

Manual “How to generate a XFEM4U model using Python”

This manual describes the functions needed to generate a **XFEM4U** model. It will be illustrated by some examples which the user can use of course.

Don't be afraid. Just get started. In short time you will be familiar with the use of class functions we have made.

1. Add “Struct4U.py” to your Python environment. This is our Python library.
2. Copy from or use “ExamplesXFEM4U.py”. This will give you a quick start.

In all the examples it starts with creating the object **XFEM4U** followed by several class method (or function) call's. All methods (functions) are described here in this manual. Note that the number of parameters divers. Some are mandatory, some are not and set by a default value. The main reason is to keep it very simple and enabling you to get the job done easy.

1. Create the class object **XFEM4U**
2. Use **XFEM4U** class methods to add the several items needed. Note that the sequence of calling these methods is crucial since checks are built in. For example. While adding a beam you cannot refer to a node that does not exist yet.

If you have any remarks and/or questions please let us know.

XFEM4U (Description, ProjectNumber, ConsequenceClass, DesignWorkingLife, ExportDateTime)
Create the XFEM4U object.

- `[Description]` (string): Description of the project
- `[ProjectNumber]` (string): Project number
- `[ConsequenceClass]` (ConsequenceClass): ConsequenceClass (**CC1**/CC2/CC3)
- `[DesignWorkingLife]` (int): Design working life in years
- `[ExportDateTime]` (string): Export date and time

XFEM4U:AddGrid (x, x_label, y, y_label, z, z_label)

Create grid lines

- `x` (string): distances [syntax](#): "distance distance numberxdistance ..and so on"
- `x\_label` (string): labels
- `y` (string): distances
- `y\_label` (string): labels
- `z` (string): distances
- `z\_label` (string): labels

XFEM4U:AddNode (nr, x, y, z, type)

Create and add a node

- `nr` (int): node number; if nr==0 generates node number
- `x` (float): x- coordinate
- `y` (float): y- coordinate

- `z` (float): z- coordinate
- `[type]` (SupportType): (**NoneSupport**/Custom/Pinned/Fixed/Roll_X/Roll_Y/Roll_Z/Hinge)

If the node coordinate (x, y, z) already exists the return value is the node number of the node found.

return value: node number when succeed; -1 when fails

XFEM4U: AddSupport (NodeNumber, type, Tx, Ty, Tz, Rx, Ry, Rz, Kx, Ky, Kz, Cx, Cy, Cz)

Create and add a support

- `NodeNumber` (int): node number
- `type` (SupportType): (NoneSupport/Custom/Pinned/Fixed/Roll_X/Roll_Y/Roll_Z/Hinge)
- `[Tx]` (str): tx-restraint (“ ”=free;”A”=absolute;”P”=positive;”N”=negative;”S”=spring)
- `[Ty]` (str): ty-restraint
- `[Tz]` (str): tz-restraint
- `[Rx]` (str): rx-restraint (“ ”=free;”A”=absolute;”S”=spring)
- `[Ry]` (str): ry-restraint
- `[Rz]` (str): rz-restraint
- `[Kx]` (float): kx-spring value
- `[Ky]` (float): ky-spring value
- `[Kz]` (float): kz-spring value
- `[Cx]` (float): cx-spring value
- `[Cy]` (float): cy-spring value
- `[Cz]` (float): cz-spring value

return value: node number when succeed; -1 when fails

XFEM4U: AddSteelProfile (nr, ProfileName, MaterialName, Angle, shape, h, b, tf, tw, r1, r2, slope)

Create and add a steel profile

- `nr` (int): profile number; if nr==0 generates profile number
- `ProfileName` (str): profile name (see XFEM4U for standard profiles)
- `MaterialName` (str): material name
- `[Angle]` (float): angle of profile
- `[shape]` (SteelProfileShape): (**H**,U,T,RHS,CHS,FLAT,HOLE,L,CSS,TRIANGLE,CFL,CC,T1)
- `[h]` (float): height
- `[b]` (float): width
- `[tf]` (float): flange thickness
- `[tw]` (float): web thickness
- `[r1]` (float): radius flange
- `[r2]` (float): radius flange-web

return value: profile number when succeed; -1 when fails

XFEM4U: AddConcreteProfile (nr, ProfileName, MaterialName, Angle,
shape, h, b, h1, b1, h2, b2, h3, b3, h4, b4)

Create and add a concrete profile

- `nr` (int): profile number; if nr==0 generates profile number
- `ProfileName` (str): profile name
- `MaterialName` (str): concrete grade
 - "C12/15", "C16/20", "C20/25", "C25/30", "C28/35",
 - "C30/37", "C35/45", "C40/50", "C45/55", "C50/60",
 - "C53/65", "C55/67", "C60/75", "C70/85", "C80/95", "C90/105"
- `[Angle]` (float): angle of profile
- `[shape]` (ConcreteProfileShape): (**S1**,... S20)
- `[h]` (float): height
- `[b]` (float): width
- `[h1]` (float): height top left
- `[b1]` (float): width top left
- `[h2]` (float): height top right
- `[b2]` (float): width top right
- `[h3]` (float): height bottom left
- `[b3]` (float): width bottom left
- `[h4]` (float): height bottom right
- `[b4]` (float): width bottom right

return value: profile number when succeed; -1 when fails

XFEM4U: AddBeam (nr, from_node, to_node, profile, RestrainBegin, RestrainEnd,
NlateralSupports, Angle, Angle_Profile, Z, ref)

Create and add a beam

- `nr` (int): beam number; if nr==0 generates beam number
- `from\_node` (int): from node number
- `to\_node` (int): to node number
- `profile` (int): profile number
- `[ RestrainBegin]` (BeamRestraint): (hinge/**fixed**/tension)
- `[ RestrainEnd]` (BeamRestraint): (hinge/**fixed**/tension)
- `[ NlateralSupports]` (int): number of lateral supports
- `[ Angle]` (float): angle on x-axis
- `[ Angle\_Profile]` (float): angle of profile
- `[ Z]` (float): z-distance
- `[ ref]` (Z_refence): (Top/**Center**/Bottom) reference

return value: beam number when succeed; -1 when fails

XFEM4U: AddPlate (nr, polygon, type, MaterialName, h, z)

Create and add a plate

- `nr` (int): node number; if nr==0 generates plate number
- `polygon` ([int]): list of node numbers
- `type` (MaterialType): type of material (Steel/Concrete/Timber/BuiltupSection/Masonry)
- `MaterialName` (str): material name
- `h` (float): plate thickness
- `[z]` (float): z-distance
- `[ref]` (Z_refence): (Top/**Center**/Bottom) reference

return value: plate number when succeed; -1 when fails

XFEM4U: AddPanel (nr, Description, polygon, Direction, type)

Create and add a panel

- `nr` (int): node number; if nr==0 generates panel number
- `polygon` ([int]): list of node numbers
- `Direction` (LoadBearingDirection): load bearing direction (all/X/Y)
- `type` (SurfaceType): surface type (wall/free_standing_wall/flat_roof/
mono_pitch_roof/saddle_roof_positive_pitch/
saddle_roof_negative_pitch/unknown/not_determined)

return value: panel number when succeed; -1 when fails

XFEM4U: AddCase (nr: int, Description : str, type: LoadTyp, psi0: float, psi1: float, psi2: float)

Create and add a load case

- `nr` (int): node number; if nr==0 generates case number
- `Description` ([int]): list of node numbers
- `type` (LoadTyp): load type (Permanent_incl_own_weight/A_domestic/B_office/
C_congregation/D_shops/E_storage/F_traffic/
G_traffic/H_roofs/Snow/Wind/Temp/live_load_medium/
Imperfection/Permanent_excl_own_weight/live_load_short)
- `psi0` (float): momentary factor psi0
- `psi1` (float): momentary factor psi1
- `psi2` (float): momentary factor psi2

return value: case number when succeed; -1 when fails

XFEM4U: AddNodeLoad (LoadCaseNumber, NodeNumber, Fx, Fy, Fz, Mx, My, Mz, Description)

Create and add a node load

- `LoadCaseNumber` (int): load case number
- `NodeNumber` (int): node number
- `Fx` (float): Force in global x-direction
- `Fy` (float): Force in global y-direction
- `Fz` (float): Force in global z-direction
- `[Mx]` (float): Moment about on global x-axis
- `[My]` (float): Moment about on global x-axis

