OpenStruc: Static Analysis of Two- and Three-dimensional Structures

Version 1.0

User Guide

Edgar F. Black[*]

DRAFT: August 4, 2025

[*]`edgarfblack@gmail.com`

1 Introduction

Welcome to ***OpenStruc***, and thank you for using it! ***OpenStruc*** is a system for the static analysis of two- and three-dimensional structures. ***OpenStruc*** can be used to analyze several types of structures: plane trusses and frames, space trusses and frames, and grids – plane structures with loads applied orthogonally to their plane. The objective of this report is not to comprehensively cover all the commands included in ***OpenStruc*** but to provide a quick review of basic ones that will allow you to analyze structures quickly.

OpenStruc is written entirely in C++. At its core, ***OpenStruc*** uses a direct solver algorithm based on Gauss Elimination, including a variation of the *Active Column Solution* method, as described by Bathe[1].

The most straightforward way to use ***OpenStruc***, although not the only one, is to write the commands required to analyze a given structure in a script file and feed that file into ***OpenStruc***.

In what follows, we will introduce some basic ***OpenStruc*** commands using five examples, each corresponding to one of the types of structures that ***OpenStruc*** is capable of analyzing.

Running Examples

The most common mode of running ***OpenStruc*** is by reading its commands from an input script file. For example, to execute ***OpenStruc*** from an input script file called *myExample.in*, execute the following command at the prompt:

```
$ OpenStruc myExample.in
```

As a result of executing the command shown above, the output will go to the stdout device, usually the computer screen. If instead of the screen you want to send the output to a file, execute the following command:

```
$ OpenStruc myExample.in myExample.out
```

In this case, the output goes to a file called *myExample.out*.

As a final note, be aware that ***OpenStruc*** does not allow reading its input or writing its output by redirection.

Scripted Examples

The examples selected to introduce ***OpenStruc's*** scripting are:

- **Analysis of a plane truss**.
- **Analysis of a Space Truss**.
- **Analysis of a Plane Frame**.
- **Analysis of a Space Frame**.
- **Analysis of a Grid**.

2 Plane Truss Analysis

This example describes the data preparation for the static analysis of a plane truss structure. This plane truss corresponds to an example presented in Chapter 5, pages 330-331, of the book by Weaver and Gere[2].

Figure 1 shows a plane truss having four nodes and six elements. The truss is restrained at its two lower nodes by hinge supports that prevent translations in both the x and y directions. The area of the horizontal elements is 6 square inches, the vertical ones 8 square inches, and the diagonals 10 square inches.

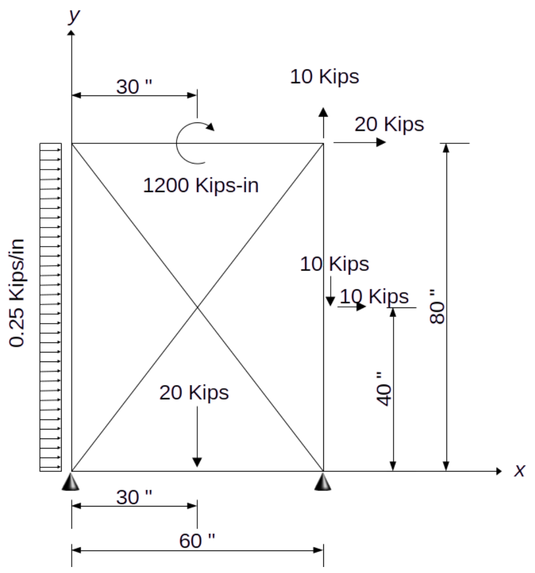

Figure 1: Plane Truss

OpenStruc Script for the Plane Truss

Figure 2 presents a script containing the set of ***OpenStruc***'s commands used to model the truss shown in Figure 1.

Line 1 in Figure 2 contains the **`Structure`** command. All ***OpenStruc*** scripts must begin with this command. Anything following the **`Structure`** command up to the end of the line is considered the problem's title. The problem's title is optional.

Line 2 illustrates the use of comments inside ***OpenStruc*** scripts. Everything following the '#' character is considered a comment. Therefore, the entire content of line 2 is a comment and does not affect the analysis.

Although in this case it has been commented out, line 2 contains the **`echo`** command, which is used as a binary '`on` | `off`' flag. When turned `on`, which is its default value, all the remaining script's commands go to the output device (file or stdout). The principal use of the **`echo`** command is for debugging purposes while developing the input script.

