Appendices

A

The Grasshopper model: from design to work procedure

This appendix describes the model made in grasshopper. The model is a combination of three models as described in the report. The first part consists of the design model, the second part is the engineering model, the third part is the robotic model. The input is shown on the left as purple, the models are shown as red, green and blue respectively, and the output is shown on the right in orange.

The input parameters can be found in table 11.1, with a few additional inputs for the transition between models and programs, like the file imports. The output consists of the visualisation of the vault's design, the information for each following model, the input for the structural analysis in Python and the poses and additional information for RoboDK.

The following images are present in this appendix to illustrate and show the Grasshopper model:

- Figure A.1: The entire canvas in the Grasshopper model, also showing the work space created in Grasshopper.
- Figure A.2: The canvas is represented as a process in a diagram.
- Figure A.3: A detail of the map projection cluster from chapter 4.
- Figure A.4: The entire canvas again, with enougb resolution for visible components.

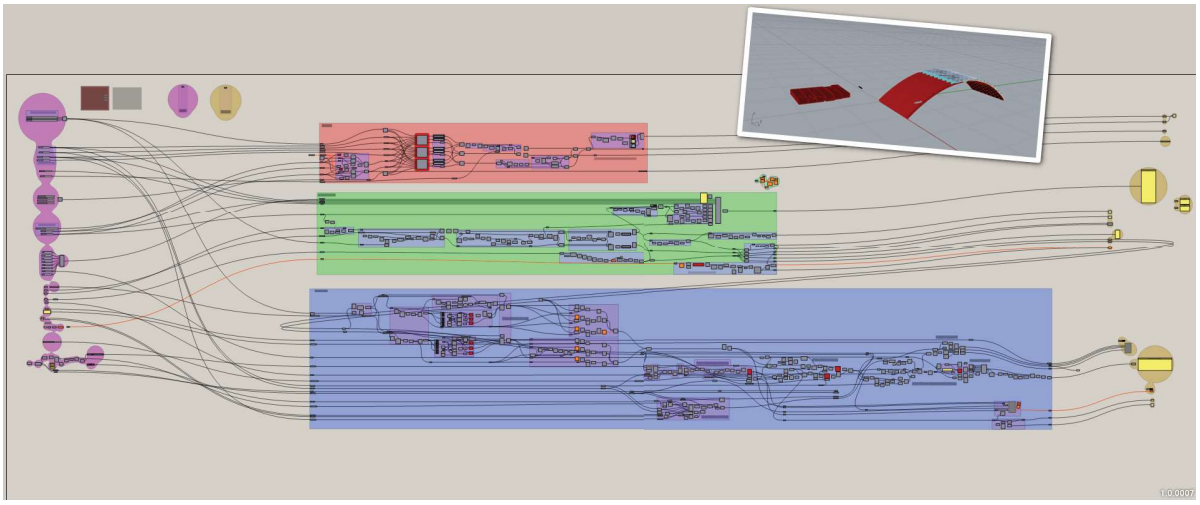


Figure A.1: The complete canvas in Grasshopper. On the left is the input, on the right is the output. The red box is the design model, the green box is the engineering and the blue box is the robotics.

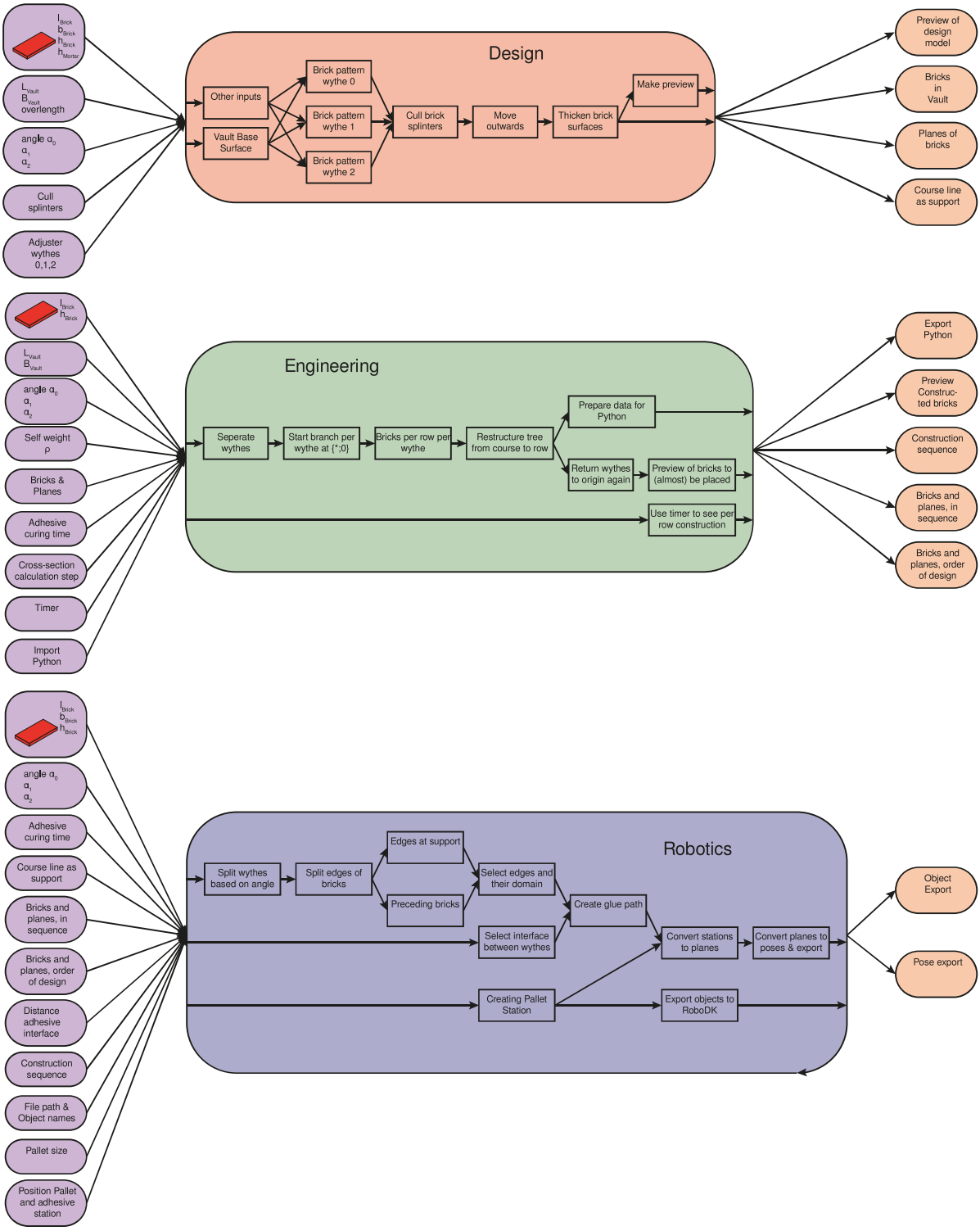
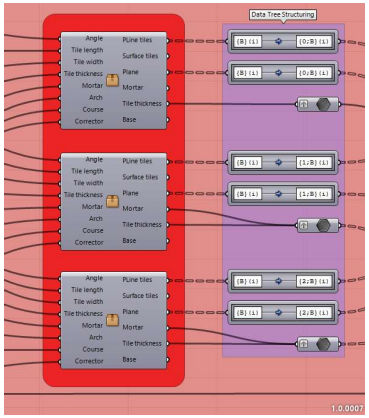
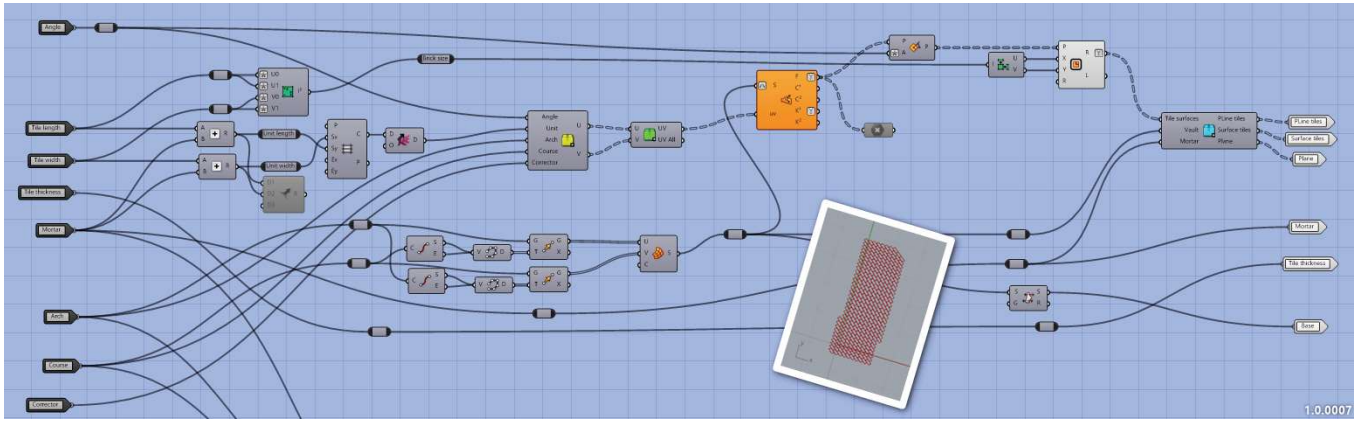


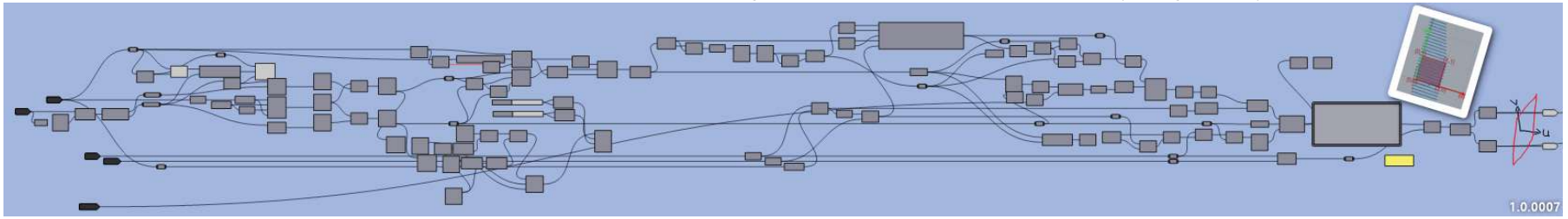
Figure A.2: The workflow of the Grasshopper model based on the groups created on the Grasshopper canvas



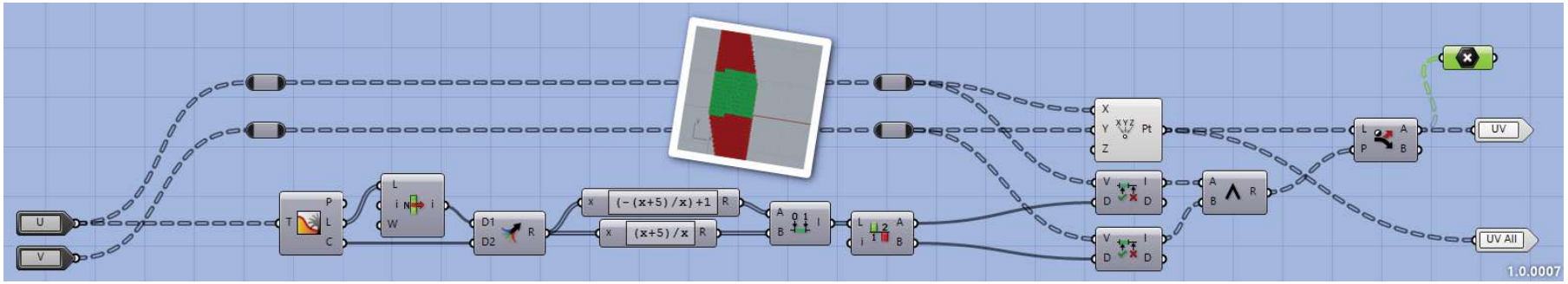
(a) The tiling cluster within the GH canvas;



