```
# AUTHOR:XIN CHEN

# This is an example code for my additional thesis.

# This code can only be worked in Diana 10.2.

# A series of finite element models of bridges with different span length can be obtained by running this code.



from math import *

# accumulation function

def accumu(list):

    total = 0

    for x in list:

        total += x

        yield total



p = 0                                # a counter to select specific case

for counter in range(17):

    scale       =
[0.60,0.65,0.70,0.75,0.80,0.85,0.90,0.95,1.00,1.05,1.10,1.15,1.20,1.25,1.30,1.35,1.40] # A

list of scale to change the span length

    # variables about loads

    selfweightList =

[0.01,0.011,0.012,0.014,0.015,0.017,0.019,0.021,0.023,0.025,0.027,0.030,0.032,0.035,0.037,0.040,0.043]

    selfweight          = -selfweightList[p]
```

```
dis_load_line1         = -0.014                              # load applied to national lane 1
dis_load_line2         = -0.004                              # load apploed to national lane 2 and
remaining area
tandemLoad_line1       = -300000/2*1.5
tandemLoad_line2       = -200000/2*1.5



# variables about bridge
a                    = 90                          # angle
ES                   = 100                         # element size
List_of_plate_width     = [1000, 3000, 3000, 1400, 1000]     # a list of width for different lanes
bh                   = list(accumu(List_of_plate_width))

bsp1                 = int(10800*scale[p])            # length of span1
bsp2                 = int(15400*scale[p])            # length of span2
Lbh                  = [0]*5
for i in range(5):
        Lbh[i]             = int(bh[i]/tan(a*pi/180))

# variables about cross beams
cbw                  = 800                         # width of cross beam
ind                  = 500                         # indent for support
ch                   = [ind, bh[4]/2, bh[4]-ind]
Lch                  = [0]*3
for i in range(3):
        Lch[i]             = int(ch[i]/tan(a*pi/180))

# thickness
```

```
tspan1 = int(470*scale[p]**2) # basic thickness of span1

tspan2 = int(470*scale[p]**2) # basic thickness of span2

ext = 200



a1list = [112263 ,114746 ,116904,118799 ,120479 ,121980 ,123330
,124554 ,125670

,142810 ,127635 ,128508 ,129320 ,130079 ,130790
,131460 ,132093]

r1list = [112094 ,114547 ,116674,118535 ,120178 ,121640 ,122950
,124130 ,125200

,142292 ,127067 ,127887 ,128644 ,129345 ,129996
,130603 ,131172]

m2list = [11500 ,12425 ,13350 ,14275 ,15200 ,16125 ,17050 ,17975 ,18900

,18900 ,20750 ,21675 ,22600 ,23525 ,24450 ,25375 ,26300 ]

a2list = [62076 ,63023 ,63849 ,64578 ,65229 ,65815 ,66347 ,66834 ,67283

,76428 ,68088 ,68454 ,68800 ,69127 ,69440 ,69740 ,70028 ]

r2list = [61907 ,62824 ,63619 ,64314 ,64928 ,65475 ,65966 ,66409 ,66813

,75910 ,67520 ,67833 ,68123 ,68393 ,68646 ,68883 ,69107 ]


m1 = 800 # geometry information for circular distribution of
thickenss

a1 = a1list[p] # geometry information for circular distribution of
thickenss

r1 = r1list[p] # geometry information for circular distribution of
thickenss

m2 = m2list[p] # geometry information for circular distribution of
thickenss
```

```
a2 = a2list[p]                  # geometry information for circular distribution of thickenss
r2 = r2list[p]                  # geometry information for circular distribution of thickenss
# thickness function of span1
fx1 = range(0, (bsp1+Lbh[4])+1000, 100)          # extra 1000mm to make sure fuction field bigger

than geometry field
fy1 = range(0, bh[4]+1000, 100)
Lfy1 = [0]*len(fy1)
fac1matx = [[0 for x in range(len(fy1))] for y in range(len(fx1))]
fac1list = [0]*len(fx1)*len(fy1)
k = 0
for j in range(len(fy1)):
    Lfy1[j] = int(fy1[j]/tan(a*pi/180))
    for i in range(len(fx1)):
        fac1matx[i][j] = (((-r1)**2-(fx1[i]-m1-Lfy1[j])**2)**0.5-a1)/-tspan1
        fac1list[k] = fac1matx[i][j]
        k += 1


# ---------------------------------------------------
# thickness of sidewalk1
fac3matx = [[0 for x in range(len(fy1))] for y in range(len(fx1))]
fac3list = [0]*len(fx1)*len(fy1)