- `[Mz]` (float): Moment about on global x-axis
- `[Description]` (str): Description of the load

return -1 when fails

XFEM4U: AddBeamLoadQ (LoadCaseNumber, BeamNumber, coordinate_system, dir, Load1, Load2, a, bCustomL=FALSE, L=0, Angle=0, Description=" ")

Create and add a distributed beam load

- `LoadCaseNumber` (int): load case number
- `BeamNumber` (int): beam number
- `dir` (Direction): direction of the load (X/Y/Z)
- `Load1` (float): Load at begin
- `Load2` (float): Load at the end
- `[a]` (float): distance begin node and q-load
- `[bCustomL]` (bool): setting whether the length can be adjusted
- `[L]` (float): Length of the q-load
- `[Angle]` (float): Angle with beam x-axis
- `[coordinate]` (CoordinateSystem): coordinate system (**Local**/Global/GlobalProjective)
- `[Description]` (str): Description of the load

return -1 when fails

XFEM4U: AddBeamLoadF (LoadCaseNumber, BeamNumber, coordinate_system, dir, Load, a, Angle=0, Description=" ")

Create and add a point load on a beam

- `LoadCaseNumber` (int): load case number
- `BeamNumber` (int): beam number
- `dir` (Direction): direction of the load (X/Y/Z)
- `Load` (float): Load
- `a` (float): distance begin node and F-load
- `[Angle]` (float): Angle with beam x-axis
- `[coordinate]` (CoordinateSystem): coordinate system (**Local**/Global/GlobalProjective)
- `[Description]` (str): Description of the load

return -1 when fails

XFEM4U: AddBeamLoadM (LoadCaseNumber, BeamNumber, coordinate_system, dir, Load, a, Description=" ")

Create and add a moment load on a beam

- `LoadCaseNumber` (int): load case number
- `BeamNumber` (int): beam number
- `dir` (Direction): direction of the load (X/Y/Z)
- `Load` (float): Load
- `a` (float): distance begin node and M-load

- `[coordinate]` (CoordinateSystem): coordinate system (**Local**/Global/GlobalProjective)
- `[Description]` (str): Description of the load

return -1 when fails

XFEM4U: AddSurfaceLoad (LoadCaseNumber: int, polygon, Direction,coordinate_system,
dir, q1, q2=0, q3=0, LoadConstantOrLinear, iq1=0, iq2=0, iq3=0,
Description = " " Description=" ")
Create and add surface load

- `LoadCaseNumber` (int): load case number
- `polygon` ([xyz]): list of xyz - coordinates x,y,z
- `Direction` (LoadBearingDirection): load bearing direction (all/X/Y)
- `dir` (Direction): direction of the load (X/Y/Z)
- `q1` (float): Load
- `[q2]` (float): Load
- `[q2]` (float): Load
- `[LoadConstantOrLinear]` (LoadConstantOrLinear): load type (constant/linear/waterheight)
- `[q1]` (int): location q1
- `[q2]` (int): location q2
- `[q2]` (int): location q3
- `[coordinate]` (CoordinateSystem): coordinate system (Local/Global/GlobalProjective)
- `[Description]` (str): Description of the load

return -1 when fails

XFEM4U: AddCombination (Descriptionr, type, case1, loadfactor1, psi1,
case2, loadfactor2, psi2,
case3, loadfactor3, psi3,
case4, loadfactor4, psi4,
case5, loadfactor5, psi5)

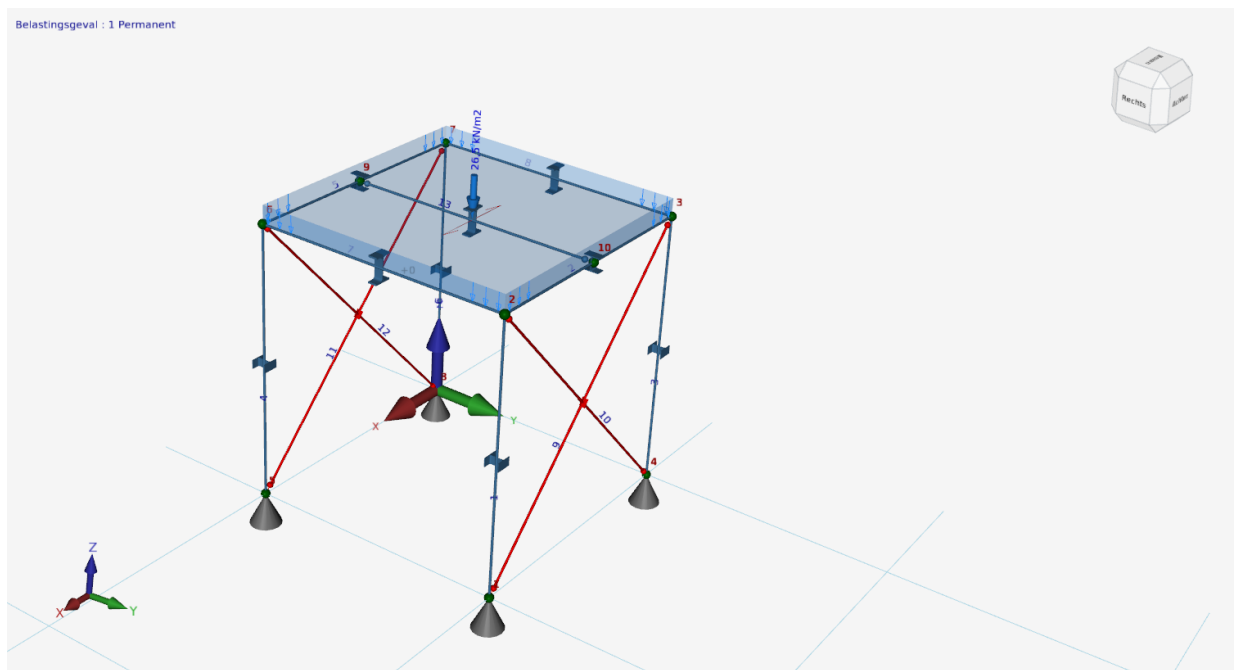
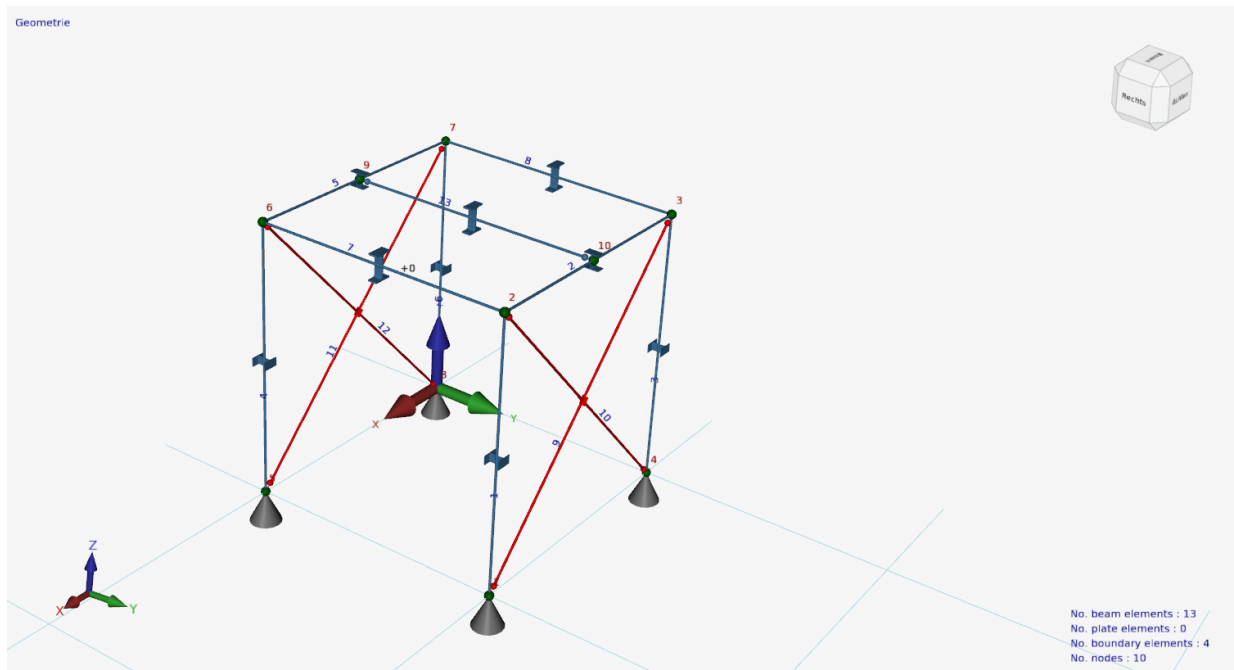
Create and add a combination