(b) The tiling cluster exploded, with three more clusters visible in yellow, green and cyan;

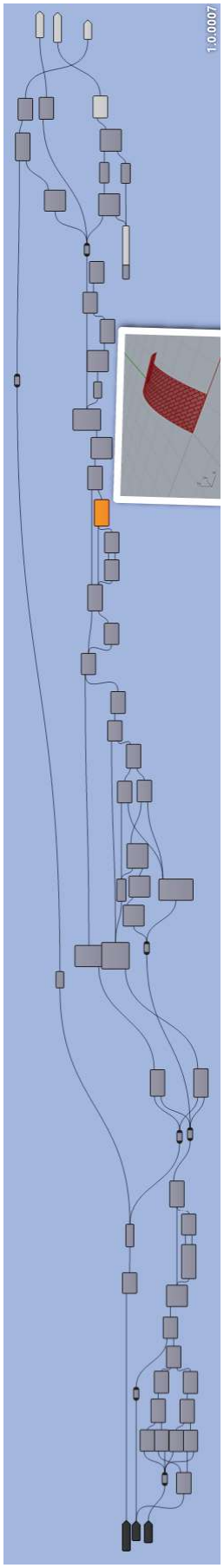


(c) Tiling the surface: the yellow cluster;



(d) Reducing points to compute: the green cluster;

Figure A.3: The tiling cluster exploded.



(e) Project points from unit surface to base vault surface: the cyan cluster;

Figure A.3: The tiling cluster exploded.

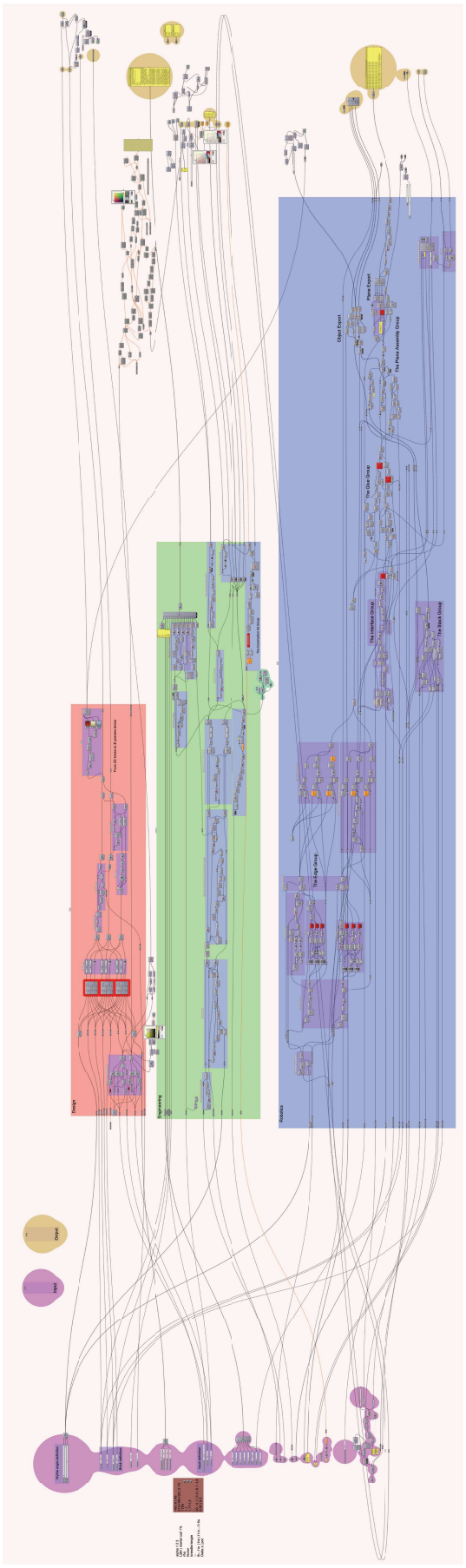


Figure A.4: A detailed and (digitally) zoomable figure of the canvas.

B

DIANA: modelling the monolithic property

This chapter provides the full DIANA model used for section 6.4.1

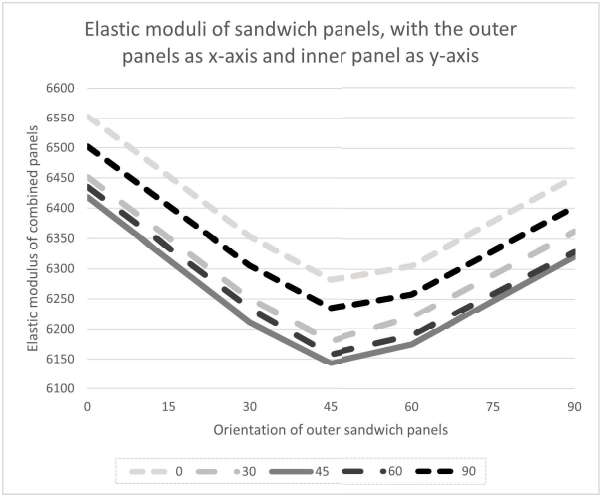


Figure B.1: The elastic modulus at different orientations in a three wythe sandwich panel found with DIANA.

Report 3 panels 1

Contents

Chapter 1

Project information

Diana project name	W:/student-homes/w/jwelles/Documents/Master_Thesis/Rhino Files/Untitled.dpf
Analysis aspects	[‘Structural’]
Model dimension	[‘Three dimensional’]
Default mesher type	HEXQUAD
Default mesher order	QUADRATIC

Diana version	Diana 10.4, Latest update: 2021-03-05 13:13:13
System	Windows NT 6.2 Build 9200
Model sizebox	10.0

Units

The following units are applied

Quantity	Unit	Symbol
Length	meter	m
Mass	kilogram	kg
Force	newton	N
Time	second	s
Temperature	kelvin	K
Angle	radian	rad

Directions

The following directions are defined:

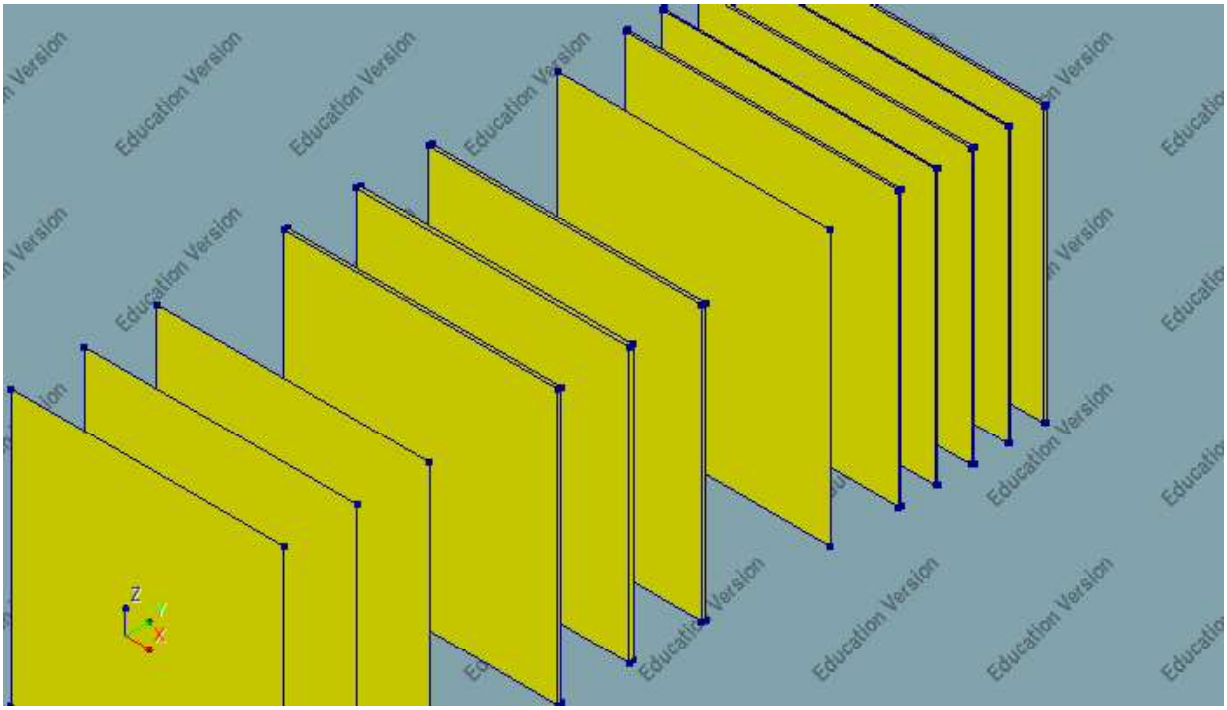
Name	X	Y	Z
X	1	0	0
Y	0	1	0
Z	0	0	1

Definitions

Name	Value
Acceleration of gravity	-9.81 m/s2
Fluid density	1000 kg/m3
Reference point for total head	0 0 0

Rayleigh damping coefficients	a: 0 b: 0
Design safety factor concrete compressive strength	1
Design safety factor concrete uniax. tensile strength	1
Design safety factor concrete stiffness	1
Design safety factor steel yield stress	1
Design safety factor steel stiffness	1
Direction of gravity	Z

Chapter 2



The model consists of the following shapes, reinforcements, piles and interfaces:

Shapes

Name	Set	Element Class	Material	Geometry	Seeding method	Element size [m]	Division
Wythe 1	Shapes	CURSHL	Brick 1	Masonry wall x=(1;0;0)	Divisions	0	19
Wythe 2	Shapes	CURSHL	Brick 1	Masonry wall x=(1;0;var)	Divisions	0	19
Wythe 3	Shapes	CURSHL	Brick 1	Masonry wall x=(1;0;0)	Divisions	0	19
Sheet 1	Shapes 1	CURSHL	Brick	Masonry wall x= 1	Divisions	0	19
Block 1	Shapes 2	STRSOL	Brick no gap	Wall no gap		0	0
Block 2	Shapes 2	STRSOL	Brick no gap	Wall no gap		0	0
Block 3	Shapes 2	STRSOL	Brick no gap	Wall no gap	Divisions	0	9
Block 4	Shapes 3	STRSOL	Brick no gap	Wall no gap		0	0
Block 5	Shapes 3	STRSOL	Bed no gap	Wall no gap		0	0
Block 6	Shapes 3	STRSOL	Brick no gap	Wall no gap		0	0
Block 7	Shapes 3	STRSOL	Bed no gap	Wall no gap		0	0
Block 8	Shapes 3	STRSOL	Brick no gap	Wall no gap		0	0