for k in range(len(fx1)*len(fy1)):
    fac3list[k] = fac1list[k]+ext/tspan1


# ---------------------------------------------------
```

```
# thickness function of span2

fx2 = range(bsp1, bsp1+bsp2+Lbh[4]+1000, 100)

fy2 = range(0, bh[4]+1000, 100)

Lfy2 = [0]*len(fy2)

fac2matx = [[0 for x in range(len(fy2))] for y in range(len(fx2))]

fac2list = [0]*len(fx2)*len(fy2)

k = 0

for j in range(len(fy2)):

    Lfy2[j] = int(fy2[j]/tan(a*pi/180))

    for i in range(len(fx2)):

        fac2matx[i][j] = (((-r2)**2-(fx2[i]-m2-Lfy2[j])**2)**0.5-a2)/-tspan2

        fac2list[k] = fac2matx[i][j]

        k += 1

# --------------------------------------------------

# thickness of sidewalk2

fac4matx = [[0 for x in range(len(fy2))] for y in range(len(fx2))]

fac4list = [0]*len(fx2)*len(fy2)

for k in range(len(fx2)*len(fy2)):

    fac4list[k] = fac2list[k]+ext/tspan2


# thickness monitor

ab1R        = -int(((-r1)**2-(cbw-m1)**2)**0.5-a1       )

ab2L        = -int(((-r1)**2-(bsp1-m1)**2)**0.5-a1       )

ab2R        = -int(((-r2)**2-(cbw+bsp1-m2)**2)**0.5-a2  )

ab3L        = -int(((-r2)**2-(bsp1+bsp2-m2)**2)**0.5-a2 )

tabeam1       = ab1R+200

tabeam2       = ab2L+250
```

```
tabeam3              = ab3L+250


start                = 3000-1100                   # start of axle load
end                  = 7000-1100                   # end   of axle load
step                 = 200                         # step size
p=p+1

# -------------------------------------------------------------------------------

outputDir    = 'D:/AdditionalThesis/model_csv/No6/'            # path of csv file
name_csv     = "angle"+str(a)+","+"bscthc"+str(tspan1)

with open( outputDir + name_csv + '.csv', 'a' ) as file:
  file.writelines('\n'+'angle, '+str(a)+'\n')
  file.writelines('element size, '+str(ES)+'\n')
  file.writelines('length,
'+str(cbw)+','+str(bsp1)+','+str(cbw+bsp1)+','+str(bsp1+bsp2)+','+str(cbw

+bsp1+bsp2)+'\n')
  file.writelines('width,'+str(bh)+'\n')
  file.writelines('basic thickness,'+str(tspan1)+','+str(tspan2)+'\n')
  file.writelines('thickness of across
beam,'+str(tabeam1)+','+str(tabeam2)+','+str(tabeam3)+'\n')
  file.writelines('TS start,'+str(start+1100)+'  end,'+str(end+1100)+'  step,'+str(step)+'\n')
  file.writelines('selfweight,'+str(selfweight)+'\n')

# -------------------------------------------------------------------------------
```

```
for D in range(start, end, step):

        nameDiana    = "angle"+str(a)+","+"bscthc"+str(tspan1)+","+"D"+str(D+1100)
        DianaFileDir =
"D:/AdditionalThesis/model/No6/"+"a"+str(a)+"bscthc"+str(tspan1)+"step"+str

(step)+"/"+nameDiana  # path of Diana file
        analysis     = nameDiana
        output       = "Output linear static analysis"



        newProject( DianaFileDir, 1000 )
        setModelAnalysisAspects( [ "STRUCT" ] )
        setModelDimension( "3D" )
        setDefaultMeshOrder( "LINEAR" )
        setDefaultMesherType( "HEXQUAD" )
        setUnit( "LENGTH", "MM" )
        setUnit( "FORCE", "N" )



        # create the first span
        createSheet( "span1_1" , [[ cbw       , 0     , 0 ],[ bsp1       , 0     , 0 ],[ bsp1+Lch[0] ,
ch[0] , 0 ],[

cbw+Lch[0] , ch[0] , 0 ]] )
        createSheet( "span1_2" , [[ cbw+Lch[0] , ch[0] , 0 ],[ bsp1+Lch[0] , ch[0] , 0 ],[
bsp1+Lbh[0] ,

bh[0] , 0 ],[ cbw+Lbh[0] , bh[0] , 0 ]] )
```

```
		createSheet( "span1_3" , [[ cbw+Lbh[0] , bh[0] , 0 ],[ bsp1+Lbh[0] , bh[0] , 0 ],[
bsp1+Lbh[1] ,

bh[1] , 0 ],[ cbw+Lbh[1] , bh[1] , 0 ]] )
		createSheet( "span1_4" , [[ cbw+Lbh[1] , bh[1] , 0 ],[ bsp1+Lbh[1] , bh[1] , 0 ],[
bsp1+Lch[1] ,

ch[1] , 0 ],[ cbw+Lch[1] , ch[1] , 0 ]] )
		createSheet( "span1_5" , [[ cbw+Lch[1] , ch[1] , 0 ],[ bsp1+Lch[1] , ch[1] , 0 ],[
bsp1+Lbh[2] ,

bh[2] , 0 ],[ cbw+Lbh[2] , bh[2] , 0 ]] )
		createSheet( "span1_6" , [[ cbw+Lbh[2] , bh[2] , 0 ],[ bsp1+Lbh[2] , bh[2] , 0 ],[
bsp1+Lbh[3] ,

bh[3] , 0 ],[ cbw+Lbh[3] , bh[3] , 0 ]] )
		createSheet( "span1_7" , [[ cbw+Lbh[3] , bh[3] , 0 ],[ bsp1+Lbh[3] , bh[3] , 0 ],[
bsp1+Lch[2] ,

ch[2] , 0 ],[ cbw+Lch[2] , ch[2] , 0 ]] )