In line 3, the **`type`** command defines the structural type for the problem at hand. This command optionally accepts the **`sections`** sub-command, which takes an integer parameter. By default, ***OpenStruc*** solves the resulting forces and moments only at the ends of each element of a given problem. The **`sections`** sub-command instructs ***OpenStruc*** to logically divide each element into as many segments as the integer parameter, and to solve the resulting forces and moments at the end of each segment.

Line 4 contains the **`number`** command followed by sub-commands: **`nodes`**, **`elements`**, **`materials`**, **`Section_Types`**, and **`Load_Cases`**, each used to define internal ***OpenStruc*** variables with the same name. Alternatively, each of these sub-commands can be issued individually, in a separate line, after the **`number`** command.

In line 5, the **`coordinates`** command begins a sequence that allows you to define the coordinates of all the nodes. Before issuing the **`coordinates`** command, it is necessary to set the number of nodes. The **`support`** sub-command, shown in lines 8 and 9, can follow the definition of a node's coordinates and is used to constrain particular degrees of freedom of the node.

In line 10, the **`incidences`** command begins a sequence that defines the elements of the structure, based on the nodes it connects. Before issuing this command, it is necessary to set the number of elements.

The **`materials`** command in line 17 starts the definition of the material types used for the analysis. Their input order identifies each material type. Notice that at least one material **must** be defined. As shown in line 18, four parameters define each material: the elasticity modulus (**`e`**), the shear modulus or modulus of rigidity (**`g`**), the coefficient of thermal expansion (**`ct`**), and the weight (**`weight`**).

In line 19, the **`SecType`** command allows defining different section types. The input order identifies each section type. In general, the material type **`mt`**, the area **`ar`**, the second polar moment of area (a.k.a. polar moment of inertia) **`Ix`**, and the second moment of area **`Iy`** and **`Iz`** fully define a section type. Depending on the type of structure, some of these parameters could remain undefined. If the material type is missing, ***OpenStruc*** uses material type 1 (mt 1). For the plane truss in this example, each section type only requires defining its element's cross-sectional area, as shown in lines 20 to 22.

In line 23, the **`elements`** command initiates a sequence that enables you to associate a list of elements with a specific section type. The list of elements can be a plain sequence of numbers, as shown in line 24, or a range, as shown in two different ways in lines 25 and 26.

By the end of line 26, we have entered all the necessary information to describe this plane truss, and the definition of the load cases can begin. However, it is always a good idea to perform a visual confirmation of the correct location of nodes and the elements' incidences. The **`plot`** command, shown in line 27, serves this purpose.

As shown in Figure 1, the loads for this example consist of one loaded node (upper-right node) and four loaded elements. All these loads are grouped into a single loading case, starting at line 28, using the **`loading case`** command followed by a parameter to indicate loading case number and by an optional loading case title.

In line 30, the **`loaded`** command, followed by the **`nodes`** sub-command, precedes the prescription of nodal loads. Following this line, there are as many new lines as loaded nodes

```
Structure Weaver & Gere book, Chapter 5, Plane Truss Example, pages 330−331
#echo off
type plane truss # sections 4
number of nodes 4 element 6 materials 1 Section_Types 3 load_Cases 1
coordinates
    1  0.0 80.0
    2 60.0 80.0
    3  0.0  0.0 support dx dy
    4 60.0  0.0 support dx dy
incidences
    1 1 2
    2 3 4
    3 3 1
    4 4 2
    5 1 4
    6 3 2
materials
   e 10000 g 1.0 ct 6.5e−6 weight 0.2836  # this is material 1
SecType
    ar  6.0                               # this is SecType 1, implicitly using mt 1
    ar  8.0                               # this is SecType 2, implicitly using mt 1
    ar 10.0 mt 1                          # this is SecType 3, explicitly using mt 1
elements
    1 2         SecType 1                 # Elements 1 and 2 use SecType 1
    3−4         SecType 2                 # Elements 3 and 4 use SecType 2
    5 to 6 by 1 SecType 3                 # Elements 5 and 6 use SecType 3
plot
loading case 1 "First Load Case"

    loaded nodes
    2 fx 20.0  fy 10.0

    loaded elements
    1  # parameters related to local temperature changes could follow here.
    couple local    mz −1200.0 la 30.0
    #couple local    mz −1200.0 la −0.5

    2  # parameters related to local temperature changes could follow here.
    concentrated local   py −20.0  la 30.0
    #concentrated global  py −20.0  la −0.5

    3  # parameters related to local temperature changes could follow here.
    uniform local wy −0.25 la 0.0 lb  −1.0
    #uniform global wx 0.25

    4  # parameters related to local temperature changes could follow here.
    concentrated local   px −10.0 py −10.0  la 40.0
    #concentrated global px 10.0 py −10.0  la −0.5

    solve
stop
```

Figure 2: Plane Truss Script.

