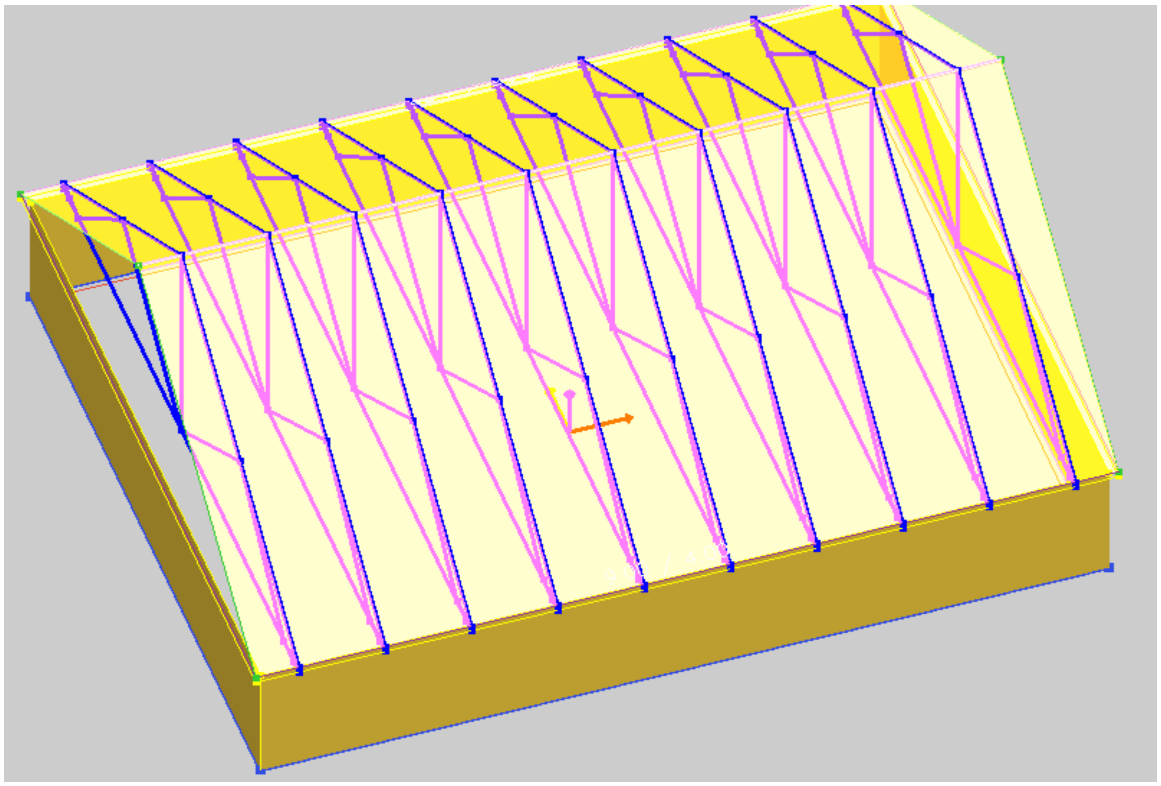

Figure 19: A simple building with trusses running parallel to each other and having equal spacing

Next, what fraction of the loads on a panel is being supported by which particular truss must be determined. Consider the simplest case, a four sided building with two roof panels making a gable on each end. Assume that the trusses are running perpendicular to the panels with equal spacing. Also, the distance between the end trusses and the end of the panel is half the spacing between the trusses.

In this case, the panel loads are equally supported by all of the trusses. Therefore the calculation of each sub-panel, which transfers all its loads into one particular truss, is very simple. Panels need to be divided at the mid-line of each truss line to calculate sub-panels.

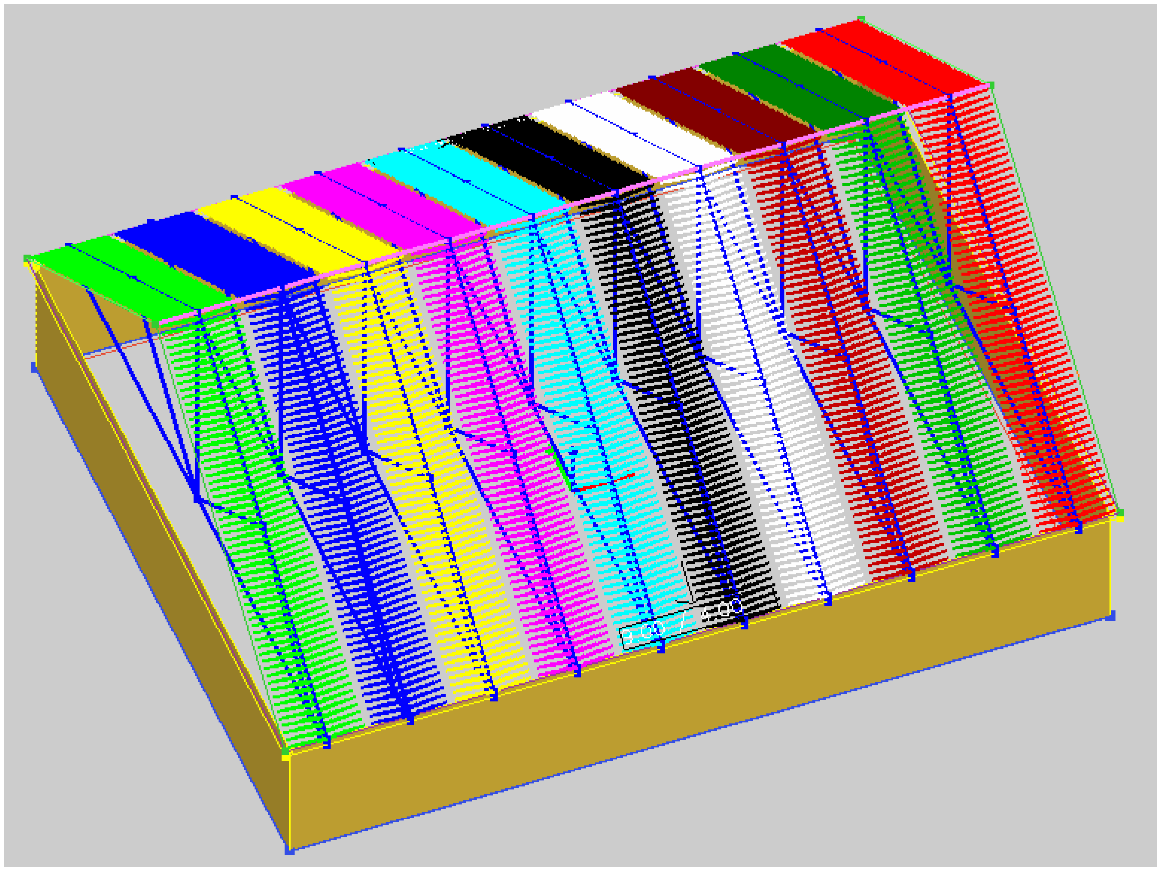

Figure 20: Truss regions of all the trusses on a simple building

The solution looks good for simple cases like the one discussed above, in which each truss calculates the geometry of its panel area or sub-panel from which it will get its loads. But the process becomes very complicated when spacing between trusses is not equal and trusses do not run parallel to each other. The problem becomes even more complicated in the case of girder trusses (when trusses cross each other). Such a case is shown below.

