Final assignment modpro 21-22

Jos Havinga, edited by B. Bets, D. de Jong, T. Slangen

October 2021

Course: ModPro Semester 1
Date: 15th of October - 29th of October
Hand in before: October 29, 23:59

- You may work in pairs
- Teacher assistants are available to support you!
- You are not allowed to copy any code that was not written by yourself in your solution, unless stated in the assignment.
- You are allowed to consult any resource (course exercises and solutions, books, the internet) related to Matlab and Statics.
- Hand in your assignment via Canvas!
  - You will pass the case by finishing the assignments described below.
  - You can receive 0.5 points for the block 1 statics grade by finishing assignment one.
  - You will receive an additional 0.4 points for finishing assignment two.
  - You can receive the last 0.1 point for handing in a complete set of files, and commenting in your files about how your code works. The files should be in a .zip file, and when extracted, they should work on their own without modifications by the examinators.
- After finishing you must:
  - Hand in all of the scripts and functions you used in a .zip file by canvas.
  - Please hand in everything that is required to run your file properly since the ModPro staff is not going to search or ask for missing files.

Assignment 1 - Statically determinate truss system

In this assignment we are going to write a script for calculating the unknown forces in a statically determinate truss system. The truss system that will be used to write and test the script can be seen in Figure 1. Note that once the script contains no errors and works as intended, we can practically think of any truss system and calculate its unknowns, as long as the system is statically determinate.

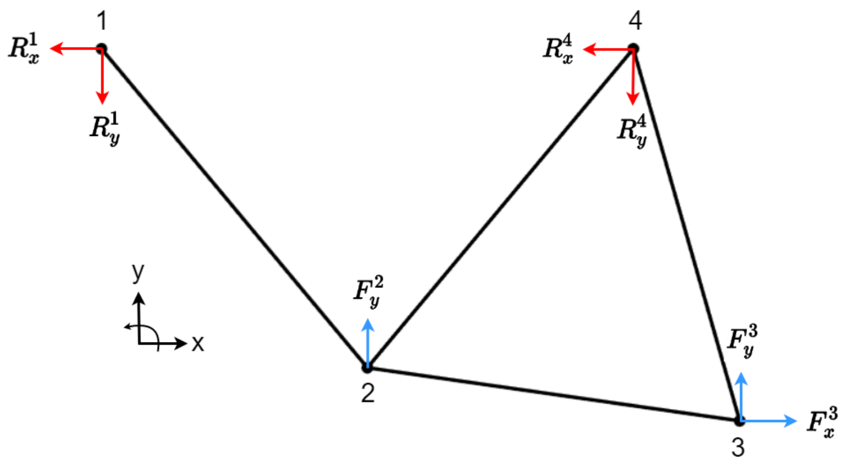


Figure 1: Schematic overview of the truss system. Reaction forces are shown in red and external forces are shown in blue.

The coordinates of the joints, the magnitudes of the external forces and the numbering of the trusses of this system can be found in Table 1.

Joint	Coordinates [m]
1	(0 , 0)
2	(0.5 , -0.6)
3	(1.2 , -0.7)
4	(1 , 0)

External force	Magnitude [N]
F_y^2	-100
F_x^3	-50
F_y^3	30

Truss number	Joints
1	1 - 2
2	2 - 4
3	2 - 3
4	3 - 4

Table 1: Properties that define the truss system.

The method of joints will be used to calculate the unknowns, which are the truss forces F^{1-2}, F^{2-4}, F^{2-3} and F^{3-4}, and the reaction forces R_x^1, R_y^1, R_x^4 and R_y^4. At each joint, equilibrium equations in x- and y-direction have to be determined. For example, the equilibrium equation in x-direction at joint 3 is:

$$-F_x^{2-3} - F_x^{3-4} + F_x^3 = 0 \tag{1}$$

This can be rewritten to:

$$0 * F^{1-2} + 0 * F_x^{2-4} - F_x^{2-3} - F_x^{3-4} + 0 * R_x^1 + 0 * R_y^1 + 0 * R_x^4 + 0 * R_y^4 = -F_x^3 \tag{2}$$

