

General Structural Analysis Programs

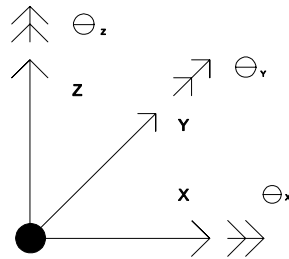
Objectives

- 1) Set-up the DOF for a structure
- 2) Follow the steps for the analysis of a structure
- 3) Develop the required load cases and combinations

General structural analysis programs perform exactly the same steps as outlined before.

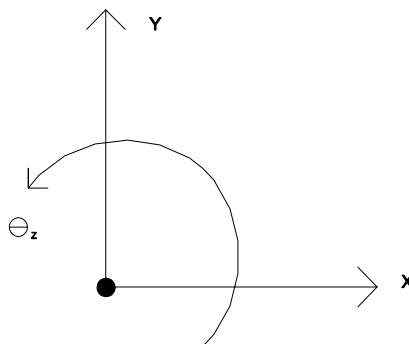
Set up the problem geometry.

The problem geometry consists of defining points in space usually called nodes. The coordinates define the location of a node. (X and Y in 2-D, X,Y and Z in 3-D). Each node is assigned boundary conditions that tell what directions the node is allowed to move. In 2-D, a node is allowed to move in the X and Y directions and rotate about the Z-axis (assuming X-Y plane). In 3-D, a node is allowed to move in the X, Y, and Z directions and rotate about all three axes. All nodes are generally assumed to have the six DOF shown below.



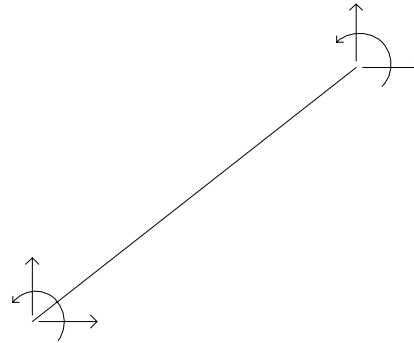
Six possible DOF for a node.

For a 2-D problem in the X-Y plane, you only need to consider the following DOF:



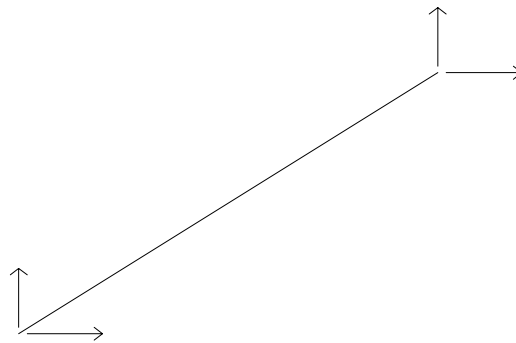
Three possible DOF in 2-D for a node.

When using a 3-D program to perform a 2-D analysis, it is mandatory that the additional DOF not required in 2-D be fixed (not allowed to move). This requirement is a specialized form of the rule that only the DOF that have stiffness can be allowed to move. For example, our 2-D frame element in a general X-Y configuration has the following DOF shown in Figure 6.3.



Actual DOF for general 2-D frame element.

Therefore, these are the only DOF to which the element can add stiffness in the global stiffness matrix. Since nodes in general 3-D programs are allowed to move and rotate in the X, Y and Z directions, we need to prevent that movement where no stiffness exists. The same concepts are true for truss elements. A general 2-D version of a truss element would have DOF as shown below.



Actual DOF for general 2-D truss element.

Form all element matrices.

Formation of all element matrices consists of forming the element stiffness in global directions (\mathbf{K}_{xy}), the force transformation matrix ($\mathbf{FT}_a = \mathbf{K}_e \mathbf{a}_a$), the LV vector (the pointer form of the \mathbf{a}_L matrix), and the member loads in global and local directions (\mathbf{S}^f and $\mathbf{a}_a^T \mathbf{S}^f$). This step requires the numbers of the nodes to which the element is connected in order to acquire all the information to form the element matrices. By knowing the nodes, the coordinates at the ends of the member can be found and the rotation angle and

member length calculated. The properties of the element and member load are the only additional information needed in order to form the required matrices.

Assemble the global stiffness and loads.