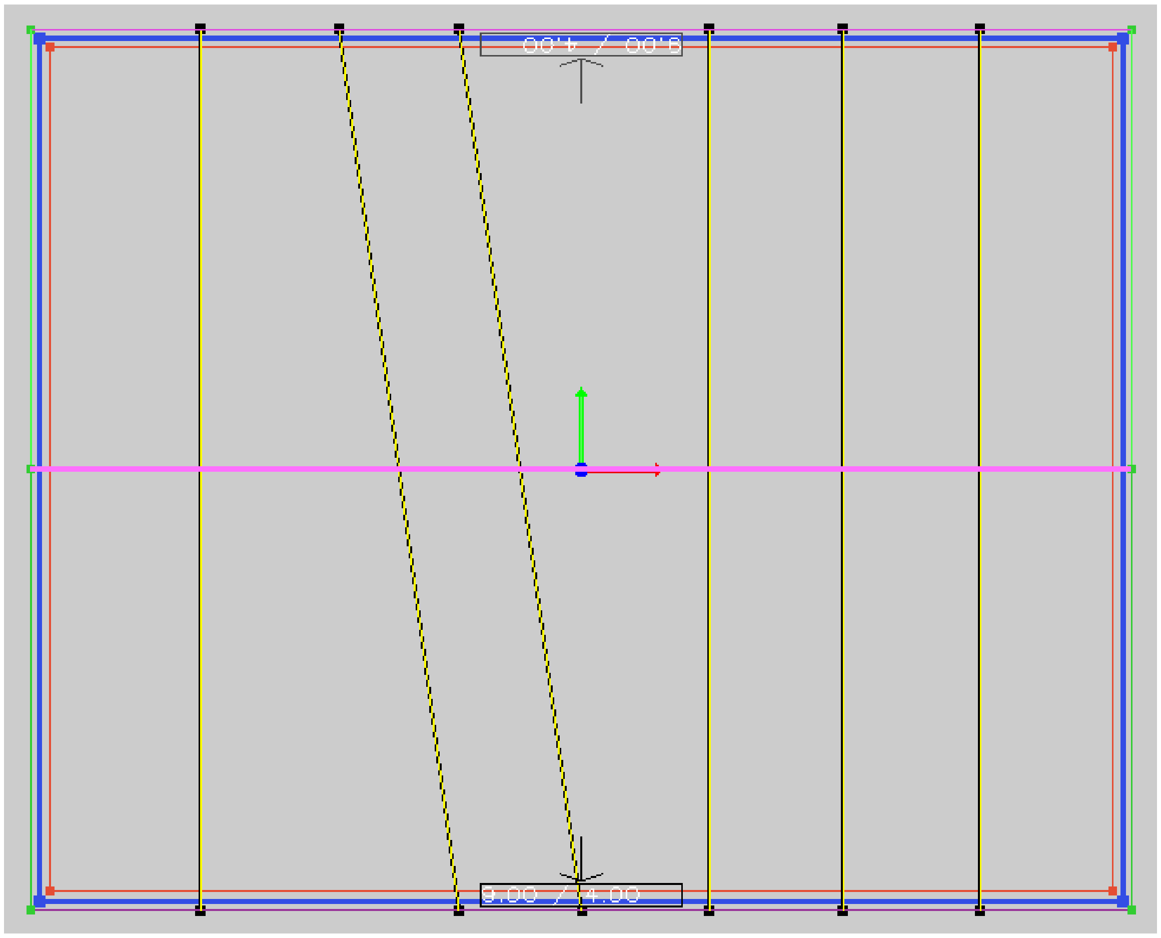

Figure 21: Non-parallel truss arrangement

However, rather than using a segregation approach to calculate the geometry of a subpanel, an accumulation approach can be used to determine the same thing. A panel is first divided into a large number of micro-panels. The area of a micro-panel is so small that pressure variation on it is negligible and load applied on each micro-panel can be considered as a point load. Each micro-panel is associated to its nearest truss in the system and all micro-panels associated with a particular truss will cumulatively form a sub-panel for that truss. Results generated with the above method are shown in the figure.

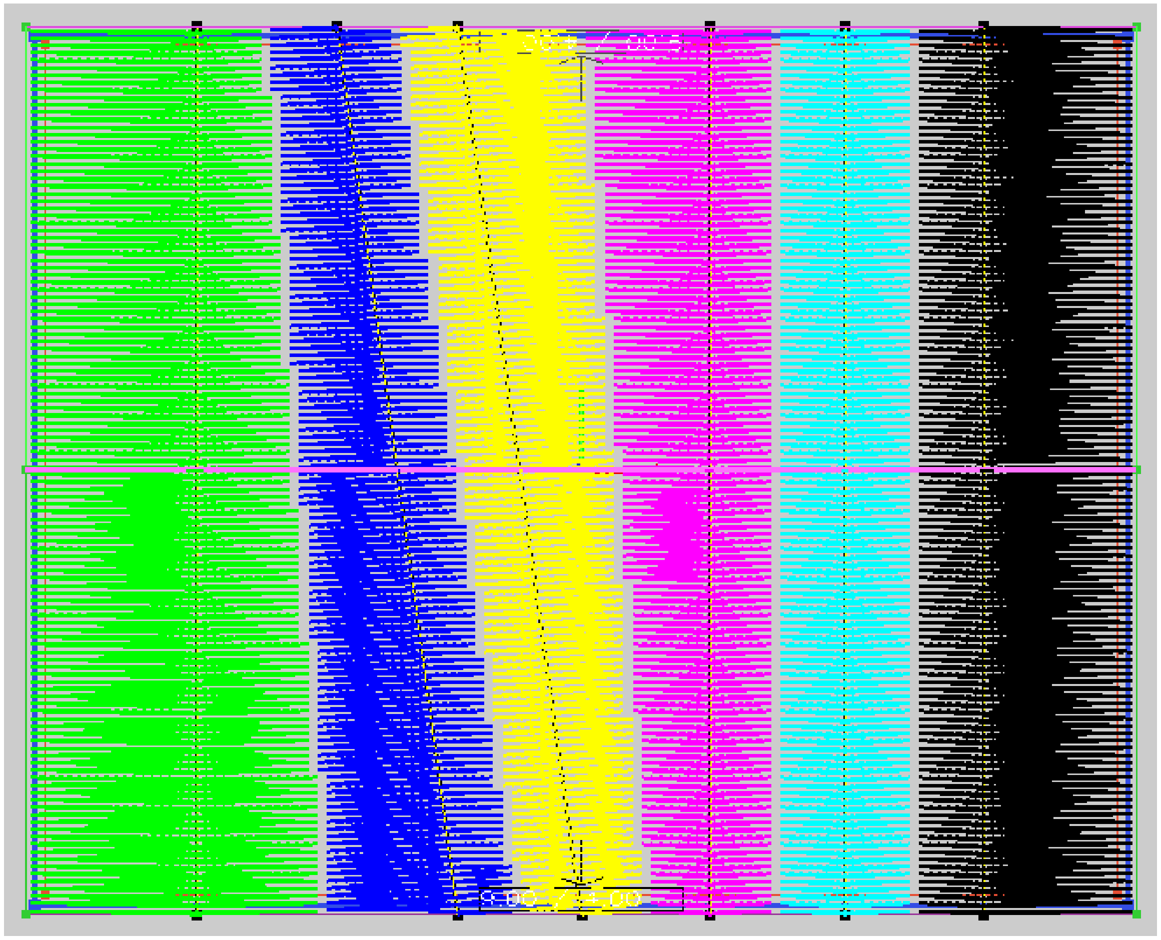

Figure 22: Truss regions in non-parallel truss arrangement

Consider an imaginary truss alignment as shown in the figure which is very complicated in nature. The method described above will handle the problem easily and will find the solution as shown below. However, sub-panels generated with curvilinear boundaries will be very difficult to calculate with the previous approach.

