The Ideal Beam Design Software

MER 498 Senior Project

Jacob Laufer
3/11/2016

This report includes a full description of the current version of the Ideal Beam Design Software. The idea for this software stemmed from a beam design competition in MER 214. The software was developed in MER 497 and MER 498 under the guidance of Professor Andrew Rapoff, Ph.D.

Contents

Introduction

The Ideal Beam Design Software was developed for civil engineers and home owners to save time and money while undertaking construction projects. This software enables a user to design cost-effective beams by defining certain beam parameters through a friendly user interface. The software supports statically determinate beams of rectangular cross-sections. The user interface consists of a main graphical user interface (a GUI) which links to other GUIs through push buttons. In some of these linked GUIs, the user defines input parameters that define the beam they wish to create. These parameters define how the beam is loaded, how the beam is supported, the accuracy in which the software analyzes the beam, the material used to construct the beam, and the base dimensions that the software converts all values of different dimensions. Other linked GUIs make it possible for the user to view the outputs of the software in the form of diagrams and three dimensional plots. The outputs of the software include diagrams of the internal forces along the length of the beam and a three dimensional plot of the ideal beam. The software uses known mechanical equations for statically determinate beams and stresses to solve and generate the outputs. Analytical solutions for several ideal beam problems were performed and compared to the software's solution for the same problems. In all cases, the analytical solutions matched the software's solution, validating the software's method of creating the ideal beams.

In MER 497, the first version of the Ideal Beam Design Software was developed in Microsoft Excel using VBA code. In MER 498, the Microsoft Excel development was migrated to MATLAB. In addition, the software's mechanics, user interface, and outputs were improved to create the current version of the software. The current version includes a user friendly interface, a much quicker resolve time and several additional features that were not available in the first version of the software. As the Ideal Beam Design Software evolved, certain functionalities were removed while others were added. In the future, more functionality can be implemented to the software, making it more desirable to potential customers.

Problem Definition and Design Requirements

Large corporations that design and build structures often hire engineers to ensure that the beams within the structures are sturdy and will not fail. Likewise, home owners without technical engineering backgrounds who work on home improvement projects must find ways to ensure that the beams used in their projects do not yield and fail. Safety is of the upmost importance when designing structures. If careful consideration is not given to the design of every component within the structure then the structure has the potential to fail. A structure with failing components puts the safety of others at risk and jeopardizes the hard work and money that was required to build the structure. To ensure that everyone remains safe and money is not lost, both homeowners and engineers must validate that the beams used in their structures are capable of withstanding the largest amounts of stress that they could possible encounter. For many, the cost of the beam as well as the time that it takes to design the beam plays a large role in determining the type of beam that is constructed. The Ideal Beam Design Software makes it possible for engineers and home owners alike to design cost effective and safe beams in minimal time.

Big corporations spend significant resources on hiring civil engineers to design safe, cost-effective structures. A crucial part of the civil engineer's job is designing the beams that make up these structures. Traditionally, a civil engineer solves complicated beam problems by hand to design cost effective and safe beams. Often these engineers compute failure analyses on the blueprints of the structures to determine if any particular component within the structure will fail once built. There is free software available online that can conduct a failure analysis of a loaded beam. These software tools prompt the user to define loading of a beam with a predetermined cross-section and informs the user whether or not the beam would fail under such loading. The method of designing a structure first and then checking for failure is neither cost nor time efficient. A much more efficient method of designing structures is to hire an engineer during the initial design phase to help design the shapes of the structural components. This is a more cost effective method because the engineer can help decide the optimum cross-section of the components. Using the optimum cross-section reduces the amount of material used during the manufacturing phase which results in a reduction of the overall cost. The time until manufacturing is also reduced when this method is used. A failure analysis is not necessary when using this method since the components within the structure are designed to withstand the

applied loading. Also there is no longer the potential of needing to redesign components that do not pass the failure analysis. Engineers will typically design optimal beams by running calculations by hand however these are typically very complicated calculations which take substantial time to complete. Engineers and architects alike can use the Ideal Beam Design Software to design any optimal statically determinate beam of a rectangular cross-section.

Many home owners attempt to save money by taking on small scale construction projects for their homes opposed to hiring professionals. The average home owner who decides to pursue one of these projects does not possess the appropriate engineering background to ensure that every component of the project is designed safely. One option for home owners in this position is to refer to a beam manufacturer's catalog. These catalogs rate the beams that the manufacturer carries in terms of allowable loading magnitudes. Although the beams from these catalogs have a safety guarantee, disadvantages do exist with buying beams from these manufacturers. An individual who chooses to buy raw material from a home improvement store to construct beams on their own will have more flexibility in the design of their project and save money. When purchasing beams from a catalog the buyer is limited to the stock that is carried by the manufacturer and will not likely find all of the best suited components for their particular project. The buyer will also face drastic price mark-ups on components bought from a catalog in comparison to the cost of the raw materials. The Ideal Beam Design Software makes it possible for home owners to safely design any statically determinate beam of rectangular cross-sections with solely the raw materials and power tools.

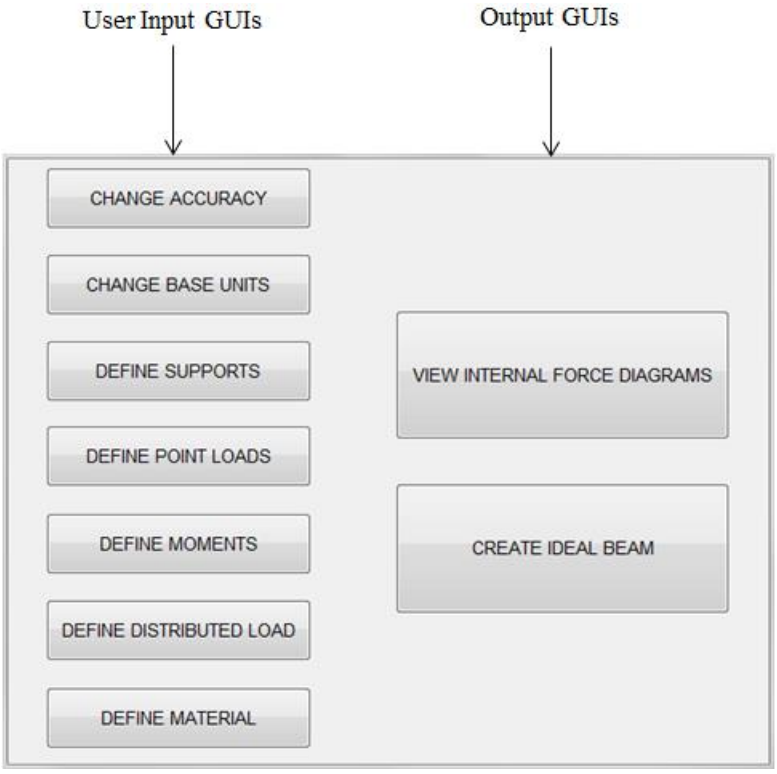
Computer aided-design (CAD) simulation tools exist which can run optimization analysis of any designed part. These tools have the ability to run analysis on significantly more complicated parts than a simple beam. The advanced nature of these simulation tools make them very expensive. In addition, significant training is required for an individual to know how to correctly use them. It is not practical for an architectural firm or a home owner to purchase an entire CAD package solely for beam design. The Ideal Beam Design Software serves as an easy to use optimization tool for who that are only concerned with optimizing beams.

Interface Design

The interface that was created in MER 498 was designed to optimize the user's experience while creating ideal beams using the Ideal Beam Design Software. The first version of the software was built on Microsoft Excel. The graphical user interface (GUI) functionality that MATLAB offers makes designing the software's interface in MATLAB a more desirable option than in Microsoft Excel. A graphical user interface design environment (GUIDE) exists within MATLAB which provides the tools necessary for MATLAB users to easily create GUIs. Instead of writing code to define objects within a GUI, GUIDE allows the user to interactively define and place objects such as static text boxes, edit text boxes, pushbuttons and popup menus as needed. MATLAB's GUIDE function was used to neatly layout and organize the different components of the final Ideal Beam Design Software's interface.

To launch the Ideal Beam Design Software one must run the Main_GUI.m file by pressing run in the MATLAB's editor tab with the appropriate file is open. Running this file launches the Main GUI, shown in **Figure 1**, which is comprised of several different pushbuttons. Activating each pushbutton launches the GUI that is described by the text on that particular pushbutton. The seven pushbutton located on the left side of the Main GUI are linked to user input GUIs. In a user input GUI the user is prompted to define certain parameters which the software saves and uses to run the appropriate analyses on the beam. The parameters that are defined by the seven user input GUIs are as follows: the system accuracy, the base units of the user inputs, the type, magnitude and location of the supports, the magnitude and location of any point loads located on the beam, the magnitude and location of any moments located on the beam, the starting point, ending point and equation of the line which defines a distributed load and the type of material that the beam is constructed of. The right side of the Main GUIs interface consists of two pushbuttons that link to output GUIs. The output GUIs runs calculations of beam analyses and displays them in a desirable fashion. The user must define the appropriate parameters through the user input GUIs in order to launch a particular output GUI. If not all of the appropriate parameters are defined and an output GUI is activated through the Main GUI interface, MATLAB will be unable to launch the activated output GUI. The user is able to view both the internal shear force and bending moment diagrams as well as the three dimensional ideal beam plot which represents the optimized beam by navigating the output GUIs. The Main GUI adds structure and ease of use to the Ideal Beam Design Software.