- `Description` (str): Description of the combination
- `type` (CombTyp): combination type (ULS/SLS/QPERM/PERM/FREQUENT)
- `case1` (int): load case number
- `loadfactor1` (float): load factor
- `psi1` (int): momentary factor
- `[case2]` (int): load case number
- `[loadfactor2]` (float): load factor
- `[psi2]` (int): momentary factor
- `[case3]` (int): load case number
- `[loadfactor3]` (float): load factor
- `[psi3]` (int): momentary factor
- `[case4]` (int): load case number
- `[loadfactor4]` (float): load factor
- `[psi4]` (int): momentary factor
- `[case5]` (int): load case number

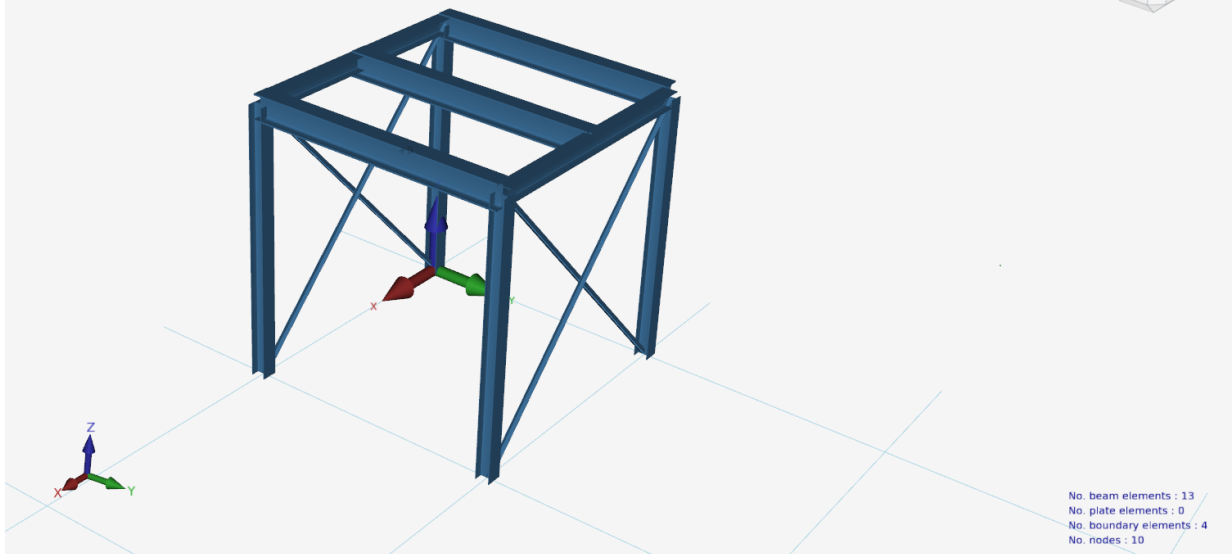
- `[loadfactor5]` (float): load factor
- `[psi5]` (int): momentary factor

return -1 when fails

Simple steel structure



Geometrie



```
xfem = XFEMF4U("Example in Python","12345", ConsequenceClass.CC2, 100, "")

xfem.AddGrid("0 4x3000 2000 2x500","A B C D E F G H I J K L M N O P Q R S T U V W X Y Z", \
            "0 4x3000", "1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25", \
            "0", "+0")

xfem.AddCase(1,"Permanent", LoadTyp.Permanent_incl_own_weight, 1.0,1.0,1.0)
xfem.AddCase(2,"Veranderlijk", LoadTyp.E_storage, 1.0,0.9,0.8)
xfem.AddCase(3,"Wind", LoadTyp.Wind, 1.0,0.0,0.0)

xfem.AddNode(1,3000,3000,0,SupportType.Pinned)
xfem.AddNode(2,3000,3000,3200)
xfem.AddNode(3,0,3000,3200)
xfem.AddNode(4,0,3000,0,SupportType.Pinned)
xfem.AddNode(5,3000,0,0,SupportType.Pinned)
xfem.AddNode(6,3000,0,3200)
xfem.AddNode(7,0,0,3200)
xfem.AddNode(8,0,0,0,SupportType.Pinned)
xfem.AddNode(9,1500,0,3200)
xfem.AddNode(10,1500,3000,3200)

xfem.AddSteelProfile(1,"HE180A","S235")
xfem.AddSteelProfile(2,"L50X50X5","S235")
xfem.AddSteelProfile(3,"MyOwnShape","S235",0, SteelProfileShape.H,300,200,20,12,0,30)

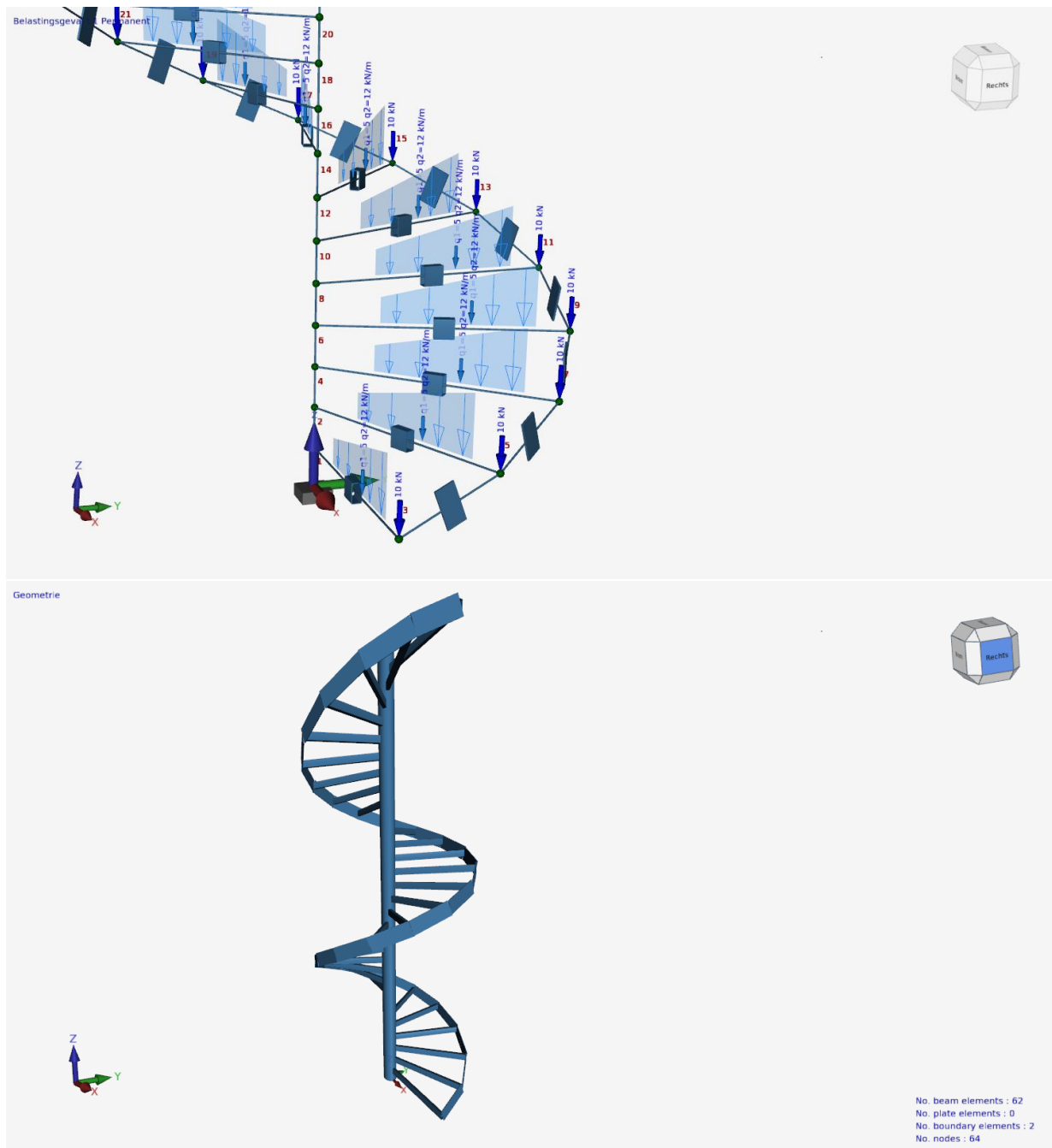
xfem.AddBeam(1,1,2,1)
xfem.AddBeam(2,2,3,1)
xfem.AddBeam(3,4,3,1)
xfem.AddBeam(4,5,6,1)
xfem.AddBeam(5,6,7,1)
xfem.AddBeam(6,8,7,1)
xfem.AddBeam(7,6,2,3)
xfem.AddBeam(8,7,3,3)
xfem.AddBeam(9,1,3,2,BeamRestraint.tension)
xfem.AddBeam(10,4,2,2,BeamRestraint.tension)
xfem.AddBeam(11,5,7,2,BeamRestraint.tension)
xfem.AddBeam(12,8,6,2,BeamRestraint.tension)
xfem.AddBeam(13,9,10,3,BeamRestraint.hinge,BeamRestraint.hinge)

pol = [6,7,3,2] # list of node numbers
xyz_poly = [xyz]
xyz_poly = xfem.GetNodeXYZList(pol) # get coordinates x,y and z from this list of node numbers
xfem.AddSurfaceLoad(1,xyz_poly,LoadBearingDirection.X, CoordinateSystem.Local, Direction.Z,-26.5)

xfem.AddCombination("Permanent",CombTyp.ULS,          1,1.35,1.00,    2,1.50,1.00)
xfem.AddCombination("Veranderlijk",CombTyp.ULS,        1,1.20,1.00,    2,1.50,1.00)
xfem.AddCombination("Wind",CombTyp.ULS,                1,1.20,1.00,    3,1.50,1.00)
xfem.AddCombination("Permanent",CombTyp.SLS,          1,1.00,1.00,    2,1.00,1.00)
xfem.AddCombination("Wind",CombTyp.SLS,                1,1.00,1.00,    3,1.00,1.00)
xfem.AddCombination("BGT Blijvend",CombTyp.PERM,       1,1.00,1.00,    2,1.00,1.00)
xfem.AddCombination("BGT Quasi-blijvend",CombTyp.PERM,1,1.00,1.00,    2,1.00,0.80)

temp_dir = tempfile.gettempdir()
filename = temp_dir + "\\steel_frame.xml"
with open(filename, "w", encoding="ascii") as file:
    file.writelines(xfem.XML())
writeDirectCommandsfile(filename)
```