Interfaces

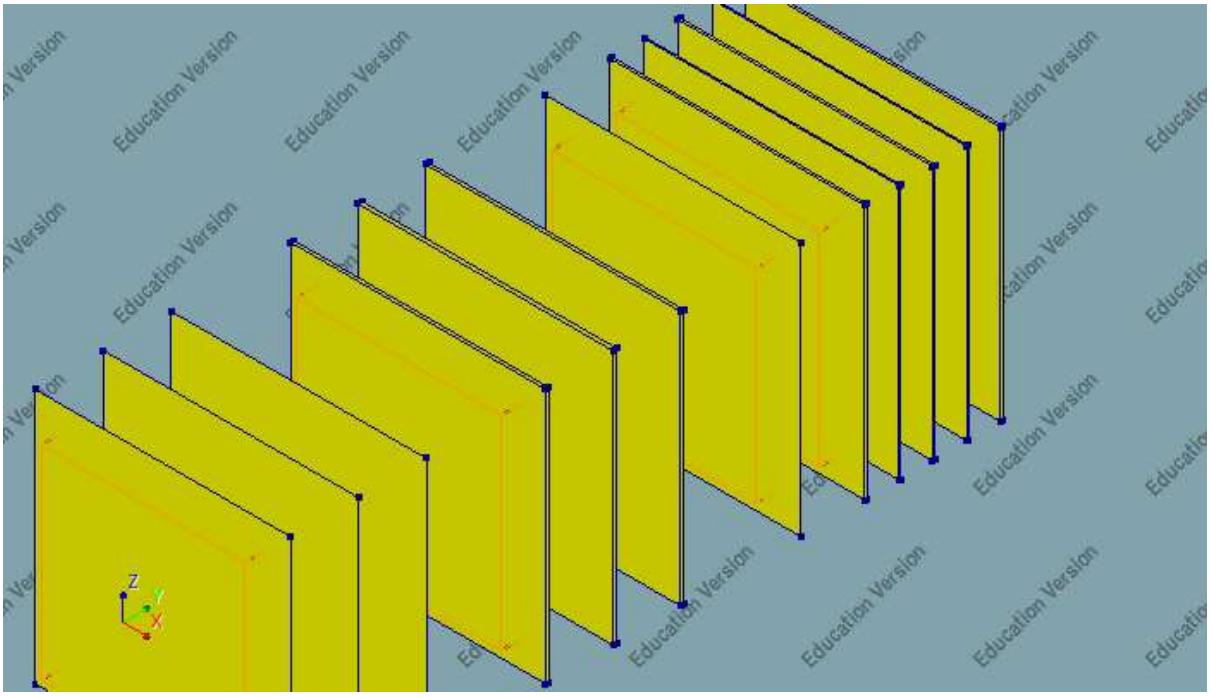
Name	Interface Type	Element Class	Material
Wythe 1	Interface	STPLIF	Wythe bed
Wythe 2	Interface	STPLIF	Wythe bed
Wythe 3			
Block 1	Interface	STPLIF	Wythe bed
Block 2			
Block 2	Interface	STPLIF	Wythe bed
Block 3			

Dimensions

Axes	Minimum coordinate [m]	Maximum coordinate [m]
X	-1	1
Y	-2	2.6
Z	-1	1

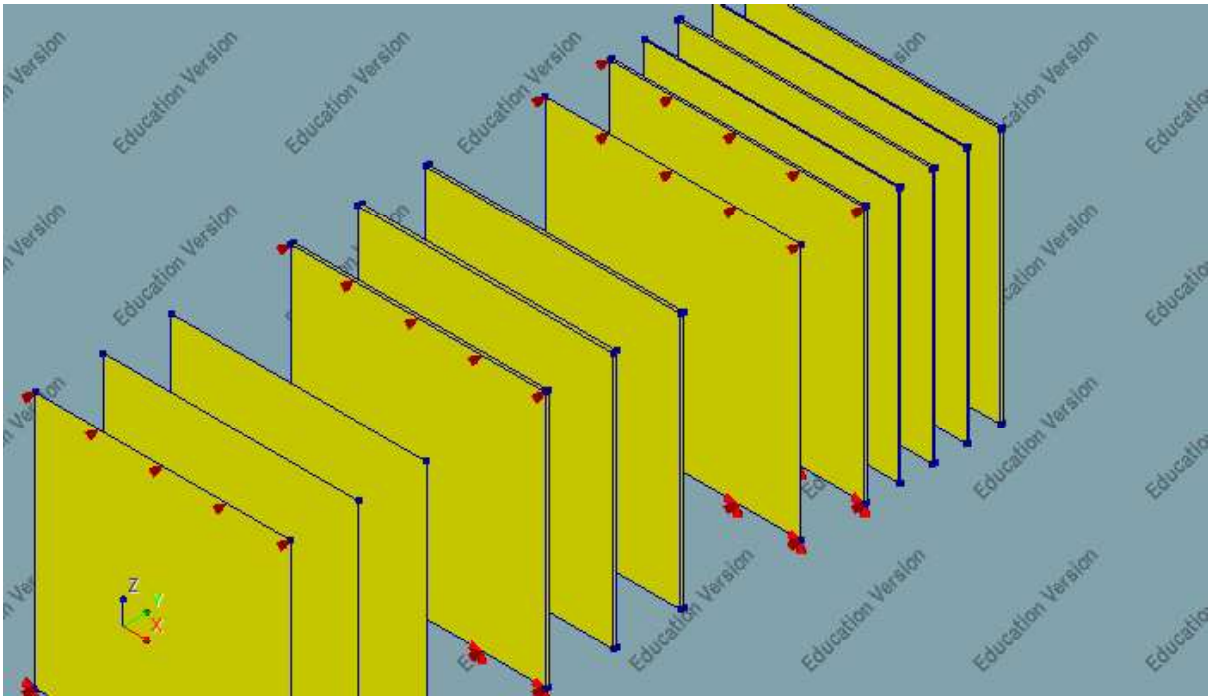
Geometry load ‘Bending’

Name	Target	Type	Direction	DOF	Value	Unit
Load Bending wythe 1	SURFAC	FORCE	NORMAL		1000	N/m2



Geometry support ‘Supports’

Name	Target	Translation	Rotation
Roller	LINE	Y	
Simple support	LINE	X,Y,Z	



Geometry: Masonry wall x=(1;0;0)

Name	Value
Geometry class	Sheets
Geometry model	Regular curved shell elements
Thickness	0.028 m
Element x axis	1 0 0

Geometry: Masonry wall x=(1;0;var)

Name	Value
Geometry class	Sheets
Geometry model	Regular curved shell elements
Thickness	0.028 m
Element x axis	1 0 0

Geometry: Wythe bed

Name	Value
Geometry class	Sheets
Geometry model	Structural surface interface elements
Element x axis	1 0 0

Geometry: Masonry wall x= 1

Name	Value
Geometry class	Sheets
Geometry model	Regular curved shell elements
Thickness	0.084 m
Element x axis	1 0 0

Material: Brick

Name	Value
Material class	Concrete and masonry
Material model	Linear elastic orthotropic
Color	grey
Element type	Curved shell
Young’s modulus	6.5e+09 6.5e+09 6.5e+09 N/m2
Poisson’s ratio	0.15 0.15 0.15
Shear modulus	2.82e+09 2.82e+09 2.78e+09 N/m2
Mass density	1800 kg/m3

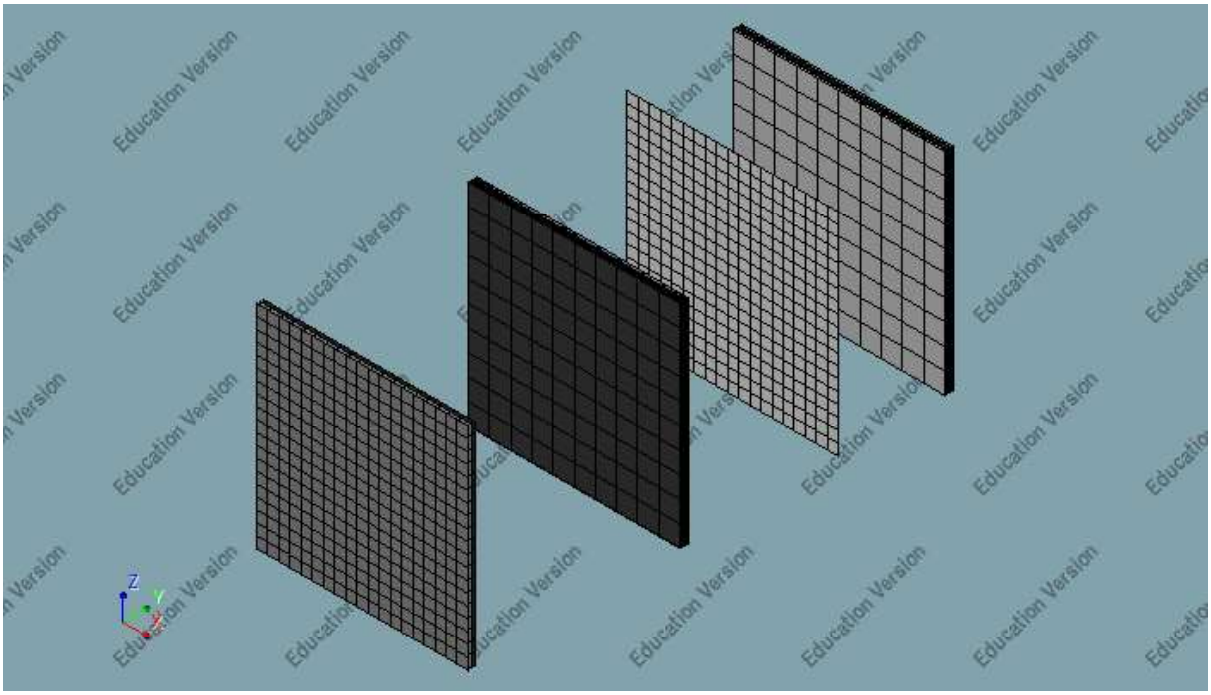
Material: Wythe bed

Name	Value
Material class	Interface elements
Material model	Linear elasticity
Color	silver
Type	3D surface interface
Normal stiffness modulus-z	9.17e+10 N/m3
Shear stiffness modulus-x	3.36e+10 N/m3
Shear stiffness modulus-y	3.36e+10 N/m3

Data: Element data 1

Name Value —— ——- —— ——-

Chapter 3



The Mesh consists of the following parts:

Mesh Sets

Name	# Elements	Material	geometry	Data
Wythe 1	361	Brick 1	Masonry wall x=(1;0;0)	
Wythe 2	361	Brick 1	Masonry wall x=(1;0;var)	
Wythe 3	361	Brick 1	Masonry wall x=(1;0;0)	
Sheet 1	361	Brick	Masonry wall x= 1	
Block 1	390	Brick no gap	Wall no gap	
Block 2	474	Brick no gap	Wall no gap	
Block 3	534	Brick no gap	Wall no gap	
Block 4	100	Brick no gap	Wall no gap	
Block 5	544	Bed no gap	Wall no gap	
Block 6	100	Brick no gap	Wall no gap	
Block 7	471	Bed no gap	Wall no gap	
Block 8	100	Brick no gap	Wall no gap	
Connection 1-2	361	Wythe bed	Wythe bed	
Connection 2-3	361	Wythe bed	Wythe bed	
Con12	100	Wythe bed	Wythe bed	
Con23	100	Wythe bed	Wythe bed	

Chapter 4

Analysis: Analysis1

Definition

```
Structural linear static
 Structural linear static
  Evaluate model
   Evaluate elements
    Average nodal normals
     Tolerance angle  = 0.349066
    Evaluate composed elements
   Assemble Elements
    Degree of freedom tolerance  = 1e-06
   Setup element stiffness matrices
   Setup load vectors
  Solve the system of equations
   Method  = Parallel Direct Sparse
   Convergence tolerance  = 1e-08
   Parallel Direct Sparse
    Factorization
  Output linear static analysis
   Output linear static analysis
    Device  = DIANA native
    Option  = Binary
    Seltyp  = USER
    Select
     Blknam  = OUTPUT
     Modsel  = Complete
     Loads
      User selection  = ALL
     Casety  = LOADS
    User
     Displacements
      Displacements
       Form  = TRANSL
       Oper  = GLOBAL
       Type  = TOTAL
       Total
      Displacements
       Form  = TRANSL
       Oper  = LOCAL
       Type  = TOTAL
       Total
     Strains
     Stresses
```