Figure 1



The Main GUI

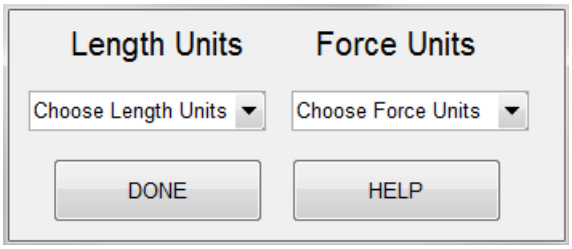
The first of the user input GUIs is the Accuracy GUI, shown in **Figure 2**, which is linked to the change accuracy pushbutton. One parameter can be defined within the Accuracy GUI. This parameter is the accuracy that that the software uses when analyzing the user defined beam, also referred to as the system's accuracy. Pressing the help pushbutton calls the Accuracy Help Dialog Box which explains how accuracy is used in the software. The system's accuracy dictates how many sections the software breaks a particular beam into per unit length during analysis. The higher the value of accuracy the more points that the software solves for along the length of the beam. Raising the accuracy of the software will result in more precise outputs, longer run times and more system memory used. The software automatically uses an accuracy of 10,000 if not otherwise specified in the Accuracy GUI. Only values between 0 and 100,000 are accepted as accuracy inputs. If a user attempts to input a value outside of this range a dialog box appears which prompts the user to define a value for accuracy which lies within the allowable range. Upon activating the done pushbutton the Accuracy GUI disappears.

Figure 2

The Accuracy GUI

The change base units pushbutton on the Main GUI activates the Base Units GUI shown in **Figure 3**. By using the Base Units GUI the user can define which units of force and length they would like the system to convert their inputs to. The user can choose from meters, millimeters, feet or inches for the length's base units and from newtons, kilonewtons or pounds for the force's base units. Upon entering a value to any parameter that has units associated with it, the user is given the option to choose which units they desire to attach to the value through the unit's pop-up menu. The default option in every unit's pop-up menu is the "Base Unit" option. If the "Base Unit" option is chosen from the unit's pop-up menu than the value that is attached to that unit is unchanged. If any other option is chosen from the unit's pop-up menu then the value that is attached to that unit's pop-up menu is converted to the particular base unit that was defined in the Base Units GUI. The internal force diagrams and ideal beam plots display results in the defined base units. If the user does not define the base units in the Base Units GUI then the user does not have the option to attach units to any parameters. In this case the user must work entirely with dimensionless values. If a user is working with dimensionless values and tries to attach a unit to any parameter then the user will encounter an error. The help pushbutton activates the Base Units Help Dialog Box and activating the done pushbutton makes the Base Units GUI disappear.

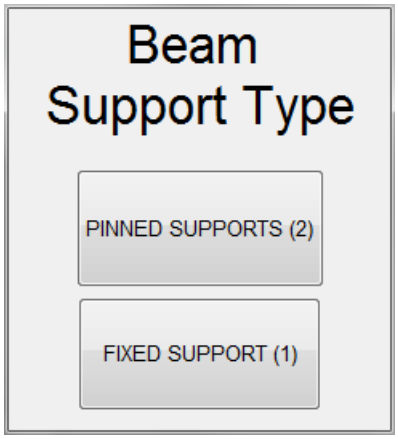
Figure 3



The Base Units GUI

The Define Supports pushbutton in the Main GUI links to the Support Type GUI shown in **Figure 4**. The Support Type GUI is a simple GUI which consists of two pushbuttons, each pushbutton linked to a separate GUI. In the Support Type GUI the user must choose between a beam that is supported by either pinned or fixed supports. Since the Ideal Beam Design Software does not support buckling analyses, only loads perpendicular to the beams can be applied. Since it is not possible to apply any type of load that contains a parallel component to the beam, no support reactions parallel to the beam exist. Without parallel support reactions roller supports and pin supports serve the same purpose for the sake of the Ideal Beam Design Software. Therefore users that wish to analyze beams that are supported by roller supports can replace them with pin supports without hindering the effects of the outputs. In addition to launching a new GUI and closing the current one when one of the two pushbuttons in the Support Type GUI is activated, the string value of the support type is saved to the MATLAB Workspace. The MATLAB Workspace is an area where a user can view the inputs that they have entered using the user input GUIs. If the input that is displayed in the MATLAB Workspace is a value with a unit attached to it that is different from the defined base units, the value is converted to the base units before it is displayed.

Figure 4

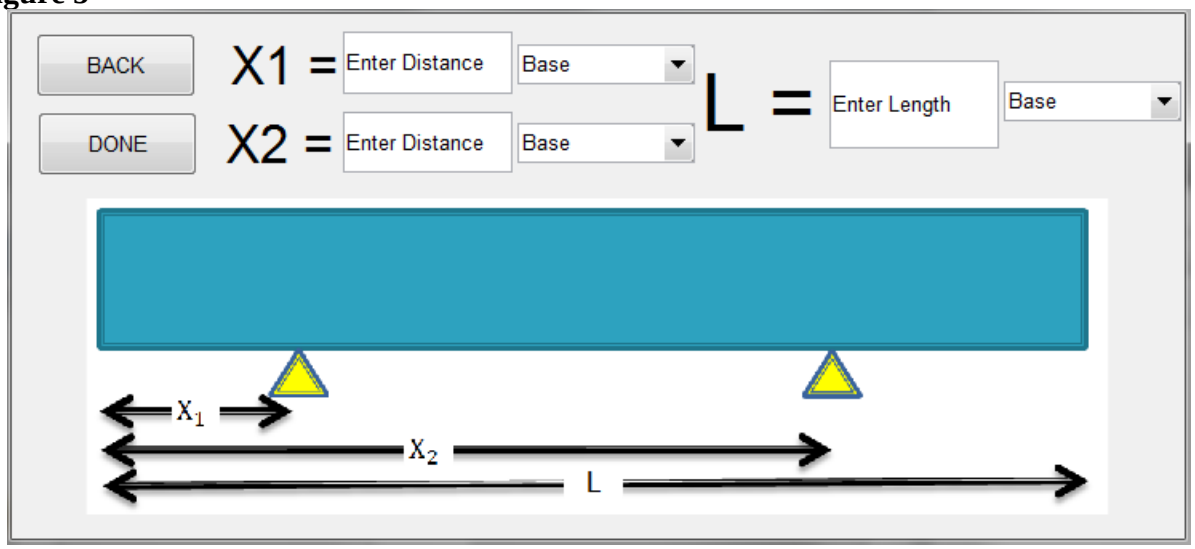


The Support Type GUI

The pinned supports pushbutton in the Support Type GUI links to the Pinned Supports GUI shown in **Figure 5**. The Pinned Supports GUI prompts the user to define the location of two pin supports and the length of the beam. The only type of pin supported beam that is supported by the software is a beam with two pin supports. The reason why the software only supports beams of this type is because beams that are supported by more or less than two pin

supports are not statically determinate. If the beam was only supported by one pinned support the beam has the potential to spin around the support making the beam dynamic. If the beam was supported by any more than two supports then too many unknowns would exist at the support locations and the beam would be indeterminate. The image at the bottom of the Pinned Support GUI guides the user to enter the appropriate values into the edit text box that lies next to the static text box which describes that particular parameter. If the user defines the values of the distances to the pinned supports to be greater than the length of the beam or less than zero and the user activates the done pushbutton than a message box appears which informs the user that the supports must be placed on the beam. If the supports are properly placed on the beam and the done pushbutton is activated then the Pinned Support GUI simply disappears. The back pushbutton closes the Pinned Support GUI and reactivates the Support Type GUI.

Figure 5



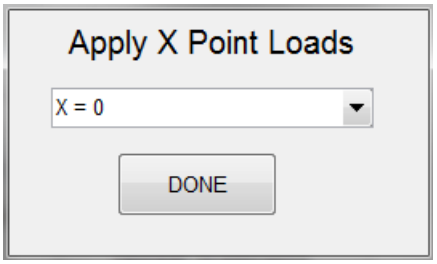
The Pinned Support GUI

The fixed support pushbutton in the Support Type GUI closes the Support Type GUI and opens the Fixed Support GUI. The Fixed Support GUI has the same general layout as the Pinned Support GUI. The only difference between these two GUIs is that in the Fixed Support GUI the user is prompted to define the location of a single fixed support opposed to in the Pinned Support GUI where the user must define the location of two pinned supports. The same reason exists for why the user may only define a single fixed support as to why the user may only define two pin supports. If more than a single fixed support is added to a particular beam then the beam is

unable to be analyzed determinately due to the excess amount of unknowns at the support locations.

