Dynamo-based displacement visualization

1. 共轭梁算法

简而言之，就是构造出了一个特定边界条件下的虚梁，使得其受力下的弯矩和剪力数值上等同于实梁的挠度和转角。

我将这部分算法编成了三个部分，主函数main、引用函数functions、生成图表tables

1.1主函数main

生成beam类后，依次引用各个函数计算虚梁上均布荷载、支座反力、各点弯矩和剪力，最后输出生成图表。

```
from time import sleep
from function import beam
from table import plot1, plot2

beam = beam(12, 6, 0.16)
data = beam.read_data()
list = []
for i in range(1, data.shape[0]):
# for i in range(1, 3):
# for i in range(7000, 7001):
    q = beam.q_calculation(data, i)
    sumQ = beam.R_calculation(q)
    list_M = beam.M_calculation(sumQ, q)
    list_F = beam.F_calculation(sumQ, q)
    if i > 5000 and i < 10000:
        list.append(list_M[5])
    if i ==1:
        plot1(list_M)
    sleep(0.1)
plot2(list)
```

1.2引用函数functions

写了一个beam类，底下是根据共轭梁算法所写的函数

```
import pandas as pd

class beam():
    def __init__(self, number, L, H):
        self.number = number
        self.L = L
        self.H = H
        self.Lm = self.L / self.number
    def read_data(self):
        data = pd.read_excel('DynamicStrain.xlsx', sheet_name='Sheet1')
        return data

    def get_data(self, data, i):
        data_required = data.loc[i]
```

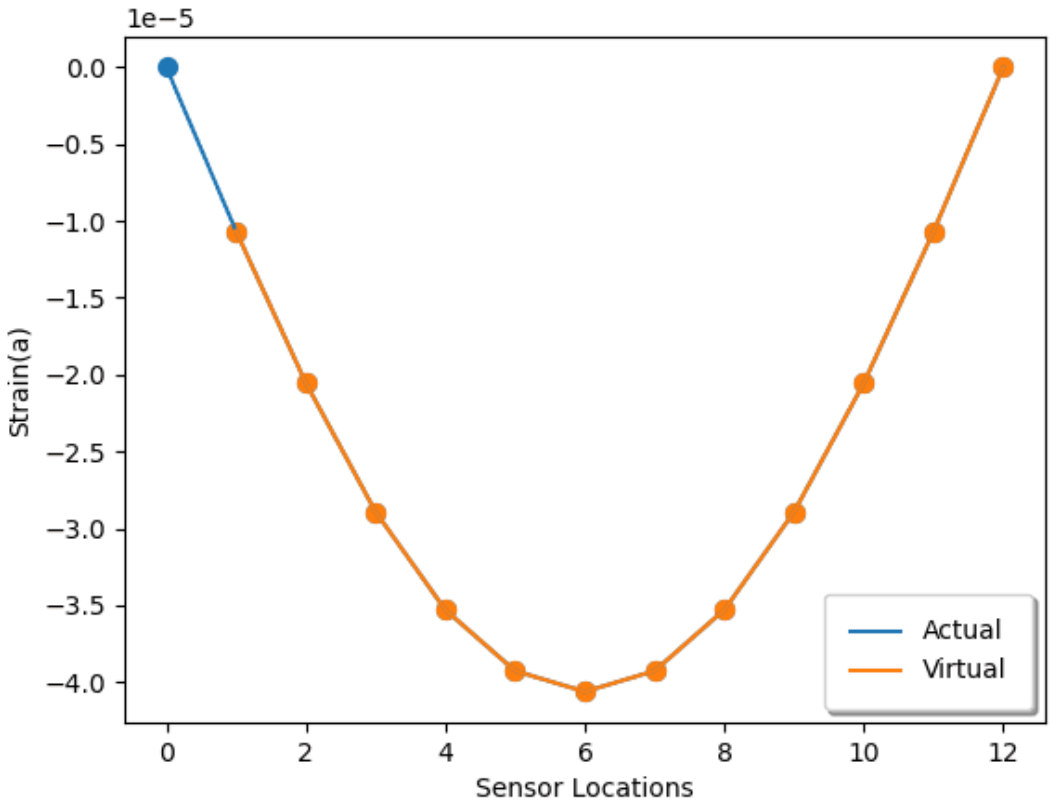
```
        print('\n', f"\033[31mLoop NO.{i+1} times {'='*100}\033[0m")
        return data_required

    def q_calculation(self, data):
        # Strain = beam.get_data(self, data, i)
        # print('\n', f"\033[31mLoop NO.{i+1} times {'='*100}\033[0m")
        Hm = self.H / 2
        q = []
        for i in range(len(data)-1):
            q.append(data[i+1]/Hm)
        return q

    def R_calculation(self, q,):
        SumQ = 0
        Lm=0.5
        number=12
        for i in range(self.number):
            SumQ += q[i] * Lm * (number - i - 0.5)/12
        return SumQ
    def M_calculation(self, SumQ, q):
        Lm=0.5
        # print('SumQ:', SumQ)
        # print('q:', q)
        list_M = []
        for i in range(self.number):
            SumM = -SumQ * Lm * (i+1)
            for j in range(i+1):
                SumM += q[j] * Lm * Lm * (i-j+0.5)
            list_M.append(SumM)
        print('list_M:', list_M)
        return list_M

    def F_calculation(self, SumQ, q):
        Lm=0.5
        list_F = []
        for i in range(self.number):
            SumF = -SumQ
            for j in range(i+1):
                SumF += q[j] * Lm
            list_F.append(SumF)
        # print('list_F:', list_F)
        return list_F
```