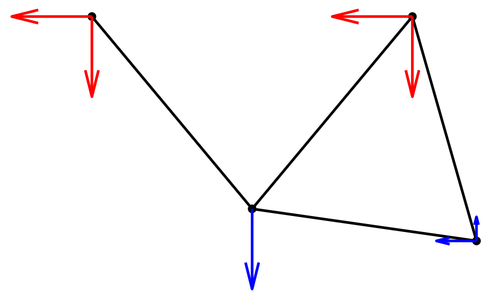
Where F_x^{i-j} is the x-component of truss force F^{i-j}. To recap how to decompose truss forces and how to apply the joint method, read Section 2 (Statically determinate truss system) from the explanation file on Canvas (ModPro_Mod1_S5_Reader.pdf). After decomposing, the equation becomes:

$$0 * F^{1-2} + 0 * F^{2-4} - \frac{x_3 - x_2}{L_{2-3}} F^{2-3} + \frac{x_4 - x_3}{L_{3-4}} F^{3-4} + 0 * R_x^1 + 0 * R_y^1 + 0 * R_x^4 + 0 * R_y^4 = -F_x^3 \quad (3)$$

i) Start with creating two script files: a main file (to perform all calculations) and a separate file for the truss definition. The truss is defined in a separate file, because then it will be easier to evaluate different truss systems with the same script, just by running the frame definition script of choice before performing the calculations. In the main file, add the command to run the truss initialization file by adding the file name as a command (`Mod1_Ass1_simpleTruss` if your definition file is called Mod1_Ass1_simpleTruss.m). In the frame definition script, define the following variables (see Table 1):

   a) `jointCoordinates`: a 4 by 2 matrix, with the x-coordinates in the first column and the y-coordinates in the second column.
   b) `trusses`: a 4 by 2 matrix, with each row containing the joint numbers that are connected by each truss.
   c) `supportReactions`: a 4 by 2 matrix. Each row contains one reaction force, with the first column representing the joint number, and the second column the direction of the reaction force (1 for x-direction and 2 for y-direction).
   d) `externalForces`: a 3 by 3 matrix. Each row contains one external force, with the first column representing the joint number, the second column the direction of the external force (1 for x-direction and 2 for y-direction) and the third column the magnitude of the external force.

   hint: Check whether the frame was defined correctly by running the `Mod1_Ass1_plotFrame` *script (available on Canvas). The result should look like:*



ii) Continue working in the main file. Check whether a statically determinate truss system was defined. Use the `size()` function to determine the number of joints (assign to variable `nJoints`), of trusses (assign to `nTrusses`) and of reaction forces (assign to `nSupports`). Determine the number of unknowns (`numberOfUnknowns`) and of equations (`numberOfEquations`). If the number of equations is unequal to the number of unknowns, throw an error using the `error()` function (`error('Defined truss system is not statically determinate')`).

hint: Use an `if end` *sequence to throw the error conditionally.*

iii) Initialize the matrix `A` with equations using the `zeros()` function. Use the variables `numberOfEquations` and `numberOfUnknowns` to define the size of matrix `A`. Rows 1 to `nJoints` will be used for the equilibrium equation in x-direction, and rows (`nJoints` + 1) to (`2 * nJoints`) will be used for the equilibrium equations in y-direction. Columns 1 to `nTrusses` belong to the unknown truss forces, and columns `(nTrusses + 1)` to `nTrusses + nSupports` belong to the unknown reaction forces.

iv) Create a `for` loop with index `iTruss` that runs over all trusses (`for iTruss = 1:nTrusses ... end`). In the loop, the contribution of each truss force to all equilibrium equations will be determined. To do so, perform the following actions:

a) Use the `trusses` variable to determine which joints are connected to the current truss. Store these indexes in `iJoint` and `jJoint`.

b) Calculate the `deltaX` and `deltaY` of the truss ($\Delta x = x_j - x_i$ and $\Delta y = y_j - y_i$) using the variables `jointCoordinates`, `iJoint` and `jJoint`.

c) Calculate the length of the truss and store the length in the variable `L`.