The define point loads pushbutton in the Main GUI activates the Point Load Number GUI shown in **Figure 6**. The Point Load Number GUI contains a single static text box, a pop-up menu and a pushbutton. The point load pop-up menu is used to indicate how many point loads are to be applied to the beam. A user can add between zero and five point loads using the point load pop-up menu. Each option that is selected from the point load pop-up menu opens a unique GUI in which the user is prompted to define the location and magnitude of the point loads. A value must be chosen from the point load pop-up menu otherwise the software will be unable of analyzing the user defined beam. Upon activating the done pushbutton in the Point Load Number GUI the system places all five forces and force distance parameters in the MATLAB Workspace and then calls upon the appropriate Define Point Load GUIs. Of these ten parameters the parameters that the Define Point Load GUI prompts the user to define show up as empty vectors in the MATLAB Workspace while all other variables display an 'NA' value.

Figure 6

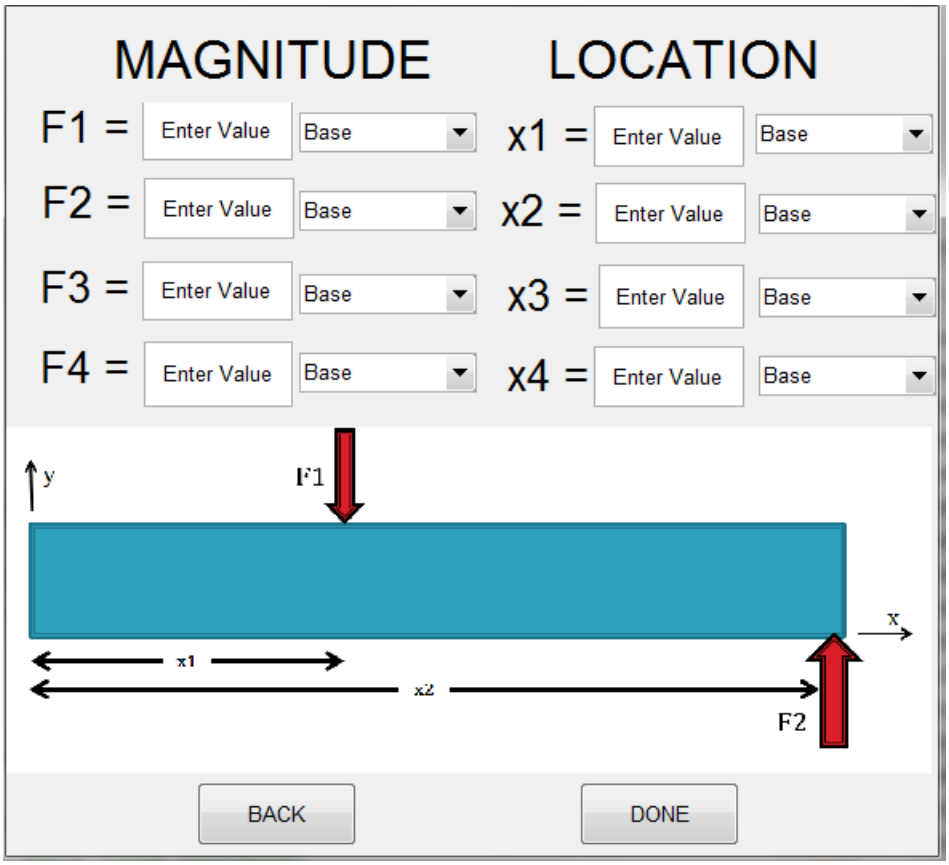


The Point Load Number GUI

There are five separate Define Point Load GUIs. The particular Define Point Load GUI that is called is dependent on the option that is chosen from the point load pop-up menu. Each of the options available in the point load pop-up menu, with the exception of the X = 0 option, calls a different Define Point Load GUI upon activation of the done pushbutton. **Figure 7** shows the particular Define Point Load GUI that is activated when the "4" option is chosen from the point load pop-up menu in the Point Load Number GUI. When this GUI is first activated forces and force distances one through four are set to empty vector values in the MATLAB Workspace while the fifth force and force distance are set to an 'NA' value. These empty vector values are updated using the corresponding edit text boxes in the Define Point Load GUI. A coordinate system is present in the image that appears at the bottom of the Define Point Load GUI. The

user must follow this coordinate system when defining point loads or the point loads may be applied in the wrong direction. The back pushbutton in this GUI closes the Define Point Load GUI and reopens the Point Load Number GUI while the done pushbutton closes the Define Point Load GUI.

Figure 7



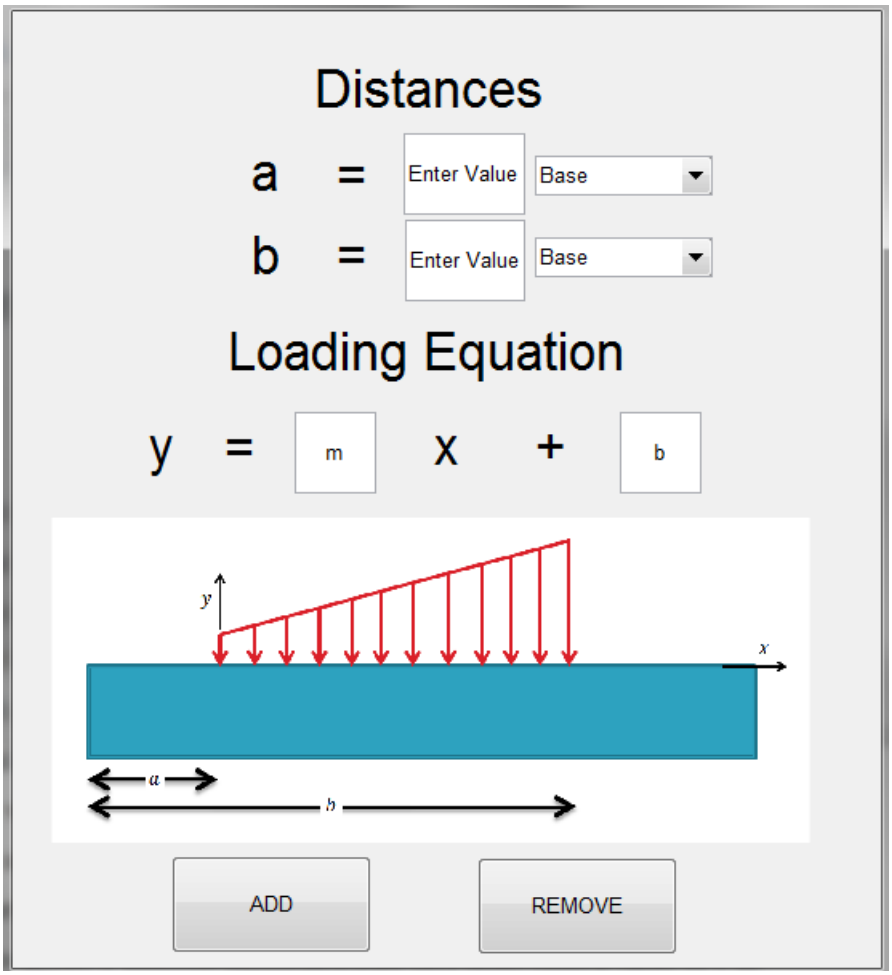
The Define Point Load GUI

The define moment pushbutton in the Main GUI interface begins a very similar sequence of GUIs that the define point load pushbutton begins. The define moment pushbutton links to the Moment Number GUI where a user can choose the number of moments they would like to load the beam with. Like the Point Load Number GUI a user can choose to apply up to five moments in the Moment Number GUI. Upon the completion of the Moment Number GUI one of five Define Moment GUIs appears unless the user chose to apply zero moments. The Define Moment GUI appears in a very similar manner as the Define Point Load GUI with the exception that a user is inputting the values for moments opposed to point loads. There is no axis

associated with the image at the bottom of the Define Moment GUI however two moments are displayed on a beam. The moment that is applied in the counterclockwise direction is defined as positive while the moment that is applied in clockwise direction is defined as negative. A user must keep these directions in mind when defining moments or the user could accidently apply a moment with the wrong orientation.

To activate the Distributed Load GUI, shown in **Figure 8**, the define distributed load pushbutton located on the Main GUI interface must be activated. Using the Distributed Load GUI a user can define one distributed load with a linear distribution. The user must define the starting and ending points of the distributed load in relation to the coordinate system shown in the image at the bottom of the Distributed Load GUI. Both the starting and ending locations of the distributed load are saved and can be viewed in the MATLAB Workspace. Once the user defines the span over which the distributed load acts the user must input the equation of the line that linearly distributed load follows. The Ideal Beam Design Software supports uniform distributed loads and linearly increasing or decreasing distributed loads. For a user to define a uniform distributed load they must enter a value of zero for the slope of the line and the value of the uniform load per unit length as the y-intercept. If the user chooses to define a linearly increasing or decreasing distributed load they must enter the appropriate slope and the appropriate y-intercept that defines the load's distribution. Both the slope and the y-intercept of the line that defines the distributed load's distribution are saved to the MATLAB Workspace. Once the distributed load is fully defined the add pushbutton discretizes the distributed load into a predetermined amount of equally spaced smaller loads across the span that the distributed load stretches and creates a vector with these values. The remove pushbutton sets this vector to a zero vector. Either the remove or add pushbuttons must be activated within the Distributed Load GUI for the software to run any of the output GUIs.

