1. How to set up a truss

1.1. Define the nodes

You can define the nodes that make up the structure by clicking the Add, Edit, or Remove button under the Nodes section. The nodes are called "keypoints" in Ansys.

Please note that the GUI will automatically assign the node indices. The node indices are always consecutive integers, starting with the number 1.

To add a node, please specify the following:

x-coordinate	the (global) x-coordinate of the node
y-coordinate	the (global) y-coordinate of the node

To edit an existing node, please specify the following:

Index of the node	the index of the node that you want to modify
x-coordinate	the (global) x-coordinate of the node
y-coordinate	the (global) y-coordinate of the node

To remove an existing node, please specify the following:

Index of the node	the index of the node that you want to remove

1.2. Define the elements

You can define the elements that make up the structure by clicking the Add or Remove button under the Elements section. The truss elements are called "lines" in Ansys.

Please note that the GUI will automatically assign the element indices. The element indices are always consecutive integers, starting with the number 1.

To add an element, please specify the following:

Indices of the two nodes	the indices of the start node and the end node
Young's modulus	a material parameter
Cross-sectional area	a material parameter

To remove an existing element, please specify the following:

Index of the element	the index of the element that you want to remove

1.3. Define the boundary conditions (BCs)

You can define the boundary conditions on the nodes by clicking the Add or Remove button under the Boundary conditions section.

Please note that you must specify the BC components in x- and y-directions together when adding a BC. When you export the assembly file, all nodes without a boundary condition will be assigned a zero force BC in the x- and y-directions.

To add a boundary condition, please specify the following:

Index of the node	the index of the node where you want to impose the BC
Type of boundary condition	indicate whether you want a displacement or force BC
Value of the x-component	the amount of displacement/force in the x-direction
Value of the y-component	the amount of displacement/force in the y-direction

Some common boundary conditions:

To specify a fixed pin, you would select "Displacement, Displacement" for the type of boundary condition, and enter the number 0 for the value of the x-component and the value of the y-component. (This is an example of a pure Dirichlet BC.)

To specify a node with no external force, you would select "Force, Force" for the type of boundary condition, and enter the number 0 for the value of the x-component and the value of the y-component. Recall that the GUI can assign a zero force BC for you if you leave the node without a BC specified. (This is an example of a pure Neumann BC.)

To specify a roller pin that is free to move vertically, select "Displacement, Force" for the type of boundary condition, and enter the number 0 for the value of the x-component and the value of the y-component. (This is an example of a mixed BC.)

To specify a roller pin that is free to move horizontally, select "Force, Displacement" for the type of boundary condition, and enter 0 for the value of the x-component and the value of the y-component. (This is also an example of a mixed BC.)

(Currently, the GUI does not support a roller pin on an inclined surface.)

To remove a boundary condition, please specify the following:

Index of the node	the index of the node where you want to remove the BC

1.4. Display options

You can indicate whether you want to see the node indices, the element indices, the arrows representing the BCs, and the grid on the display.

To do this, you can press Ctrl + 1 (2, 3, or 4), or go to the Tools menu and click on Display options, followed by Node index, etc.

1.5. Save your workspace, come back later

You can save your current work as a file (it is saved as a .mat file), and come back later to continue working on your structure.

To do this, you can press Ctrl + s, or go to the File menu and click on Save workspace. Please enter the name of the file, and select the directory under which you want to save the file. Click the Save button.

To open a workspace file, you can press Ctrl + o, or go to File menu and click on Open workspace. Please check that you select a workspace file, and not an assembly file (which have "_assembly" appended to the name of the file) or any other .mat file. The GUI does not check whether you have entered a valid .mat file.

1.6. Create the assembly file

Once you have specified the nodes, elements, and BCs that make up the structure, you can organize all the information needed for assembly by creating an assembly file. To do this, please go to the File menu and click on Export, followed by Assembly file. The GUI will then automatically create the assembly file in the same directory as your workspace file, and name it by appending "_assembly" to the workspace file name.

The assembly file is saved under two formats: one as a single .mat file, and the other as four .txt files. While we recommend using the .mat file with Matlab (you load the file with one statement, and the values are automatically read and stored in memory), you may use instead the .txt files. You will have to use the .txt files if you are using another language for assembly and postprocessing.

The .mat file contains 4 variables, named nodes, elements, BCs_displacement, and BCs_force. Each one is a double array (while technically not correct, you may think of it as a matrix), where the rows correspond to a node, an element, a displacement BC, or a force BC, and the columns correspond to certain information.

We list what the columns stand for here:

nodes	[x-coordinate, y-coordinate]
elements	[start node index, end node index, Young's modulus, cross-sectional area]
BCs_displacement	[node index, coordinate index, BC value]
BCs_force	[node index, coordinate index, BC value]

Note that we do not store the node indices in the nodes array, the element indices in the elements array, etc., because we assume that the indices are consecutive integers starting with 1. To find the number of nodes in the nodes array (and so on), we can use Matlab's size routine, which will return the dimensions of a double array. Please see the next page for the correct syntax.

The .txt files always begin with the number of lines (i.e. the number of nodes, etc.) that will follow afterwards, but contain otherwise the same information in the same order as shown above. The multiple values in a line are separated by a whitespace, i.e. a whitespace is the delimiter.

2. Some useful Matlab routines

When you write your assembly routine in Matlab, you may find these built-in routines useful. If you have an idea of what needs to be done but aren't sure how to write it in Matlab, reading Matlab's online documentation or looking for examples on Google are a great help.

load('str'), where str is a string (must be enclosed by single quotation marks as shown)

You must use this routine to load the .mat assembly file; the four variables are then read automatically, and can be used anywhere in your code following the load statement. str is the full path to the .mat assembly file, i.e. it is the combination (concatenation) of the directory path and the name of the assembly file. If you have the .mat assembly file in the same directory as your assembly routine, then entering just the name of the assembly file in place of str is enough.

Near line 552 in main.m is an example of the load routine (it loads the .mat workspace file); the additional arguments indicate which variables we want to load from the file. You may leave out these arguments if you want to read all variables in the file.

A(a, b), where A is a matrix, and a and b are column vectors (of positive integers)

This returns a submatrix of A, with the rows specified by the index vector a, and the columns specified by the index vector b. To get a submatrix with all the rows but some of the columns of A, we use a colon and type in A(:, b). To get a submatrix with all the columns but some of the rows of A, we would type in A(a, :).

find(expr), where expr is a logical expression involving matrices vectors, and/or scalars

This returns a vector of indices where the logical expression is a true statement.

rank(A), where A is a matrix

This returns the rank of A. Recall that, if A is a $n \times n$ square matrix appearing in a matrix equation $A\vec{x} = \vec{b}$ (assume that $\vec{b}$ is in the column space of A, i.e. a solution exists), then n minus the rank tells us the minimum number of constraints that we need to impose on the vector $\vec{x}$ if we want $\vec{x}$ to be a unique solution.

size(A, 1), where A is a matrix, vector, or scalar

By setting the second argument to be 1, the routine finds the row dimension of A.

sort(a), where a is a column vector

This returns a column vector of the same size as a, but with the entries sorted in ascending order. You can also use [b, permutation] = sort(a), which returns two vectors: b is the sorted vector, and permutation is a vector that indicates how the rows of a have been permuted (switched around) in order to get to b.