		createSheet( "span1_8" , [[ cbw+Lch[2] , ch[2] , 0 ],[ bsp1+Lch[2] , ch[2] , 0 ],[
bsp1+Lbh[4] ,

bh[4] , 0 ],[ cbw+Lbh[4] , bh[4] , 0 ]] )




		# create the second span
		createSheet( "span2_1" , [[ bsp1+cbw       , 0     , 0 ],[ bsp1+bsp2       , 0     , 0 ],[
bsp1+Lch

[0]+bsp2 , ch[0] , 0 ],[ bsp1+Lch[0]+cbw , ch[0] , 0 ]] )
```

```
			createSheet( "span2_2" , [[ bsp1+Lch[0]+cbw , ch[0] , 0 ],[ bsp1+Lch[0]+bsp2 , ch[0] ,
0 ],[

bsp1+Lbh[0]+bsp2 , bh[0] , 0 ],[ bsp1+Lbh[0]+cbw , bh[0] , 0 ]] )
			createSheet( "span2_3" , [[ bsp1+Lbh[0]+cbw , bh[0] , 0 ],[ bsp1+Lbh[0]+bsp2 , bh[0] ,
0 ],[

bsp1+Lbh[1]+bsp2 , bh[1] , 0 ],[ bsp1+Lbh[1]+cbw , bh[1] , 0 ]] )
			createSheet( "span2_4" , [[ bsp1+Lbh[1]+cbw , bh[1] , 0 ],[ bsp1+Lbh[1]+bsp2 , bh[1] ,
0 ],[

bsp1+Lch[1]+bsp2 , ch[1] , 0 ],[ bsp1+Lch[1]+cbw , ch[1] , 0 ]] )
			createSheet( "span2_5" , [[ bsp1+Lch[1]+cbw , ch[1] , 0 ],[ bsp1+Lch[1]+bsp2 , ch[1] ,
0 ],[

bsp1+Lbh[2]+bsp2 , bh[2] , 0 ],[ bsp1+Lbh[2]+cbw , bh[2] , 0 ]] )
			createSheet( "span2_6", [[ bsp1+Lbh[2]+cbw , bh[2] , 0 ],[ bsp1+Lbh[2]+bsp2 , bh[2] ,
0 ],[

bsp1+Lbh[3]+bsp2 , bh[3] , 0 ],[ bsp1+Lbh[3]+cbw , bh[3] , 0 ]] )
			createSheet( "span2_7", [[ bsp1+Lbh[3]+cbw , bh[3] , 0 ],[ bsp1+Lbh[3]+bsp2 , bh[3] ,
0 ],[

bsp1+Lch[2]+bsp2 , ch[2] , 0 ],[ bsp1+Lch[2]+cbw , ch[2] , 0 ]] )
			createSheet( "span2_8", [[ bsp1+Lch[2]+cbw , ch[2] , 0 ],[ bsp1+Lch[2]+bsp2 , ch[2] , 0
],[

bsp1+Lbh[4]+bsp2 , bh[4] , 0 ],[ bsp1+Lbh[4]+cbw , bh[4] , 0 ]] )


			# create the first across beam
			createSheet( "cb1" , [[ 0      , 0     , 0 ],[ cbw/2       , 0     , 0 ],[ cbw/2+Lch[0] , ch[0] , 0
],[ Lch

[0] , ch[0] , 0 ]] )
```

```
createSheet( "cb2" , [[ Lch[0] , ch[0] , 0 ],[ cbw/2+Lch[0] , ch[0] , 0 ],[ cbw/2+Lbh[0] ,
bh[0] , 0

],[ Lbh[0] , bh[0] , 0 ]] )
createSheet( "cb3" , [[ Lbh[0] , bh[0] , 0 ],[ cbw/2+Lbh[0] , bh[0] , 0 ],[ cbw/2+Lbh[1] ,
bh[1] , 0

],[ Lbh[1] , bh[1] , 0 ]] )
createSheet( "cb4" , [[ Lbh[1] , bh[1] , 0 ],[ cbw/2+Lbh[1] , bh[1] , 0 ],[ cbw/2+Lch[1] ,
ch[1] , 0

],[ Lch[1] , ch[1] , 0 ]] )
createSheet( "cb5" , [[ Lch[1] , ch[1] , 0 ],[ cbw/2+Lch[1] , ch[1] , 0 ],[ cbw/2+Lbh[2] ,
bh[2] , 0

],[ Lbh[2] , bh[2] , 0 ]] )
createSheet( "cb6" , [[ Lbh[2] , bh[2] , 0 ],[ cbw/2+Lbh[2] , bh[2] , 0 ],[ cbw/2+Lbh[3] ,
bh[3] , 0

],[ Lbh[3] , bh[3] , 0 ]] )
createSheet( "cb7" , [[ Lbh[3] , bh[3] , 0 ],[ cbw/2+Lbh[3] , bh[3] , 0 ],[ cbw/2+Lch[2] ,
ch[2] , 0

],[ Lch[2] , ch[2] , 0 ]] )
createSheet( "cb8" , [[ Lch[2] , ch[2] , 0 ],[ cbw/2+Lch[2] , ch[2] , 0 ],[ cbw/2+Lbh[4] ,
bh[4] , 0

],[ Lbh[4] , bh[4] , 0 ]] )


arrayCopy( [ "cb1", "cb2", "cb3", "cb4", "cb5", "cb6", "cb7", "cb8"],
        [ cbw/2, 0, 0 ], [ 0, 0, 0 ], [ 0, 0, 0 ], 1 )


# create the second across beam
```

```
arrayCopy( [ "cb1", "cb2", "cb3", "cb4", "cb5", "cb6", "cb7", "cb8"],
           [ bsp1, 0, 0 ], [ 0, 0, 0 ], [ 0, 0, 0 ], 1 )
arrayCopy( [ "cb1", "cb2", "cb3", "cb4", "cb5", "cb6", "cb7", "cb8"],
           [ bsp1+cbw/2, 0, 0 ], [ 0, 0, 0 ], [ 0, 0, 0 ], 1 )

# create the third across beam
arrayCopy( [ "cb1", "cb2", "cb3", "cb4", "cb5", "cb6", "cb7", "cb8"],
           [ bsp1+bsp2, 0, 0 ], [ 0, 0, 0 ], [ 0, 0, 0 ], 1 )

setViewPoint( "ISO1" )

# create support
# vertical support
addSet( GEOMETRYSUPPORTSET, "VSset" )
createPointSupport( "verticalSupport", "VSset" )
setParameter( GEOMETRYSUPPORT, "verticalSupport", "AXES", [ 1, 2 ] )
setParameter( GEOMETRYSUPPORT, "verticalSupport", "TRANSL", [ 0, 0, 1 ] )
setParameter( GEOMETRYSUPPORT, "verticalSupport", "ROTATI", [ 0, 0, 0 ] )

attach( GEOMETRYSUPPORT, "verticalSupport", "cb1",  [[ cbw/2+Lch[0]      , ch[0], 0
]] )
attach( GEOMETRYSUPPORT, "verticalSupport", "cb4",  [[ cbw/2+Lch[1]      , ch[1], 0
]] )
attach( GEOMETRYSUPPORT, "verticalSupport", "cb7",  [[ cbw/2+Lch[2]      , ch[2], 0
]] )
attach( GEOMETRYSUPPORT, "verticalSupport", "cb17", [[ cbw/2+Lch[0]+bsp1    ,
ch[0], 0 ]] )
attach( GEOMETRYSUPPORT, "verticalSupport", "cb20", [[ cbw/2+Lch[1]+bsp1    ,
ch[1], 0 ]] )
```

```
        attach( GEOMETRYSUPPORT, "verticalSupport", "cb23", [[ cbw/2+Lch[2]+bsp1    ,
ch[2], 0 ]] )