Figure 8



The Distributed Load GUI

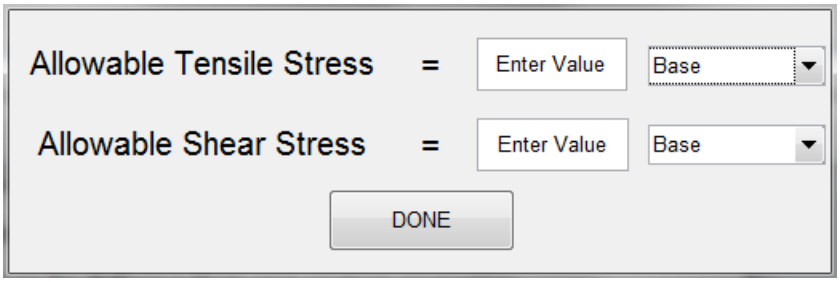
The last user input pushbutton located at the bottom left of the Main GUI activates the Materials GUI shown in **Figure 9**. The purpose of the Materials GUI is to prompt the user to define the material properties of the virtual beam they are creating. By using the choose material pop-up menu within the Materials GUI the user can choose which material they would like the beam to be constructed from the available options. A user can choose from two commonly used materials in the construction industry using the materials pop-up menu. These two materials are 6061 Aluminum and A36 Structural Steel. The allowable tensile and shear stresses associated with these two materials are programmed into the software and the values of the appropriate material properties are added to the MATLAB Workspace upon selection of that material. If the user chooses to construct their beam from a material that is not listed in the choose materials pop-up menu they can do so by defining the material properties of that material through the Properties GUI shown in **Figure 10**. To launch the Properties GUI the user must activate the

define custom properties pushbutton in the Materials GUI. The Properties GUI allows the user to enter the allowable tensile and allowable shear stress of any material that they wish. These properties are saved in the MATLAB Workspace.

Figure 9

The Materials GUI

Figure 10

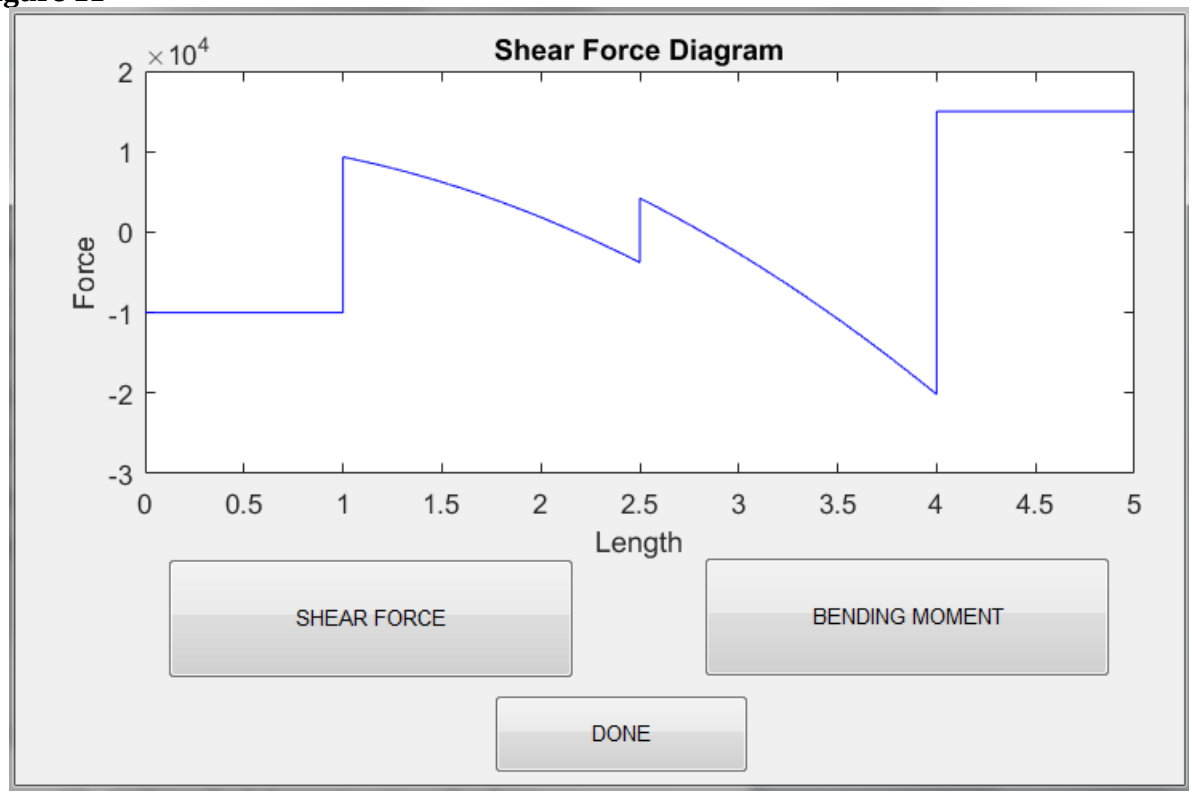


The Properties GUI

The view internal force diagrams pushbutton located on the right side of the Main GUI activates the Internal Force output GUI shown below in **Figure 11**. The loading and supports of the virtual beam must be fully defined before the software is capable of launching the Internal Force GUI. When the view internal force diagrams pushbutton is activated the software produces three vectors that are used to plot points within the Internal Force GUI which make up the internal shear and internal bending moment diagrams. The first of the three vectors that the software produces is the length vector. The length vector discretizes the entire beam into equally spaced sections. Each value within the length vector represents the distance from the starting point of the beam to the end of a particular section. The number of values that makes up the length vector depends on two user input parameters: the length of the beam and the system accuracy. The second of the three vectors is the shear force vector which consists of the for internal shear force values at every point along the length vector. The software calculates the

shear force vector using the defined point loads, a distributed load (if one exists) and the reactionary forces. The third vector, the bending moment vector, uses the shear force vector as well as applied and reactionary moments to create the bending moment vector. When the user activates the shear force pushbutton in the Internal Force GUI the length vector is plotted against the shear force vector on the axes located in the center of the Internal Force GUI. This plot is referred to as the internal shear force diagram. The shear force pushbutton is activated in the snapshot of the GUI shown in Figure 11. When the user activates the bending moment pushbutton the shear force vector is replaced by the bending moment vector and the title and y-axes label are changed to create a bending moment diagram. The software uses the shear force and bending moment vectors to create the ideal beam therefore the internal force diagrams pushbutton must be activated in order to run the final output of the Ideal Beam Design Software.

Figure 11

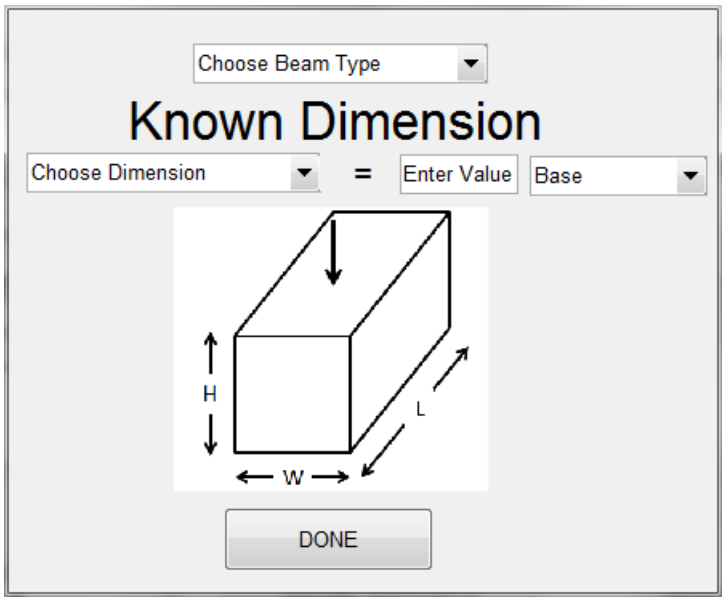


The Internal Force GUI with the shear force pushbutton activated

The Beam Type GUI, shown in **Figure 12**, is launched by activating the bottom pushbutton on the right side of the Main GUI. In the Beam Type GUI the user defines certain parameters that determine the cross-sectional dimensions of the optimized beam at every point

along the beam's length vector. In the choose beam type pop-up menu the user chooses how the cross-section of the beam will vary. The user can choose if the cross-section of the optimized beam is prismatic or varies on one side or both sides of the neutral axis. In the known dimension pop-up menu the user chooses which of the two rectangular cross-sectional beam dimensions are known and in the edit text box adjacent to that pop-up menu the user defines the value of that dimension. The image at the bottom of the Beam Type GUI advises the user on which rectangular cross-sectional dimension is considered the height and which is considered the width in relation to the applied loading. The software uses the known cross-sectional dimension along with the other user defined parameters to solve for the unknown cross-sectional dimension. If the user chooses the beam type that varies on one side of the neutral axis from the beam type pop-up menu, all positive points are plotted for the unknown cross-sectional dimension when the ideal beam is created. This method creates an optimized beam with three flat sides and one varying side. For a beam that varies on both sides of the neutral axis the values for the unknown cross-sectional dimension are divided by two and plotted as both positive and negative points. This kind of beam has two flat sides and two varying sides with each varying side having symmetry with respect to the other one. For the case of an ideal prismatic beam, the software calculates the maximum value of the unknown cross-sectional dimension and only plots that value as positive points. The software produces a rectangular beam as a result of the prismatic beam option. Upon the activation of the done pushbutton in the Beam Type GUI the software runs calculations to determine the points that make up the edges of the ideal beam and launches the Ideal Beam GUI.