d) Now, assign the contribution of each truss force to the correct equation in matrix `A`.

hint: Remind that each truss has a contribution to four different equations. See the explanation file (ModPro_Mod1_S5_Reader.pdf) *to check how to write these contributions as function of* `deltaX`, `deltaY` *and* `L`.

hint: Think about which equation (row of matrix `A`*) and which unknown (column of matrix* `A`*) corresponds to each contribution. For targeting the correct position in matrix* `A`*, the variables* `iJoint`, `jJoint`, `iTruss` *and* `nJoints` *are needed.*

note: After running the for loop, matrix `A` *should look like:*

```
A =

    0.6402         0         0         0         0         0         0         0
   -0.6402    0.6402    0.9899         0         0         0         0         0
         0         0   -0.9899   -0.2747         0         0         0         0
         0   -0.6402         0    0.2747         0         0         0         0
   -0.7682         0         0         0         0         0         0         0
    0.7682    0.7682   -0.1414         0         0         0         0         0
         0         0    0.1414    0.9615         0         0         0         0
         0   -0.7682         0   -0.9615         0         0         0         0
```

v) Create a `for` loop with index `iSupport` that runs over all reaction forces. In the loop, the contribution of each reaction force to all equilibrium equations will be determined. To do so, perform the following actions:

a) Use the `supportReactions` variable to determine at which joint the reaction force acts. Store this index in `iJoint`.

b) Create an `if else end` sequence to check whether the reaction force acts in x- or y-direction. Check the value in the second column of `supportReactions` to check the direction of the support reaction (a value of 1 represents a reaction force in x-direction, a value of 2 represents a reaction force in y-direction).

c) Assign the value -1 (we have defined positive reaction forces to point in negative direction, see Figure 1) to the correct positions in matrix `A`.

hint: For targeting the correct position in matrix `A`*, the variables* `iJoint`, `iSupport`, `nJoints` *and* `nTrusses` *must be used.*

hint: Remind that the rows 1 to `nJoints` *represent the equilibrium equations in x-direction, and that the rows* `(nJoints + 1)` *to* `(2*nJoints)` *represent the equilibrium equations in y-direction.*

hint: Remind that the first `nTrusses` *columns are used for the truss forces, and that columns* `nTrusses + 1` *to* `nTrusses + nSupports` *are used for the reaction forces.*

note: After adding the reaction forces, matrix `A` *should look like:*

```
A =

    0.6402         0         0         0   -1.0000         0         0         0
   -0.6402    0.6402    0.9899         0         0         0         0         0
         0         0   -0.9899   -0.2747         0         0         0         0
         0   -0.6402         0    0.2747         0         0   -1.0000         0
   -0.7682         0         0         0         0   -1.0000         0         0
    0.7682    0.7682   -0.1414         0         0         0         0         0
         0         0    0.1414    0.9615         0         0         0         0
         0   -0.7682         0   -0.9615         0         0         0   -1.0000
```

vi) The last step is to add all known external forces to vector `b`. First, initialize the column vector `b` using the `zeros()` function (what is the length of `b`?). Then, create a `for` loop with index `iExtForce` that runs over all external forces. In order to assign each external force to the right position in `b`, perform the following actions:

a) Use the `externalForces` variable to determine at which joint the external force acts. Store this index in `iJoint`.

b) Create an `if else end` sequence to check whether the external force acts in x- or y-direction. Check the value in the second column of `externalForces` to check the direction of the support reaction (a value of 1 represents an external force in x-direction, a value of 2 represents an external force in y-direction).

c) Assign minus the magnitude of the external force (third column of `b`) to the correct position in matrix `b`.

note: See the explanation file (ModPro_Mod1_S5_Reader.pdf) *to understand why the reaction force must be added to* `b` *with a minus sign.*

hint: For targeting the correct position in vector `b`*, the variables* `iJoint` *and* `nJoints` *must be used.*