        attach( GEOMETRYSUPPORT, "verticalSupport", "cb33", [[ cbw/2+Lch[0]+bsp1+bsp2,
ch[0], 0 ]]

)
        attach( GEOMETRYSUPPORT, "verticalSupport", "cb36", [[ cbw/2+Lch[1]+bsp1+bsp2,
ch[1], 0 ]]

)
        attach( GEOMETRYSUPPORT, "verticalSupport", "cb39", [[ cbw/2+Lch[2]+bsp1+bsp2,
ch[2], 0 ]]

)
        # horizantal support
        addSet( GEOMETRYSUPPORTSET, "HSset" )
        createPointSupport( "horizontalSupport", "HSset" )
        setParameter( GEOMETRYSUPPORT, "horizontalSupport", "AXES", [ 1, 2 ] )
        setParameter( GEOMETRYSUPPORT, "horizontalSupport", "TRANSL", [ 1, 0, 0 ] )
        setParameter( GEOMETRYSUPPORT, "horizontalSupport", "ROTATI", [ 0, 0, 0 ] )

        attach( GEOMETRYSUPPORT, "horizontalSupport", "cb39", [[
cbw/2+Lch[2]+bsp1+bsp2, ch[2], 0

]] )


        # load, selfweight
        addSet( GEOMETRYLOADSET, "selfweightSet" )
        createSurfaceLoad( "selfweight", "selfweightSet" )
        setParameter( GEOMETRYLOAD, "selfweight", "FORCE/VALUE", selfweight )
        setParameter( GEOMETRYLOAD, "selfweight", "FORCE/DIRECT", 3 )
```

```
attach( GEOMETRYLOAD, "selfweight", "cb1"   ,  [[ 0      , 0     , 0 ]] )
attach( GEOMETRYLOAD, "selfweight", "cb2"   ,  [[ Lch[0] , ch[0] , 0 ]] )
attach( GEOMETRYLOAD, "selfweight", "cb3"   ,  [[ Lbh[0] , bh[0] , 0 ]] )
attach( GEOMETRYLOAD, "selfweight", "cb4"   ,  [[ Lbh[1] , bh[1] , 0 ]] )
attach( GEOMETRYLOAD, "selfweight", "cb5"   ,  [[ Lch[1] , ch[1] , 0 ]] )
attach( GEOMETRYLOAD, "selfweight", "cb6"   ,  [[ Lbh[2] , bh[2] , 0 ]] )
attach( GEOMETRYLOAD, "selfweight", "cb7"   ,  [[ Lbh[3] , bh[3] , 0 ]] )
attach( GEOMETRYLOAD, "selfweight", "cb8"   ,  [[ Lch[2] , ch[2] , 0 ]] )

attach( GEOMETRYLOAD, "selfweight", "cb"+str(1+8)   ,  [[ 0     +cbw/2 , 0     , 0 ]] )
attach( GEOMETRYLOAD, "selfweight", "cb"+str(2+8)   ,  [[ Lch[0]+cbw/2 , ch[0] , 0 ]] )
attach( GEOMETRYLOAD, "selfweight", "cb"+str(3+8)   ,  [[ Lbh[0]+cbw/2 , bh[0] , 0 ]] )
attach( GEOMETRYLOAD, "selfweight", "cb"+str(4+8)   ,  [[ Lbh[1]+cbw/2 , bh[1] , 0 ]] )
attach( GEOMETRYLOAD, "selfweight", "cb"+str(5+8)   ,  [[ Lch[1]+cbw/2 , ch[1] , 0 ]] )
attach( GEOMETRYLOAD, "selfweight", "cb"+str(6+8)   ,  [[ Lbh[2]+cbw/2 , bh[2] , 0 ]] )
attach( GEOMETRYLOAD, "selfweight", "cb"+str(7+8)   ,  [[ Lbh[3]+cbw/2 , bh[3] , 0 ]] )
attach( GEOMETRYLOAD, "selfweight", "cb"+str(8+8)   ,  [[ Lch[2]+cbw/2 , ch[2] , 0 ]] )

attach( GEOMETRYLOAD, "selfweight", "cb"+str(1+16)  ,  [[ 0     +bsp1 , 0     , 0 ]] )
attach( GEOMETRYLOAD, "selfweight", "cb"+str(2+16)  ,  [[ Lch[0]+bsp1 , ch[0] , 0 ]] )
attach( GEOMETRYLOAD, "selfweight", "cb"+str(3+16)  ,  [[ Lbh[0]+bsp1 , bh[0] , 0 ]] )
attach( GEOMETRYLOAD, "selfweight", "cb"+str(4+16)  ,  [[ Lbh[1]+bsp1 , bh[1] , 0 ]] )
attach( GEOMETRYLOAD, "selfweight", "cb"+str(5+16)  ,  [[ Lch[1]+bsp1 , ch[1] , 0 ]] )
attach( GEOMETRYLOAD, "selfweight", "cb"+str(6+16)  ,  [[ Lbh[2]+bsp1 , bh[2] , 0 ]] )
attach( GEOMETRYLOAD, "selfweight", "cb"+str(7+16)  ,  [[ Lbh[3]+bsp1 , bh[3] , 0 ]] )
attach( GEOMETRYLOAD, "selfweight", "cb"+str(8+16)  ,  [[ Lch[2]+bsp1 , ch[2] , 0 ]] )
```

attach(GEOMETRYLOAD, "selfweight", "cb"+str(1+24) , [[0 +bsp1+cbw/2 , 0 , 0]])

attach(GEOMETRYLOAD, "selfweight", "cb"+str(2+24) , [[Lch[0]+bsp1+cbw/2 , ch[0] , 0]])

attach(GEOMETRYLOAD, "selfweight", "cb"+str(3+24) , [[Lbh[0]+bsp1+cbw/2 , bh[0] , 0]])

attach(GEOMETRYLOAD, "selfweight", "cb"+str(4+24) , [[Lbh[1]+bsp1+cbw/2 , bh[1] , 0]])

attach(GEOMETRYLOAD, "selfweight", "cb"+str(5+24) , [[Lch[1]+bsp1+cbw/2 , ch[1] , 0]])

attach(GEOMETRYLOAD, "selfweight", "cb"+str(6+24) , [[Lbh[2]+bsp1+cbw/2 , bh[2] , 0]])

attach(GEOMETRYLOAD, "selfweight", "cb"+str(7+24) , [[Lbh[3]+bsp1+cbw/2 , bh[3] , 0]])

attach(GEOMETRYLOAD, "selfweight", "cb"+str(8+24) , [[Lch[2]+bsp1+cbw/2 , ch[2] , 0]])

attach(GEOMETRYLOAD, "selfweight", "cb"+str(1+32) , [[0 +bsp1+bsp2 , 0 , 0]])

attach(GEOMETRYLOAD, "selfweight", "cb"+str(2+32) , [[Lch[0]+bsp1+bsp2 , ch[0] , 0]])

attach(GEOMETRYLOAD, "selfweight", "cb"+str(3+32) , [[Lbh[0]+bsp1+bsp2 , bh[0] , 0]])