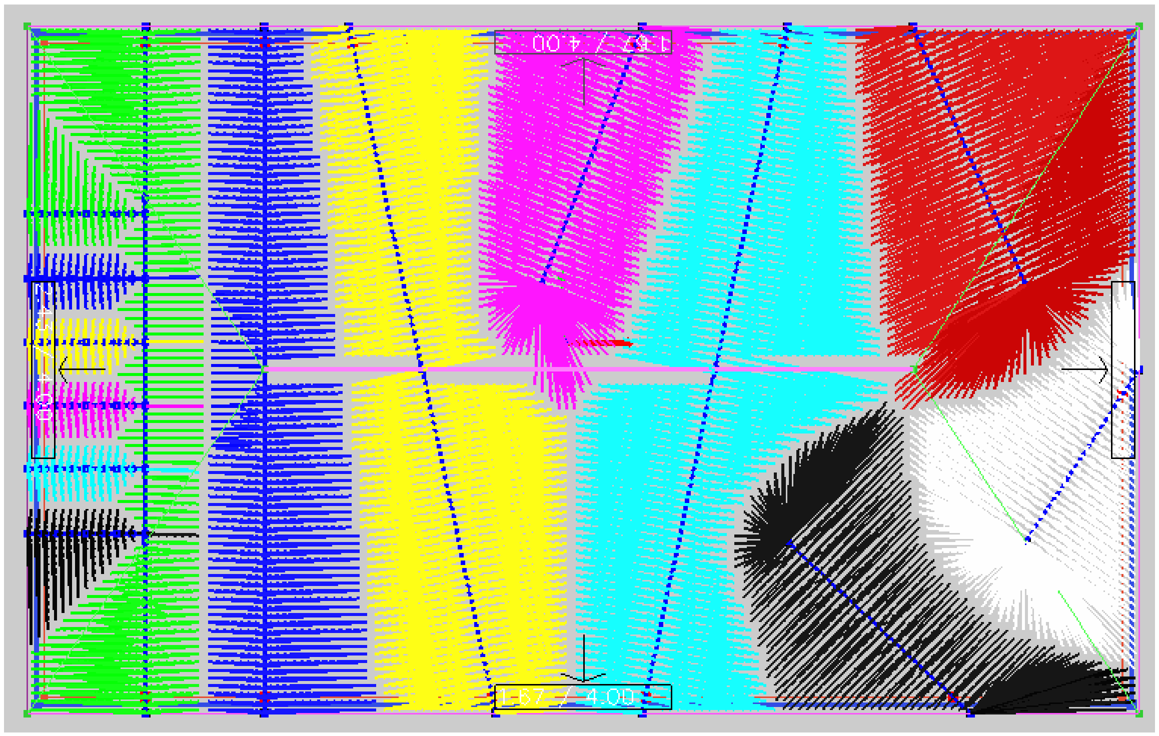

Figure 23: Truss regions of an imaginary truss arrangement in a simple square building

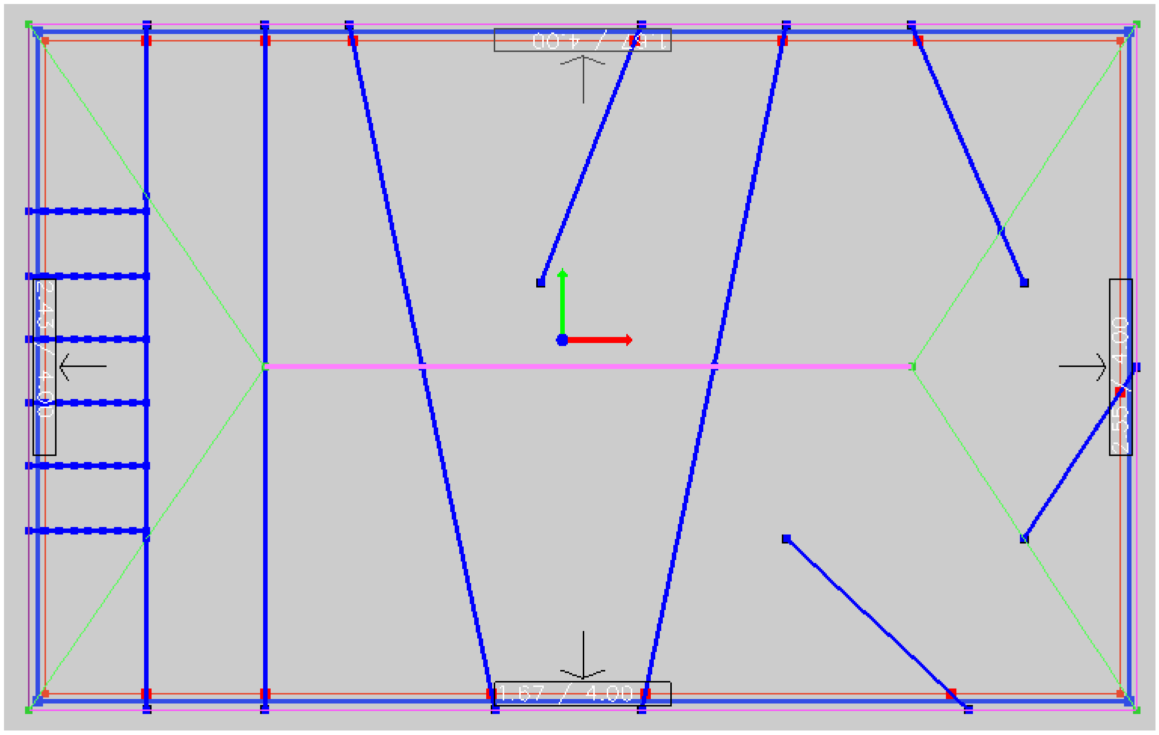

Figure 24: Imaginary truss arrangement in a simple square building

Size Truss Members:

Once the loads applied on each truss are known, a programmable method is required to calculate forces acting on each member. Trusses can then be sized to get optimized results. In programming language, dealing with matrices is easier than dealing with individual equations. [6]There are two methods to analyze trusses using a matrix.

a. Stiffness matrix method
b. Flexibility method

In the flexibility method, two separate procedures are used, on for statically determinate trusses, the other for indeterminate trusses. In the stiffness matrix method, a single procedure can be followed to handle both kinds of trusses. Moreover, the stiffness matrix method directly yields displacement and forces in the members. Thus, the stiffness matrix method is better to use this analysis process.

In the stiffness matrix method, a truss is treated as a collection of discrete finite elements called a member, which are connected at joint points called nodes. The forces and displacement properties of each member are analyzed and the global stiffness matrix 'K' of a truss is calculated. The procedure to generate the global stiffness matrix is explained in [6] "Structural Analysis" by R. C. Hibbeler. Once the global stiffness matrix is calculated, the unknown displacement of nodes can be determined. Using the force-displacement properties of each member, the internal forces can be calculated for each member. However, for determining forces in truss members, cross section properties are needed to carry out the calculation. This puts the process into iteration.

The application is using a pre-defined library of cross sections. Cross sections are sorted based on their area. The reason behind sorting them by cross sectional area is that the area is directly proportional to its weight which is closely related to its cost. The process starts with the weakest cross section in the library and determines the forces in the members. If the cross section can sustain under those forces, then that cross section can be used. Otherwise, the process selects the next cross section from the library and re-determines the forces. The loop continues until the application finds the lightest cross section which will carry the applied loads.

A method to decide, weather a truss member will carry the applied load or not is described in [1] "The Analysis of Cold-Formed Steel Roof Trusses" by Matt Ritter which completes the whole process to find out the optimum size of a truss member. Below is an example for sizing a particular truss under a specific loading condition.

Truss Geometry and loads

Sample truss geometry is given below with loads applied on different nodes as shown in the fig below. Forces and moments in each member are calculated and are given below. These data are used to determine the smallest cross section of members which will carry these loads. The results are shown below.