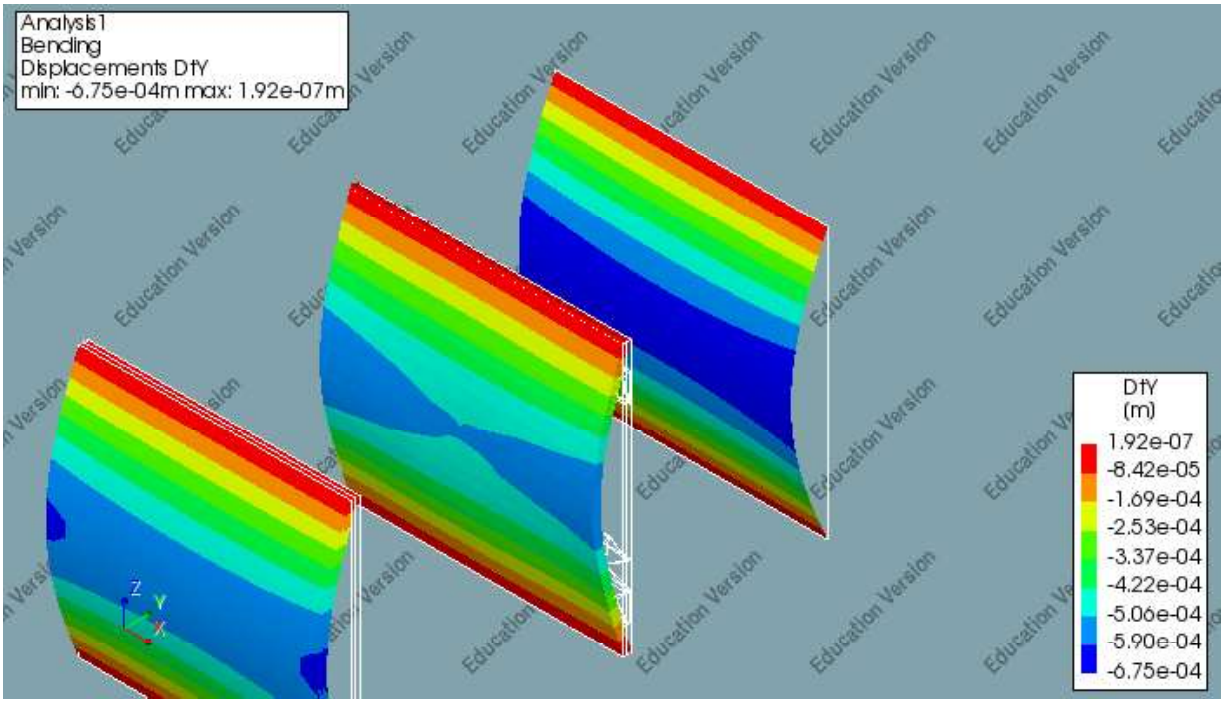
```
        Stresses
         Form  = CAUCHY
         Oper  = GLOBAL
         Type  = TOTAL
         Total
        Stresses
         Form  = CAUCHY
         Oper  = LOCAL
         Type  = TOTAL
         Total
        Stresses
         Form  = DISFOR
         Oper  = LOCAL
         Type  = TOTAL
         Total
        Stresses
         Form  = DISFOR
         Oper  = LOCAL
         Type  = ELEMEN
         Element contribution
       Forces
       Model parameters
       Element forces
```

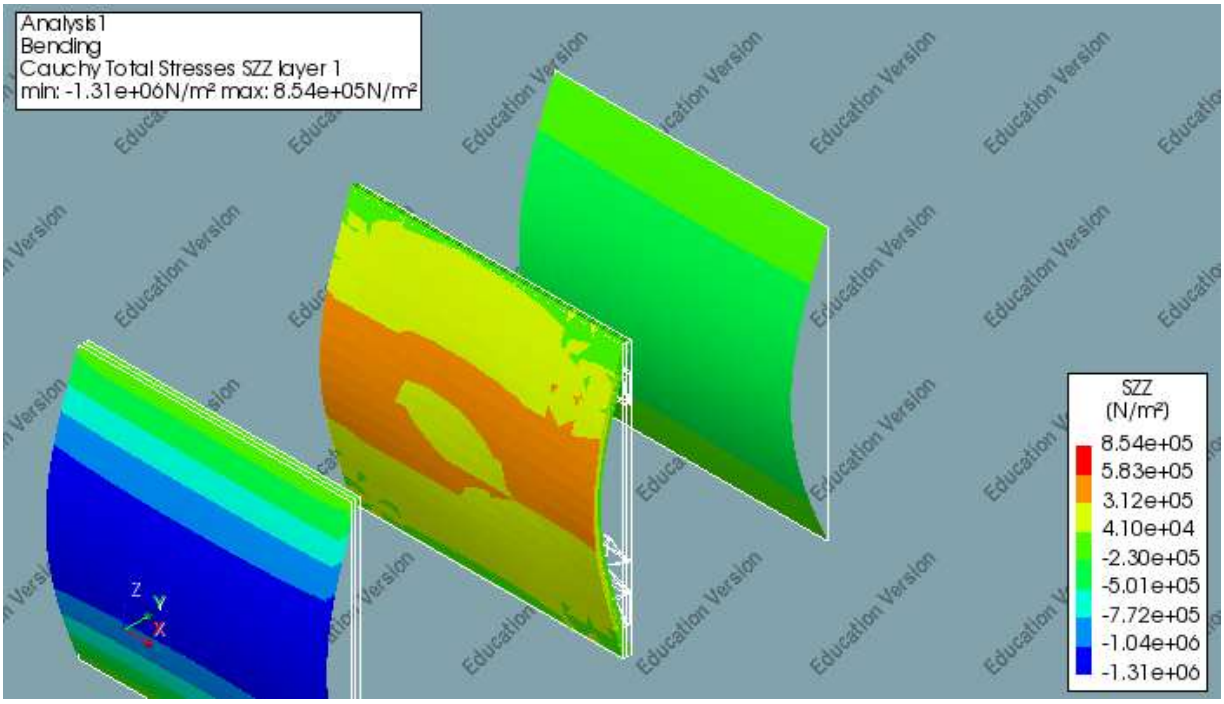
DCF Commands

```
*LINSTA LABEL="Structural linear static"
  MODEL EVALUA CHECK OFF
  SOLVE PARDIS
  BEGIN OUTPUT
    TEXT "Output linear static analysis"
    BINARY
    SELECT LOADS ALL /
    DISPLA TOTAL TRANSL GLOBAL
    DISPLA TOTAL TRANSL LOCAL
    STRESS TOTAL CAUCHY GLOBAL
    STRESS TOTAL CAUCHY LOCAL
    STRESS TOTAL DISFOR LOCAL
    STRESS ELEMEN DISFOR LOCAL
  END OUTPUT
*END
```

Phases

Chapter 5


Analysis1
Bending
Displacements DtY
min: -6.75e-04m max: 1.92e-07m
DtY
(m)
1.92e-07
-8.42e-05
-1.69e-04
-2.53e-04
-3.37e-04
-4.22e-04
-5.06e-04
-5.90e-04
-6.75e-04
Z
Y
X


Analysis1
Bending
Cauchy Total Stresses SZZ layer 1
min: -1.31e+06N/m² max: 8.54e+05N/m²
SZZ
(N/m²)
8.54e+05
5.83e+05
3.12e+05
4.10e+04
-2.30e+05
-5.01e+05
-7.72e+05
-1.04e+06
-1.31e+06
Z
Y
X

C

Brick experiment: testing the bond strength development

This chapter provides the experiment done on standard bricks from the hardware store and epoxy glue provided by the TU Delft. The epoxy glue is a Siko-Clearbond glue of 50 mL, with a 2K-Methyl methacrylaat. The following figures are present in this appendix:

- Figure C.1: Properties of the glue.
- Figure C.2: Weather of the days of experimentation.
- Figure C.3: The first brick experiment on the 24th of May.
- Figure C.4: The second brick experiment.
- Figure C.5: The third brick experiment.
- Figure C.6: The fourth brick experiment.
- Figure C.7: The fifth brick experiment.
- Figure C.8: The sixth brick experiment on the 28th of May, with cobblestone bricks.
- Figure C.9: The seventh brick experiment with hardware store bricks again.

2-component adhesive product name	Siko Clearbond
Type	Methyl methacrylate
Gap fill	2mm
Colour	Clear
Temperature resistance	-50C to +120C
Viscosity mPa.s	60000-90000
Fixture time	4 min
Final curing	24 h
Tensile strength Mpa	13-17
UV-resistance	Unknown
Shore D hardness	
Comments	It's high viscosity makes it easy to apply. Cures within 3min. Results in a clear adhesive layer.

Figure C.1: Siko-Clearbond properties

The brick did not fail with a hardening time of 7 minutes and 33 seconds (figure C.5), 9 minutes and 15 seconds (figure C.7), 7 minutes and 30 seconds (figure C.8) and 4 minutes and 23 second (figure C.9).

Weerstatistieken mei 2021 in Delft

Bekijk hieronder de weerstatistieken van mei 2021 in Delft. Lees onderaan de pagina hoe deze data is samengesteld. Je vindt hier de data per dag. Verder staat er onderaan nog een tabel met bijzondere dagen.

Let op: deze gegevens zijn niet gevalideerd. Eventuele meetfouten en/of storingen zijn niet uitgefilterd. Gebruik deze data daarom indicatief!

Datum	Tmax	Tmin	Neerslag	Max Windstoot	Opmerkingen
2021-05-31	23.8 °C	11.7 °C	0,0 mm	15.0 km/u	data tot 11:55u
2021-05-30	22.6 °C	9.4 °C	0,0 mm	14.0 km/u	data tot 11:55u
2021-05-29	18.3 °C	9.9 °C	0,0 mm	12.2 km/u	data tot 11:55u
2021-05-28	20.9 °C	7.1 °C	0,2 mm	10.0 km/u	data tot 11:55u
2021-05-27	13.2 °C	9.8 °C	0,8 mm	19.1 km/u	data tot 11:55u
2021-05-26	13.2 °C	8.7 °C	10,8 mm	22.0 km/u	data tot 11:55u
2021-05-25	13.1 °C	7.8 °C	2,4 mm	21.0 km/u	data tot 11:55u
2021-05-24	15.4 °C	10.1 °C	5,8 mm	22.0 km/u	data tot 11:55u

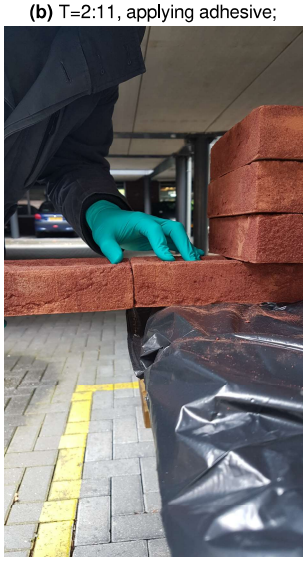
Figure C.2: The weather on the days of experimentation | https://www.hetweeractueel.nl/ weer/delft/historie/2021/05/.