Spiral stairs



```
xfem = XFEMF4U("Example in Python","12345", ConsequenceClass.CC2, 100, "")

xfem.AddGrid("0 4x3000 2000 2x500","A B C D E F G H I J K L M N O P Q R S T U V W X Y Z", \
    "0 4x3000", "1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25", \
    "0", "+0")

xfem.AddCase(1,"Permanent", LoadTyp.Permanent_incl_own_weight, 1.0,1.0,1.0)
xfem.AddCase(2,"Veranderlijk", LoadTyp.E_storage, 1.0,0.9,0.8)
xfem.AddCase(3,"Wind", LoadTyp.Wind, 1.0,0.0,0.0)

optrede = 200
ltrede = 1200
ntrede360 = 15
alpha = 360.0 / ntrede360
nspil = 2
a = 0
x = 0
y = 0
z = 0

xfem.AddSteelProfile(1,"HFCHS219x125","S235")
xfem.AddSteelProfile(2,"HFRHS100x50x4","S235")
xfem.AddSteelProfile(3,"S200x10","S235",90)
```

```

k1 = xfem.AddNode(0,x,y,z,SupportType.Fixed)
z = optrede
vnode = 0

for i in range(nspil*ntrede360+1) :
    x = ltrede * cos(a * pi/180)
    y = ltrede * sin(a * pi/180)
    node1 = xfem.AddNode(0,0,0,z)
    node2 = xfem.AddNode(0,x,y,z)
    beam = xfem.AddBeam(0,node1,node2,2)
    xfem.AddNodeLoad(1,node2,0,0,-10)
    xfem.AddBeamLoadQ(1,beam, CoordinateSystem.Local, Direction.Z, -5,-12, 300,TRUE,750)

    a += alpha
    z += optrede
    if vnode!=0:
        beam = xfem.AddBeam(0,vnode,node2,3)
        vnode = node2

k2 = xfem.AddNode(0,0,0,z,SupportType.Roll_Z)
xfem.AddBeam(0,k1,k2,1)

xfem.AddCombination("Permanent",CombTyp.ULS,          1,1.35,1.00,    2,1.50,1.00)
xfem.AddCombination("Veranderlijk",CombTyp.ULS,       1,1.20,1.00,    2,1.50,1.00)
xfem.AddCombination("Wind",CombTyp.ULS,               1,1.20,1.00,    3,1.50,1.00)
xfem.AddCombination("Permanent",CombTyp.SLS,          1,1.00,1.00,    2,1.00,1.00)
xfem.AddCombination("Wind",CombTyp.SLS,               1,1.00,1.00,    3,1.00,1.00)
xfem.AddCombination("BGT Blijvend",CombTyp.PERM,      1,1.00,1.00,    2,1.00,1.00)
xfem.AddCombination("BGT Quasi-blijvend",CombTyp.PERM,1,1.00,1.00,    2,1.00,0.80)

temp_dir = tempfile.gettempdir()
filename = temp_dir + "\\spiral_stairs.xml"

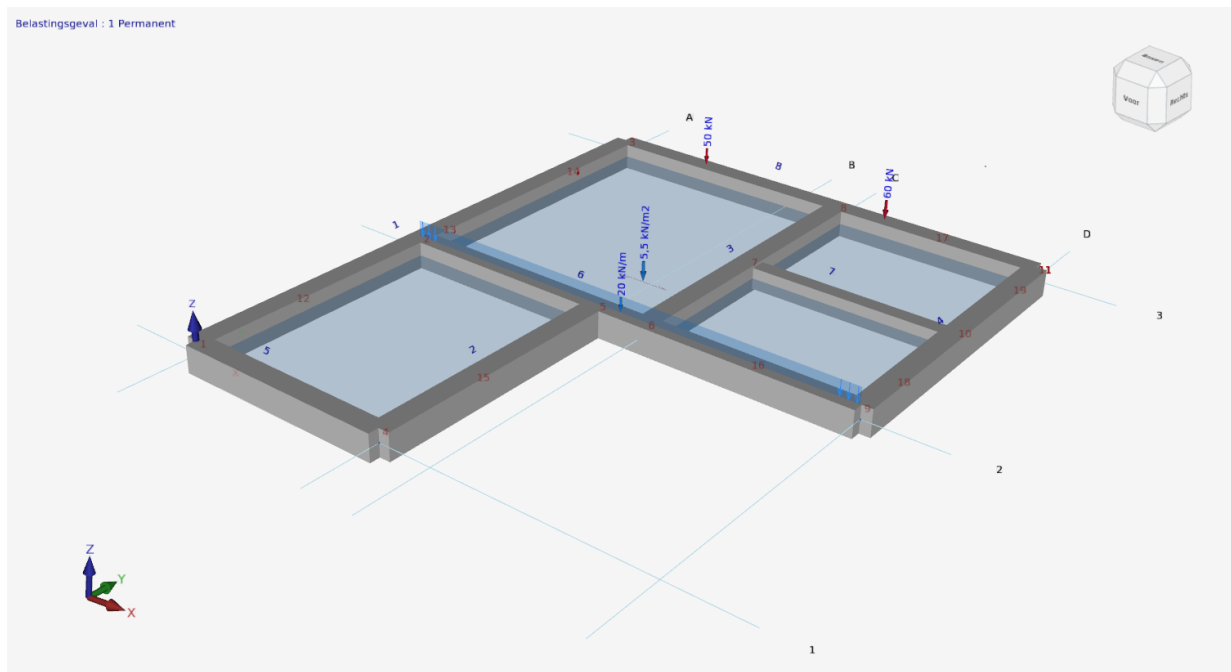
with open(filename, "w", encoding="ascii") as file:
    file.writelines(xfem.XML())

writeDirectCommandsfile(filename)

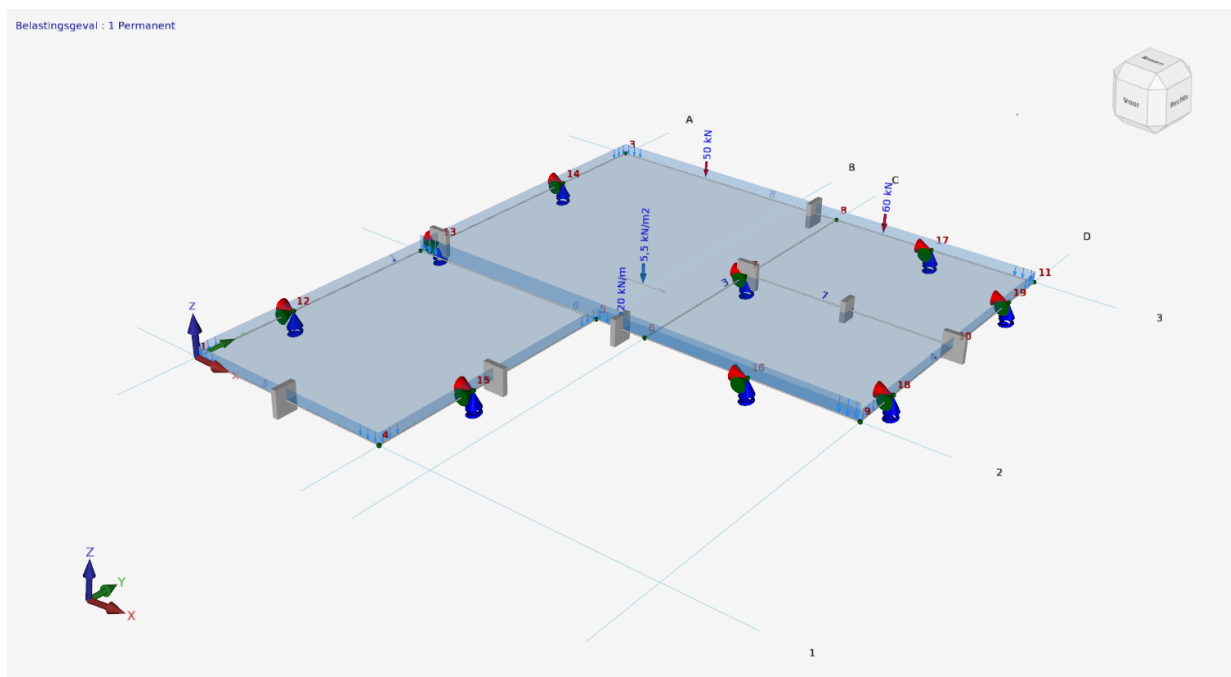
```