attach(GEOMETRYLOAD, "selfweight", "cb"+str(4+32) , [[Lbh[1]+bsp1+bsp2 , bh[1] , 0]])

attach(GEOMETRYLOAD, "selfweight", "cb"+str(5+32) , [[Lch[1]+bsp1+bsp2 , ch[1] , 0]])

attach(GEOMETRYLOAD, "selfweight", "cb"+str(6+32) , [[Lbh[2]+bsp1+bsp2 , bh[2] , 0]])

```
attach( GEOMETRYLOAD, "selfweight", "cb"+str(7+32)  ,  [[ Lbh[3]+bsp1+bsp2 , bh[3] ,
0 ]] )
attach( GEOMETRYLOAD, "selfweight", "cb"+str(8+32)  ,  [[ Lch[2]+bsp1+bsp2 , ch[2] ,
0 ]] )



attach( GEOMETRYLOAD, "selfweight", "span1_1" ,  [[ cbw       , 0     , 0 ]] )
attach( GEOMETRYLOAD, "selfweight", "span1_2" ,  [[ cbw+Lch[0] , ch[0] , 0 ]] )
attach( GEOMETRYLOAD, "selfweight", "span1_3" ,  [[ cbw+Lbh[0] , bh[0] , 0 ]] )
attach( GEOMETRYLOAD, "selfweight", "span1_4" ,  [[ cbw+Lbh[1] , bh[1] , 0 ]] )
attach( GEOMETRYLOAD, "selfweight", "span1_5" ,  [[ cbw+Lch[1] , ch[1] , 0 ]] )
attach( GEOMETRYLOAD, "selfweight", "span1_6" ,  [[ cbw+Lbh[2] , bh[2] , 0 ]] )
attach( GEOMETRYLOAD, "selfweight", "span1_7" ,  [[ cbw+Lbh[3] , bh[3] , 0 ]] )
attach( GEOMETRYLOAD, "selfweight", "span1_8" ,  [[ cbw+Lch[2] , ch[2] , 0 ]] )



attach( GEOMETRYLOAD, "selfweight", "span2_1" ,  [[ cbw       , 0     , 0 ]] )
attach( GEOMETRYLOAD, "selfweight", "span2_2" ,  [[ cbw+Lch[0] , ch[0] , 0 ]] )
attach( GEOMETRYLOAD, "selfweight", "span2_3" ,  [[ cbw+Lbh[0] , bh[0] , 0 ]] )
attach( GEOMETRYLOAD, "selfweight", "span2_4" ,  [[ cbw+Lbh[1] , bh[1] , 0 ]] )
attach( GEOMETRYLOAD, "selfweight", "span2_5" ,  [[ cbw+Lch[1] , ch[1] , 0 ]] )
attach( GEOMETRYLOAD, "selfweight", "span2_6" ,  [[ cbw+Lbh[2] , bh[2] , 0 ]] )
attach( GEOMETRYLOAD, "selfweight", "span2_7" ,  [[ cbw+Lbh[3] , bh[3] , 0 ]] )
attach( GEOMETRYLOAD, "selfweight", "span2_8" ,  [[ cbw+Lch[2] , ch[2] , 0 ]] )
```

```
# load, lane 1

addSet( GEOMETRYLOADSET, "lane1Set" )
createSurfaceLoad( "lane1", "lane1Set" )
setParameter( GEOMETRYLOAD, "lane1", "FORCE/VALUE", dis_load_line1 )
setParameter( GEOMETRYLOAD, "lane1", "FORCE/DIRECT", 3 )

attach( GEOMETRYLOAD, "lane1", "cb3"          ,  [[ Lbh[0] , bh[0] , 0 ]] )
attach( GEOMETRYLOAD, "lane1", "cb"+str(3+8)   ,  [[ Lbh[0]+cbw/2 , bh[0] , 0 ]] )
attach( GEOMETRYLOAD, "lane1", "cb"+str(3+16)  ,  [[ Lbh[0]+bsp1 , bh[0] , 0 ]] )
attach( GEOMETRYLOAD, "lane1", "cb"+str(3+24)  ,  [[ Lbh[0]+bsp1+cbw/2 , bh[0] , 0
]] )
attach( GEOMETRYLOAD, "lane1", "cb"+str(3+32)  ,  [[ Lbh[0]+bsp1+bsp2 , bh[0] , 0 ]]
)

attach( GEOMETRYLOAD, "lane1", "span1_3" ,  [[ cbw+Lbh[0] , bh[0] , 0 ]] )

attach( GEOMETRYLOAD, "lane1", "span2_3" ,  [[ bsp1+Lbh[0]+cbw , bh[0] , 0 ]] )

# load, lane2
addSet( GEOMETRYLOADSET, "lane2Set" )
createSurfaceLoad( "lane2", "lane2Set" )
setParameter( GEOMETRYLOAD, "lane2", "FORCE/VALUE", dis_load_line2 )
setParameter( GEOMETRYLOAD, "lane2", "FORCE/DIRECT", 3 )

attach( GEOMETRYLOAD, "lane2", "cb4"   ,  [[ Lbh[1] , bh[1] , 0 ]] )
```

```
attach( GEOMETRYLOAD, "lane2", "cb5"   ,  [[ Lch[1] , ch[1] , 0 ]] )
attach( GEOMETRYLOAD, "lane2", "cb6"   ,  [[ Lbh[2] , bh[2] , 0 ]] )


attach( GEOMETRYLOAD, "lane2", "cb"+str(4+8)   ,  [[ Lbh[1]+cbw/2 , bh[1] , 0 ]] )
attach( GEOMETRYLOAD, "lane2", "cb"+str(5+8)   ,  [[ Lch[1]+cbw/2 , ch[1] , 0 ]] )
attach( GEOMETRYLOAD, "lane2", "cb"+str(6+8)   ,  [[ Lbh[2]+cbw/2 , bh[2] , 0 ]] )

attach( GEOMETRYLOAD, "lane2", "cb"+str(4+16)  ,  [[ Lbh[1]+bsp1 , bh[1] , 0 ]] )
attach( GEOMETRYLOAD, "lane2", "cb"+str(5+16)  ,  [[ Lch[1]+bsp1 , ch[1] , 0 ]] )
attach( GEOMETRYLOAD, "lane2", "cb"+str(6+16)  ,  [[ Lbh[2]+bsp1 , bh[2] , 0 ]] )

attach( GEOMETRYLOAD, "lane2", "cb"+str(4+24)  ,  [[ Lbh[1]+bsp1+cbw/2 , bh[1] , 0
]] )
attach( GEOMETRYLOAD, "lane2", "cb"+str(5+24)  ,  [[ Lch[1]+bsp1+cbw/2 , ch[1] , 0 ]]
)
attach( GEOMETRYLOAD, "lane2", "cb"+str(6+24)  ,  [[ Lbh[2]+bsp1+cbw/2 , bh[2] , 0
]] )

attach( GEOMETRYLOAD, "lane2", "cb"+str(4+32)  ,  [[ Lbh[1]+bsp1+bsp2 , bh[1] , 0 ]]
)
attach( GEOMETRYLOAD, "lane2", "cb"+str(5+32)  ,  [[ Lch[1]+bsp1+bsp2 , ch[1] , 0 ]] )
attach( GEOMETRYLOAD, "lane2", "cb"+str(6+32)  ,  [[ Lbh[2]+bsp1+bsp2 , bh[2] , 0 ]]
)

attach( GEOMETRYLOAD, "lane2", "span1_4" ,  [[ cbw+Lbh[1] , bh[1] , 0 ]] )