(a) T=14:33:15, set up;

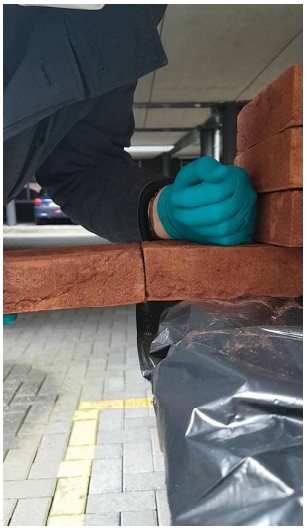
(b) T=2:11, applying adhesive;

(c) T=2:18, attaching;

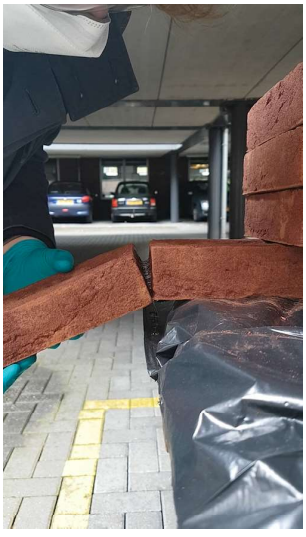
(d) T=4:13, pressure for placement;



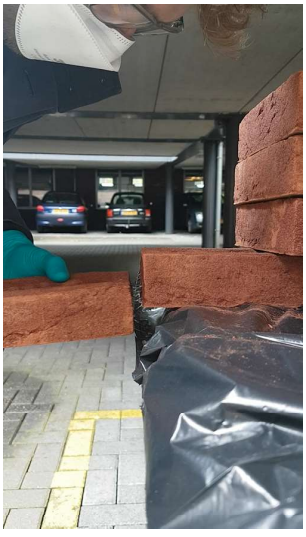
(e) T=6:24, pressure for placement, sideways;



(f) T=7:42, failure;



(g) T=7:51, check failure;



(h) T=8:51, ending the experiment;

(i) T=9:14, headers of bricks;

Figure C.3: First brick experiment.

(a) T=14:51:09, applying adhesive;

(b) T=1:06, pressing for placement;

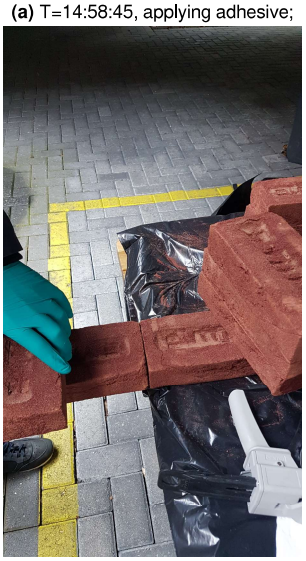
(c) T=4:54, failure;

Figure C.4: Second brick experiment.

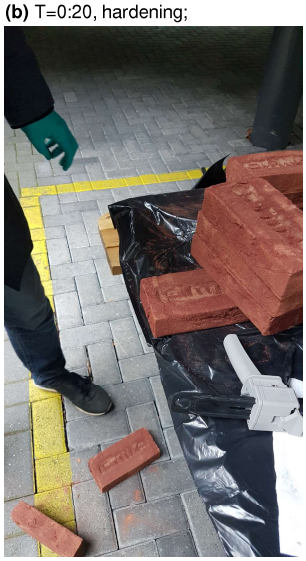
(a) T=14:58:45, applying adhesive;

(b) T=0:20, hardening;

(c) T=7:33, place horizontally;



(d) T=7:47, extra loading;



(e) T=7:57, failure;

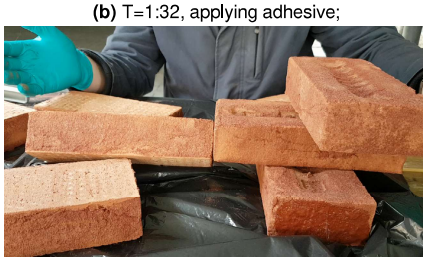
(f) T=8:29, headers;

Figure C.5: Third brick experiment.

(a) T=15:27:40, set up;

(b) T=1:32, applying adhesive;

(c) T=4:53, removing support;



(d) T=5:05, failure;

Figure C.6: Fourth brick experiment.

(a) T=15:35:11, applying adhesive;

(b) T=5:36, trying to remove support;

(c) T=5:39, put support back before failure;

(d) T=9:15, removing support;

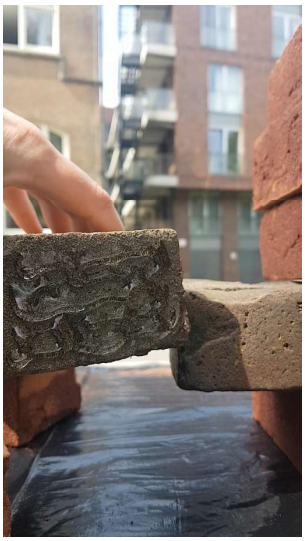
(e) T=10:28, failure;

(f) T=10:30, failure, close up;

(g) T=12:19, headers;

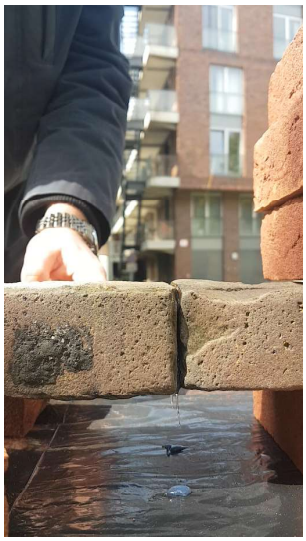
(h) T=21:31, ripping off adhesive;

Figure C.7: Fifth brick experiment.



(a) T=11:17:18, applying adhesive;

(b) T=1:03, applying additional adhesive;



(c) T=7:30, removing support;

(d) T=8:28, adding weight, but more counterweight is needed;

(e) T=9:04, adding the weight again, with additional counterweight;

(f) T=9:17, failure;

(g) T=10:12, headers, close up;

(h) T=10:47, headers;

Figure C.8: Sixth brick experiment.

(a) T=11:31:02, sanding;

(b) T=5:15, applying adhesive;

(c) T=9:38, removing support;

(d) T=13:09, additional load;

(e) T=16:22, failure with two bricks;

(f) T=16:27, failure, overview;

(g) T=21:52, headers;

(h) T=22:37, comparison with figure C.8;

Figure C.9: Seventh brick experiment.

D

Phased Structural Analysis in Python

The phased structural anlysis has been done with an import file from Grasshopper, resulting in an export file to Grasshopper. The workflow is given. In the script afterwards not only the calculation is included, but also the making of the visualisations.

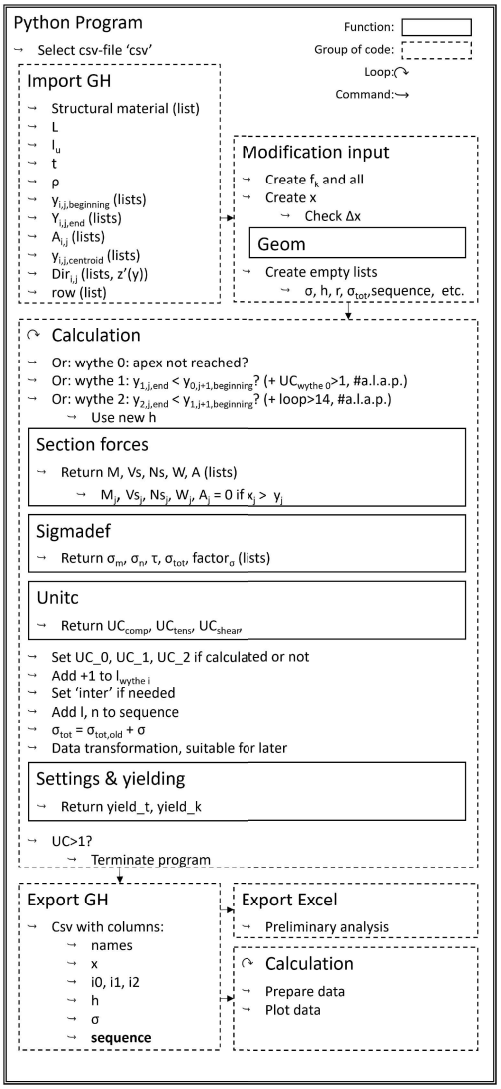


Figure D.1: Python workflow of the structural analysis.

In [1]:
```
__author__ = "Joris"
__version__ = "2021.08.26"
#
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.animation as ani
from matplotlib.animation import FuncAnimation as Fani
from IPython.core.display import display, HTML
display(HTML("<style>.container { width:45% !important; }</style>"))
# %config InlineBackend.figure_formats = ['retina']
mpl.rcParams['figure.dpi'] = 600
print(mpl.rcParams['figure.dpi'])
import csv
print("INPUT:")
```

```
<IPython.core.display.HTML object>


600.0
INPUT:
```

In [2]:
```
#
GH = pd.read_csv(
    "C:/Users/Joris/Documents/Master Thesis quick access/GH_to_Python.csv",
    error_bad_lines=False)
GH = pd.read_csv(
    "C:/Users/Joris/Documents/Master Thesis quick access/GH_to_Python.csv",
    header=None, sep='\n')
GH = GH[0].str.split(';', expand=True)

GH = GH.values.tolist()
GH = [[ele for ele in GH[ind] if str(ele) != 'nan'] for ind,ele in enumerate(GH)]
GH = [[ele for ele in GH[ind] if str(ele) != 'None'] for ind,ele in enumerate(GH)]

GH_top = GH[0]
GH = GH[1:]

for i in range(len(GH_top)):
    if len(GH[i])>1:
        exec(GH_top[i] + " = [float(ele) for ele in GH[i]]")
    else:
        exec(GH_top[i] + " = float(GH[i][0])")
    exec("print(" + "GH_top[i])\nprint(" + GH_top[i] + ")\nprint()")

print("INPUT^")
print("FUNCTIONS:")
```

```
sm
[20.0, 0.1, 0.3, 0.1, 2.0, 0.04, 0.5, 40.0]

L
3.6

lu
0.21

t
0.02

rho
1750.0

y0b
[0.001942, 0.004363, 0.12204, 0.24266, 0.366599, 0.494935, 0.626936, 0.762191, 0.900692, 1.042161, 1.186268, 1.332786...

y1b
[-0.013907, -0.001186, 0.117076, 0.239484, 0.364786, 0.494101, 0.627385, 0.764148, 0.90285, 1.045686, 1.191479, 1.339...

y2b
[-0.029768, 0.017589, 0.190165, 0.371265, 0.559851, 0.756109, 0.959366, 1.169021, 1.383607, 1.601516, 1.821049, 2.040...

y0e
[0.067261, 0.185694, 0.307957, 0.433913, 0.564524, 0.698096, 0.834715, 0.974734, 1.117627, 1.262972, 1.410713, 1.5597...

y1e
[0.061141, 0.180306, 0.304216, 0.430439, 0.562536, 0.698043, 0.836294, 0.977588, 1.122094, 1.268772, 1.418042, 1.5688...

y2e
[0.096149, 0.272971, 0.457876, 0.649016, 0.848577, 1.055176, 1.266654, 1.482252, 1.700828, 1.92046, 2.139227, 2.35532...
```

```
A0
[0.040683, 0.139165, 0.139165, 0.139165, 0.139165, 0.139165, 0.139165, 0.139165, 0.139165, 0.139165, 0.139165, 0.1391...

A1
[0.052435, 0.139165, 0.139165, 0.139165, 0.139165, 0.139167, 0.139172, 0.139178, 0.139183, 0.139186, 0.139184, 0.1391...

A2
[0.103116, 0.1995, 0.1995, 0.1995, 0.1995, 0.1995, 0.1995, 0.1995, 0.1995, 0.1995, 0.1995, 0.1995, 0.1995, 0.1995, 0.1...

y0
[0.021823, 0.095505, 0.215188, 0.338746, 0.465814, 0.596443, 0.730637, 0.868174, 1.00878, 1.152152, 1.297881, 1.445516...

y1
[0.008651, 0.087761, 0.208485, 0.332874, 0.461473, 0.593586, 0.729094, 0.868036, 1.010159, 1.154967, 1.302475, 1.45177...

y2
[0.012579, 0.144002, 0.323122, 0.509154, 0.703541, 0.904715, 1.112227, 1.325243, 1.542039, 1.76096, 1.980261, 2.198153...

Dir0
[0.817056, 0.778442, 0.726475, 0.666186, 0.589992, 0.5287, 0.472582, 0.411248, 0.349787, 0.288724, 0.220057, 0.159523,...

Dir1
[0.81852, 0.776537, 0.713992, 0.669917, 0.591574, 0.52023, 0.460946, 0.401482, 0.339271, 0.282871, 0.214852, 0.151768,...

Dir2
[0.817891, 0.754174, 0.646288, 0.568826, 0.469041, 0.379546, 0.29128, 0.198586, 0.10745, 0.017397, -0.073506, -0.16695...

row
[8.0, 8.0, 11.0]

INPUT^
FUNCTIONS:
```