(one in this case), each containing the node number, the type and direction of the applied loads (e.g., fx for force in x), and their magnitude.

In line 33, the **`loaded`** command, followed by the **`elements`** sub-command, starts the prescription of element loads. Then, for each loaded element, use a line to set the element number. Be aware that some parameters related to local temperature changes, but not applicable to the present example, can also be set in this line. Then, for each load type applied to the element, a new line contains the load type (concentrated, uniform, couple, etc.), the coordinate system used (local, global), the type and direction of the load (e.g., mz for moment in z), and parameters describing the position of the load (la, lb, etc.).

To demonstrate the different input options available to the user for prescribing the loads acting in elements, Figure 2 presents, for each of the four loaded elements, two equivalent versions used to define the same load. Notice that the second version is a comment. Feel free to try each version.

For example, in line 35, **`la`** prescribes the absolute distance of the applied load, measured from the initial end of the element. On the other hand, in line 36, the parameter **`la`** contains a negative number. This negative number indicates that the applied load is at a relative (to the length) distance, measured from the initial end of the element. Both lines, 35 and 36, are equivalent.

Notice how the **`global`** versus **`local`** parameters affect the input of the data. While the **`local`** parameter refers to the element's system of coordinates, the **`global`** parameter refers to the structure's system of coordinates. Look, for example, at the equivalent lines 43 and 44. While line 43 prescribes a **`local`** uniform load, using **`wy`** -0.25, line 44 prescribes the same load as a **`global`** load, this time using **`wx`** 0.25. Also, notice the meaning of **`la`** and **`lb`** for the **`uniform`** load case. While the **`la`** parameter means the distance (absolute or relative) measured from the initial end of the element to the beginning of the load, the **`lb`** parameter prescribes the (absolute or relative) length of the uniform load. In the frequent case that the uniform load acts along the whole length, both **`la`** and **`lb`** can be omitted, as shown in line 44.

In line 47 (and its equivalent line 48), we can see that several loads can be specified simultaneously, as long as they are at the same point. If an element has more than one load applied at different locations, you must use a new line for each load.

Once you finish entering the data corresponding to a load case, it can be solved using the **`solve`** command, as shown in line 50.

Finally, use the **`stop`** command, as shown in line 51, to end the analysis.

3 Plane Frame Analysis

This example describes the data preparation for the static analysis of a plane frame structure. This plane frame corresponds to an example presented in Chapter 5, pages 337-339, of the book by Weaver and Gere[2].

Figure 3 shows a plane frame having four nodes and three elements. The lower-left node is a fixed-support, while the lower-right node is a hinge support that prevents translations in both the x and y directions. The three elements have an area, ***ar***, equal to 0.02 m^2, and a second moment of area, ***iz***, equal to 0.003 m^4. The elasticity modulus (e) is 70x10^6 kN/m^2.

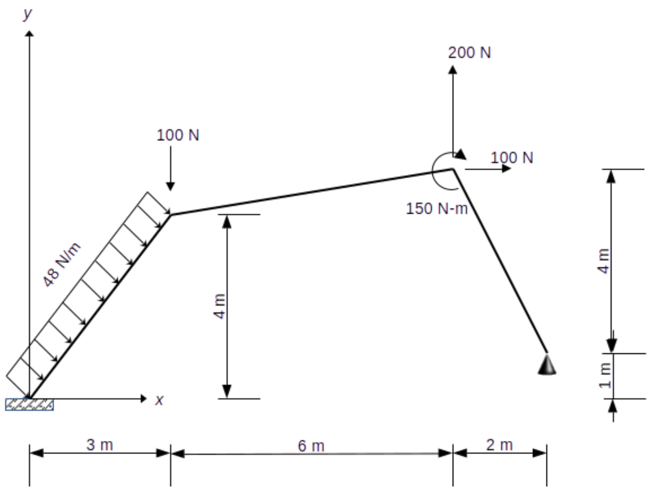

Figure 3: Plane Frame

OpenStruc Script for the Plane Frame

Figure 4 presents a script containing the set of ***OpenStruc***'s commands used to model the frame shown in Figure 3. Notice that, in what follows, we will describe *only* commands and parameters not covered in the Plane Truss example.