attach( GEOMETRYLOAD, "lane2", "span1_5" ,  [[ cbw+Lch[1] , ch[1] , 0 ]] )
attach( GEOMETRYLOAD, "lane2", "span1_6" ,  [[ cbw+Lbh[2] , bh[2] , 0 ]] )
```

```
attach( GEOMETRYLOAD, "lane2", "span2_4" ,  [[ bsp1+Lbh[1]+cbw , bh[1] , 0 ]] )
attach( GEOMETRYLOAD, "lane2", "span2_5" ,  [[ bsp1+Lch[1]+cbw , ch[1] , 0 ]] )
attach( GEOMETRYLOAD, "lane2", "span2_6" ,  [[ bsp1+Lbh[2]+cbw , bh[2] , 0 ]] )


# Quadrilateral force
td          = [ D, D+1000, D+1200, D+2200 ]
createSheet( "tandem1" , [[td[0], bh[0]     , 0 ],[td[1], bh[0]     , 0 ],[td[1], bh[0]+1000,
0 ],[td[0],

bh[0]+1000, 0 ]] )
createSheet( "tandem2" , [[td[0], bh[1]-1000, 0 ],[td[1], bh[1]-1000, 0 ],[td[1], bh[1]
, 0 ],[td

[0], bh[1]     , 0 ]] )
createSheet( "tandem3" , [[td[0], bh[1]     , 0 ],[td[1], bh[1]     , 0 ],[td[1], bh[1]+1000,
0 ],[td[0],

bh[1]+1000, 0 ]] )
createSheet( "tandem4" , [[td[0], bh[2]-1000, 0 ],[td[1], bh[2]-1000, 0 ],[td[1], bh[2]
, 0 ],[td

[0], bh[2]     , 0 ]] )


arrayCopy( [ "tandem1", "tandem2", "tandem3", "tandem4"],  [ 1200, 0, 0 ], [ 0, 0, 0
], [ 0, 0, 0

], 1 )


addSet( GEOMETRYLOADSET, "tandem1Set" )
createSurfaceLoad( "TL1", "tandem1Set" )
```

```
setParameter( GEOMETRYLOAD, "TL1", "LODTYP", "QUADFO" )
setParameter( GEOMETRYLOAD, "TL1", "QUADFO/VALUE", tandemLoad_line1 )
setParameter( GEOMETRYLOAD, "TL1", "QUADFO/DIRECT", 3 )
attachTo( GEOMETRYLOAD, "TL1", "QUADFO/AREA", "span1_3", [[ cbw+Lbh[0] ,
bh[0] , 0 ]] )
attachTo( GEOMETRYLOAD, "TL1", "QUADFO/AREA", "cb3", [[ Lbh[0] , bh[0] , 0 ]] )
attachTo( GEOMETRYLOAD, "TL1", "QUADFO/AREA", "cb11", [[ Lbh[0]+cbw/2 , bh[0] ,
0 ]] )
attach( GEOMETRYLOAD, "TL1", "tandem1", [[td[0]     , bh[0]     , 0]] )
attach( GEOMETRYLOAD, "TL1", "tandem2", [[td[0]     , bh[1]-1000, 0]] )
attach( GEOMETRYLOAD, "TL1", "tandem5", [[td[0]+1200, bh[0]     , 0]] )
attach( GEOMETRYLOAD, "TL1", "tandem6", [[td[0]+1200, bh[1]-1000, 0]] )


addSet( GEOMETRYLOADSET, "tandem2Set" )
createSurfaceLoad( "TL2", "tandem2Set" )
setParameter( GEOMETRYLOAD, "TL2", "LODTYP", "QUADFO" )
setParameter( GEOMETRYLOAD, "TL2", "QUADFO/VALUE", tandemLoad_line2 )
setParameter( GEOMETRYLOAD, "TL2", "QUADFO/DIRECT", 3 )
attachTo( GEOMETRYLOAD, "TL2", "QUADFO/AREA", "span1_4", [[ cbw+Lbh[1] ,
bh[1] , 0 ]] )
attachTo( GEOMETRYLOAD, "TL2", "QUADFO/AREA", "span1_5", [[ cbw+Lch[1] , ch[1]
, 0 ]] )
attachTo( GEOMETRYLOAD, "TL2", "QUADFO/AREA", "cb4", [[ Lbh[1] , bh[1] , 0 ]] )
attachTo( GEOMETRYLOAD, "TL2", "QUADFO/AREA", "cb5", [[ Lch[1] , ch[1] , 0 ]] )
attachTo( GEOMETRYLOAD, "TL2", "QUADFO/AREA", "cb12", [[ Lbh[1]+cbw/2 , bh[1] ,
0 ]] )
attachTo( GEOMETRYLOAD, "TL2", "QUADFO/AREA", "cb13", [[ Lch[1]+cbw/2 , ch[1] ,
0 ]] )
attach( GEOMETRYLOAD, "TL2", "tandem3", [[td[0]     , bh[1]     , 0]] )
```

```
attach( GEOMETRYLOAD, "TL2", "tandem4", [[td[0]     , bh[2]-1000, 0]] )
attach( GEOMETRYLOAD, "TL2", "tandem7", [[td[0]+1200, bh[1]     , 0]] )
attach( GEOMETRYLOAD, "TL2", "tandem8", [[td[0]+1200, bh[2]-1000, 0]] )

setViewPoint( "TOP" )

# set load combination
setDefaultGeometryLoadCombinations(  )
addGeometryLoadCombination( "" )
setGeometryLoadCombinationFactor( "Geometry load combination 6",
"selfweightSet", 1 )
setGeometryLoadCombinationFactor( "Geometry load combination 6", "lane1Set", 1
)
setGeometryLoadCombinationFactor( "Geometry load combination 6", "lane2Set", 1
)
setGeometryLoadCombinationFactor( "Geometry load combination 6",
"tandem1Set", 1 )
setGeometryLoadCombinationFactor( "Geometry load combination 6",
"tandem2Set", 1 )

# material property
plates = [ "span1_1", "span1_2", "span1_3", "span1_4", "span1_5", "span1_6",
"span1_7",

"span1_8",
           "span2_1", "span2_2", "span2_3", "span2_4", "span2_5", "span2_6",

"span2_7", "span2_8",
```

```
                    "cb1", "cb2", "cb3", "cb4", "cb5", "cb6", "cb7", "cb8",
                    "cb9", "cb10", "cb11", "cb12", "cb13", "cb14", "cb15", "cb16",
                    "cb17", "cb18", "cb19", "cb20", "cb21", "cb22", "cb23", "cb24",
                    "cb25", "cb26", "cb27", "cb28", "cb29", "cb30", "cb31", "cb32",
                    "cb33", "cb34", "cb35", "cb36", "cb37", "cb38", "cb39", "cb40" ]