In [3]:
```
#
def lim(y,l,i,j=0):
    try:
        return y[l[i]+j]
    except IndexError:
        return y[0]
#return:    either y[l[i]+j] or y[0]

def ll(l,i):
    try:
        return l[i]
    except IndexError:
        return 0
#return:    either l[i] or 0

def geom(x,yi, diri):
    ii = []; dsdxi = []; dzdxi = []
    ii.append( [np.argmax(yi > ele) +(len(yi))*(yi[-1] <  ele) for ele in x] )
    ii = ii[0]
    dzdxi.append([-lim(diri,ii,ele[0]) for ele in enumerate(x)])
    dzdxi = dzdxi[0]
    dsdxi.append([(1+dzdxi[ele[0]]**2)**0.5 for ele in enumerate(x)])
    dsdxi = dsdxi[0]
    ia = ii
    if max([ele<0 for ele in yi]):
        ia.extend( [ele for ele, a in enumerate([not(ele>0) for ele in yi]) if a] )
    if not ( all( x in ia for x in range(len(yi)) ) ):
        print('not every brick is represented in linspace x')
        print('x:')
        print(x)
        print(ia)
        print(set(ia).difference(range(len(yi))))
        exit()
    return ii,dsdxi,dzdxi
#return:    ii,dsdxi,dzdxi

def zero(i):
    zeros = np.zeros((i,len(x) ))
    return zeros
#return:    zeros

def new(l,h_old):
    n = min( len([l0,l1,l2][l]) , len([y0,y1,y2][l])-1 )
    ii = [i0,i1,i2][l]
    h = np.array([ ele+t if ii[ind]==n else ele for ind,ele in enumerate(h_old) ])
    return(h,n)

def section_forces(n,yi,Ai,dsdx,dzdx,h):
    weight = rho*9.81*t #N/mš
    y=yi[n] #m
```

```
    A=Ai[n] #mš/mž (Sum(l*b)/L)

    Vx = np.piecewise(x, [x<=y,x>y], [weight*A,0])
    M = weight*(y-x)*A
    M = [0 if ele < 0 else ele for ele in M]

    Vs = Vx/dsdx
    Ns = Vx*dzdx/dsdx

    W = 1/6 * h**2
    A = h
    return M,Vs,Ns,W,A
#return:    M,Vs,Ns,W,A

def sigmadef():
    dist = [[
        np.linspace( -1,1,int(h[j]/t+1) )[i]
        if int(h[j]/t)>=i
        else 0
        for j in range(len(x))]
        for i in range(lmax+1)]
    sigma_m = np.array(
        [[
        M[j]/W[j] *10**(-6) * dist[i][j]
        if h[j] != 0
        else 0
        if h[j]/t>=i
        else 0
        for j in range(len(x))]
        for i in range(lmax+1)] )
    sigma_n = np.array(
        [[
        Ns[j]/A[j] *10**(-6)
        * (h[j]/t>=i)
        if h[j] != 0
        else 0
        for j in range(len(x)) ]
        for i in range(lmax+1)] )
    tau =  np.array(
       [[
       Vs[j]*(6*dist[i][j]**2 - 3/2)/h[j] *10**(-6)
       if (h[j] != 0 and h[j]/t>=i)
       else 0
       for j in range(len(x)) ]
       for i in range(lmax+1)] )
    return dist,sigma_m,sigma_n, tau
#return:    dist, sigma_m, sigma_n, tau

def settings(l,n,setting):
    setting = [[
        min(i+[r0,r1,r2][l],1) #number of bricks
                #placed since this brick has been placed. similar to i+= row placed
        if i !=0
        else i
        for i in setting[j] ]
        for j in range(3)] #setting has shape of wythe Œ x
    setting[l] = [ [r0,r1,r2][l] if n==ele else setting[l][ind]
                  for ind,ele in enumerate([i0,i1,i2][l]) ]
    setting = np.array(setting)
    return setting
#return: setting

def yielding(setting,h_old,t,set):
    for j in range(4):
        for i in range(len(h_old)):
            if (int(j <= h_old[i]/t)) and (h_old[i] >0):
                sump = max( int(h_old[i]/t) -1,0)
                set[j][i] = setting[min(sump,j)][i]
    yield_t = [[fti+ele*(ftk-fti) for ele in set[j]] for j in range(4)]
    yield_k = [[fm+ele*(fk-fm) for ele in set[j]] for j in range(4)]
    yield_t,yield_k = np.array(yield_t),np.array(yield_k)
    return yield_t,yield_k
#return: yield_t,yield_k

def unitc(sigma,inter,strength):
    unitsc = np.array([sigma[0]/strength[0],
             sigma[1]/strength[0],
             (sigma[1]-inter[1])/strength[1],
             sigma[2]/strength[1],
             (sigma[2]-inter[2])/strength[2],
             sigma[3]/strength[3]])
    return unitsc
#return: uccomp,uctens,uctau
```

```
        def annot_max(x,y, ax=None,textp=[0.75,0.95]):
            xmax = x[np.argmax(y)] #xvalue
            ymax = y.max() #yvalye
            text= "@x:{:.2f}, uc={:.2f}".format(xmax, ymax) #text
            textpo = [textp[0],textp[1]] # textbox position
            if not ax:
                ax=plt.gca()
            bbox_props = dict(boxstyle="square,pad=0.3", fc="w", ec="k", lw=0.72) # textbox
            arrowprops=dict(arrowstyle="->",connectionstyle="angle,angleA=0,angleB=120")
                    # arrow
            kw = dict(xycoords='data',textcoords="axes fraction",
                      arrowprops=arrowprops, bbox=bbox_props, ha="left", va="top")
            ax.annotate(text, xy=(xmax, ymax), xytext=(textpo[0],textpo[1]), **kw)

        print("FUNCTIONS")
        print("SCRIPT")
        print("INPUT,OUTPUT AND PARAMETERS")
```

```
FUNCTIONS
SCRIPT
INPUT,OUTPUT AND PARAMETERS
```

```
In [4]: #        ##directe bewerking input
        L = L   #float
        [fb,fm,fvk,fti,ftf,deltax,pot,cure] = sm    #list of floats
        # fti,fm = .13,.13
        fk = -0.75*fb**0.7
        fm = -fm
        ftk = ftf
        fy = fk                             #####!!!!!
        fyt = ftk
        # ftk = 5
        fyv = fvk
        # y0 = [ele for ele in y0 if ele<L];    #list of floats
        # y1 = [ele for ele in y1 if ele<y0[-1]-lu/4]
        # y2 = [ele for ele in y2 if ele<y1[-1]-lu/4]
        #t & rho is also one float number   A0,A1,A2,row are lists of floats

                ##output
        seq=["layer;brick"]

                ##overige parameters

        l0=[];l1=[];l2=[]; lmax=3

        ymin = min(
        min([j-i for i, j in zip(y0[:-1], y0[1:])]),
        min([j-i for i, j in zip(y1[:-1], y1[1:])]),
        min([j-i for i, j in zip(y2[:-1], y2[1:])]))
        CHECK = (deltax < ymin)
        if CHECK==False:
            print("Reduce deltax!")
            print(ymin)

        x = np.arange(0,max(y0[-1],y1[-1],y2[-1]),deltax)
        i0,dsdx0,dzdx0 = geom(x,y0,Dir0)    #list of floats
        i1,dsdx1,dzdx1 = geom(x,y1,Dir1)
        i2,dsdx2,dzdx2 = geom(x,y2,Dir2)

        sigma_M = zero(4); sigma_N = zero(4); Tau = zero(4)
        sigma_m = zero(4); sigma_n = zero(4); tau = zero(4); dist = zero(4)
        h_old = np.zeros(len(x))

        setting = zero(3); set = zero(4)
        [r0,r1,r2] = row #number of bricks per row
        r0,r1,r2 = r0*pot/cure,r1*pot/cure,r2*pot/cure
        r = zero(4)
        yield_t,yield_k = yielding(setting,h_old,t,set)
        inter = zero(4)

        UCseries = []
        UCseries_ = []

        Distr = []
        Sigres = []
        Cstrres = []
        Tstrres = []
        Cucres = []
        Tucres = []
        ut=[]

        Sigd0 = [[],[]]
        Sigd9 = [[],[]]
```

```
        Sigd18 = [[],[]]
        Yied0 = [[],[]]
        Yied9 = [[],[]]
        Yied18 = [[],[]]
        UCd = [[],[]]
        apexl0=False
        print("INPUT,OUTPUT AND PARAMETERS^")
        print("FOR LOOP:")

INPUT,OUTPUT AND PARAMETERS^
FOR LOOP:


In [5]: #
        for runloop in range(len(y0)+len(y1)+len(y2)):
        # Layer 0 (1st) ########################################
            print('layer 0:',apexl0==False) # check if y0 is over the apex
            if apexl0==False:
        #     if lim(y0,l0,-1,1)<=(L+2*lu)/2 or len(l0)<1:
              l=0
              h,n = new(l,h_old)

              M,Vs,Ns,W,A = section_forces(n,y0,A0,dsdx0,dzdx0,h)
              dist, sigma_m, sigma_n, tau = sigmadef()

              sigma = sigma_M + sigma_m + sigma_N + sigma_n

              ttau = Tau + tau

              uccomp = unitc(sigma,inter,yield_k)
              uctens = unitc(sigma,inter,yield_t)
              uctau = ttau/fvk
            #     uccomp = (sigma-inter)/yield_k
              uccomp[uccomp < 0],uctens[uctens < 0] = 0,0
              UCmax = max( [np.max(ele) for ele in [uccomp,uctens,uctau] ] )

              ucs = UCmax; UCm1='na'; UCm2='na'; UCm0 = UCmax


            print('layer 1:',lim(y0b,l0,-1,1),' > ',
                  lim(y1e,l1,-1,1),
                  lim(y0b,l0,-1,1) > lim(y1e,l1,-1,1)) # check if new brick in
                             #2nd layer gives overlap for a new brick in 1st layer
        # Layer 1 (2nd) ########################################
            if lim(y0b,l0,-1,1) > lim(y1e,l1,-1,1):# and (UCm0>1 or apexl0==True):
                  # a.l.a.p. += and (UCm0>1 or apexl0==True)
        #     if y0[max(len(l0)-1,0)]-lu/2 > y1[max(len(l1)-1,0)] or UCmax>1:
              l=1
              h,n = new(l,h_old)

              M,Vs,Ns,W,A = section_forces(n,y1,A1,dsdx1,dzdx1,h)
              dist, sigma_m, sigma_n, tau = sigmadef()

              sigma = sigma_M + sigma_m + sigma_N + sigma_n
              ttau = Tau + tau

              uccomp = unitc(sigma,inter,yield_k)
              uctens = unitc(sigma,inter,yield_t)
              uctau = ttau/fvk

              uccomp[uccomp < 0],uctens[uctens < 0] = 0,0
              UCmax = max( [np.max(ele) for ele in [uccomp,uctens,uctau] ] )

              ucs = UCmax; UCm1 = UCmax

            print('layer 2:',lim(y1b,l1,-1,1),' > ',
                  lim(y2e,l2,-1,1),
                  lim(y1b,l1,-1,1) > lim(y2e,l2,-1,1)) # check if new brick in
                        #3rd layer gives overlap for a new brick in 2nd layer

        # Layer 2 (3rd) ########################################
            if lim(y1b,l1,-1,1) > lim(y2e,l2,-1,1):# and runloop>14:
                  # a.l.a.p. += and runloop>14
              l=2
              h,n = new(l,h_old)

              M,Vs,Ns,W,A = section_forces(n,y2,A2,dsdx2,dzdx2,h)
              dist, sigma_m, sigma_n, tau = sigmadef()

              sigma = sigma_M + sigma_m + sigma_N + sigma_n
              ttau = Tau + tau

              uccomp = unitc(sigma,inter,yield_k)
              uctens = unitc(sigma,inter,yield_t)
```

```
            uctau = ttau/fvk

            uccomp[uccomp < 0],uctens[uctens < 0] = 0,0
            UCmax = max( [np.max(ele) for ele in [uccomp,uctens,uctau] ] )

            ucs = UCmax; UCm2 = UCmax

        #Apex reached?
            if runloop>30:
                print(lim(y2e,l2,-1,1),lim(y1b,l1,-1,1),' ',
                      lim(y1e,l1,-1,1),lim(y0b,l0,-1,1),' ',
                      lim(y0,l0,-1,1),(L+2*lu)/2,' ', len(l0)>0)
            if lim(y2e,l2,-1,1)>lim(y1b,l1,-1,1) \
            and lim(y1e,l1,-1,1)>lim(y0b,l0,-1,1) and lim(y0,l0,-1,1)>(L+2*lu)/2 \
            and len(l0)>0:
                print("Apex has been reached! No more bricks can be added!")
                break
            if lim(y0,l0,-1,1)>(L+2*lu)/2 and len(l0)>0 and apexl0==False:
                print("Apex has been reached! The first layer is done!")
                apexl0=True
        # Print UC's
            if l == 0:
                if isinstance(UCm0,np.float64):
                    UC_0 = np.round(UCm0,3)
                else:
                    UC_0 = 'finished'
                print('UC0 = %s' % (UC_0))

                UC_0
            elif l == 1:
                if isinstance(UCm0,np.float64):
                    UC_0 = np.round(UCm0,3)
                else:
                    UC_0 = 'finished'
                if isinstance(UCm1,np.float64):
                    UC_1 = np.round(UCm1,3)
                else:
                    UC_1 = 'no try'
                print('UC0, UC1 = %s, %s' % (UC_0,UC_1))

                UC_0
            elif l == 2:
                if isinstance(UCm0,np.float64):
                    UC_0 = np.round(UCm0,3)
                else:
                    UC_0 = 'finished'
                if isinstance(UCm1,np.float64):
                    UC_1 = np.round(UCm1,3)
                else:
                    UC_1 = 'no try'
                if isinstance(UCm2,np.float64):
                    UC_2 = np.round(UCm2,3)
                else:
                    UC_2 = 'no try'
                print('UC0, UC1, UC2 = %s, %s, %s' % (UC_0,UC_1,UC_2))
                UC_0

            if len([l0,l1,l2][l])<len([y0,y1,y2][l]):
                [l0,l1,l2][l].append(n)
            if l!=0:                #   inter   #
                inte = [ind for ind,ele in enumerate([i0,i1,i2][l]) if ele==n]
                inter[l,inte[0]:inte[-1]+1] = sigma[l,inte[0]:inte[-1]+1]

            seq.append(str(l) + ";" + str(n))

            h_old = h

            sigma_M = sigma_M + sigma_m     #is correct sign for position on cross-section
            sigma_N = sigma_N + sigma_n     #negative is correct
            sigma_norm = sigma_M+sigma_N
            Tau = Tau + tau #positive is correct

            UCseries.append(ucs)
            UCseries_.append([runloop,UCm0,UCm1,UCm2])

            Distr.append(runloop)