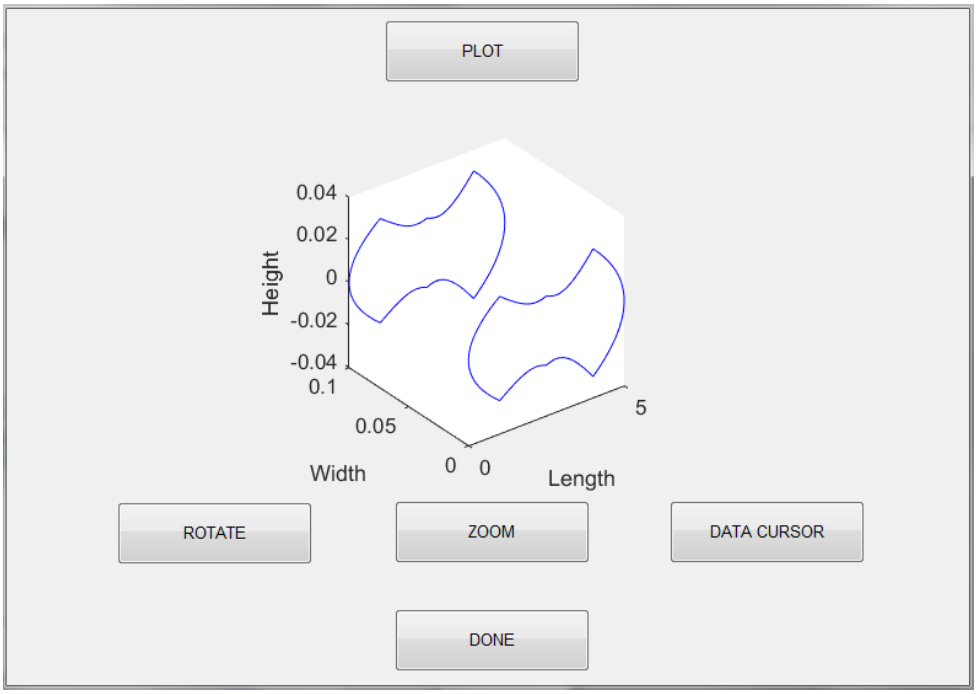
Figure 12



The Beam Type GUI

The Ideal Beam GUI, shown **in Figure 13**, is the final output of the Ideal Beam Design Software. To launch the Ideal Beam GUI all of the parameters that were defined to launch the Internal Force GUI, the material property parameters and the parameters within the Beam Type GUI must be defined. In the Ideal Beam GUI the software uses all of these user inputted parameters to create a plot of the ideal beam. The example in Figure 13 shows the plot of an ideal beam which varies on both sides of the neutral axis with the width of the beam being the known cross-sectional dimension. When the Ideal Beam GUI is first launched the plot pushbutton must be activated to display the current ideal beam. The rotate, zoom and data cursor pushbuttons activate certain modes within the Ideal Beam GUI. By utilizing these different modes the user is capable of viewing the ideal beam from different orientations and receiving exact coordinate positions of any point along the edges of the ideal beam. The done pushbutton closes the Ideal Beam GUI and returns the user to the Main GUI.

Figure 13



The Ideal Beam GUI

The Ideal Beam Design Software's user interface maximizes the ease of use and minimizes potential problems that a user may face while navigating through the software. The original interface, developed in Microsoft Excel for MER 497, was difficult to use and follow. In the new interface built using MATLAB GUIs a user can easily follow the natural flow of designing and loading a beam through user input GUIs, view their inputs in an organized fashion through the MATLAB Workspace and visualize their ideal beam through a three dimensional interactive plot. If the software does produce an error when an output is launched most likely all the necessary parameters are not defined that are required to launch that particular output. For best practice a user should start at the top left pushbutton on the Main GUI and work their way down and then to the right, defining all of the parameters along the way. Follow this method will ensure that the user's place is not forgotten and all the necessary parameters are defined.

System Mechanics

The mechanical method that the Ideal Beam Design Software uses to create ideal beams is based on known equations and validated analytical methods. The software uses all of the parameters that the user defines through user input GUIs to run calculations and create a plot that represents an optimized beam. The virtualized mechanical process used by the software to create the ideal beam can be broken down into six steps. First the software breaks the beam up into equally spaced sections through a process referred to as discretization. This step controls how many times and at what locations along the length of the beam the software calculates the cross-sectional dimensions of the rectangular beam. For the second step the software breaks up the magnitude of a distributed load into proportional point loads at every discretized point on the beam along the span of the distributed load. The software must also sum the entire distributed load and add it as a single point load at the centroid of the distributed load for the purpose of the support reaction analysis. In the third step the software calculates the unknown support reactions. The analytical equivalent to this step is drawing a free body diagram of the loaded beam and solving for all the unknown forces and moments that exist at the beam's supports. In the fourth step the software uses the previously solved reactionary forces and moments, the user defined point loads, the broken up distributed load (if applicable) and the user defined moments to create a shear force and a bending moment vector. These vectors are then graphed along the length of the beam to create internal shear force and internal bending moment diagrams. In the fifth step the software uses the internal shear force and bending moment vectors to create a new vector which defines the dimensions of the ideal cross-section at every discretized point along the beam. The ideal cross-section is a cross-section with the smallest possible area that creates a large enough moment of inertia that prevents yielding at that location due to the stresses caused by the applied loads. The final step that the software takes is to use the vector which defines the dimensions of the ideal cross-section and create a three dimensional image of the optimized beam.

Step 1: Discretizing the Length of the Beam

Before the software is able to make use of any mechanical equations for beam analysis purposes, it must create a length vector. This length vector is used to determine at what locations along the beam the software will run calculations. Since the internal forces on a loaded beam are not constant, solving for the dimensions of the beam's ideal cross-section at any arbitrary point

on the beam will not provide an accurate representation of the ideal cross-section at any other particular point on the beam. When the software ran on Microsoft Excel the number of discretized points was a predetermined value that could not be controlled by the user. One benefit to using a flexible coding environment such as MATLAB is that numbers output to matrixes which can be easily manipulated unlike the cells that are used in Microsoft Excel. This flexible coding environment makes it possible to vary the number of discretized points depending on certain beam parameters.

The number of components in the length vector is determined by the product of the non-dimensionalized value of the beam's length and the system's accuracy. A beam with a large non-dimensionalized length that is solved with a high system accuracy will have a length vector with many components. A large length vector may result in a long system run time. In the case of a beam with a large non-dimensionalized length it is recommended that either the large non-dimensionalized length parameter is converted to a smaller non-dimensionalized value by changing the system's base units or the system's accuracy is lowered to avoid a long system run time. **Equation 1** is used by the software to determine the distance from one end of the beam to any discretized point along the beam. The length vector is created in MATLAB by solving this equation until the distance to the discretized point is equal to the length of the beam.

$$\mathrm{DP} = \mathrm{DP}_{\mathrm{Offset}(-1,0)} + \frac{1}{A} \qquad \textbf{Equation 1}$$

Where DP is the distance to the discretized point of interest, the subscript Offset(X, Y) refers to a value in a matrix offset X number of columns and Y number of rows and A is the accuracy of the system

Step 2: Incorporating the Distributed Load

The software treats the user defined distributed load in two different ways for the purpose of analytical methods that occur in different steps of the software's analysis. In one of these ways the software takes the magnitude of the entire distributed load and accounts for it as a single point load acting at the centroid of the distributed load. The distributed load is treated in this manner for the purpose of support reaction analysis. The reactionary moments and forces that a point load causes are much simpler to analyze then attempting to analyze the reactions that a discretized distributed load causes. The other way that the software treats the distributed load is as a distributed load vector. In the distributed load vector the magnitude of the distributed load is broken into several tiny point loads proportional to the distributed load distribution. The

distributed load vector is useful when incorporating the distributed load to the shear force vector. Incorporating a point load representation of the distributed load to the shear force vector does not produce accurate results.