setElementClassType( SHAPE, plates, "PLATE" )

addMaterial( "concrete", "CONCR", "LEI", [] )
setParameter( MATERIAL, "concrete", "LINEAR/ELASTI/YOUNG", 40000 )
setParameter( MATERIAL, "concrete", "LINEAR/ELASTI/POISON", 0.2 )
clearReinforcementAspects( plates )
assignMaterial( "concrete", SHAPE, plates )

# thickness
span1  = [ "span1_3", "span1_4", "span1_5", "span1_6" ]

span2  = [ "span2_3", "span2_4", "span2_5", "span2_6" ]

abeam1 = [ "cb1", "cb2", "cb3", "cb4", "cb5", "cb6", "cb7", "cb8",
                    "cb9", "cb10", "cb11", "cb12", "cb13", "cb14", "cb15", "cb16"]

abeam2 = ["cb17", "cb18", "cb19", "cb20", "cb21", "cb22", "cb23", "cb24",
                    "cb25", "cb26", "cb27", "cb28", "cb29", "cb30", "cb31", "cb32"]

abeam3 = ["cb33", "cb34", "cb35", "cb36", "cb37", "cb38", "cb39", "cb40"]
```

```
sidewalk1 = [ "span1_1", "span1_2", "span1_7", "span1_8" ]

sidewalk2 = [ "span2_1", "span2_2", "span2_7", "span2_8" ]

# --------------------------------------------
addGeometry( "span1", "SHEET", "PLATE", [] )
setFunctionValues( "fsp1", fx1, fy1, [  ], fac1list )
setGeometryFunction( "span1", "THICK", "fsp1" )
setParameter( GEOMET, "span1", "THICK", tspan1 )
setParameter( GEOMET, "span1", "LOCAXS", True )
assignGeometry( "span1", SHAPE, span1 )

# --------------------------------------------
addGeometry( "span2", "SHEET", "PLATE", [] )
setFunctionValues( "fsp2", fx2, fy2, [  ], fac2list )
setGeometryFunction( "span2", "THICK", "fsp2" )
setParameter( GEOMET, "span2", "THICK", tspan2 )
setParameter( GEOMET, "span2", "LOCAXS", True )
assignGeometry( "span2", SHAPE, span2 )

# --------------------------------------------
addGeometry( "sidewalk1", "SHEET", "PLATE", [] )
setFunctionValues( "fsidewalk1", fx1, fy1, [  ], fac3list )
setGeometryFunction( "sidewalk1", "THICK", "fsidewalk1" )
setParameter( GEOMET, "sidewalk1", "THICK", tspan1 )
setParameter( GEOMET, "sidewalk1", "LOCAXS", True )
assignGeometry( "sidewalk1", SHAPE, sidewalk1 )
```

```
# ---------------------------------------------
addGeometry( "sidewalk2", "SHEET", "PLATE", [] )
setFunctionValues( "fsidewalk2", fx2, fy2, [  ], fac4list )
setGeometryFunction( "sidewalk2", "THICK", "fsidewalk2" )
setParameter( GEOMET, "sidewalk2", "THICK", tspan2 )
setParameter( GEOMET, "sidewalk2", "LOCAXS", True )
assignGeometry( "sidewalk2", SHAPE, sidewalk2 )

# ---------------------------------------------
addGeometry( "abeam1", "SHEET", "PLATE", [] )
setParameter( GEOMET, "abeam1", "THICK", tabeam1 )
setParameter( GEOMET, "abeam1", "LOCAXS", True )
assignGeometry( "abeam1", SHAPE, abeam1 )

addGeometry( "abeam2", "SHEET", "PLATE", [] )
setParameter( GEOMET, "abeam2", "THICK", tabeam2 )
setParameter( GEOMET, "abeam2", "LOCAXS", True )
assignGeometry( "abeam2", SHAPE, abeam2 )

addGeometry( "abeam3", "SHEET", "PLATE", [] )
setParameter( GEOMET, "abeam3", "THICK", tabeam3 )
setParameter( GEOMET, "abeam3", "LOCAXS", True )
assignGeometry( "abeam3", SHAPE, abeam3 )
# ------------------------------------------------------

resetElementData( SHAPE, plates )
```

```
# mesh
setElementSize( plates, ES, -1, True )
setMesherType( plates, "HEXQUAD" )
generateMesh( [] )
hideView( "GEOM" )
showView( "MESH" )

# run
addAnalysis( analysis )
addAnalysisCommand( analysis, "LINSTA", "Structural linear static" )
runSolver( analysis )
showView( "RESULT" )

# set results
setResultCase( [ analysis, "Output linear static analysis", "Load-combination 6" ] )
setResultPlot( "contours", "Distributed Moments/node", "Mxx" )

# post-process
# results process
rCase         = resultCases( analysis, output )
rCase         = rCase[5]
rLabel        = 'Distributed Moments'
rComp          = 'Mxx'
rLayer        = resultLayer()
rElem1         = elementsInElementSet( "span1_1" )
rElem2         = elementsInElementSet( "span1_2" )
```

```
rElem3          = elementsInElementSet( "span1_3" )
rElem4          = elementsInElementSet( "span1_4" )
rElem5          = elementsInElementSet( "span1_5" )
rElem6          = elementsInElementSet( "span1_6" )