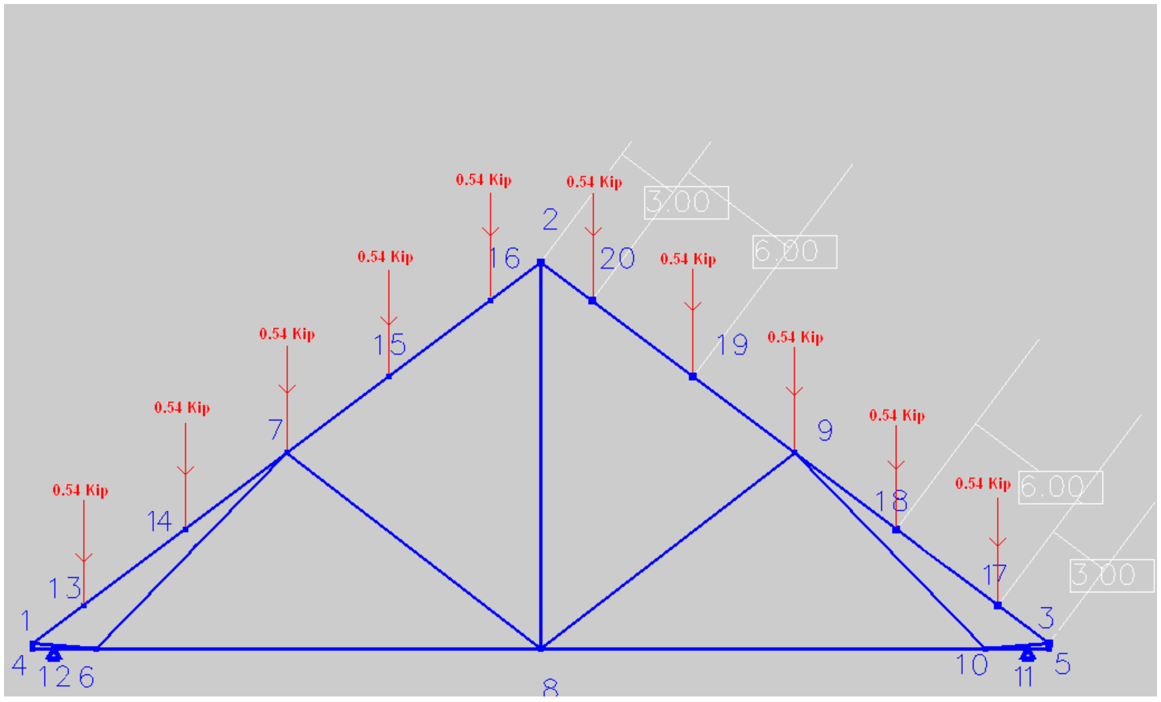

Figure 25: Schematic diagram of truss with loads applied

Resultant forces and moments

Near	Far	A	I	L	NX	NY	NZ	FX	FY	FZ
		in * in	in ^ 4	in	Kip	Kip	Kip	Kip	Kip	Kip
1	13	0.7215	2.8791	36.0000	-0.5684	0.0947	-0.0001	0.5684	-0.0947	15.0032
3	20	0.7215	2.8791	36.0001	1.8169	-0.7364	15.1725	-1.8169	0.7364	2.8649
5	6	0.8356	4.7280	12.0000	-0.0001	0.0000	-0.0003	0.0001	0.0000	-0.0003
7	8	0.8356	4.7280	252.0000	-2.2629	-0.3832	-64.8003	2.2629	0.3832	-31.7696
7	1	0.8356	0.0000	36.4966	-0.5684	0.0947	0.0000	0.5684	-0.0947	0.0000
2	15	0.7215	2.8791	72.0000	1.8168	1.8163	18.2999	-1.8168	-1.8163	7.8334
7	2	0.8356	0.0000	157.0350	2.8312	2.9885	0.0000	-2.8312	-2.9885	0.0000
8	9	0.8356	4.7280	252.0000	-2.2629	0.3832	31.7696	2.2629	-0.3832	64.8007
8	2	0.8356	0.0000	183.6628	-0.4461	0.3531	0.0000	0.4461	-0.3531	0.0000
8	3	0.8356	0.0000	222.0000	0.0000	-1.4727	0.0000	0.0000	1.4727	0.0000
4	18	0.7215	2.8791	72.0000	-0.5683	0.9853	18.3006	0.5683	-0.9853	13.9044
8	4	0.8356	0.0000	183.6628	0.4461	0.3531	0.0000	-0.4461	-0.3531	0.0000
9	11	0.8356	4.7280	24.0000	0.0000	-2.7000	-64.8007	0.0000	2.7000	0.0004
4	9	0.8356	0.0000	157.0350	2.8312	-2.9885	0.0000	-2.8312	2.9885	0.0000
9	10	0.8356	0.0000	36.4966	0.5684	0.0947	0.0000	-0.5684	-0.0947	0.0000
11	12	0.8356	4.7280	12.0000	0.0000	0.0000	-0.0003	0.0000	0.0000	-0.0001
6	7	0.8356	4.7280	24.0000	0.0000	2.7000	-0.0003	0.0000	-2.7000	64.8003
13	14	0.7215	2.8791	72.0000	-0.5683	-0.4453	-15.0034	0.5683	0.4453	13.9042
14	2	0.7215	2.8791	72.0000	-0.5683	-0.9853	-13.9042	0.5683	0.9853	-18.2999
15	16	0.7215	2.8791	72.0000	1.8169	1.2764	-7.8334	-1.8169	-1.2764	2.8648
16	3	0.7215	2.8791	36.0001	1.8168	0.7363	-2.8647	-1.8168	-0.7363	-15.1726
17	10	0.7215	2.8791	36.0000	-0.5683	-0.0947	-15.0035	0.5683	0.0947	0.0001
18	17	0.7215	2.8791	72.0000	-0.5683	0.4453	-13.9044	0.5683	-0.4453	15.0037
19	4	0.7215	2.8791	72.0000	1.8168	-1.8163	-7.8339	-1.8168	1.8163	-18.3005
20	19	0.7215	2.8791	72.0000	1.8168	-1.2763	-2.8649	-1.8168	1.2763	7.8339

Cross section dimensions

Fig below shows the 'C' shaped cross sections of members. The three sizes for top, bottom, and web members are listed below.

Singly Symmetric Channel Section (Reference 2002 AISI Manual Section 3.3.2

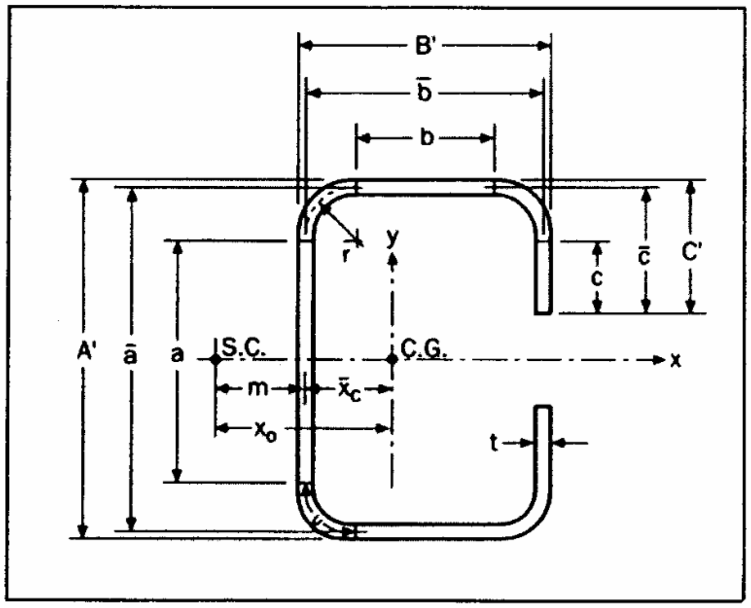

Fig 26: Properties of a 'C' shaped cross section

Top Members		
A'	5	in
B'	2.3	in
C'	0.525	in
t	0.0713	in
R	0.107	in
alpha	1	
Fy	50	ksi
Fu	40	ksi
mu	0.3	

Bottom Members		
A'	6	in
B'	2.5	in
C'	0.625	in
t	0.0713	in
R	0.107	in
alpha	1	
Fy	70	ksi
Fu	60	ksi
mu	0.3	

Web Members		
A'	6	in
B'	2.5	in
C'	0.625	in
t	0.0713	in
R	0.107	in
alpha	1	
Fy	70	ksi
Fu	60	ksi
mu	0.3	

CHAPTER V

DATA STRUCTURE

The central idea of any CAD application is to display three dimensional data and provide an easy way to manipulate that three dimensional data. However, efficiency of demonstration and manipulation of 3D data is heavily dependent on the model for storing the data. This section will focus on structures and techniques that are used internally to store and manipulate organized units of information.