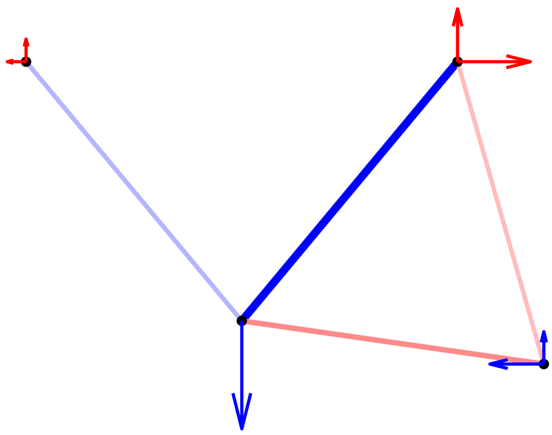
hint: Remind that the positions 1 to `nJoints` *represent the equilibrium equations in x-direction, and that the rows* `(nJoints + 1)` *to* `(2*nJoints)` *represent the equilibrium equations in y-direction.*

vii) Now it is time to solve the set of linear equations (perform a check: have all forces on all joints been added to matrix `A`? Have all external forces been added to vector `b`?). The truss forces and reaction forces can be determined with `Finternal = A\b`. What is the meaning of the entries of the result vector `Finternal`? The final result should be:

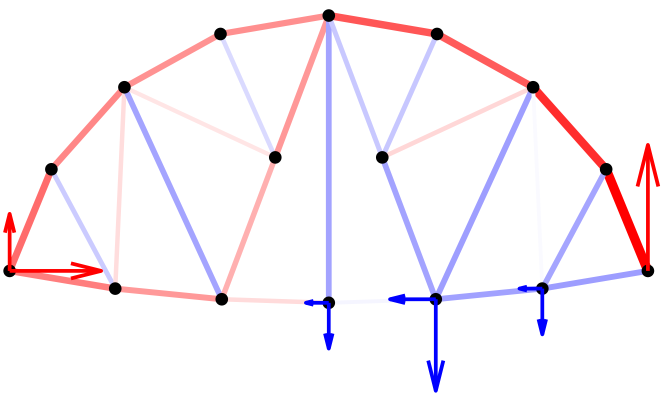
```
Finternal =

   27.3359
   94.8031
  -43.6300
  -24.7834
   17.5000
  -21.0000
  -67.5000
  -49.0000
```

note: Plot the frame again using the script `Mod1_Ass1_plotFrame`*. Now, the forces will be visible in the plot. Red colors represent compressive forces, and blue colors represent tensile forces. The plot should look like:*



viii) Finally, you can test your script with a different frame. Download the definition of a bridge frame from Canvas (Mod1_Ass1_bridge.m), and run the your code with the bridge definition instead of the simple frame. After plotting, the final result should look like:



Assignment 2 - A truss system in 3 dimensions

In this second assignment we are going to write a script for calculating the unknown forces in a statically determinate truss system in three dimensions. The truss system that will be used to write and test the script can be seen in Figure 2.
For this second assignment you need to download the following file from canvas: `Mod1_Ass_2_plotFrame3D.m`