The first step that software takes in the analysis of the distributed load is to take the parameters that describe the distributed load and create a distributed load vector. Once the distributed load vector is created it is summed with another vector which accounts for the point loads and support reaction forces, the sum of these two vectors creates the shear force vector. The distributed load vector has an equal amount of components as the length vector. Values of zero are assigned to the components of the distributed load vector whose positions do not correspond to the span of the distributed load along the length vector. For the components of the distributed load vector that do correspond to the locations on the length vector in which the distributed load spans, **Equation 2** is used to determine each component's value.

$$y = \frac{mx^* + b}{A}$$ **Equation 2**

Where y is to the distributed load vector, m is the user defined slope of the distributed load, x^* refers to every component of the length vector between the starting and ending points of the distributed load, b is the user defined y-intercept of the distributed load and A is the accuracy of the system

Once the distributed load vector is defined, the software calculates the magnitude of the entire distributed load and the horizontal distance from the beginning of the beam to the centroid of the distributed load. Once these two values are calculated the software can treat the distributed load as a single point load which is beneficial to solving for the unknown support reactions. To determine the magnitude of the entire distributed load the software sums the values of all the components within the distributed load vector. To determining the horizontal distance from the start of the beam to the centroid of the distributed load, the software adds the distance between the start of the beam and the start of the distributed load to the horizontal distance between the start of the distributed load and the centroid of the distributed load. The distance from the y-axis to the centroid depends on the shape that the distributed load takes. Distributed loads that are capable of being defined using the Ideal Beam Design Software may take the shape of a rectangle, a triangle or a triangle sitting atop a triangle. The software breaks up any shape that the distributed load takes into a triangle and a rectangle. The software proceeds to calculate the area of each shape and the horizontal distance from the starting point of the distributed load to the centroid of each shape. The software then takes the values for these calculated areas and

distances and plugs them into **Equation 3** which solves for the horizontal distance between the beginning of the distributed load and the centroid of the entire distributed load.

$$C_x = \frac{\sum_n A_n C_{x_n}}{\sum_n A_n} \qquad \textbf{Equation 3}$$

Where C_x is the distance from the y-axis to the centroid, A is the area of a particular shape and the subscript n represents the two shapes that multiple shapes that make up the distributed load

Step 3: Solving for the Support Reactions

Once the beam is successfully discretized and the distributed load is properly accounted for the software uses the user defined loading, support types and support locations to solve for the unknown support reactions. Since only determinate beams and loading perpendicular to a beam is supported by the software, every case of a supported and loaded beam results in two unknown support reactions. The software uses two known principles of static beams to solve for the unknown support reactions: the sum of the forces and the sum of the moments applied a static beam must equate to zero. Using these static beam principles the software creates two equations which include the two unknown support reactions. Two equations with two unknown variables form a system of equations. The software solves the system of equations to determine the values of the unknown support reactions.

The software defines the two pin supports in a pin supported beam as Support A and Support B. Since pin supports do not support reactionary moments and the software does not support components of force parallel to a beam, the two support reactions for a pin supported beam case are vertical reactionary forces at each pin support. The laws of a static beam in conjunction with the unknown vertical reactionary forces at the pin supports are used to create **Equation 3** and **Equation 4**. Unlike pin supports fixed supports support reactionary moments. The single fixed support on the fixed supported beam is defined as Support C. For the case of a fixed supported beam the unknown reactions at support C are described as a vertical reactionary force and a reactionary moment. Since a fixed supported beam is always static **Equation 5** and **Equation 6** were developed for a beam of this nature. The software solves these two sets of equations simultaneously to determine the unknown support reactions.

Pair of Equations for Beams with Two Pin Supports

$$\sum F_Y = 0 = \sum F_{UD} + F_{R_A} + F_{R_B}$$ **Equation 3**

Where F is a force, the subscript Y refers to the y direction, the subscript UD refers to a user defined input, the subscript R refers to an unknown reaction at a support location, the subscript A refers to Pin Support A and the subscript B refers to Pin Support B

$$\sum M_{R_A} = 0 = \sum M_{UD} + \left(\sum_{n=1}^{x} F_{n,UD} * (D_{n,UD} - D_{R_A})\right) + F_{R_B} * (D_{R_B} - D_{R_A})$$ **Equation 4**

Where M is a moment, D is the distance from the beginning of the beam and the point of interest, the subscript n refers to the number of the parameter it references and x refers to the last value of n

Pair of Equations for Beams with One Fixed Support

$$\sum F_Y = 0 = \sum F_{UD} + F_{R_C}$$ **Equation 5**

Where the subscript C refers to Fixed Support C

$$\sum M_{R_C} = 0 = \sum M_{UD} + \left(\sum_{n=1}^{x} F_{n,UD} * (D_{n,UD} - D_{R_C})\right) + M_{R_C}$$ **Equation 6**

To solve for the unknown support reactions the software first creates three matrixes out of the appropriate support case's pair of equations. The three matrixes that are formed are matrix A, matrix x and matrix B. Matrix A is defined as a two-by-two matrix filled with the coefficients that are attached to each unknown support reaction, matrix x is a two-by-one matrix containing the unknown reaction variables and matrix C is a two-by-one matrix that contains the opposite sign of the sum of the constants in each equation. These three matrixes are solved using matrix multiplication, a method exhibited in **Equation 7**. **Table 1** demonstrates how each of these three matrixes are constructed by the software for both support cases.

$$x = A^{-1}B$$ **Equation 7**

Where x is defined as matrix x, A^{-1} is defined as the inverse of matrix A and B is defined as matrix B

Table 1

Matrix	Pin Support Case	Fixed Support Case
A	$\begin{matrix} 1 & 1 \\ 0 & D_{R_B} - D_{R_A} \end{matrix}$	$\begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix}$
X	$\begin{matrix} F_{R_A} \\ F_{R_C} \end{matrix}$	$\begin{matrix} F_{R_C} \\ M_{R_C} \end{matrix}$
C	$\begin{matrix} -\sum F_{UD} \\ -\sum M_{UD} - \left(\sum_{n=1}^{x} F_{n,UD} * (D_{n,UD} - D_{R_A})\right) \end{matrix}$	$\begin{matrix} \sum F_{UD} \\ -\sum M_{UD} - \left(\sum_{n=1}^{x} F_{n,UD} * (D_{n,UD} - D_{R_C})\right) \end{matrix}$

Summary of the matrixes that are formed from the beam statics equations

Step 4: Calculating Internal Shear and Bending Moment

Once the unknown reactionary forces and moments are determined on a beam, the software creates a shear force and bending moment vector which represent the internal forces along the length of the beam. These two vectors are used to create the Internal Shear Force Diagram and Internal Bending Moment Diagram system outputs. Both internal force vectors are created by calculating either the shear force or bending moment at every location along the beam's length vector. The software calculates the internal shear force at any particular location along the length vector by summing all the forces that lie between that particular location and the start of the beam. The software calculates the internal bending moment at any particular location along the length vector by summing every moment that lies between that particular location and the start of the beam with the value of the integrated shear force at that location.

The internal shear force vector is the first of the two internal force vectors created by the software. When creating the internal shear force vector the software first creates an initial shear force vector which includes the user defined point loads and the calculated reaction forces. It is not until after this initial vector is created that the distributed load vector is added to create the shear force vector. To create the initial shear force vector the software first creates a two-by-seven matrix. The first row of this matrix includes all the user defined point loads and reactionary force magnitudes. Each component in the second row of the matrix contains the distance from the start of the beam to the force magnitude located in that component's column. The maximum number of forces that a beam in the Ideal Beam Design Software may encounter is seven, five user defined and two reactionary. If the beam does not contain the maximum number of forces than the components of the matrix that represent the forces and the force distances that do not exist equal zero. The software then sorts each column of this matrix by distance values. Using the sorted matrix the software is able to construct the initial shear force vector which incorporates all of the reactionary and user defined forces.

The shear force at any location along the length of a beam is related to the bending moment at that same location by a known mechanical correlation expressed in **Equation 8**. The software uses this correlation to create an initial moment vector using the previously determined shear force vector. For every user defined or reactionary moment on a beam the software creates an additional moment vector. Each additional moment vector is comprised of two parts, the first part contains components with all zero values and the second part contains components with all

values that are equal to that particular user defined or reactionary moment. The length of the first part of any additional moment vector is equal to the number of components in the length vector that exists between the start of the beam and the point where the moment occurs. The length of the second part is equal to the number of components in the length vector that exists between the point where the moment occurs and the end of the beam. The sum of all the moment vectors and the initial moment vector produces the bending moment vector.

$$dM = Vdx \qquad \textbf{Equation 8}$$

Where dM is the change in internal bending moment at a given point, V is the internal shear force at that point and dx is the change in length at that point

Step 5: Creating the Ideal Beam Profile

Once the software has calculated the internal force vectors, it uses these internal forces to determine the ideal beam profile. The ideal beam profile is a vector containing the values of the ideal unknown cross-sectional dimension of the beam at every location along the beam's length vector. The software determines what the allowable stresses are within the beam based upon the beam's material property parameters. The software accounts for two types of allowable stresses, allowable tensile stress and allowable shear stress, when determining the ideal beam profile. These two allowable stresses create two separate vectors containing the unknown cross-sectional dimension of the ideal beam. The maximum value at each position along the two vectors is taken to create the ideal beam profile.