In any CAD application, the basic topological items are vertices, edges and faces. Now a model is considered to store these items and relate them internally.

In any programming language, arrays are the basic structure to store multiple items of the same kind. However, they do not provide dynamic memory allocation, which is a significant requirement of any application. In this case, Link List is the best option, which provides an ability to store multiple objects with dynamic memory allocation. Moreover, Link List is supported in many programming languages. The next task is to find a model to relate the data internally.

There are three basic categories of models to represent the relationships between vertices, edges and polygons [13].

1. Vertex Centric Relationship model
2. Edge Centric Relationship model

3. Polygon Centric Relationship model

Vertex Centric Relationship model

In this type of model, vertices are stored in a list and all polygons and all edges which use these vertices will point to specific vertices in a list. However, this is a very inefficient way of storing the relationships because the relationships between edges and polygons are not directly provided. The model is calculation intensive in manipulating edges and polygons.

Edge Centric Relationship model

In this type of model, edges are stored in a list and each edge, points to two vertices and two polygons. The two vertices being pointed to by an edge are the two end points of that edge. The polygons being pointed to by an edge are the two polygons which include that edge as their boundaries. One of the oldest data structures for this type of model is the [2]*Winged Edge* data structure.

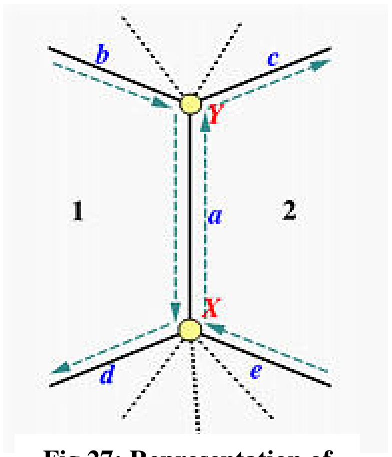

Fig 27: Representation of the winged edge data structure

The *Winged Edge* data structure is the most sophisticated type of Boundary Representation (B-Rep) model. As shown in the figure, each edge in the data structure points to:

a. Two vertices, which are the end points of that edge

b. Two polygons, which use that edge as one of their boundary lines.

c. Four edges which emanates from the end points and are associated with the two polygons pointed to.

The data structure for Winged Edge is shown above. There are two more types of B-Rep model.

a. Quad Edge Data structure
b. Half Edge Data structure

Both of them are a little different from the winged edge data structure. But, the core idea is the same. The method of storing the data structure discussed above makes them an excellent choice for many applications.

Polygon Centric Relationship model

The edge Centric Relationship model is sufficient for implementing a data structure, but it can be improved significantly by enhancing the data structure for polyhedral geometry. The model is designed in such a way that all properties of a polygon are directly accessible from the data structure. Polygons are stored in a linked List. This linkage is purely for an ease in implementing the algorithm that requires frequent computation related to all polygons.

<u>Polygon Structure</u>

Each polygon in a Link List will point to:

d. A Link List of pointers to an edge which form a boundary for that polygon. These edges are organized in a specific order to give outward normal of a polygon. Moreover, the specific order is important to calculate the loops involved in that polygon.

e. A Link List of pointers to vertices which lie on the boundary of a polygon. These lists of vertices can be accessed by an edge list. But a pre-formed list will increase the speed of computation. These vertices are organized in an order to give the outward normal of a polygon.

f. Extra pre-calculated information can be stored such as the normal vector, plane vector or area.

<u>Edge Structure</u>

Each edge in a Link List will point to:

a. Two vertices which are end points of an edge
b. Two polygons which contain the edge
c. Extra pre-calculated information can be stored such as the length or line vector.

<u>Vertex Structure</u>

Each vertex in a Link List will point to:

a. A Link List of pointers to polygons which contain that vertex
b. A link List of pointers to edges which contain that vertex
c. It also stores the x, y, z location

Advantages

1. The model contains heavy use of pointers which removes the problem of duplicate data in memory. The solution not only saves memory, but also reduces the complexity of updating that duplicate data.
2. When a value of any one vertex changes, updates in properties of only those edges and polygons which contains that vertex can be done. Thus, it saves time in calculating property of all other edges and polygons which are not affected by that vertex.
3. The model is not specific to any programming language.
4. When drawing the whole geometry, a list of edges can be drawn rather than drawing a list of polygons. This saves time in overdrawing edges which are being shared by two polygons.

Data structure for the Truss

Any truss is a collection of members which are connected at joints. These joints are called nodes. Thus, any truss is a collection of nodes and members, and analysis of a truss is an analysis of these nodes and members only. A data structure of a truss should have an ability to store the information of its members and nodes and also express the relationships between them.

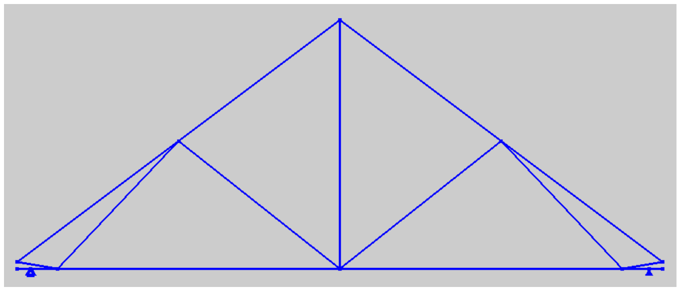
Figure 28: Schematic diagram of a truss

A schematic diagram of a sample truss shown in the figure shows the fact that a truss can be visualized as a collection of edges and points, where each edge represents a member of a truss and the end point of an edge represents a node in a truss. Thus, the same polyhedral data structure can be used for a truss with little modification. Moreover, members and nodes are not shared across different trusses. The data structures for nodes, members and truss are as described below.

<u>Node Data structure</u>

a. It contains x and y coordinates from a reference point in the truss
b. It points to a Link List of pointers to members which contain that node.

<u>Member Data structure</u>

a. It contains pointers to two nodes: Near node and Far node.
b. It contains some extra information like length, cross section, weight etc.

A polyhedral data structure is a very good model to store information for any CAD application. Moreover, the model is easy to implement, supports faster computation and is less error prone. Also the same model can be used for both topological and truss geometry.

CHAPTER VI

TEXT IN 3D

All engineering drawings will result in a meaningless collection of lines and curves if it does not include associated data with it like dimensions, labels, and other information. The same thing holds true if such drawings are being rendered on a computer screen. The information associated with each element in CAD geometry needs to be displayed on screen. Generally this information will be displayed in the form of some text in the drawing. This scenario leads to a requirement for the ability to render text in a three dimensional drawing. The problem might look simple, but there are many concerns associated with rendering text in a three dimensional drawing. This adds significant complexity to the problem. The basis of computer fonts is discussed in the next section to find the solution.

In typography, the shape of each symbol is known as a *glyph*[12]. In this case, a symbol means character set A – Z, a – z, 0 – 9 and other symbols like comma, colons, punctuation etc. A method is required to display these set of glyphs on a computer screen. A data file font is nothing but an electronic version of such methods. There are three basic formats of font files.

- Bitmap fonts
- Outline fonts
- Stroke based fonts

The advantages and disadvantages of all three types of fonts are considered below.

Bitmap fonts

In this type of format, each glyph is stored in an array of pixels (i.e. bitmap). For this reason sometimes they are called raster fonts. It stores each glyph in form of small images, which makes it extremely fast in rendering. But there are many major disadvantages to this style. One of the most important is that they are not scalable, which means different sets of small images are required for each different size of glyph. One more problem in 3D rendering is that these fonts cannot be rotated in 3D space. Thus, it is not appropriate to use bitmap fonts in the application.