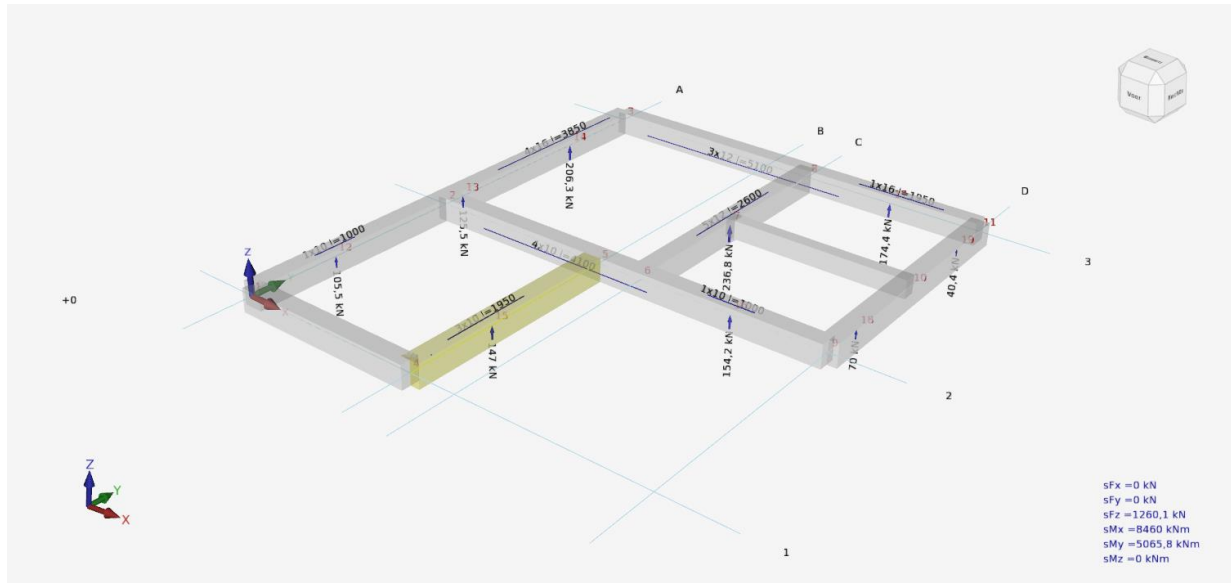
Simple grid foundation

Belastingsgeval : 1 Permanent



Belastingsgeval : 1 Permanent





```
xfem = XFEMF4U("Example in Python","12345", ConsequenceClass.CC2, 100, "")

xfem.AddGrid("0 4000 1000 4000","A B C D E F G H I J K L M N O P Q R S T U V W X Y Z", \
    "0 5000 6000", "1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25", \
    "0", "+0")

xfem.AddCase(1,"Permanent", LoadTyp.Permanent_incl_own_weight, 1.0,1.0,1.0)
xfem.AddCase(2,"Veranderlijk", LoadTyp.E_storage, 1.0,0.9,0.8)
xfem.AddCase(3,"Wind", LoadTyp.Wind, 1.0,0.0,0.0)

xfem.AddNode(1,0,0,0)
xfem.AddNode(2,0,5000,0)
xfem.AddNode(3,0,11000,0)
xfem.AddNode(4,4000,0,0)
xfem.AddNode(5,4000,5000,0)
xfem.AddNode(6,5000,5000,0)
xfem.AddNode(7,5000,8000,0)
xfem.AddNode(8,5000,11000,0)
xfem.AddNode(9,9000,5000,0)
xfem.AddNode(10,9000,8000,0)
xfem.AddNode(11,9000,11000,0)

xfem.AddConcreteProfile(1,"500x400",ConcreteElement.Beam,"C30/37", ConcreteProfileShape.S1,500,400)
xfem.AddConcreteProfile(2,"400x300",ConcreteElement.Beam,"C30/37", ConcreteProfileShape.S1,400,300)

xfem.AddBeam(1,1,3,1)
xfem.AddBeam(2,4,5,1)
xfem.AddBeam(3,6,8,1)
xfem.AddBeam(4,9,11,1)
xfem.AddBeam(5,1,4,1)
xfem.AddBeam(6,2,9,1)
xfem.AddBeam(7,7,10,2)
xfem.AddBeam(8,3,11,1)

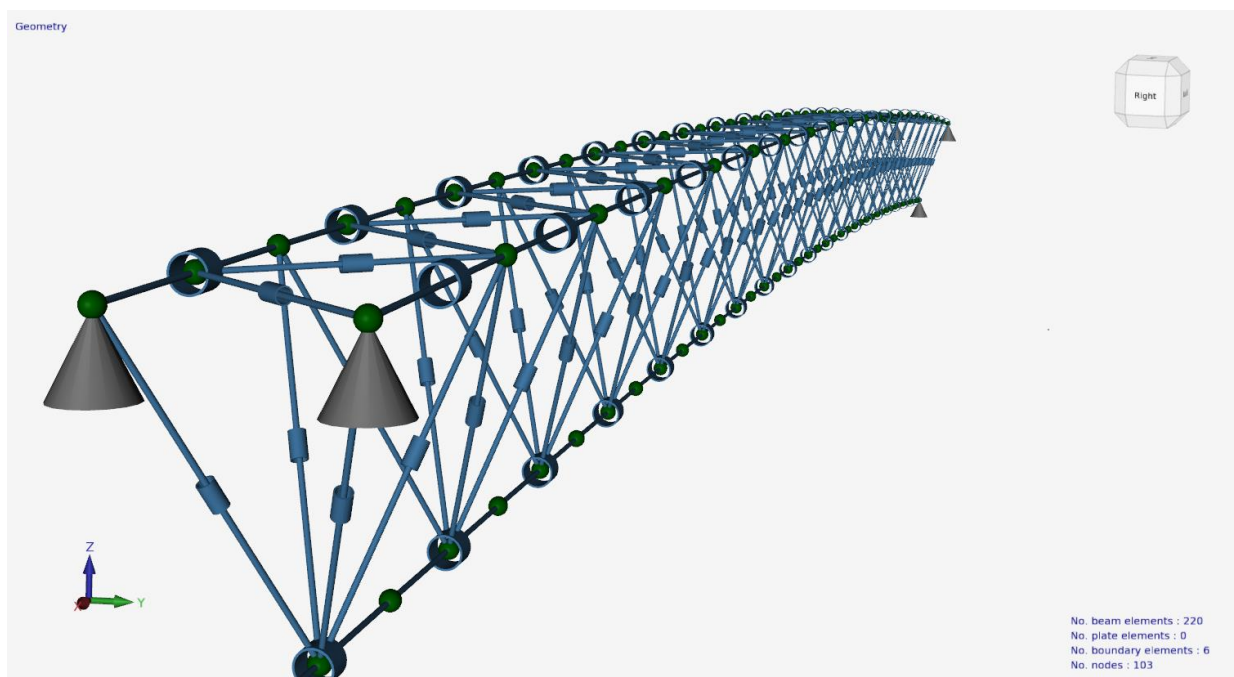
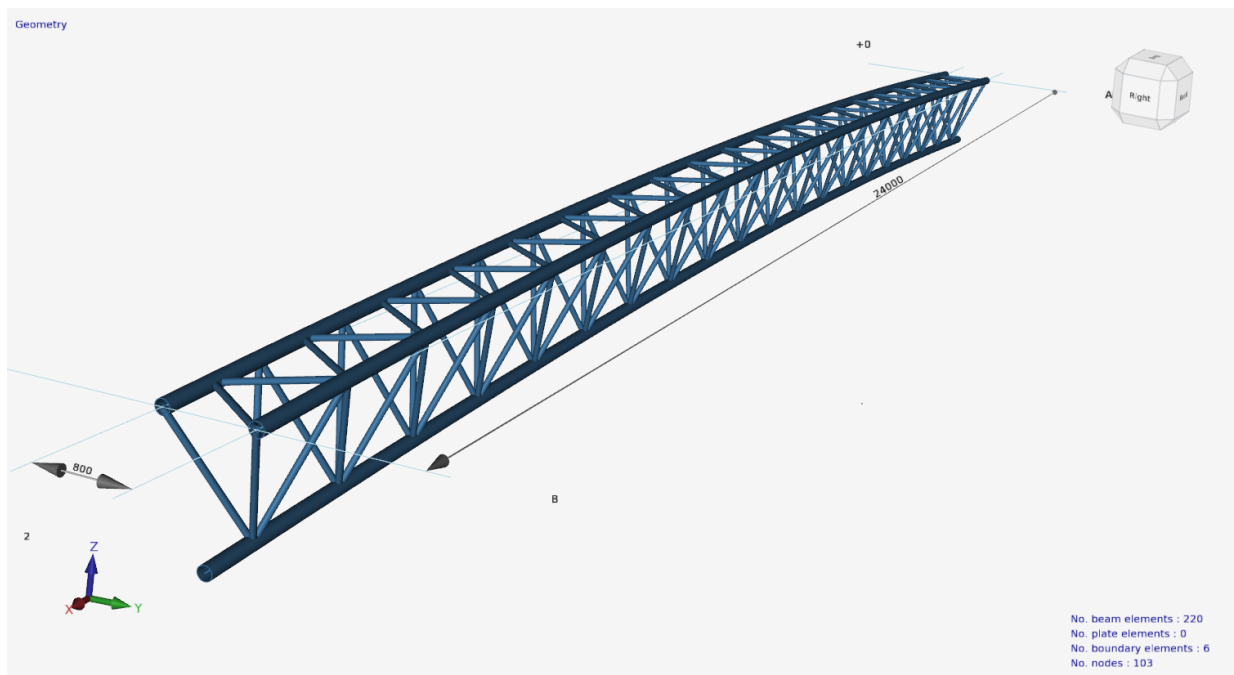
# piles
xfem.AddSupport(xfem.AddNode(0,0,2000,0),SupportType.Custom,"A","A","S","","",0,0,10000)
xfem.AddSupport(xfem.AddNode(0,0,5500,0),SupportType.Custom,"A","A","S","","",0,0,10000)