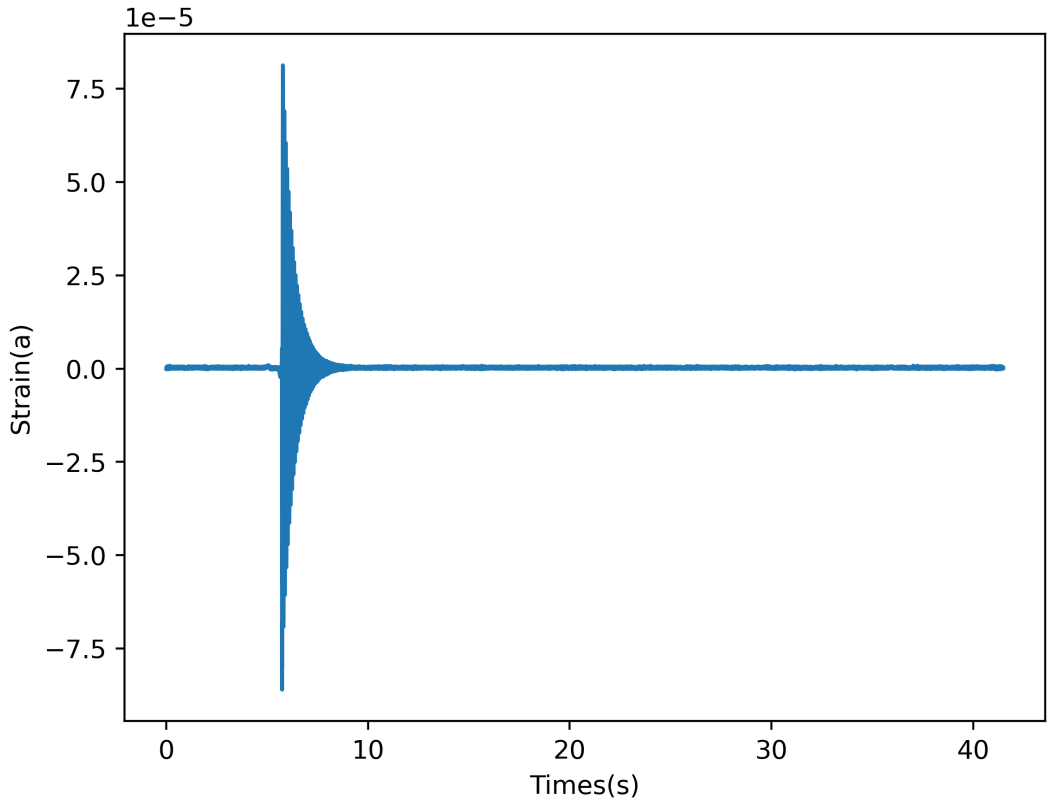
1.3生成图表tables

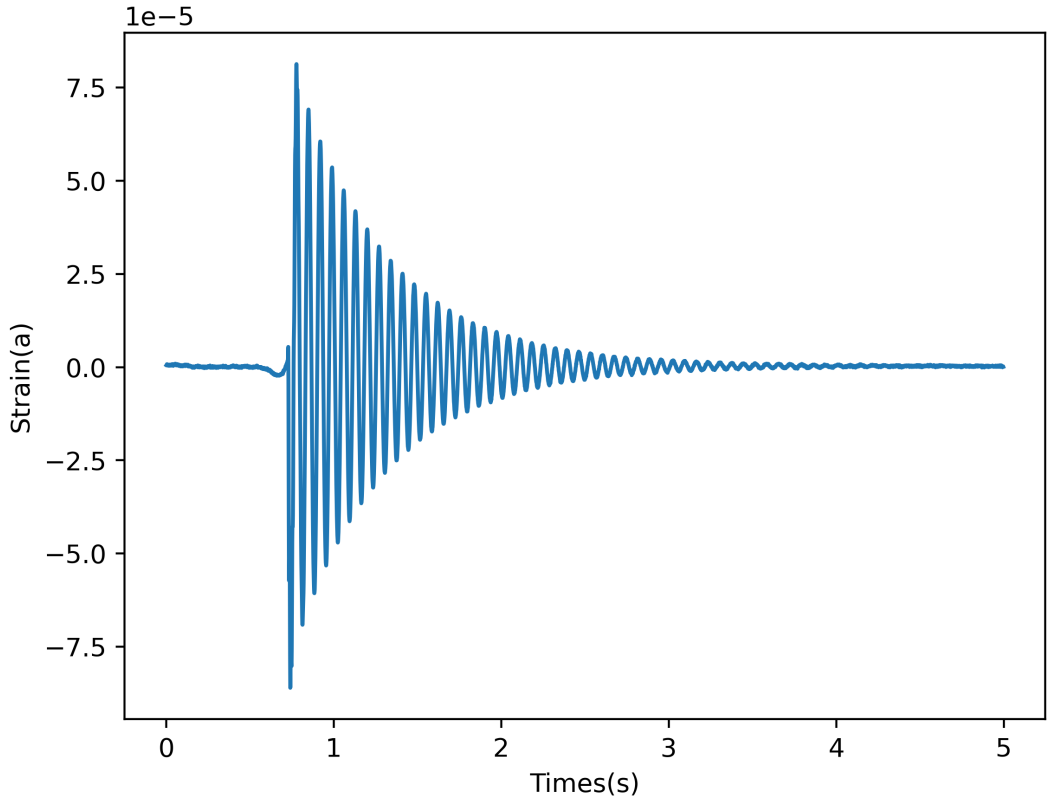
这里生成图表1比对计算数据和实际数据，结果是完全吻合的。图表2表示了一个点随时间变化的位移。图表三为图表二局部放大图。

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

def plot1(M_list):
    data = pd.read_excel('StaticStrain_Displacement.xlsx')
    Displacement = data['Displacement(m)']
    # print(Strain.values)
    plt.plot(Displacement.values)
```

```
    plt.plot(range(1, 13), M_list, label='-r')
    plt.scatter(range(13), Displacement.values, s=50)
    plt.scatter(range(1, 13), M_list, s=50)
    plt.xlabel('Sensor Locations')
    plt.ylabel('Strain(a)')
    plt.legend(
        [ 'Actual', 'Virtual'], loc='lower right', fancybox=True, framealpha=1,
shadow=True, borderpad=True
    )
    plt.show()