Outline fonts

In this format, the outline of a font is defined as set of lines and curves (Bezier curve). These fonts are heavily used in vector monitors and vector plotters. The main advantage of these fonts is that it allows an infinite level of scaling in a single definition. Formats like *PostScript* and *OpenType* are widely used in drawing applications like adobe and others.

However, the accuracy of curves on a raster display is not very good and it requires considerable processing power. This disadvantage becomes significant when used in a real time application like CAD. Even though, major CAD applications like AutoCAD and

SolidWorks use this type of format, called TrueType, SHX; the main reason for the selection is to provide an additional functionality and not to provide a core requirement.

Stroke based fonts

In this type of format, fonts are defined by strokes of the pen. Each stroke is a straight line between two vertices and the stroke profile defines the complete glyph. It eliminates the use of curves which was a major drawback in outline fonts. Moreover, an infinite level of scaling is still available, rotation in 3D is possible and the requirement of processing power is very moderate. Thus, this type of method is most useful in real time application like CAD, because drawing a series of lines in a CAD application is highly optimized. A particular type of stroke based font known as *Hershey fonts* is used in the present application.

Hershey fonts [4]

The "Hershey fonts" were developed in 1960s by Dr. A. V. Hershey at the U. S. Naval Weapon Laboratory. There were 1377 glyphs in the original version. Each glyph was assigned by a number in the range 1 – 3926, all of which are not used. The files defining those glyphs are freely available on the internet. Due to this availability, the format of the file has changed over time and many forms are available today. In all types of the forms, a file will contain a series of the definition of glyphs. The particular format which is being using is described below. The file can be referred to in the appendix.

- A line starts with a definition of each glyph. The first five spaces contain a number associated with a glyph.
- Next three spaces contain a number, which represents the number of coordinate pairs involved in the definition.
- The following part of the string represents a series of alphabetical characters. Each represents a value relative to an ASCII value of 'R'.
    - The first pair of characters in the remaining part of the string indicates the most left and the most right coordinate of the glyph.
    - Successive pair of characters represents the (x, y) coordinates of the vertices.
    - The pen up option is indicated by "R".

A sample example of a glyph is shown below.

"   8 9MWOMOV RUMUV ROQUQ"

- The first five spaces give "   8" which is associated with the number of that glyph.
- Next three spaces give "9" which tells the number of coordinate pairs, which means there will be 9 pairs of coordinates involved in the definition.
- The first pair gives the most left and most right value.

  ASCII value of 'M' – ASCII value of 'R' = 77 – 82 = -5

  ASCII value of 'W' – ASCII value of 'R' = 87 – 82 = 5

  Thus the width of the glyph is 10 units.

- The next pair gives coordinates of the first vertices.

  ASCII value of ‘O’ – ascii value of ‘R’ = 79 – 82 = -3

  ASCII value of ‘M’ – ASCII value of ‘R’ = 77 – 82 = -5

  Thus the coordinates are (-3, -5).

- The next pair “OV” represents coordinate (-3, 4).
- The successive pair is “R” which means pen needs to raise up.
- Each consecutive pair of characters represents coordinates (3, -5), (3, 4), pen up, (-3, -1), (3, 1), respectively.
- This will draw three lines in such a way that it will look like a character ‘H’ on screen.

Thus, all glyphs can be rendered by a series of lines. Moreover, different characters can be rendered side by side to render a string. As the width of each character is known, rendering characters with varying widths will not be a problem. A sample string rendered in the application is displayed here.

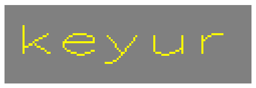

Figure 29: Sample text rendering

Hershey Font is very easy to implement, fast enough to render, and allows rotation and scaling in 3D. This makes it very valuable to use in this CAD application.

CHAPTER VII

CONSTRAINTS

The main motivation behind the creation of a CAD application was to provide a tool to generate computer based diagram to replace the paper based diagrams. Therefore, a tool should provide at least those features which are available for paper based drawings. When an engineer draws a straight horizontal line on paper, he uses a ruler or a drafter for that. But, these tools cannot be used to draw a line on a computer screen. Moreover, some behaviors are expected by users that are not automatic for the computer. For example, most users expect that certain lines should remain horizontal even if either of its end points changes its position. There are many constrain like this, e.g. Vertical, Parallel, perpendicular, tangent, constant length. This scenario leads to the requirement of an ability to create and maintain such constrains. An ability to create such a constraint is done by including a snapping feature in the drawing application. Maintaining such constraints is generally done by geometric modeling, in which these constraints are represented by equations and by solving the set of equations, the constraints are maintained.

However, the model used in the application is not a geometric model and in such cases the method described above is very difficult to implement. This is outside the scope of this project. Another method for snapping and constraints has been used. This method is easy to implement and works best with the requirements of this project. There may be many disadvantages for this method and it is not claimed to be good for other applications.

Snapping

The algorithm is very basic in nature. When a user moves a point in a 3D OpenGL space, the application takes coordinates of that point and finds its distance from all possible snap locations, like snap to horizontal, snap to vertical, snap to mid-point of a line etc. If the nearest possible snap location is within the vicinity of a point then a point will be snapped to that location.

A significant factor in the algorithm is the definition of vicinity, which means the program must decide whether the snap location is near enough or not. The definition of the vicinity should be defined in terms of distance on screen and not a distance in OpenGL space.

Constraints

Constraints are nothing but conditions which must be satisfied even after changes in the drawing dimensions occur. There are two ways a user can change the drawing entity on screen.

- Dragging a point
- Dragging an edge

However, dragging of an edge can be seen as the simultaneous dragging of both end points equally. Thus, theoretically there is only one phenomenon which changes the dimension of a drawing, changing a position of a point. So, whenever a position of any point changes, it is required to make sure that all constraint conditions are satisfied.

There can be many types of conditions which can be applied to a diagram. In the beginning, efforts have been made to include many constraints in the program. However, considering the time line of the project, constraints which are not used very often in the program are omitted. Below is the list of constraints which are implemented in the program.

- Fix the 'X' dimension of a point
- Fix the 'Y' dimension of a point
- Fix the 'Z' dimension of a point
- Horizontal Edge
- Vertical Edge
- Fixed Length Edge

The first three conditions are associated with the point itself. When an attempt is made to change the position, the above conditions will be checked and if any of the x, y, or z coordinates are fixed then it will not allow any change in that coordinate.

Now consider a Horizontal Edge. If one of the end points of that edge is moved then another end point needs to be moved in such a way that it will fulfill the condition of horizontality. Thus, the only action required is to figure out which point needs to move and to where it needs to be moved. The same phenomenon applies to the other three conditions as well. Below is the algorithm to solve any condition.

1. If any point is attempted to be moved then first it will check weather any of its coordinate is fixed or not. If any coordinate is fixed then change in that coordinate will not be allowed and rest of coordinates will be changed.
2. The next step is to check all conditions applied to edges which are pointing to this point.

In a point data structure, there is a Link List of pointers to edges which points to that point, and the same Link List can be used to check conditions applied to those edges.

Special note

It may seem like the algorithm will break if one of the fixed points is moved in effect due to the movement of some other point, but that is not true. Suppose there are two points, P1 & P2, and the position of point P2 is fixed. Now, there is some condition that exists which relates point P1 & P2. So if the position of point P1 is changed, then condition will make an attempt to change the position of point P2. However, as point P2 is fixed, the only way to fulfill the condition is to change position of point P1 back to its original position. Thus, algorithm works well in such complex conditions. The algorithm also works even if a fixed point is encountered at any level in loop. It will revert back all its effects to fulfill all the conditions.