xfem.AddSupport(xfem.AddNode(0,0,9000,0),SupportType.Custom,"A","A","S","","",0,0,10000)
xfem.AddSupport(xfem.AddNode(0,4000,2000,0),SupportType.Custom,"A","A","S","","",0,0,10000)
xfem.AddSupport(7,SupportType.Custom,"A","A","S","","",0,0,10000)
xfem.AddSupport(xfem.AddNode(0,7000,5000,0),SupportType.Custom,"A","A","S","","",0,0,10000)
xfem.AddSupport(xfem.AddNode(0,7000,11000,0),SupportType.Custom,"A","A","S","","",0,0,10000)
xfem.AddSupport(xfem.AddNode(0,9000,6000,0),SupportType.Custom,"A","A","S","","",0,0,10000)
xfem.AddSupport(xfem.AddNode(0,9000,10000,0),SupportType.Custom,"A","A","S","","",0,0,10000)

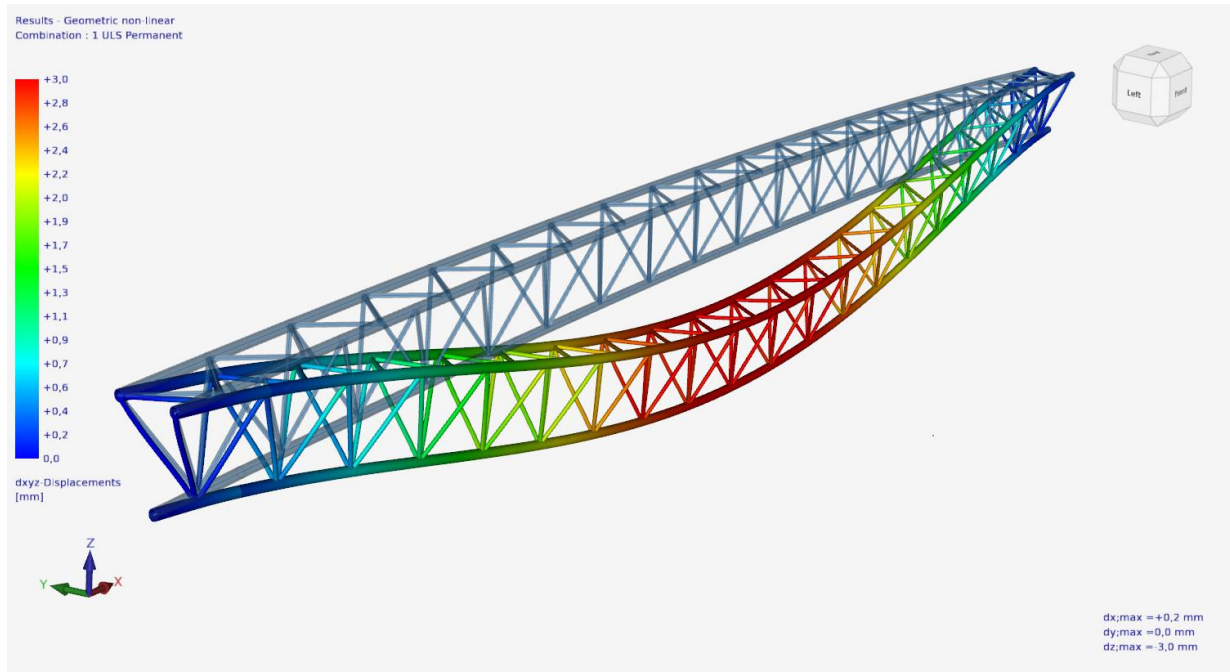
pol = [1,3,11,9,5,4] # list of node numbers
xyz_poly = [xyz]
xyz_poly = xfem.GetNodeXYZList(pol) # get coordinates x,y and z from this list of node numbers
xfem.AddSurfaceLoad(1,xyz_poly,LoadBearingDirection.Y, Direction.Z,-5.5)
xfem.AddBeamLoadQ(1,6, Direction.Z, -20, -20)
xfem.AddBeamLoadF(1,8, Direction.Z, -50,2000)
xfem.AddBeamLoadF(1,8, Direction.Z, -60,6000)

xfem.AddCombination("Permanent",CombTyp.ULS, 1,1.35,1.00, 2,1.50,1.00)
xfem.AddCombination("Veranderlijk",CombTyp.ULS, 1,1.20,1.00, 2,1.50,1.00)
xfem.AddCombination("Wind",CombTyp.ULS, 1,1.20,1.00, 3,1.50,1.00)
xfem.AddCombination("Permanent",CombTyp.SLS, 1,1.00,1.00, 2,1.00,1.00)
xfem.AddCombination("Wind",CombTyp.SLS, 1,1.00,1.00, 3,1.00,1.00)
xfem.AddCombination("BGT Blijvend",CombTyp.PERM, 1,1.00,1.00, 2,1.00,1.00)
xfem.AddCombination("BGT Quasi-blijvend",CombTyp.PERM,1,1.00,1.00, 2,1.00,0.80)

xfem.Save2XFEM4U("foundation_grid")
```