The **`type`** command in line 3 defines this problem as a *plane frame*.

Line 5 introduces the ***shear deformation*** command. Similar to the ***echo*** command, this command works as a binary '`on` | `off`' flag. Turning it on, instruct ***OpenStruc*** to include shear deformation in the calculations. Its default value is off.

Following the issue of the ***coordinates*** command in line 8, a sequence to define the coordinates of nodes and their support condition begins. In line 9, after defining the coordinates of node 1, notice how we modeled the fixed-support condition using the **`support`** sub-command to constrain two orthogonal linear displacements, and the rotational displacement utilizing the **`dx`**, **`dy`**, and **`rz`** parameters.

In line 38, notice the use of the **`mz`** parameter to apply the 150 N-m moment to the node.

Finally, it is worth mentioning the two equivalent ways of applying the distributed load to element 1, shown in lines 42 and 43. Can you verify that they are equivalents?

```
Structure Weaver & Gere book, Chapter 5, Plane Frame Example, pages 337−339
echo off
type plane frame  sections 4

shear deformation off

number of nodes 4
coordinates
    1   0.0      0.0      support dx dy rz
    2   3.0      4.0
    3   9.0      5.0
    4  11.0      1.0      support dx dy

number of element 3
incidences
    1 1 2
    2 2 3
    3 3 4

number of materials 1
materials
    e 70e6 g 1.0  ct  2.0  weight 3.0

number of section_Types 1
sectype
    ar 0.02  Iz  0.003  mt 1

elements
    1−3 secType 1

plot

number of load_Cases 1
loading case 1 "First Load Case"

    loaded nodes
    2 fy  −100.0
    3 fx  100.0  fy  200.0  mz  −150.0

    loaded elements
    1   # parameters related to local  temperature changes could follow here.
    uniform local   wy −48.0
    #uniform global  wy −28.8 wx 38.4
    solve
stop
```

Figure 4: Plane Frame Script.

4 Grid Analysis

This example describes the data preparation for the static analysis of a grid structure. A grid structure resembles a plane frame in several ways. Every node and element in both types of structures is contained in the same plane. Additionally, the elements and nodes are considered rigidly connected. The main difference between a plane frame and a grid is that in a plane frame, all forces lie in the frame's plane, whereas the forces applied to a grid are orthogonal to its plane. The grid structure discussed here corresponds to an example presented in Chapter 5, pages 342-345, of the book by Weaver and Gere[2].

Figure 5 shows a grid structure having six nodes and five elements. The element connecting nodes 1 and 2 is six meters long. All other elements are 4 meters long. Nodes 1, 3, 4 and 6 are of the fixed-support type, restraining the rotation in the x and y directions, as well as the displacement in the z direction. Nodes 2 and 5 are totally unconstrained.

All five elements have a second polar moment of area (a.k.a. polar moment of inertia) ***ix***, equal to 0.002 m^4, and second moment of area, ***iy***, equal to 0.001 m^4. The elasticity modulus (e), and the shear modulus or modulus of rigidity (g) are 200x10^6 and 80x10^6 kN/m^2 respectively.

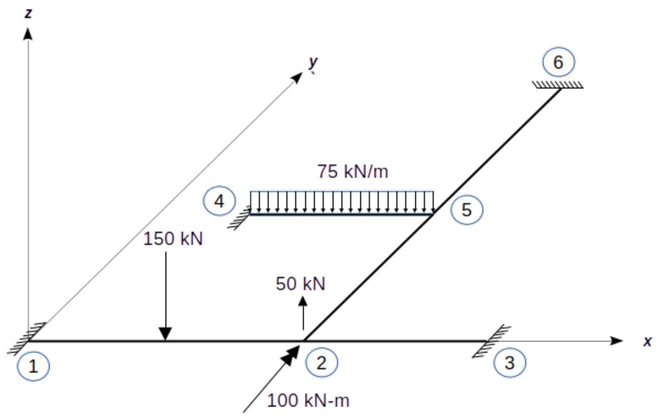

Figure 5: Grid

OpenStruc Script for the Grid Structure

Figure 6 presents a script containing the set of ***OpenStruc***'s commands used to model the grid shown in Figure 5. Once again, we will describe *only* commands and parameters not covered in the previous examples.