rElem7          = elementsInElementSet( "span1_7" )
rElem8          = elementsInElementSet( "span1_8" )
rElements       = list( rElem1 + rElem3 + rElem3 + rElem4 + rElem5 + rElem6 + rElem7
+

rElem8 )
rTab            = [ analysis, output, rCase, rLabel, rComp ]      # this is the "resultTable"
mxxTable        = resultData( rTab, rElements )
mxxList         = [ row[3] for row in mxxTable ]              # A list of distributed
moment
nodeIdList      = [ row[1] for row in mxxTable ]              # A list of nodeid
mxx_min         = min( mxxList )
index_mxx_min   = mxxList.index(mxx_min)                      # search for index based
on

mxx_min
mxx_nodeid      = nodeIdList[index_mxx_min]
coordinateList  = nodeCoordinates([int(mxx_nodeid)])          # return coordinate of

mxx_min


# create probe curve
coordinate      = coordinateList[0]
xcoord          = int(coordinate[0])                          # x-coordinate of max_min
ycoord                  = int(coordinate[1])                  # y-coordinate of max_min
rangeX          = 500                                         # horizontal range of probe curve
```

```
                rangeY                    = 500                          # verticle range of probe
curve
                boundaryLeft     = xcoord - rangeX
                boundaryRight    = xcoord + rangeX
                boundaryTop      = ycoord + rangeY
                boundaryBottom   = ycoord - rangeY

        # set probe curve
                setViewSettingValue( "view setting", "RESULT/PROBE/CURVES(1)/COORDS", [
boundaryLeft,
ycoord, 0, boundaryRight, ycoord, 0 ])
                setViewSettingValue( "view setting", "RESULT/PROBE/CURVES(2)/COORDS", [ xcoord,
boundaryBottom, 0, xcoord, boundaryTop, 0 ])


        # retrieve data from probe curve
                PCcoordTable_1   = probeCurveSamplePoints('probe-curve')
                PCcoordX1        = [row[0] for row in PCcoordTable_1]
                PCcoordY1        = [row[1] for row in PCcoordTable_1]
                PCvalue_1        = probeCurveSampleValues('probe-curve')
                PCcoordTable_2   = probeCurveSamplePoints('probe-curve 2')
                PCcoordX2        = [row[0] for row in PCcoordTable_2]
                PCcoordY2        = [row[1] for row in PCcoordTable_2]
                PCvalue_2        = probeCurveSampleValues('probe-curve 2')

                # decide corresponding thickness of probe curve points
```

```
PCthickness_1    = [0] * 11           # A list to store thickness data of probe curve 1
PCthickness_2    = [0] * 11           # A list to store thickness data of probe curve 2
LPCcoordY1       = [0] * 11
LPCcoordY2       = [0] * 11
final_mxxList_1  = [0] * 11           # A list to store mxx/d of probe curve 1
final_mxxList_2  = [0] * 11           # A list to store mxx/d of probe curve 2

for i in range(11):
    LPCcoordY1[i] = int(PCcoordY1[i]/tan(a*pi/180))
    if PCcoordX1[i] <= bsp1+cbw/2:
        PCthickness_1[i] = ((-r1)**2-(PCcoordX1[i]-m1-
LPCcoordY1[i])**2)**0.5-a1
    else:
        PCthickness_1[i] = ((-r2)**2-(PCcoordX1[i]-m2-
LPCcoordY1[i])**2)**0.5-a2
for i in range(11):
    LPCcoordY2[i] = int(PCcoordY1[i]/tan(a*pi/180))
    if PCcoordX2[i] <= bsp1+cbw/2:
        PCthickness_2[i] = ((-r1)**2-(PCcoordX2[i]-m1-
LPCcoordY2[i])**2)**0.5-a1
    else:
        PCthickness_2[i] = ((-r2)**2-(PCcoordX2[i]-m2-
LPCcoordY2[i])**2)**0.5-a2

for i in range(11):
    if PCcoordY1[i] <= bh[0] or PCcoordY1[i] >= bh[3]:
        PCthickness_1[i] = PCthickness_1[i]-ext
for i in range(11):
    if PCcoordY2[i] <= bh[0] or PCcoordY2[i] >= bh[3]:
```

```
PCthickness_2[i] = PCthickness_2[i]-ext

for i in range (11):
    final_mxxList_1[i] = int(PCvalue_1[i]/(PCthickness_1[i]))
    final_mxxList_2[i] = int(PCvalue_2[i]/(PCthickness_2[i]))

final_mxxList_all = final_mxxList_1 + final_mxxList_2
final_mxx = sum(final_mxxList_all) / float(len(final_mxxList_all))

with open( outputDir + name_csv + '.csv', 'a' ) as file:
  file.writelines('('+str(int(xcoord))+','+str(int(ycoord))+','+str(int(-final_mxx))+','+str(D
+1100)+'),')
```