Simple truss with camber





```
xfem = XFEMF4U("Example in Python", "12345", ConsequenceClass.CC2, 100, "")

xfem.AddGrid("0 1x24000", "A B C D E F G H I J K L M N O P Q R S T U V W X Y Z", \
    "-400 800", "1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25", \
    "0", "+0")

xfem.AddCase(1, "Permanent", LoadTyp.Permanent_incl_own_weight, 1.0, 1.0, 1.0)
xfem.AddCase(2, "Veranderlijk", LoadTyp.E_storage, 1.0, 0.9, 0.8)
xfem.AddCase(3, "Wind", LoadTyp.Wind, 1.0, 0.0, 0.0)

xfem.AddSteelProfile(1, "HFCBS603X5", "S235", 90)
xfem.AddSteelProfile(2, "HFCBS1397X8", "S235")

h = 1200
b = 800
dx = 600
nsections = 20
camber = 300 # camber in the middle
span = 2*nsections*dx

x = 0

# determine the parabolic coefficients for the camber
ac, bc, cc = parabolic(0, 0, span/2, camber, span, 0)

for n in range(nsections) :

    # camber - camber(x)
    dz1 = ac*math.pow(x, 2) + bc*x + cc
    dz2 = ac*math.pow(x + 2*dx, 2) + bc*(x + 2*dx) + cc
    dz3 = ac*math.pow(x + dx, 2) + bc*(x + dx) + cc

    k1 = xfem.AddNode(0, x, b/2, dz1)
    k2 = xfem.AddNode(0, x, -b/2, dz1)
    k3 = xfem.AddNode(0, x, 0, -h+dz1)
    k4 = xfem.AddNode(0, x + dx, 0, -h+dz1)
    if n==0 :
        xfem.AddSupport(k1, SupportType.Pinned)
        xfem.AddSupport(k2, SupportType.Pinned)
        xfem.AddSupport(k3, SupportType.Pinned)

    xfem.AddBeam(0, k1, k4, 1)
    xfem.AddBeam(0, k2, k4, 1)
    k5 = xfem.AddNode(0, x + 2*dx, b/2, dz2)
    k6 = xfem.AddNode(0, x + 2*dx, -b/2, dz2)
    k7 = xfem.AddNode(0, x + 2*dx, 0, -h+dz2)
    if n==nsections-1 :
        xfem.AddSupport(k5, SupportType.Pinned)
        xfem.AddSupport(k6, SupportType.Pinned)
        xfem.AddSupport(k7, SupportType.Pinned)

    xfem.AddBeam(0, k4, k5, 1)
    xfem.AddBeam(0, k4, k6, 1)
    k8 = xfem.AddNode(0, x + dx, -b/2, dz1)
    xfem.AddBeam(0, k1, k8, 1)
    xfem.AddBeam(0, k8, k5, 1)
```

```

xfem.AddBeam(0,k1,k5,2)
xfem.AddBeam(0,k2,k6,2)
xfem.AddBeam(0,k3,k7,2)

x += 2*dx

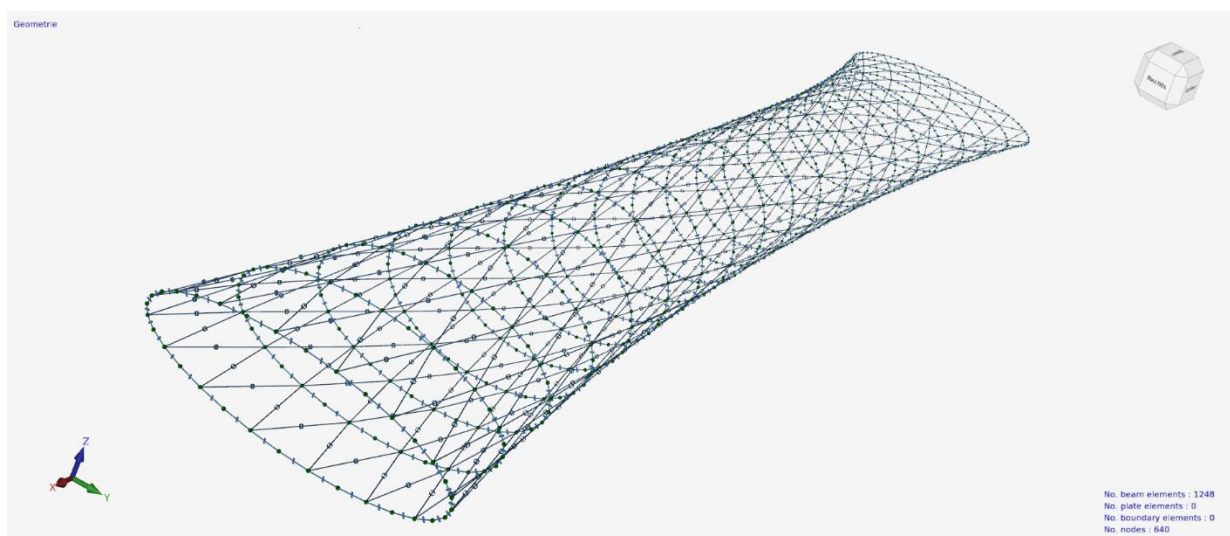
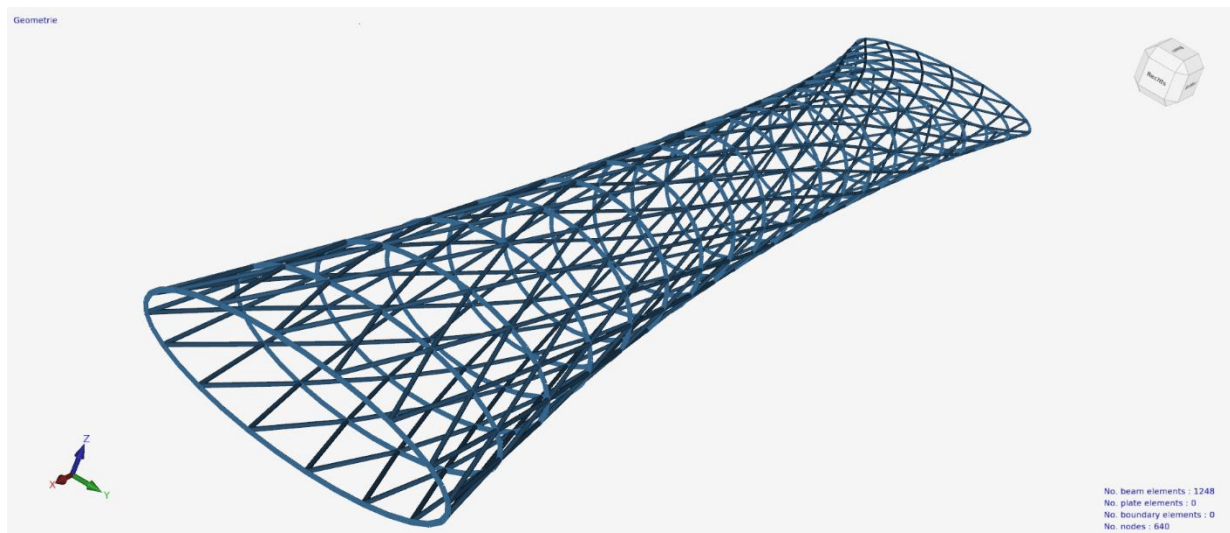
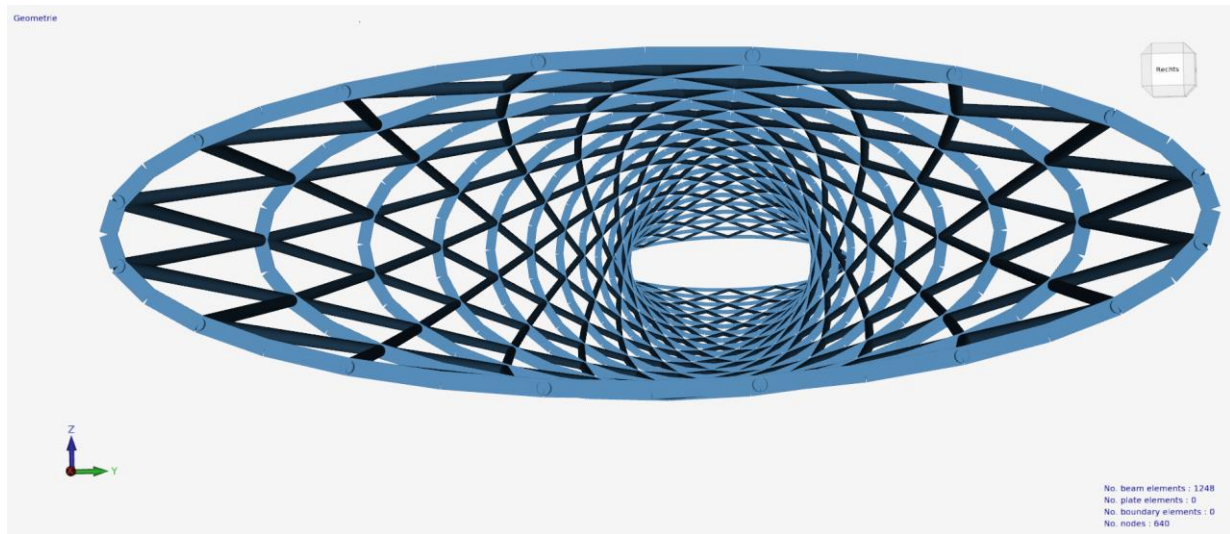
xfem.AddCombination("Permanent",CombTyp.ULS,          1,1.35,1.00,    2,1.50,1.00)
xfem.AddCombination("Veranderlijk",CombTyp.ULS,        1,1.20,1.00,    2,1.50,1.00)
xfem.AddCombination("Wind",CombTyp.ULS,                1,1.20,1.00,    3,1.50,1.00)
xfem.AddCombination("Permanent",CombTyp.SLS,          1,1.00,1.00,    2,1.00,1.00)
xfem.AddCombination("Wind",CombTyp.SLS,                1,1.00,1.00,    3,1.00,1.00)
xfem.AddCombination("BGT Blijvend",CombTyp.PERM,       1,1.00,1.00,    2,1.00,1.00)
xfem.AddCombination("BGT Quasi-blijvend",CombTyp.PERM,1,1.00,1.00,    2,1.00,0.80)

xfem.Save2XFEM4U("truss")