def plot2(M_list):
    plt.plot(np.arange(0, len(M_list)*0.001, 0.001), M_list, label='-r')
    plt.xlabel('Times(s)')
    plt.ylabel('Strain(a)')
    plt.savefig(r'C:\Users\夏哲\Desktop\学习文件\学期作业\bim作业\图表2', dpi=400)
    plt.show()
```

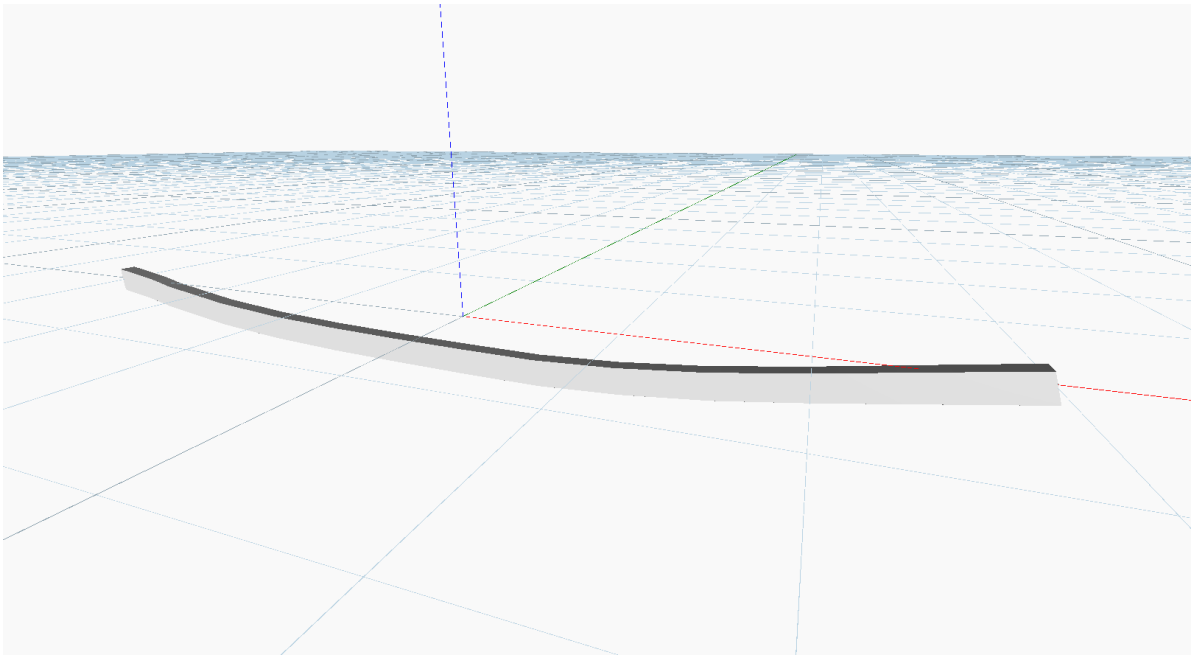