To solve for the unknown cross-sectional dimension along the length of an ideal rectangular cross-sectional beam, the software manipulates the fundamental equations for internal tensile stress due to bending moment and internal shear stress due to shear force. The two vectors which dictate the unknown cross-sectional dimension are calculating by setting the internal tensile stress due to bending moment equal to the yielding tensile strength of the material and the internal shear stress due to shear force equal to the yielding shear strength of the material. Knowing that the maximum tensile stress of a beam's cross-section occurs on the neutral axis and the maximum shear stress of a beam's cross-section occurs at the farthest point from the neutral axis, the fundamental equations for allowable stresses are manipulated to solve for the moment of inertia of an ideal cross-section. Using the equation for the moment of inertia of rectangular cross-section, the two equations for allowable stresses are further manipulated to solve for the unknown cross-sectional dimension of an ideal beam. The software uses **Equation 9** and **Equation 10** to solve for the unknown cross-sectional dimension due to both tensile and

shear stresses at any point along the ideal beam if the known cross-sectional dimension is the width. If the known cross-sectional dimension is the height the software uses **Equation 11** and **Equation 12** to solve for the unknown cross-sectional dimension. The appropriate pair of equations is solved at every location along the beam's length vector to create two cross-sectional dimensional vectors based on the allowable stresses in the beam. These two cross-sectional dimensional vectors are the necessary components in creating the ideal beam profile.

Equations used when the known cross-sectional dimension is the beam's width

$$h(x)_b = \sqrt{\frac{6M(x)}{w\sigma}} \qquad \textbf{Equation 9}$$

Where h(x) is the ideal cross-sectional height dimension vector, the subscript b refers to a particular cross-sectional dimension due to bending, M(x) is the bending moment vector, w is the known value for the beam's width and σ is the allowable tensile strength of the material

$$\mathrm{h(x)_s} = \frac{3}{2}\frac{\mathrm{V(x)}}{\mathrm{w\tau}} \qquad \textbf{Equation 10}$$

Where the subscript s refers to a particular cross-sectional dimension due to shear stress, V(x) is the shear force vector and τ is the allowable shear stress of the material

Equations used when the known cross-sectional dimension is the beam's height

$$\mathrm{w(x)_b} = \frac{6M(x)}{h^2\sigma} \qquad \textbf{Equation 11}$$

Where b(x) is the ideal cross-sectional width vector and h is the known value for the beam's height

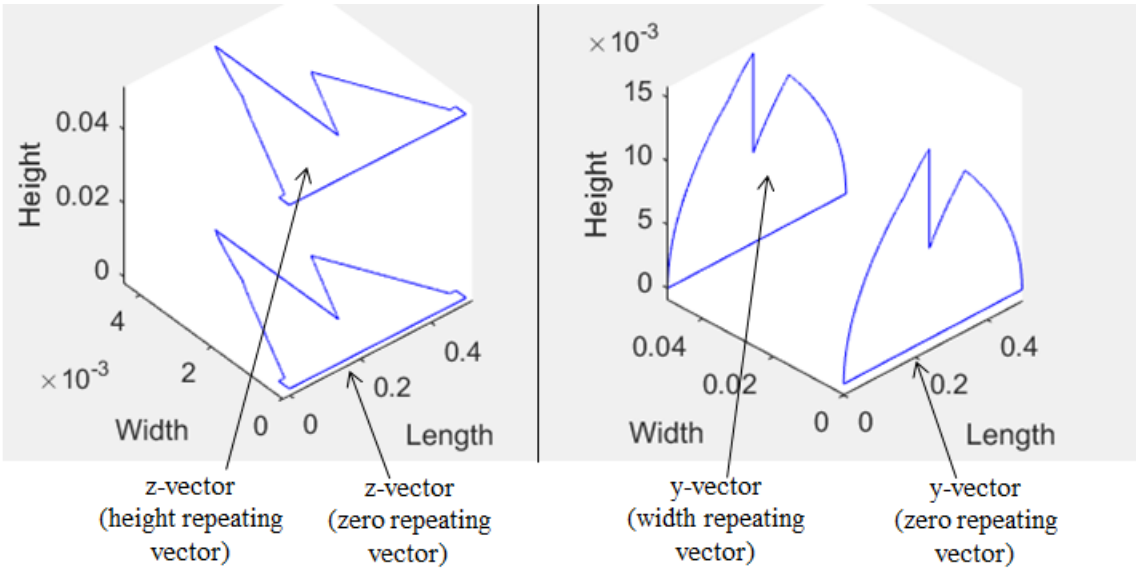
$$\mathrm{h(x)_s} = \frac{3}{2}\frac{\mathrm{V(x)}}{\mathrm{h\tau}} \qquad \textbf{Equation 12}$$

Step 6: Creating the Three Dimensional Ideal Beam Plot

The way that the Ideal Beam Design Software constructs the three dimensional ideal beam plot depends on the type of beam that the user chooses in the Beam Type GUI. The three dimensional ideal beam plot is composed of a combination of ideal lines. An ideal line is a line defined within three dimensional space which denotes an edge of a virtualized three dimensional beam. An ideal beam plot that varies on both sides of the neutral axis contains four ideal lines while an ideal beam plot of a prismatic beam or a beam that varies on a single side of the neutral axis contains only two ideal lines. In the three dimensional coordinate system that contains the ideal beam plot the x-axis represents the length of the beam, the y-axis the width and the z-axis the height. The software uses x y and z-vectors to create coordinate points which when connected form an ideal line. The x-vector of an ideal line is always the length vector. Depending on which cross-sectional dimension of the ideal beam is known, the ideal beam

profile or a manipulated version of the ideal beam profile is designated as either the y or z-vector of any ideal line. If the known cross-sectional dimension is defined as the height then the ideal beam profile or a manipulated version of the ideal beam profile will be designated as the y vector however if the known cross-sectional dimension is the width it will be designated as the z-vector. The remaining vector of any particular ideal line is a repeating value vector which can take two forms. The two forms that the remaining vector may take are either a zero vector or a repeating value vector with values equal to the known cross-sectional dimension. **Figure 14** compares two single side varying ideal beam plots: one whose unknown cross-sectional dimension is the height and one whose unknown cross-sectional dimension is the width. Also note how in this figure different types of repeating vectors are used to offset ideal lines of the same two dimensional shape.

Figure 14

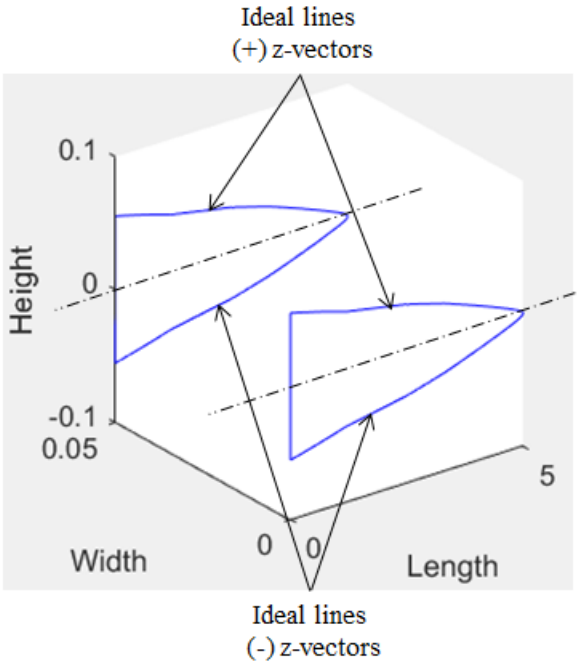


Comparison of two ideal beam plots with different known cross-sectional dimensions

In an ideal beam plot that only varies on one side of the neutral axis, the original version of the ideal beam profile defines either the y or z-vector of an ideal line. This is not the case for a prismatic beam or a beam that varies on both sides of the neutral axis. When creating these types of beam plots the software manipulates the ideal beam profile before it is used as a vector of an ideal line. In a prismatic beam plot the ideal beam profile is manipulated by creating a repeating vector of the maximum value of the original ideal beam profile. For a beam that varies on both sides of the neutral axis the software creates two different manipulated ideal beam profiles. Every value in one of these manipulated ideal beam profiles is equal to half the values

of the original ideal beam profile while the other is equal to the opposite sign of half the values of the original ideal beam profile. **Figure 15** shows an ideal beam plot of a beam that varies on both sides of the neutral axis. Note how the negative and positive versions of the ideal beam profiles give the beam symmetry about the neutral axis and how the plot is composed of four ideal lines.