```

Bridge

Inspired by the “Netkous (=mesh stocking)” bridge in Rotterdam



```

def BridgeSection(xfem, x: float, y: float, z: float, a: float, b: float, num : int):
    node = [int]
    node.clear()
    alfa = 0
    xn = x

    for i in range(num):
        yn = y + a * cos(alfa * pi/180)
        zn = z + b * sin(alfa * pi/180)
        node.append(xfem.AddNode(0,xn,yn,zn))
        alfa += 360/num
    for i in range(num):
        j = i+1
        if j == num:
            j=0
        xfem.AddBeam(0,node[i],node[j],1)
    return node
# End of Function

def Linterpolation(x1: float, y1: float, x2: float, y2: float, x: float):
    if x1 == x2:
        return y1
    else :
        return (y2 - y1) / (x2 - x1) * (x - x1) + y1;
# End of Function

def parabolic(x0, y0, x1, y1, x2, y2):
    a = b = c = 0
    if x1 == x2 or x0 == x1:
        return a, b, c
    teller = ((y1 - y2) / (x1 - x2)) - ((y0 - y1) / (x0 - x1))
    noemer = ((x1**2 - x2**2) / (x1 - x2)) - ((x0**2 - x1**2) / (x0 - x1))

    if abs(noemer) < math.ulp(1.0):
        return a, b, c
    a = teller / noemer
    b = (y1 - y2) / (x1 - x2) - ((x1**2 - x2**2) / (x1 - x2)) * a
    c = y0 - a * x0**2 - b * x0
    return a, b, c
# End of Function

def example_bridge():
    # net kous brug
    xfem = XFEMF4U("Example in Python","12345", ConsequenceClass.CC2, 100, "")

    xfem.AddGrid("0 10x10000","A B C D E F G H I J K L M N O P Q R S T U V W X Y Z", \
        "-10000 4x5000", "1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25",\
        "0", "+0")

    xfem.AddCase(1,"Permanent", LoadTyp.Permanent_incl_own_weight, 1.0,1.0,1.0)
    xfem.AddCase(2,"Veranderlijk", LoadTyp.E_storage, 1.0,0.9,0.8)
    xfem.AddCase(3,"Wind", LoadTyp.Wind, 1.0,0.0,0.0)

    xfem.AddSteelProfile(1,"S300x20","S235",90)
    xfem.AddSteelProfile(2,"HFCBS2445X12","S235")
    hbegin = 2200
    hmiddle = 5000
    wbegin = 10000
    wmiddle = 5000
    span = 100000
    nsections = 20
    x = 0
    flag = 1

    # determine the parabolic coefficients for the heigth and width
    ah, bh, ch = parabolic(0,hbegin, span/2, hmiddle, span, hbegin)
    aw, bw, cw = parabolic(0,wbegin, span/2, wmiddle, span, wbegin)

    for n in range(nsections+1) :
        num = 32 # number of nodes in each section
        # heigth - h(x)
        h = ah*math.pow(x,2) + bh*x + ch
        # width - w(x)
        w = aw*math.pow(x,2) + bw*x + cw
        # generate the section(x)
        a = BridgeSection(xfem, x, 0, 0, w, h, num)
        x += span/nsections
        # generate the connecting beam between sections
        if n > 0 :
            for i in range(num):
                if i % 2 == 0:
                    j1 = i+1
                    j2 = i-1
                    if j1 == num:
                        j1= 0

                    if j1 < 0 :
                        j2 = num - 1
                if flag == 1 :
                    xfem.AddBeam(0,a[i],b[j1],2)
                    xfem.AddBeam(0,a[i],b[j2],2)
                else :
                    xfem.AddBeam(0,a[j1],b[i],2)
                    xfem.AddBeam(0,a[j2],b[i],2)

```

```

    flag = -flag
    b = a

    xfem.AddCombination("Permanent",CombTyp.ULS,          1,1.35,1.00,    2,1.50,1.00)
    xfem.AddCombination("Veranderlijk",CombTyp.ULS,        1,1.20,1.00,    2,1.50,1.00)
    xfem.AddCombination("Wind",CombTyp.ULS,                1,1.20,1.00,    3,1.50,1.00)
    xfem.AddCombination("Permanent",CombTyp.SLS,          1,1.00,1.00,    2,1.00,1.00)
    xfem.AddCombination("Wind",CombTyp.SLS,                1,1.00,1.00,    3,1.00,1.00)
    xfem.AddCombination("BGT Blijvend",CombTyp.PERM,       1,1.00,1.00,    2,1.00,1.00)
    xfem.AddCombination("BGT Quasi-blijvend",CombTyp.PERM,1,1.00,1.00,    2,1.00,0.80)

    temp_dir = tempfile.gettempdir()
    filename = temp_dir + "\\bridge.xml"
    with open(filename, "w", encoding="ascii") as file:
        file.writelines(xfem.XML())
    writeDirectCommandsfile(filename)
# End of Function

```