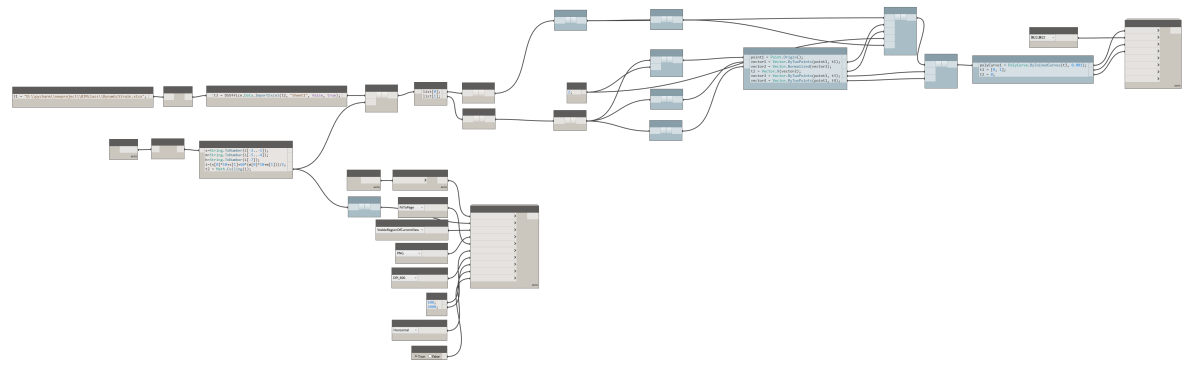


2. Dynamo建模静态梁

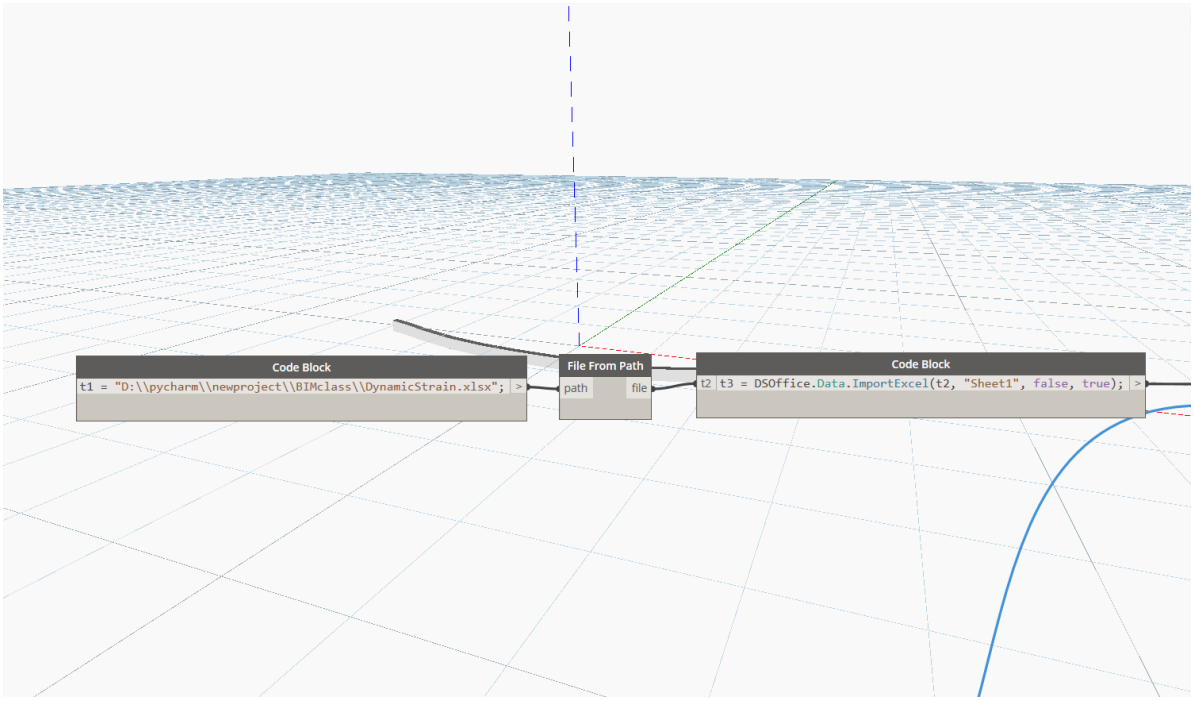
生成梁模型如图。



工作空间如图。



2.1数据导入



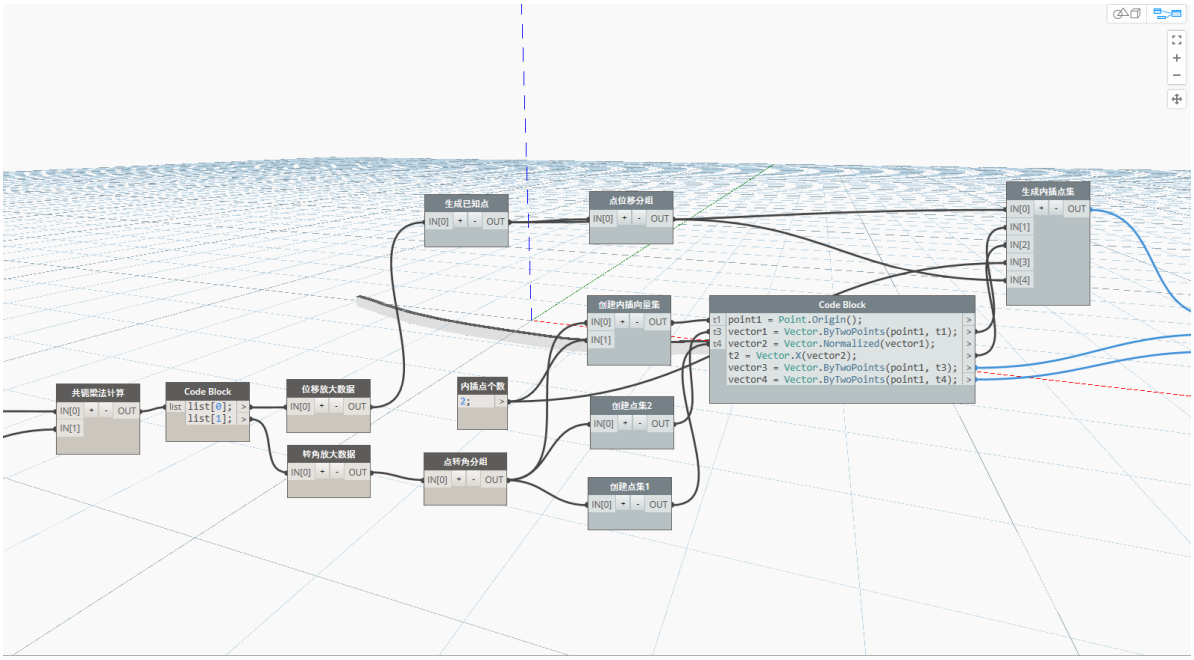
```
t1 = "D:\\pycharm\\newproject\\BIMclass\\DynamicStrain.xlsx";
```

file to path