In lines 7, 9, 10, and 12, after defining the coordinates of nodes, notice how we modeled the fixed-support condition using the **`support`** sub-command. In this case, the fixed-support condition requires constraining two orthogonal rotational displacements and one linear displacement using the **`rx`**, **`ry`**, and **`dz`** parameters.

```
 1 Structure Weaver & Gere book, Chapter 5, Grid Example 2, pages 342−345
 2 echo off
 3 type grid # sections 4
 4 shear deformation off
 5 number of nodes 6
 6 coordinates
 7     1    0.0  0.0 support dz rx ry
 8     2    6.0  0.0
 9     3   10.0  0.0 support dz rx ry
10     4   2.0   4.0 support dz rx ry
11     5   6.0   4.0
12     6   6.0   8.0 support dz rx ry
13
14 number of element 5
15 incidences
16     1 1 2
17     2 2 3
18     3 2 5
19     4 4 5
20     5 5 6
21
22 number of materials 1
23 materials
24    e 200.0e6  g 80.0e6 ct 2.0 weight 3.0
25
26 number of section_Types 1
27 sectype
28     Ix 2.0e−3 Iy  1.0e−3 mt 1
29
30 elements
31     all secType 1
32     #1−5 secType 1
33
34 plot
35
36 number of load_Cases 1
37 loading case 1 "First Load Case"
38     loaded nodes
39     2 fz 50.0  my 100.0
40     loaded elements
41     1 # parameters related to local temperature changes could follow here.
42     concentrated local  pz  −150.0 la −0.5
43     4 # parameters related to local temperature changes could follow here.
44     uniform local  wz −75.0
45     solve
46 stop
```

Figure 6: Grid Script.

Line 31 shows a way of associating *all* the elements with a specific section type, using the **`all`** command. Notice that line 32 shows an alternative way of achieving the same using a list range.

5 Space Truss Analysis

This example describes the data preparation for the static analysis of a space truss structure. This space truss corresponds to an example presented in Chapter 5, pages 345-347, of the book by Weaver and Gere[2].

Figure 7 shows a space truss having five nodes and seven elements. The truss is restrained at three of its five nodes by hinge supports that prevent translations in all three x, y, and z directions. The area of all the elements is 10 square inches. The elasticity modulus (e) and the shear modulus or modulus of rigidity (g) are 30000 and 11538.4 ksi, respectively.

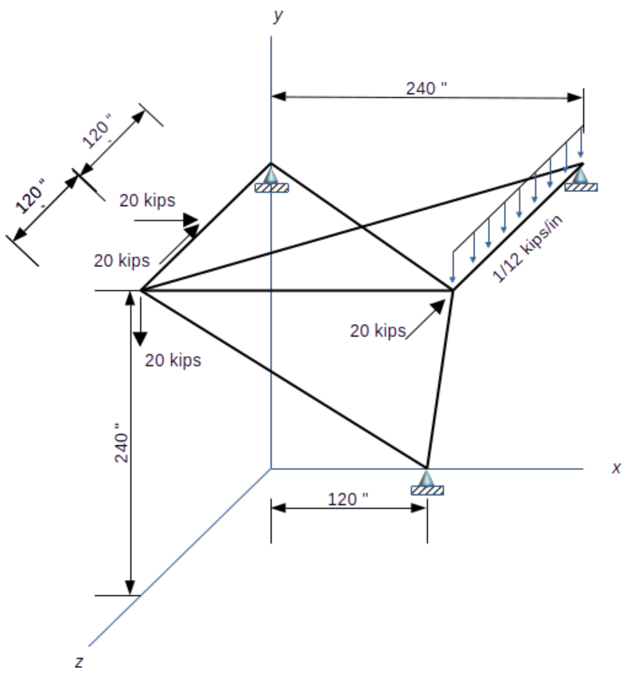

Figure 7: Space Truss

OpenStruc Script for the Space Truss Structure

Figure 8 presents a script containing the set of ***OpenStruc***'s commands used to model the space truss shown in Figure 7. By now, we have reached a stage where no new commands are needed to model this space truss. Only a few new sub-command parameters are introduced.

For example, lines 7, 10, and 11 utilize the new **`dz`** parameter, in addition to the previously used **`dx`**, and **`dy`** parameters, to model the fully constrained hinge-support condition of nodes 1, 4, and 5.