CHAPTER VIII

UNDO / REDO

In the early days, the undo/redo function was considered an additional function to enhance the performance of an application. But nowadays, it is a very basic and essential function for any application. Undo means to nullify an effect of an action which has just been performed. Redo means to replicate an effect of an action which has just been undone. Even though the function is very basic and very old, its method of implementation varies substantially for different applications. One of the main reasons is that, it is extremely influenced by the data structure. [7]The main idea behind the function is to store information, which is required to perform the Redo and Undo actions. This information is generally stored in either of the two ways described below.

1. Directly storing a state of an application after each action is performed
2. Information regarding each action which has been performed is stored in a sequence.

Each of the above method is explained below in detail.

Storing the state of an application

In this method the state of an application is pushed on the Undo stack after each action is performed. As sequences of actions are performed, the Undo stack grows. If at any point in time, the Undo operation is performed, the current state of an application will pushed on to redo stack and the last state of the application from the Undo stack will become the current state of an application. The same scenario will be reversed to perform the Redo operation. Thus, maintaining two lists of the states of the application, the Undo/Redo function can be implemented very easily. The ease in this method serves as a core reason for selecting this method.

However, there are some disadvantages involved in this method.

a. As the sequence of operation is performed, the list grows substantially, and if a state of an application requires a significant amount of memory, then the application may soon run out of memory.
b. This method is not applicable when storing a state of an application requires storing a pointer. If a state of an application has a pointer which points somewhere in memory, then that pointer will be of no use when a copy of a state has been pushed on the stack.

These two disadvantages are very significant and thus this method is not useful in a CAD application.

Storing information regarding each action

Whenever any action is performed, this method will store a sufficient amount of information about that action to re-perform it any time. There are two ways to perform an Undo/Redo operation in this method.

1. Information about each action is stored, which is capable of calculating its reversible action, which will nullify any effect of an original action.

A simple example is that if an original action creates some entity then its reversible action will delete that entity. Thus, whenever an Undo action is required after a series of actions then the application will perform the reversible action for the last action performed. For performing a Redo action, the application will replicate the last undone action from the information stored.

Many applications use this methodology, however sometimes calculating a reversible action becomes a highly complicated task. In such case the chance of errors is very high and many times the methodology described below is used.

2. Another way of performing an Undo operation from information stored is to take a fresh copy of an application and replicate all the actions performed from the information stored except the last one. The application will then be in a state as it was before the last action was performed. The Redo operation will be performed by simply replicating the last undone action from the information stored.

Advantages

- It eliminates the requirement of calculating a reversible action
- The methodology is easy to implement and less error prone.
- The same list of information can be used as a "file save", when file is reopened, all actions will be performed again to reach the latest stage.

Thus, this methodology is very useful in an application like CAD. Implementation of a methodology in the application is explained below.

Implementation of a methodology

As explained above, enough information needs to be stored about an action so that it can be replicated. There are two basic categories of actions which can be performed in the application

i. Mouse click on drawing

ii. General task actions

General task actions are all actions except mouse clicks on the drawing area. Storing of these actions is very easy, as the only information required to store is the occurrence of an event and user input parameters. Mouse click actions can be stored by using storing locations of mouse clicks whenever they are performed. The information stored is enough to replicate the mouse click actions. However, there are two options for storing a location of a mouse click i.e. screen coordinates or OpenGL coordinates. Screen coordinates relates to different OpenGL coordinates when the screen size is different or when the OpenGL mapping screen is different. Moreover, as entities are generated in OpenGL coordinates, it is better to store the location of a mouse click in OpenGL coordinates.

Problem

Consider the scenario when a user performs a series of actions and stores the file while working in a higher resolution. One of the actions is to select a point. It is discussed in the previous section that snapping is performed in screen size coordinates and a screen distance defines the vicinity of a point. So a difference in resolution may cause a difference in vicinity value which may produce different results of an action.

<u>Solution</u>

The easiest solution is to calculate the ratio of screen distance to the height of the screen and use that ratio as a mean of defining the vicinity distance.

CHAPTER IX

CONCLUSION AND RECOMMENDATION

Conclusion

The project is still under progress, however all major landmarks have been achieved.
The computer program lets user to generate a building in a 3D application. Snapping functionality and constraint feature is working correctly in the program. Once the user has defined that building geometry, the application allows the user to define the location of trusses in the plan view. The user can also define group of trusses by defining a region and spacing between those trusses. The application will calculate the truss envelope based on the roof profile defined. The user can define a custom truss or a pre-defined truss can be added from the available library. The standard truss from the library has an ability to fit itself to fill the truss envelope.

Once the trusses are defined, the program will generate the loads on roof panel. These loads on the panel are then transferred to trusses in an appropriate manner. Once the loads on each truss are known, the program calculates the internal forces and moments. Based on these forces and moments, the program will determine the smallest cross section available in the library which will carry these loads.

The results generated by the program are validated and a good match is found in the results. Another major feature still under development is to find out the optimum number of panels in a particular truss. The program will calculate the optimum number of panels required in any given truss to reduce the cost of that truss.

Recommendation

Currently the optimization is working properly in the program, but study can be extended to determine an even faster algorithm for optimization. Additional user interface can be added to increase the user friendliness of the program. The program can be extended by adding an AutoCAD compatibility feature, which allows reading and writing an AutoCAD file. Automatically determining a type of truss for given envelope will be very useful feature of the program.

REFERENCES

1. Ritter, M. (2008). The analysis of cold-formed steel roof trusses
2. http://www.cs.mtu.edu/~shene/COURSES/cs3621/NOTES/model/winged-e.html
3. http://en.wikipedia.org/wiki/Nelder-Mead_method
4. http://emergent.unpythonic.net/software/hershey
5. http://mbinfo.mbdesign.net/CAD-History.htm
6. Hibbeler, R.C. Structural analysis sixth edition
7. http://www.geocities.com/evilsnack/undo.htm
8. http://mbinfo.mbdesign.net/CAD-History.htm
9. www.opengl.org
10. http://www.gamesforwindows.com/en-US/AboutGFW/Pages/DirectX10.aspx
11. Curtis F. Gerald, Patrick O. Wheatley, Applied Numerical Analysis
12. http://en.wikipedia.org/wiki/Computer_font
13. http://www.flipcode.com/archives/The_Half-Edge_Data_Structure.shtml