```
t3 = DSOffice.Data.ImportExcel(t2, "Sheet1", false, true);
```

2.2数据处理

思路是根据已知点的转角和挠度控制整个梁的变形，在已知的两个点之间根据转角等分内插一定数量的点来对梁中轴线进行拟合。以下是数据处理的程序，重点在**分开处理挠度和转角，得到已知点和内插点的坐标和每个点的切向量。**



2.2.1共轭梁算法

分隔线上部分为通用模板，之后代码中省略。

```
# 加载 Python Standard 和 DesignScript 库
import sys
import clr
from time import sleep
# 导入 RevitAPI 和 RevitAPIUI
clr.AddReference("RevitAPI")
clr.AddReference("RevitAPIUI")
import Autodesk
from Autodesk.Revit.DB import *
from Autodesk.Revit.UI import *
#导入dyamo中的几何图元
clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *
#导入revit节点
clr.AddReference("RevitNodes")
from Revit.Elements import *
import Revit
# 导入几何体转换方法(将dynamo中输出的几何体转为revit中的几何体)
clr.ImportExtensions(Revit.GeometryConversion)
# 导入元素转换(revit转dynamo)
clr.ImportExtensions(Revit.Elements)
```

```
================================================================================
================================================

lib=r"D:\pycharm\newproject\BIMclass";
sys.path.append(lib)
from function import *
# 该节点的输入内容将存储为 IN 变量中的一个列表。
dataEnteringNode = IN
# 将代码放在该行下面
beam = beam(12, 6, 0.16)
i=IN[1]
data = IN[0][i+1]
q = beam.q_calculation(data)
sumQ = beam.R_calculation(q)
list_M = beam.M_calculation(sumQ, q)
list_M.insert(0,0)
list_F = beam.F_calculation(sumQ, q)
Insert_data = list_F[len(list_F)-1]
list_F.insert(0,-Insert_data)
OUT = list_M,list_F
```

2.2.2数据放大

为了让结果更明显，将数据等倍放大。两者代码一样。

```
dataEnteringNode = IN
# 将代码放在该行下面
scale=-1E6
list = []
for i in range(len(IN[0])):
    list.append(IN[0][i]*scale)
# 将输出内容指定给 OUT 变量。
OUT = list
```

2.2.3生成点集以及分组

生成已知点

```
dataEnteringNode = IN

# 将代码放在该行下面
out_points = []

for i in range(len(IN[0])):
    x = 0.5*i-3
    z = IN[0][i]
    out_points.append(Point.ByCoordinates(x, 0, z))
OUT = out_points
```

将已知点分组

```
dataEnteringNode = IN