            Sigres.append(sigma_norm-inter)
            Tstrres.append(yield_t)
            Cstrres.append(yield_k)
            Cucres.append(uccomp)
            Tucres.append(uctens)

            setting = settings(l,n,setting)
```

```
            yield_t,yield_k = yielding(setting,h_old,t,set)

            indc = [[0,lst,ind] for (lst,ind),ele in np.ndenumerate(uccomp) if ele==UCmax]
            indt = [[1,lst,ind] for (lst,ind),ele in np.ndenumerate(uctens) if ele==UCmax]
            indv = [[2,lst,ind] for (lst,ind),ele in np.ndenumerate(uctau) if ele==UCmax]
            ut.append((indc or indt or indv)[0])

            if UCmax>1.0:
                print("")
                if UCmax==UCm2:
                    if type(UCm1)!='float': UCm1 = 3
                    print('Check this:      ',
                          'UC0, UC1, UC2 = %s, %s, %s' % (np.round(UCm0,3),
                                                          np.round(UCm1,3),np.round(UCm2,3)))
                print("UC too big!")
                if y0[n]>(L+2*lu)/2:
                    print('Apex has been reached in layer 1!')
                l_last,n_last = l,n
                del seq[-1]
                break
            print('brick %s placed in layer %s as row %s' % (str(runloop+1),str(l),str(n)))

            nope = False
            if nope:
                Sigd0[0].append(sigma_M[0,0])
                Sigd0[1].append(sigma_M[1,0])
                Sigd9[0].append(sigma_M[0,9])
                Sigd9[1].append(sigma_M[1,9])
                Sigd18[0].append(sigma_M[0,18])
                Sigd18[1].append(sigma_M[1,18])
                Yied0[0].append(yield_k[0,0])
                Yied0[1].append(yield_t[1,0])
                Yied9[0].append(yield_k[0,9])
                Yied9[1].append(yield_t[1,9])
                Yied18[0].append(yield_k[0,18])
                Yied18[1].append(yield_t[1,18])
                UCd[0].append(uccomp[0,0])
                UCd[1].append(uctens[1,0])

            uccomp = sigma/yield_k
            uctens = sigma/yield_t
            uctau = ttau/fvk

        print()
        print("FOR LOOP^")
        print("OUTPUT:")
```

```
layer 0: True
layer 1: 0.001942  >  0.061141 False
layer 2: -0.013907  >  0.096149 False
UC0 = 0.05
brick 1 placed in layer 0 as row 0
layer 0: True
layer 1: 0.004363  >  0.061141 False
layer 2: -0.013907  >  0.096149 False
UC0 = 0.412
brick 2 placed in layer 0 as row 1
layer 0: True
layer 1: 0.12204  >  0.061141 True
layer 2: -0.013907  >  0.096149 False
UC0, UC1 = 0.696, 0.147
brick 3 placed in layer 1 as row 0
layer 0: True
layer 1: 0.12204  >  0.180306 False
layer 2: -0.001186  >  0.096149 False
UC0 = 0.696
brick 4 placed in layer 0 as row 2
layer 0: True
layer 1: 0.24266  >  0.180306 True
layer 2: -0.001186  >  0.096149 False
UC0, UC1 = 0.766, 0.241
brick 5 placed in layer 1 as row 1
layer 0: True
layer 1: 0.24266  >  0.304216 False
layer 2: 0.117076  >  0.096149 True
UC0, UC1, UC2 = 0.721, no try, 0.188
brick 6 placed in layer 2 as row 0
layer 0: True
layer 1: 0.24266  >  0.304216 False
layer 2: 0.117076  >  0.272971 False
UC0 = 0.721
brick 7 placed in layer 0 as row 3
layer 0: True
layer 1: 0.366599  >  0.304216 True
```

```
layer 2: 0.117076  >  0.272971 False
UC0, UC1 = 0.793, 0.239
brick 8 placed in layer 1 as row 2
layer 0: True
layer 1: 0.366599  >  0.430439 False
layer 2: 0.239484  >  0.272971 False
UC0 = 0.771
brick 9 placed in layer 0 as row 4
layer 0: True
layer 1: 0.494935  >  0.430439 True
layer 2: 0.239484  >  0.272971 False
UC0, UC1 = 0.846, 0.342
brick 10 placed in layer 1 as row 3
layer 0: True
layer 1: 0.494935  >  0.562536 False
layer 2: 0.364786  >  0.272971 True
UC0, UC1, UC2 = 0.846, no try, 0.267
brick 11 placed in layer 2 as row 1
layer 0: True
layer 1: 0.494935  >  0.562536 False
layer 2: 0.364786  >  0.457876 False
UC0 = 0.846
brick 12 placed in layer 0 as row 5
layer 0: True
layer 1: 0.626936  >  0.562536 True
layer 2: 0.364786  >  0.457876 False
UC0, UC1 = 0.947, 0.284
brick 13 placed in layer 1 as row 4
layer 0: True
layer 1: 0.626936  >  0.698043 False
layer 2: 0.494101  >  0.457876 True
UC0, UC1, UC2 = 0.947, no try, 0.216
brick 14 placed in layer 2 as row 2
layer 0: True
layer 1: 0.626936  >  0.698043 False
layer 2: 0.494101  >  0.649016 False
UC0 = 0.947
brick 15 placed in layer 0 as row 6
layer 0: True
layer 1: 0.762191  >  0.698043 True
layer 2: 0.494101  >  0.649016 False
UC0, UC1 = 0.979, 0.319
brick 16 placed in layer 1 as row 5
layer 0: True
layer 1: 0.762191  >  0.836294 False
layer 2: 0.627385  >  0.649016 False
UC0 = 0.784
brick 17 placed in layer 0 as row 7
layer 0: True
layer 1: 0.900692  >  0.836294 True
layer 2: 0.627385  >  0.649016 False
UC0, UC1 = 0.931, 0.423
brick 18 placed in layer 1 as row 6
layer 0: True
layer 1: 0.900692  >  0.977588 False
layer 2: 0.764148  >  0.649016 True
UC0, UC1, UC2 = 0.931, no try, 0.329
brick 19 placed in layer 2 as row 3
layer 0: True
layer 1: 0.900692  >  0.977588 False
layer 2: 0.764148  >  0.848577 False
UC0 = 0.931
brick 20 placed in layer 0 as row 8
layer 0: True
layer 1: 1.042161  >  0.977588 True
layer 2: 0.764148  >  0.848577 False
UC0, UC1 = 0.985, 0.339
brick 21 placed in layer 1 as row 7
layer 0: True
layer 1: 1.042161  >  1.122094 False
layer 2: 0.90285  >  0.848577 True
UC0, UC1, UC2 = 0.81, no try, 0.378
brick 22 placed in layer 2 as row 4
layer 0: True
layer 1: 1.042161  >  1.122094 False
layer 2: 0.90285  >  1.055176 False
UC0 = 0.81
brick 23 placed in layer 0 as row 9
layer 0: True
layer 1: 1.186268  >  1.122094 True
layer 2: 0.90285  >  1.055176 False
UC0, UC1 = 0.993, 0.464
brick 24 placed in layer 1 as row 8
layer 0: True
```

```
layer 1: 1.186268  >  1.268772 False
layer 2: 1.045686  >  1.055176 False
UC0 = 0.993
brick 25 placed in layer 0 as row 10
layer 0: True
layer 1: 1.332786  >  1.268772 True
layer 2: 1.045686  >  1.055176 False
UC0, UC1 = 1.043, 0.561
brick 26 placed in layer 1 as row 9
layer 0: True
layer 1: 1.332786  >  1.418042 False
layer 2: 1.191479  >  1.055176 True
UC0, UC1, UC2 = 0.903, no try, 0.612
brick 27 placed in layer 2 as row 5
layer 0: True
layer 1: 1.332786  >  1.418042 False
layer 2: 1.191479  >  1.266654 False
UC0 = 0.903
brick 28 placed in layer 0 as row 11
layer 0: True
layer 1: 1.481037  >  1.418042 True
layer 2: 1.191479  >  1.266654 False
UC0, UC1 = 0.986, 0.721
brick 29 placed in layer 1 as row 10
layer 0: True
layer 1: 1.481037  >  1.568822 False
layer 2: 1.339571  >  1.266654 True
UC0, UC1, UC2 = 0.823, no try, 0.784
brick 30 placed in layer 2 as row 6
layer 0: True
layer 1: 1.481037  >  1.568822 False
layer 2: 1.339571  >  1.482252 False
UC0 = 0.848
brick 31 placed in layer 0 as row 12
layer 0: True
layer 1: 1.630597  >  1.568822 True
layer 2: 1.339571  >  1.482252 False
1.482252 1.339571   1.568822 1.630597   1.7444 2.0100000000000002   True
UC0, UC1 = 1.036, 0.905
brick 32 placed in layer 1 as row 11
layer 0: True
layer 1: 1.630597  >  1.720447 False
layer 2: 1.489541  >  1.482252 True
1.482252 1.489541   1.720447 1.630597   1.7444 2.0100000000000002   True
UC0, UC1, UC2 = 1.036, no try, 0.981
brick 33 placed in layer 2 as row 7
layer 0: True
layer 1: 1.630597  >  1.720447 False
layer 2: 1.489541  >  1.700828 False
1.700828 1.489541   1.720447 1.630597   1.7444 2.0100000000000002   True
UC0 = 1.05

UC too big!

FOR LOOP^
OUTPUT:
```

```
In [6]: # export Grasshopper
        # print(np.round(sigma,2))
        sig_com = min([min(ele) for ele in sigma])
        sig_ten = max([max(ele) for ele in sigma])
        taut = max([max(ele) for ele in Tau])

        #location of highest compressive stresses compared to strength there
        sind = [[lst,ind] for (lst,ind),ele in np.ndenumerate(sigma) if ele==sig_com][0]
        print('l:',sind[0],'br:',sind[1]), print(
            'f_k',sigma[sind[0]][sind[1]]), print('f_R',yield_k[sind[0]][sind[1]])

        #location of highest tensile stresses compared to strength there
        sind = [[lst,ind] for (lst,ind),ele in np.ndenumerate(sigma) if ele==sig_ten][0]
        print('l:',sind[0],'br:',sind[1]), print(
            'f_tk',sigma[sind[0]][sind[1]]), print('f_tR',yield_t[sind[0]][sind[1]])

        #location of highest shear stresses compared to strength there
        sind = [[lst,ind] for (lst,ind),ele in np.ndenumerate(Tau) if ele==taut][0]
        print('l:',sind[0],'br:',sind[1]), print(
            'f_vk',Tau[sind[0]][sind[1]]), print('f_vR',fvk)

        maxcomp, maxtens, maxshear = np.max(uccomp), np.max(uctens), np.max(uctau)
        sind = [[lst,ind] for (lst,ind),ele in np.ndenumerate(uctens) if ele==maxtens][0]
        print(sind),print()

        print('maximum unity checks:',"maxcomp: ",np.round(maxcomp,3),
```

```
            " maxtens: ", np.round(maxtens,3),
            " maxshear: ", np.round(maxshear,3)), print()
        print(seq),print(len(seq)),print()
        print (np.round(np.array(UCseries),3)),print()

        UCseries_ = [[np.round(ele,1) if
                      type(ele)==np.float64 else ele for ele in UCseries_[j] ]
                     for j in range(len(UCseries_))]
        for i in range(len(UCseries_)):
            print(UCseries_[i])
        layer = [ele.split(';')[0] for ele in seq]
        brick = [ele.split(';')[1] for ele in seq]
        print()
        names = ['names','x','i0','i1','i2','h','sigma','seq']
        output = [names,x,i0,i1,i2,h,sigma,seq]
        output = [ele.tolist() if type(ele)==np.ndarray else ele for ele in output]

        ###                                  export

        np.savetxt('C:/Users/Joris/Documents/Master Thesis quick access/Python_to_GH.csv',
                   output, delimiter = ';', fmt = '%s')
        #print(output)

        a = [['   x','  y0','  y1','  y2','   h','sigm0','sigm1','sigm2','sigm3']]
        b = np.transpose([x,i0,i1,i2,h,sigma[0],sigma[1],sigma[2],sigma[3]])
        b = [[str("{:.2f},".format(ele)) for ele in b[j]] for j in range(len(b))]
        a += b
        # print(np.array(a))

l: 0 br: 1
f_k -2.3276665855266727
f_R -6.106357973053566
l: 3 br: 0
f_tk 2.09957120344815
f_tR 2.0
l: 0 br: 1
f_vk 0.11499622611018762
f_vR 0.3
[5, 0]

maximum unity checks: maxcomp:  1.036  maxtens:  1.05  maxshear:  0.383

['layer;brick', '0;0', '0;1', '1;0', '0;2', '1;1', '2;0', '0;3', '1;2', '0;4', '1;3', '2;1', '0;5', '1;4', '2;2', '0;6',
 '1;5', '0;7', '1;6', '2;3', '0;8', '1;7', '2;4', '0;9', '1;8', '0;10', '1;9', '2;5', '0;11', '1;10', '2;6', '0;12',
  '1;11', '2;7']
34

[0.05  0.412 0.147 0.696 0.241 0.188 0.721 0.239 0.771 0.342 0.267 0.846
 0.284 0.216 0.947 0.319 0.784 0.423 0.329 0.931 0.339 0.378 0.81  0.464
 0.993 0.561 0.612 0.903 0.721 0.784 0.848 0.905 0.981 1.05 ]

[0, 0.1, 'na', 'na']
[1, 0.4, 'na', 'na']
[2, 0.7, 0.1, 'na']
[3, 0.7, 'na', 'na']
[4, 0.8, 0.2, 'na']
[5, 0.7, 'na', 0.2]
[6, 0.7, 'na', 'na']
[7, 0.8, 0.2, 'na']
[8, 0.8, 'na', 'na']
[9, 0.8, 0.3, 'na']
[10, 0.8, 'na', 0.3]
[11, 0.8, 'na', 'na']
[12, 0.9, 0.3, 'na']
[13, 0.9, 'na', 0.2]
[14, 0.9, 'na', 'na']
[15, 1.0, 0.3, 'na']
[16, 0.8, 'na', 'na']
[17, 0.9, 0.4, 'na']
[18, 0.9, 'na', 0.3]
[19, 0.9, 'na', 'na']
[20, 1.0, 0.3, 'na']
[21, 0.8, 'na', 0.4]
[22, 0.8, 'na', 'na']
[23, 1.0, 0.5, 'na']
[24, 1.0, 'na', 'na']
[25, 1.0, 0.6, 'na']
[26, 0.9, 'na', 0.6]
[27, 0.9, 'na', 'na']
[28, 1.0, 0.7, 'na']
[29, 0.8, 'na', 0.8]
[30, 0.8, 'na', 'na']
[31, 1.0, 0.9, 'na']
[32, 1.0, 'na', 1.0]
```

```
[33, 1.0, 'na', 'na']
```

```
In [7]: # export Excel
        # distr = [Distr,Sigd0[0],Sigd0[1],Sigd9[0],Sigd9[1],Sigd18[0],Sigd18[1], Yied0[0],Yied0[1],Yied9[0],Yied9[1],
        #Yied18[0],Yied18[1],UCd[0],UCd[1]]
        # for i in range(len(distr)):
        #     distr[i].insert(0,['n',0,0,9,9,18,18,0,0,9,9,18,18,0,0][i])
        print(len(Sigres))
        print(len(Sigres[0]))
        print(len(Sigres[0][0]))
        sigres = np.transpose(Sigres)
        tres = np.transpose(Tstrres)
        cres = np.transpose(Cstrres)

        distr = []
        list = [sigres[0][0],sigres[0][1],sigres[0][2],sigres[0][3],
                cres[0][0],cres[0][1],tres[0][1],tres[0][2],tres[0][3]]
        for i in range(len(list)):
            distr.append(list[i])
        print(sigres[0][3][0:50])

        np.savetxt('C:/Users/Joris/Documents/Master Thesis quick access/Stress_dist.txt',
                   distr, delimiter = ';', fmt = '%s')

        # for i in range(len(Sigres)):
        #     distr = Sigres[i]
        #     np.savetxt('C:/Users/Joris/Documents/Master Thesis quick access/Stress_dist_%s.txt' % (i), distr,
        # delimiter = ';', fmt = '%s')
        #print(output)
```