In addition to constraints in the z direction, this example also requires applying forces in that direction to nodes and elements. For example, in line 42, the **`fz`** parameter allows applying a nodal load in the z direction. Similarly, in lines 46 and 47, the **`pz`** parameter applies a concentrated load in the z direction to element 4.

```
Structure Weaver & Gere book, Chapter 5, Space Truss Example 1, pages 345−347
echo off
type space truss # sections 4

number of nodes 5
coordinates
    1 120.0      0.000      0.0   support dx dy dz
    2   0.0    240.000    240.0
    3 240.0    240.000    240.0
    4 240.0    240.000      0.0   support dx dy dz
    5   0.0    240.000      0.0   support dx dy dz

number of element 7
incidences
    1 1 2
    2 1 3
    3 2 3
    4 2 5
    5 3 4
    6 2 4
    7 3 5

number of materials 1
materials
    e 30000 g 11538.4  ct 6.5e−6 weight 2.83565e−4

number of section_Types 1
sectype
    ar 10.0

elements
    all  secType 1

plot

number of load_Cases 1

loading case 1 "First Load Case"

    loaded nodes
    2 fy −20.0
    3 fz −20.0

    loaded elements
    4
    #concentrated local    px 20.0   pz 20.0  la  −0.5
    concentrated global    px 20.0   pz −20.0 la −0.5

    5
    #uniform local  wy −0.0833333333333333
    uniform global  wy −0.0833333333333333

    solve
stop
```

Figure 8: Space Truss Script.

6 Space Frame Analysis

This example describes the data preparation for the static analysis of a space frame structure. This space frame corresponds to an example presented in Chapter 5, pages 354, 364-366 of the book by Weaver and Gere[2].

Figure 9 shows a space frame having four nodes and three elements. The lower nodes are fixed-supported. The three elements have an area, ***ar***, equal to 0.01 m^2; a second polar moment of area, ***ix***, equal to 0.002 m^4; and second moments of area, ***iy*** and ***iz***, equal to 0.001 m^4. The elasticity modulus (e) is 200x10^6 kN/m^2, and a shear modulus (g) equal to 80x10^6 kN/m^2.

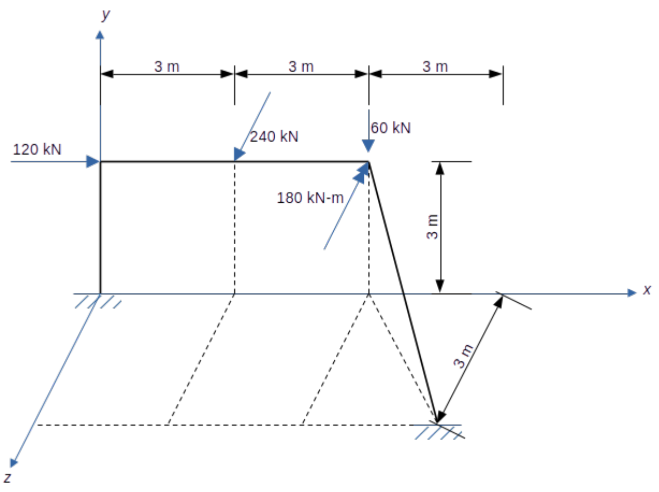

Figure 9: Space Frame

OpenStruc Script for the Space Frame

Figure 10 presents a script containing the set of ***OpenStruc***'s commands used to model the space frame shown in Figure 9. No new commands are needed to model this space frame.

```
Structure Weaver & Gere book, Chapter 5, Space Frame Example 1, pages 354, 364−366
echo off
type space frame # sections 4

shear deformation off
number of nodes 4
number of element 3
number of materials 1
number of section_Types 1

coordinates
      1 0.000       0.000       0.000 support dx dy dz rx ry rz
      2 0.000       3.000       0.000
      3 6.000       3.000       0.000
      4 9.000       0.000       3.000 support dx dy dz  rz   rx ry

incidences
      1 1 2
      2 2 3
      3 3 4

materials
      e 200.0e6 g 80.0e6 ct  2.0  weight 3.0

sectype
      ar 0.01 Ix 2.0e−3 Iy 1.0e−3 Iz 1.0e−3 mt 1

elements
      all  secType 1

plot

number of load_Cases 1
loading case 1 "First Load Case"

      loaded nodes
      2 fx 120.0
      3 fy −60.0 mz −180.0

      loaded elements
      2
      concentrated local     pz 240.0     la −0.5
      solve
stop
```

Figure 10: Space Frame Script.

References

[1] Klaus-Jürgen Bathe. *Finite Element Method*. Prentice Hall, second edition, June 1995.

[2] William Weaver Jr. and James M. Gere. *Matrix Analysis Of Framed Structures*. Van Nostrand Reinhold, third edition, August 1990.