# 将代码放在该行下面

points_list=[]
for i in range(len(IN[0])-1):
    points_sublist=[]
    points_sublist.append(IN[0][i])
    points_sublist.append(IN[0][i+1])
    points_list.append(points_sublist)
# 将输出内容指定给 OUT 变量。
OUT = points_list
```

点转角分组

```
dataEnteringNode = IN

# 将代码放在该行下面

tans_list=[]
for i in range(len(IN[0])-1):
    tans_sublist=[]
    tans_sublist.append(IN[0][i])
    tans_sublist.append(IN[0][i+1])
    tans_list.append(tans_sublist)
# 将输出内容指定给 OUT 变量。
OUT = tans_list
```

2.2.4生成向量集合

创建内插向量点集

```
dataEnteringNode = IN

# 将代码放在该行下面
out_points = []
number=IN[1]
for j in range(len(IN[0])):
    out_subpoints=[]
    for i in range(len(IN[0][1])):
        x=1
        z=math.tan(IN[0][j][i])*x
        point = Point.ByCoordinates(x, 0, z)
        out_subpoints.append(point)
        if i==0:
            aver = (IN[0][j][i+1]-IN[0][j][i])/(number+1)
            for k in range(number):
                z=math.tan(IN[0][j][i]+aver*(k+0.5))*x
                point = Point.ByCoordinates(x, 0, z)
                out_subpoints.append(point)
    out_points.append(out_subpoints)
# 将输出内容指定给 OUT 变量。
OUT = out_points
```

创建点集1

```
dataEnteringNode = IN

# 将代码放在该行下面
out_points = []
for j in range(len(IN[0])):
    out_subpoints=[]
    x=1
    z=math.tan(IN[0][j][1])*x
    point = Point.ByCoordinates(x, 0, z)
    out_subpoints.append(point)
    out_points.append(out_subpoints)
# 将输出内容指定给 OUT 变量。
OUT = out_points
```

创建点集2

```
dataEnteringNode = IN

# 将代码放在该行下面
out_points = []
for j in range(len(IN[0])):
    out_subpoints=[]
    x=1
    z=math.tan(IN[0][j][0])*x
    point = Point.ByCoordinates(x, 0, z)
    out_subpoints.append(point)
    out_points.append(out_subpoints)
# 将输出内容指定给 OUT 变量。
OUT = out_points
```

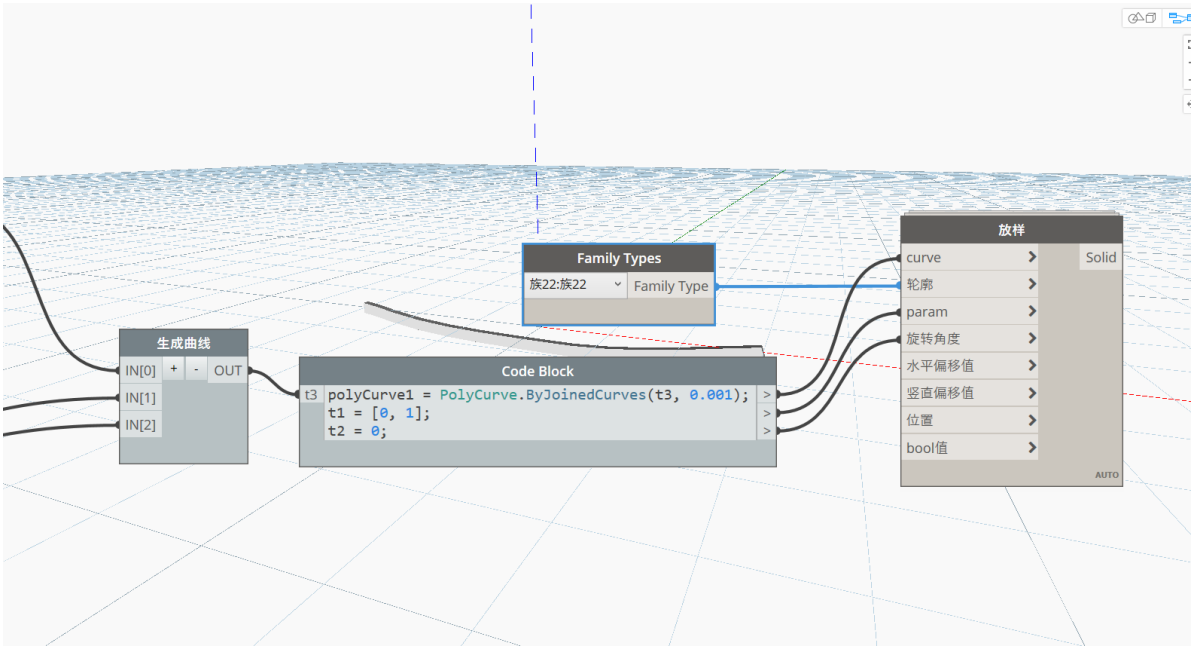
Code Block：向量生成块