Hershey Fonts: rowmans.jhf

```
  699  1JZ
  714  9MWRFRT RRYQZR[SZRY
  717  6JZNFNM RVFVM
  733 12H]SBLb RYBRb RLOZO RKUYU
  719 27H\PBP_ RTBT_ RYIWGTFPFMGKIKKLMMNOOUQWRXSYUYXWZT[P[MZKX
 2271 32F^[FI[ RNFPHPJOLMMKMIKIIJGLFNFPGSHVHYG[F
RWTUUTWTYV[X[ZZ[X[VYTWT
  734
35E_\O\N[MZMYNXPVUTXRZP[L[JZIYHWHUISJRQNRMSKSIRGPFNGMIMKNNPQUXWZY[
[[\Z\Y
  731  8MWRHQGRFSGSIRKQL
  721 11KYVBTDRGPKOPOTPYR]T`Vb
  722 11KYNBPDRGTKUPUTTYR]P`Nb
 2219  9JZRFRR RMIWO RWIMO
  725  6E_RIR[ RIR[R
  711  9MWSZR[QZRYSZS\R^Q_
  724  3E_IR[R
  710  6MWRYQZR[SZRY
  720  3G][BIb
  700 18H\QFNGLJKOKRLWNZQ[S[VZXWYRYOXJVGSFQF
  701  5H\NJPISFS[
  702 15H\LKLJMHNGPFTFVGWHXJXLWNUQK[Y[
  703 16H\MFXFRNUNWOXPYSYUXXVZS[P[MZLYKW
  704  7H\UFKTZT RUFU[
  705 18H\WFMFLOMNPMSMVNXPYSYUXXVZS[P[MZLYKW
  706 24H\XIWGTFRFOGMJLOLTMXOZR[S[VZXXYUYTXQVOSNRNOOMQLT
  707  6H\YFO[ RKFYF
  708 30H\PFMGLILKMMONSOVPXRYTYWXYWZT[P[MZLYKWKTLRNPQOUNWMXKXIWGTFPF
709 24H\XMWPURRSQSNRLPKMKLLINGQFRFUGWIXMXRWWUZR[P[MZLX
  712 12MWRMQNROSNRM RRYQZR[SZRY
  713 15MWRMQNROSNRM RSZR[QZRYSZS\R^Q_
 2241  4F^ZIJRZ[
  726  6E_IO[O RIU[U
 2242  4F^JIZRJ[
  715 21I[LKLJMHNGPFTFVGWHXJXLWNVORQRT RRYQZR[SZRY
 2273 56E`WNVLTKQKOLNMMPMSNUPVSVUUVS RQKOMNPNSOUPV
RWKVSVUXVZV\T]Q]O\L[J
YHWGTFQFNGLHJJILHOHRIUJWLYNZQ[T[WZYYZX RXKWSWUXV
  501  9I[RFJ[ RRFZ[ RMTWT
  502 24G\KFK[ RKFTFWGXHYJYLXNWOTP RKPTPWQXRYTYWXYWZT[K[
  503 19H]ZKYIWGUFQFOGMILKKNKSLVMXOZQ[U[WZYXZV
  504 16G\KFK[ RKFRFUGWIXKYNYSXVWXUZR[K[
  505 12H[LFL[ RLFYF RLPTP RL[Y[
  506  9HZLFL[ RLFYF RLPTP
           507 23H]ZKYIWGUFQFOGMILKKNKSLVMXOZQ[U[WZYXZVZS RUSZS
```

```
508  9G]KFK[ RYFY[ RKPYP
  509  3NVRFR[
  510 11JZVFVVUYTZR[P[NZMYLVLT
  511  9G\KFK[ RYFKT RPOY[
  512  6HYLFL[ RL[X[
  513 12F^JFJ[ RJFR[ RZFR[ RZFZ[
  514  9G]KFK[ RKFY[ RYFY[
  515 22G]PFNGLIKKJNJSKVLXNZP[T[VZXXYVZSZNYKXIVGTFPF
  516 14G\KFK[ RKFTFWGXHYJYMXOWPTQKQ
  517 25G]PFNGLIKKJNJSKVLXNZP[T[VZXXYVZSZNYKXIVGTFPF RSWY]
  518 17G\KFK[ RKFTFWGXHYJYLXNWOTPKP RRPY[
  519 21H\YIWGTFPFMGKIKKLMMNOOUQWRXSYUYXWZT[P[MZKX
  520  6JZRFR[ RKFYF
  521 11G]KFKULXNZQ[S[VZXXYUYF
  522  6I[JFR[ RZFR[
  523 12F^HFM[ RRFM[ RRFW[ R\FW[
  524  6H\KFY[ RYFK[
  525  7I[JFRPR[ RZFRP
  526  9H\YFK[ RKFYF RK[Y[
 2223 12KYOBOb RPBPb ROBVB RObVb
  804  3KYKFY^
 2224 12KYTBTb RUBUb RNBUB RNbUb
 2262 11JZPLRITL RMORJWO RRJR[
  999  3JZJ]Z]
  730  8MWSFRGQIQKRLSKRJ
  601 18I\XMX[ RXPVNTMQMONMPLSLUMXOZQ[T[VZXX
  602 18H[LFL[ RLPNNPMSMUNWPXSXUWXUZS[P[NZLX
  603 15I[XPVNTMQMONMPLSLUMXOZQ[T[VZXX
  604 18I\XFX[ RXPVNTMQMONMPLSLUMXOZQ[T[VZXX
  605 18I[LSXSXQWOVNTMQMONMPLSLUMXOZQ[T[VZXX
  606  9MYWFUFSGRJR[ ROMVM
  607 23I\XMX]W`VaTbQbOa RXPVNTMQMONMPLSLUMXOZQ[T[VZXX
  608 11I\MFM[ RMQPNRMUMWNXQX[
  609  9NVQFRGSFREQF RRMR[
  610 12MWRFSGTFSERF RSMS^RaPbNb
  611  9IZMFM[ RWMMW RQSX[
  612  3NVRFR[
  613 19CaGMG[ RGQJNLMOMQNRQR[ RRQUNWMZM\N]Q][
  614 11I\MMM[ RMQPNRMUMWNXQX[
  615 18I\QMONMPLSLUMXOZQ[T[VZXXYUYSXPVNTMQM
  616 18H[LMLb RLPNNPMSMUNWPXSXUWXUZS[P[NZLX
  617 18I\XMXb RXPVNTMQMONMPLSLUMXOZQ[T[VZXX
  618  9KXOMO[ ROSPPRNTMWM
  619 18J[XPWNTMQMNNMPNRPSUTWUXWXXWZT[Q[NZMX
  620  9MYRFRWSZU[W[ ROMVM
  621 11I\MMMWNZP[S[UZXW RXMX[
  622  6JZLMR[ RXMR[
  623 12G]JMN[ RRMN[ RRMV[ RZMV[
  624  6J[MMX[ RXMM[
  625 10JZLMR[ RXMR[P_NaLbKb
  626  9J[XMM[ RMMXM RM[X[
 2225 40KYTBRCQDPFPHQJRKSMSOQQ RRCQEQGRISJTLTNSPORSTTVTXSZR[Q]Q_Ra
RQSSU
SWRYQZP\P^Q`RaTb
  723  3NVRBRb
 2226 40KYPBRCSDTFTHSJRKQMQOSQ RRCSESGRIQJPLPNQPURQTPVPXQZR[S]S_Ra
RSSQU
```

```
QWRYSZT\T^S`RaPb
 2246 24F^IUISJPLONOPPTSVTXTZS[Q RISJQLPNPPQTTVUXUZT[Q[O
  718 14KYQFOGNINKOMQNSNUMVKVIUGSFQF
```

VITA

KEYUR SHARADKUMAR PANDYA

Candidate for the Degree of

Master of Science

Thesis: SELECTED TOPICS IN DEVELOPMENT OF A CAD APPLICATION FOR SIZING LIGHT GUAGE COLD ROLLED STILL ROOF TRUSSES

Major Field: Mechanical and aerospace engineering

Biographical:

Education:
Completed the requirements for the Master of Science or in Mechanical and aerospace engineering at Oklahoma State University, Stillwater, Oklahoma in July, 2008.

Name: Keyur Sharadkumar Pandya Date of Degree: July, 2008

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: SELECTED TOPICS IN DEVELOPMENT OF A CAD APPLICATION FOR SIZING LIGHT GUAGE COLD ROLLED STILL ROOF TRUSSES

Pages in Study: 55 Candidate for the Degree of Master of Science

Major Field: Mechanical and aerospace engineering

Scope and Method of Study: The goal of the current project is to develop an application which will let users easily define building and truss geometries. Then, optimum truss member sizes will be calculated based on load inputs. The application will be user friendly with required features and is expected to facilitate a wide range of users. The purpose of this document's to discuss the selected topics involved in the development of this CAD application. The selected topics cover the use of an optimization technique to generate the desired roof slopes, the procedure to size the trusses, the data structure required for the computer program, the fonts to display text in 3D, the method to associate constraints on the geometry and the implementation of Undo/Redo functionality in the application.

Findings and Conclusions: The computer program lets user to generate a building in a 3D application. Snapping functionality and constraint feature is working correctly in the program. Once the user has defined the building geometry, the application will allow the user to define the location of trusses in the plan view. The user can also define group of trusses by defining a region and spacing between those trusses. Application will calculate the truss envelope based on the roof profile defined. User can define a custom truss or a pre-defined truss can be added from the available library. The results generated by the program are validated and a good match is found in the results. The application can also determine the optimum number of panels in any truss added from the library.

ADVISER'S APPROVAL: Dr. Ronald D. Delahoussaye