Figure 15



Ideal beam plot of a beam that varies on both sides of the neutral axis

Design Evolution

The idea for creating the Ideal Beam Design Software was sparked from a beam design competition. In this competition, teams were challenged to design a beam with a predetermined length that was supported in a predetermined manner and loaded by a point load at a predetermined location. The competition was scored based on a ratio of the weight that the beam could hold to the volume of the material that comprised the beam's construction. All the beams were constructed of the same material so the competition solely depended on the shape of the beam. Upon the completion of this competition, the very first ideal beam was created using software that became the building blocks for the Ideal Beam Design Software. The first version of the software, which ran on Microsoft Excel, was primitive in comparison to the current version of the software, which runs on MATLAB. Many of the same principles that were

incorporated into the original software that designed the first ideal beam for the beam design competition are used in the current version of the software. Three components of the software evolved throughout its development, giving the software the capabilities it possesses in the current version. These three components are the system's mechanics, the user interface design, and the software's ideal beam output.

Mechanical Development

In the early development phase, Microsoft Excel proved to be a useful environment. In the Microsoft Excel spreadsheet that contained the first version of the software all of the inputs and intermediate calculations could be viewed in the easy to read format of a spreadsheet. This easy to read format simplified troubleshooting any problems that arose during the mechanical development. If a mechanical problem did arise in the software it was easy to pinpoint which cells within the spreadsheet were affected. Knowing the cells that were affected within the software would easily lead the developer to the mechanical problem that was causing the faulty results. Developing the mechanics of the software in a Microsoft Excel environment was a valuable first step in the development of the system's mechanics.

In these early stages of development, while working in the Microsoft Excel environment, the system's mechanical processes evolved to increase the efficiency of the software. In early development of the first version of the software, the bending moment and shear force vectors of a loaded beam were calculated on a case by case basis. During this stage, the internal force vectors of a particular beam were determined using unique static beam equations that varied depending on the manner in which the beam was loaded. This method was extremely inefficient from a development standpoint. An extra step was added in the mechanical process that the software used to solve for the internal force vectors. In this extra step the support reactions are solved using the static beam equation before the internal force vectors are created. The calculated support reactions are then used along with the user defined loading to create the internal force vectors. Developing the software to solve problems in this manner drastically reduced the amount of code necessary to solve for the internal force vector. This method of solving for the internal force vectors is the method that is used in the current version of the software.

There are some differences between the features that were available in the first version of the software and the features that are available in the current version. During the development of the first version of the software, significant time was spent researching the effect adding a hole to a loaded beam would have on the shape of the ideal beam. In the first version of the software, the user had the capability of adding a hole of any size along the neutral axis of the beam, a feature that is no longer available in the current version of the software. This feature no longer exists in the current version because the mechanics behind this capability are not 100% accurate. Although this particular feature was dropped in the current version of the software, another more practical feature exists. This new feature gives the user the ability to define a distributed load on a beam. It was decided that architects and engineers using the software to design beams would appreciate the distributed load feature more than the hole feature.

User Interface Development

One of the major reasons that the software was migrated from Microsoft Excel to MATLAB was to improve the user interface. In the first version of the software, a user would define all the parameters of the beam through a single column of cells. Next to this single column of cells were pushbuttons that when pressed in the correct order would output coordinate points representing the ideal beam. This interface that was created in Microsoft Excel was not intuitive and errors often occurred due to the complicated nature of the interface. The interface was also very basic and not graphically pleasing to any user. When the software migrated to MATLAB, user input GUIs and output GUIs were created. These GUIs are formatted in an organized fashion and guide the user to input the appropriate parameters in the appropriate locations and press the appropriate pushbuttons at the appropriate times. In the early stages of the MATLAB interface development, the interface did not include a main GUI but rather consisted of user input GUIs and output GUIs in a sequential order. To navigate through an interface of this type, the user would define the beam in the order that the GUIs appeared. If the user decided they wanted to change the value of a parameter they had already defined they would have to navigate through the interface backwards, change the parameter, and then navigate back to their previous location. The main GUI was incorporated into the current version of the software's interface so the user could navigate to any particular GUI within the software in a single step. Adding this flexibility to the interface saves the user time and adds structure to the software's interface.

As the software's interface evolved to become the interface that runs on current version of the software, certain functionalities were added which were not available in the first version. In the first version of the software, the accuracy of the system was not a parameter that could be changed by the user. In this version, the number of discretized points along the length of the beam was a predetermined value unrelated to the length of the beam. This change in functionality is due to the way that the MATLAB represents lists of numbers in comparison to Microsoft Excel. MATLAB includes lists of numbers in a single matrix or vector while Microsoft Excel designates each value of a list to its own cell. Since a vector or a matrix is a single object, it can be easily manipulated making it possible to vary the number of sections that comprise the beam. In the current version of the software accuracy is the user defined parameter that allows a user to vary the number of sections of a beam. Another functionality added to the current version of the software was the ability for a user to define a dimension attached to an input. The existence of pop-up menus in MATLAB made it possible to prompt a user to add dimensions to their inputs.

Software Output Development

From the software that outputted the first ideal beam for the design competition to the beams designed in the current version of the software, the Ideal Beam Design Software has always outputted coordinate points to define the outline of an ideal beam. Over time what evolved was how these coordinate points were used to create three dimensional images of ideal beams. In the first version of the software, the coordinate points produced by the software were saved as text files. The idea behind creating a text file with the coordinate points was that they could be exported to a separate CAD platform where the three dimensional image of the ideal beam could be viewed. Since MATLAB possesses the functionality to create three dimensional plots, when the software was migrated over to MATLAB the coordinate points could create the ideal beam within the software itself. The first types of ideal beams that were created and viewed within the MATLAB interface were two dimensional beam profiles. As the methods of plotting the coordinate points advanced, the outputs of the system evolved to become three dimensional beam plots. In the current version of the software, where three dimensional beam plots are the system's output, the functionality of creating a text file containing the coordinate points of the ideal beam no longer exists.

Future Improvements

The functionality and user friendliness of the Ideal Beam Design Software has drastically improved over the past two releases however there is still room for improvement. Incorporating different shapes into the beams that are supported by the software is one area in which the software can increase functionality. The first version of the software accounted for the stress concentrations due to a hole however this functionality never reached the point of complete accuracy. More research and development regarding these stress concentrations due to a hole could be done to rebuild this functionality into future releases of the software with 100% accuracy. Future releases could also focus on the stress concentrations due to other shapes. Another potential feature that could be incorporated into future releases is giving the user the option to vary the cross-section of a beam. Since I-beams are so prevalent in structural architecture, these would be the most practical beams to include in future versions of the software.

Adding certain mechanical methods to the software in future releases would increase the flexibility a user has in regards to the number of support location along a beam and the direction of applied point loads on a beam. If the method of superposition was incorporated into future releases of the software, then the software would be capable of solving statically indeterminate beams. When solving statically indeterminate beams, static beam equations are not sufficient in determining all of the reactionary forces so the deflection formula must be accounted for as a means of solving the redundant reactions of a beam. Incorporating buckling analyses to future versions of the software would give the user the capability of including components of force parallel to a beam. If future releases of the software accounted for parallel components of force, columns would be added to the types of ideal structures that the software would have the capability of optimizing.

Certain aspects of the user interface can be built upon in future releases of the software to improve the user's experience. In the current version, when the user does not define all the necessary parameters of the beam through the input GUIs and tries to run an output GUI, an error appears in the MATLAB Command Window. It would be helpful if instead of a cryptic error appearing in the Command Window, error dialog boxes were programmed into the system to inform the user of exactly what the problem was that caused the error. With these messages a

user could easily fix the problem instead of being confused to why the software was not able to produce the expected outputs. Certain features could also be added to the outputs of the system in future releases which could improve the user's understanding of the software's outputs. Another improvement to the software's user interface would be to show the local minimums and maximums on the y-axes of the shear and bending moment diagrams. This improvement would make these diagrams much easier for the user to interpret. Another way to better the user's understanding of these diagrams and beam plots would be if the axes labels were marked with the units of the parameter that was displayed. Adding the option of switching the units that are displayed on the axes is a feature that could potentially help users manufacture the beam.

A user can easily create ideal beams using the current version of the Ideal Beam Design Software. The user interface of the software is easy to follow and the mechanics behind the software produce accurate results. The mechanics that the software uses to produce the ideal beams have been verified by comparing the results of the software's solution of a specific idealization problems to the analytical solution of the same problem. Potential customers such as civil engineers and home owners will only find the Ideal Beam Design Software helpful if they are trying to construct the types of beams that are supported by the software. With continued enhancements, the Ideal Beam Design Software has the potential to produce a larger range of ideal beams and as a result become more desirable to potential customers.