Now that all the element matrices have been formed, the pointer method of assembly can be used to form the global stiffness and load matrices. The load matrix has contributions from the element loads, which must also be assembled, and concentrated loads. It is at this point that the concentrated loads are usually added to the element contributions assembled into the global load vector.

Solve for the unknown displacements.

In order to solve for the unknown displacements, any type of equation solver can be used that solves simultaneous equations.

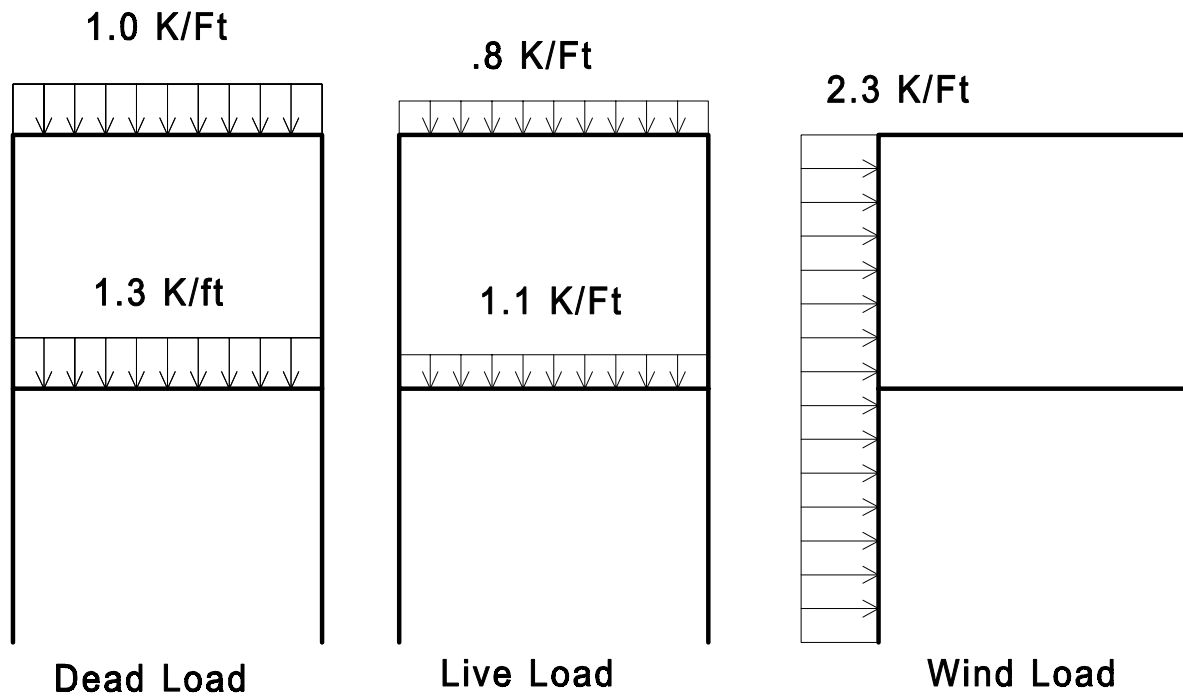
Recover the element forces.

Using the solved for displacements, we can recover the element forces. The LV pointer is again required to tell which DOF are connected to each element. Once the correct DOF are found, forces can be recovered using the force transformation matrix calculated earlier ($\mathbf{FT}_e = \mathbf{K}_e \mathbf{a}_e$) and the appropriate displacements. Again, the element load vector needs to be subtracted to correct the forces for element loadings (\mathbf{S}^f) for beam members.

Load Cases

There are two other concepts that occur when using most general analysis programs. The first is the definition of multiple load cases. In design, the building codes usually require that you check a structure for different combinations of loadings to determine that which combination is the most critical. For example, you might need to check dead plus live load, $3/4(\text{dead} + 1/2 \text{ live} + \text{wind})$ and so on. Since only the loading and not the structure changed you can solve for these multiple load cases simultaneously.

Most programs allow you to define multiple load cases, which consist of different sets of loads both concentrated and distributed. In addition, some programs allow for these basic load cases to be combined forming load combinations. For example, three load cases are defined below.



These three load cases could be combined in any fashion required to satisfy building code requirements. For example, you could combine load cases 1 and 2 (Dead + Live) to form a combination needed to perform a code design. Many programs allow you to specify combinations of these basic load cases.