```
point1 = Point.Origin();
vector1 = Vector.ByTwoPoints(point1, t1);
vector2 = Vector.Normalized(vector1);
t2 = Vector.X(vector2);
vector3 = Vector.ByTwoPoints(point1, t3);
vector4 = Vector.ByTwoPoints(point1, t4);
```

2.2.5生成内插点集

```
dataEnteringNode = IN

# 将代码放在该行下面
list=IN[4]
for i in range(len(IN[0])-1):
    point1 = IN[0][i]
    for j in range(IN[3]):
        distance = (0.5/(IN[3]+1))/IN[2][i][j+1]
        point2 = Geometry.Translate(point1,IN[1][i][j+1],distance)
        list[i].insert(j+1,point2)
        point1 =point2
# 将输出内容指定给 OUT 变量。
OUT = list
```

2.3生成拟合曲线并生成模型



生成曲线

```
dataEnteringNode = IN

# 将代码放在该行下面
NurbsCurves=[]
for i in range(len(IN[0])):
    NurbsCurves.append(NurbsCurve.ByPointsTangents(IN[0][i],IN[1][i][0],IN[2][i]
[0]))
# 将输出内容指定给 OUT 变量。
OUT = NurbsCurves
```

Code Block：

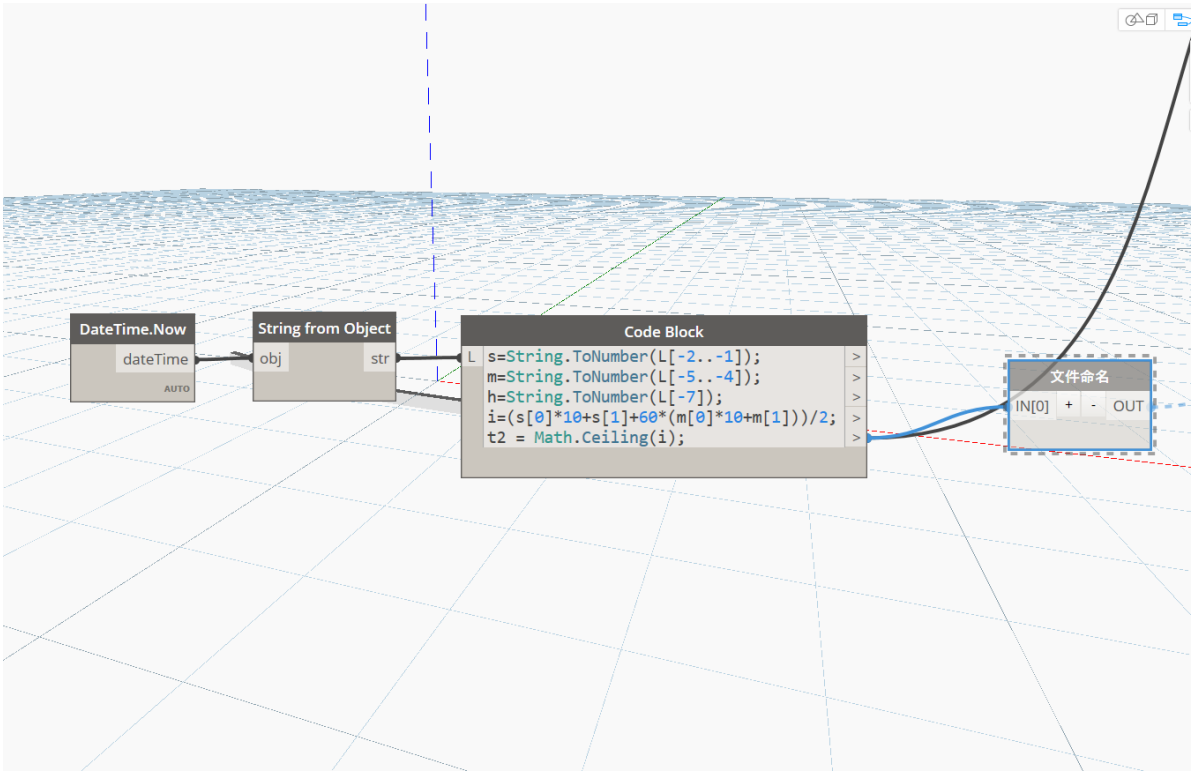
```
polyCurve1 = PolyCurve.ByJoinedCurves(t3, 0.001);
t1 = [0, 1];
t2 = 0;
```

Family Types导入了截面形状族

最后使用了C_Create节点包的放样节点生成模型。

3. Dynamo建模动态梁

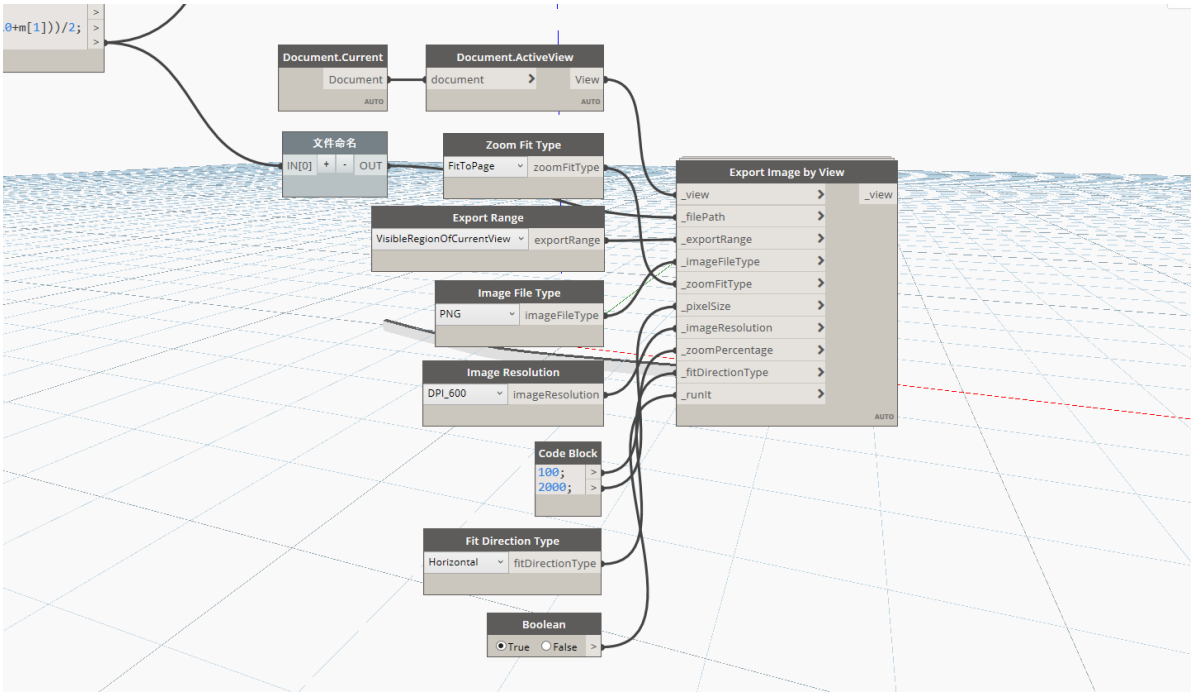
这里我的思路是使用Date Time. Now，通过时间的不断流逝，达到依次读取excel数据生成梁的效果，根据电脑性能我的设置是2秒生成一次。



Code Block：

```
s=String.ToNumber(L[-2..-1]);
m=String.ToNumber(L[-5..-4]);
h=String.ToNumber(L[-7]);
i=(s[0]*10+s[1]+60*(m[0]*10+m[1]))/2;
t2 = Math.Ceiling(i);
```

4.导出图片，制作视频



文件命名

```
dataEnteringNode = IN

# 将代码放在该行下面
t1='D:\新建文件夹\新建文件夹\time={i}'.format(i=IN[0])
# 将输出内容指定给 OUT 变量。
OUT = t1
```

其余内容使用了**archi-lab package**里的 **Package Manager**。用于导出高清图片，因为dynamo自带的图片导出相当模糊。

最后导出图片到文件夹。

视频效果：...