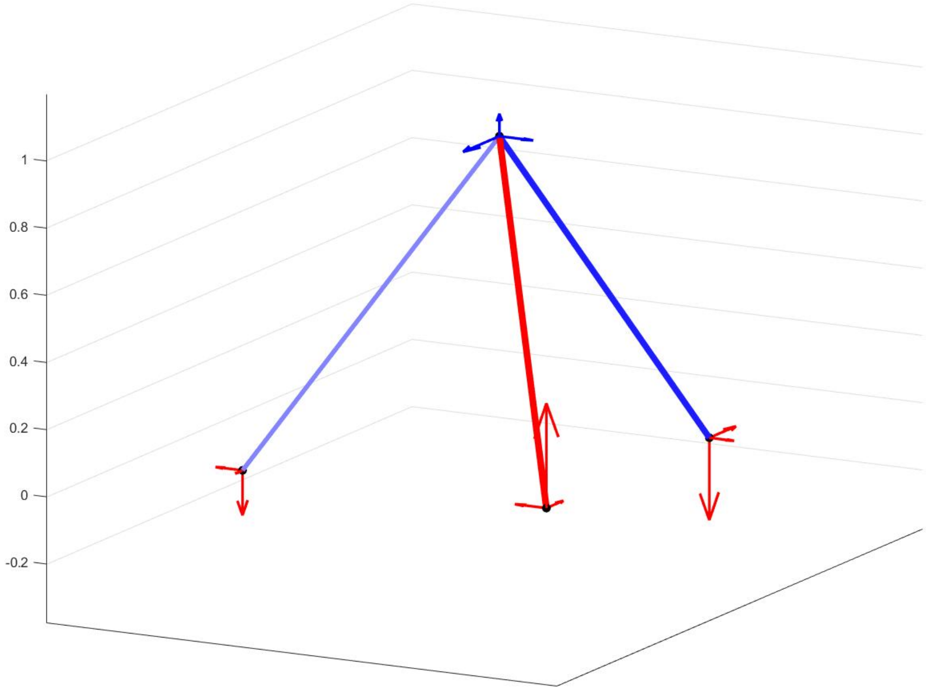


Figure 2: Schematic overview of the truss system. Reaction forces are shown in red and external forces are shown in blue.

The coordinates of the joints, the magnitudes of the external forces and the numbering of the trusses of this three dimensional system can be found in Table 2.

Joint	Coordinates [m]
1	(0 , 0 , 0)
2	(1 , 0 , 0)
3	(0 , 1 , 0)
4	(1/3 , 1/3 , 1)

External force	Magnitude [N]
F_x^4	-100
F_y^4	-50
F_z^4	30

Truss number	Joints
1	1 - 4
2	2 - 4
3	3 - 4

Table 2: Properties that define the truss system.

Just like in the first assignment, we will try to model the solving file such that it can also work in three dimensions. The three dimensions are as follows [x, y, z]. You should use your files from Assignment 1, and edit those. This will save you a lot of time. Be sure however to save Assignment 1 separately so you can hand that in Assignment 1 and 2 separately. **Do not only overwrite your assignment 1.**

i) a) `jointCoordinates`: a 4 by 3 matrix, with the x-coordinates in the first column, y-coordinates in the second column and the z-coordinates in the third column

   b) `trusses`: a 3 by 2 matrix, with each row containing the joint numbers that are connected by each truss. in this case all of the joints are connected to joint 4. *See table 2*

   c) `supportReactions`: a 9 by 2 matrix. Each row contains one reaction force, with the first column representing the joint number, and the second column the direction of the reaction force (1 for x-direction, 2 for y-direction and 3 for z-direction).

   d) `externalForces`: a 3 by 3 matrix. Each row contains one external force, with the first column representing the joint number, the second column the direction of the external force (1 for x-direction, 2 for y-direction and 3 for z-direction) and the third column the magnitude of the external force.

   hint: Check whether the frame was defined correctly by running the `Mod1_Ass2_plotFrame3D.m` *script (available on Canvas). The result should look like figure 2.*

ii) For three dimensional trusses, the rule to check if it is statically determinate changes from 2*nJoints to 3*nJoints. If this is not the case, give an error messsage.

iii) Initialize the A matrix with equations using the zeros() function. Think of a way to expand filling in the A matrix of 2-dimensional trusses to 3-dimensional trusses.

iv) create a `for` loop with index `iTruss` just like assignment 1, but make sure it also calculates the force scalar in Z direction, and gives a logical place in the A matrix. Make sure it has the correct sign.

v) create a `for` loop with index `iSupport` that runs over all reaction forces. Make sure they also check in Z - direction.

vi) Now, for the external force vector `b`, also make sure this takes into account the Z-direction.

vii) Now finally solve the A matrix and check if your answer is correct. It should look like figure 2. You can call the `Mod1_Ass_2_plotFrame3D.m` function to see the final 3D plot.

The `Finternal` should look like:

```
-154.7758
 137.1941
  74.8331
  46.6667
  46.6667
 140.0000
  73.3333
 -36.6667
-110.0000
 -20.0000
  40.0000
 -60.0000
```

You now finished this assignment, please upload your files to canvas as a .zip folder. Thank you for completing the assignments!