

# ASEN 2001 Lab Report: Computer Analysis of Structures

**Section 11, Team 2**

Davis Peirce

Abdulla Al Ameri

Daniel Gutiérrez Mendoza

### Theory Manual - 2D and 3D Equilibrium

In engineering, calculating the equilibrium of non-accelerating or constant velocity systems is essential to understand the loads and moments that system is undergoing. This is important because it allows us to think about our external loads and design our systems accordingly to withhold or behave in a certain way and provide strong enough supports to avoid failures. Therefore, for our system to be in static equilibrium, we want all of our net forces and net moments to be equal to zero. Supports on the system are responsible for having the system remaining static by exerting reaction forces and moments that interact with the applied forces and moments and cancel them out. This is what we call a system in equilibrium. For a system to be in static equilibrium, the **following conditions must be met**:

1. The sum of forces must equal 0, that is to say, no translation in any direction.

$$\Sigma F_{net} = 0$$

2. The sum of moments must equal 0, that is to say, no rotational effects in any direction. In 3D problems the sum of moments turns to be with respect to the axis of rotation, while in 2D the axis of rotation will always be the axis perpendicular to the plane in which the components of the force or moments are in.

$$\Sigma M_{net} = 0$$

3. Static equilibrium analysis assumes the the system is made out of rigid body. That is to assume the components cannot bend or deform.

The previous conditions imply that there are six equations in 3D equilibrium problem, since each force or moment can be broken down into 3 dimensional component. Thus, for a system to be determinant, i.e. there is a possible solution, the number of equations must be equal to the number of unknowns. The possible 6 equations are expressed below, where x,y,z represent direction when we sum the force, and represent axis of rotation when we sum the moment.

$$\Sigma F_x = 0$$

$$\Sigma F_y = 0$$

$$\Sigma F_z = 0$$

$$\Sigma M_x = 0$$

$$\Sigma M_y = 0$$

$$\Sigma M_z = 0$$

In this case of more equation than unknowns, the system will be over determinant. On the other hand, if there is more unknowns than equations, the system will be underestimated.

### Developer Manual

- **Matrix Theory**: After we have the six equations set up from, using the matrix theory will solve for the unknowns by matrix theory. The matrix theory is a ways of solving systems of equations via matrices. Given n number of equations as explained below, where (a) represent the coefficients, and (x) represent the unknowns. And then convert it into matrix form. Figure 1 and 2 explains the idea.

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\
&\vdots \\
a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n
\end{aligned}$$

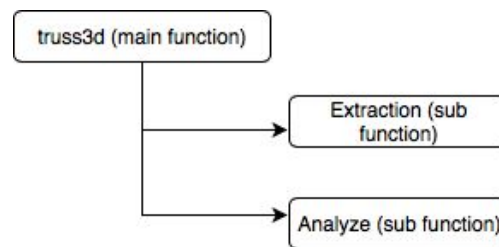
**Figure 1: The equations of equilibrium.**

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

**Figure 2: The matrices.**

There are several ways to solve a system of equations in MATLAB. We will use left division method.

- **Code Structure:** structural coding was used with one main function and two sub-functions explained below.



- **truss3d (function):** This function is the chassis of our code. truss3d will utilize two sub-functions, *Extraction*, and *Analyze*. Afterwards, truss3d will perform the matrix operations above and write out the solutions found with the problem setup. The inputs and outputs are commented inside the .m file with a brief description about the code. The code is very well commented for anyone to see the process.
- **Extraction (function):** Provided the text file with the necessary information, this function was specifically designed to extract all the raw information from the file. This function looks up first for comments and ignore them. Then look up for numerical values and problem information and stores them into a cell arrays. All this function do is convert the input text file into something MATLAB understands and can work with, and spits it out back to truss3d. There are some warnings and errors placed in Extraction deliberately to check for any mistakes that the user might do or any problems with the problem setup, this is explained in the code capabilities section. The inputs and outputs are commented inside the .m file with a brief description about the code. The code is very well commented for anyone to see the process.
- **Analyze (function) :** This function uses the information provided from *Extraction* as an input to do the math of setting up the equilibrium equation and prepare matrices A, x, b. There are some warnings and errors placed in Extraction deliberately to check for any mistakes that the user might do or any problems with the problem setup, this is explained in the code capabilities section. The inputs

and outputs are commented inside the .m file with a brief description about the code. The code is very well commented for anyone to see the process.

**Code Capabilities** : Provided a text file with the number of moments and forces applied, the magnitude, direction and point at which those vectors are applied, as well as the position of the supports and the direction of the reaction forces, the code should be capable of calculating the magnitudes of the reaction forces and moments. The code will also print the results to another text file and organize the magnitudes found to their corresponding support and direction.

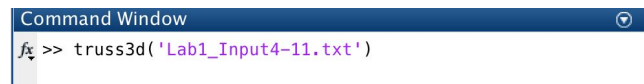
**Dynamic Naming**: The code is designed to rename the output files dynamically by given each output file the name of the input and follow it up with the word “\_Reactions” to provide dynamic naming so no overwriting is needed.

**Special Cases/common mistakes and how the code handles them**: All of these conditions are deliberately designed to avoid as many user mistakes as possible and widen the range of capabilities.

1. **Zero External Forces/moments**: The code will be able to run and output the expected outputs even if the external forces/moments are inputted as 0.
2. **Division by 0 and NaN**: The code will be aware of the cases where there will be division of 0 and NaN (Not-a-Number) in MATLAB and will correct those mistakes.
3. **Directions that are not unit vectors**: The analysis requires the direction to be expressed in unit vector form, and the code will double check that in case the user inputted the direction as non-normalized vector.
4. **Over/Under determinant matrices**: The code will check if you have different number of equation with respect to unknowns and will throw an error.
5. **If the user forgot a component of the force/moment**: The code will throw an error for the user to indicate a missing component of the force/moment.
6. **Warning if user has forgotten to input force or moment**: The code will throw an error and warning when user deletes the section of force or moment because they are 0.


### **How to call the function:**

How to call a the function from  
MATLAB command line.

A screenshot of the MATLAB Command Window. The title bar says "Command Window". The command prompt shows "fx >> truss3d('Lab1\_Input4-11.txt')".

```
Command Window
fx >> truss3d('Lab1_Input4-11.txt')
```

How the output txt file will look like.

A small icon representing a text file.

Lab1\_Input4-11\_Ractions.txt