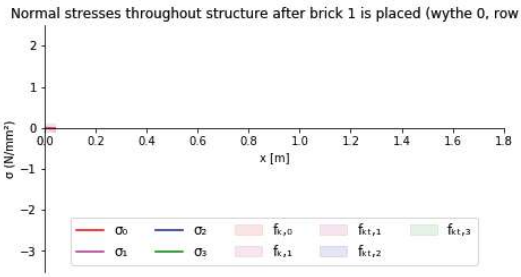
```
34
4
91
[0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 3.68681070e-04 2.68415776e-02 4.29403158e-02
 7.95325484e-02 1.05537275e-01 1.21254326e-01 1.68249483e-01
 2.04495470e-01 2.40661554e-01 2.98343542e-01 3.45111306e-01
 4.13746351e-01 4.71307812e-01 5.28712031e-01 6.08544540e-01
 6.77174465e-01 7.56770657e-01 8.48020906e-01 9.27973034e-01
 1.03082873e+00 1.12231642e+00 1.22487941e+00 1.33949234e+00
 1.44272744e+00 1.56898081e+00 1.69546215e+00 1.81058390e+00
 1.96115600e+00 2.09957120e+00]
```

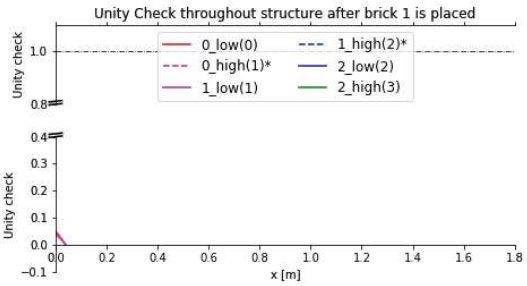
```
In [8]: def sub(x):
                normal = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+-=()"
                subs = "CDGQwZw"
                res = x.maketrans(''.join(normal), ''.join(subs))
                return x.translate(res)
```

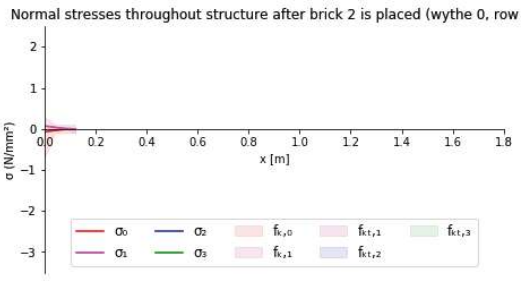
```
In [9]: # normal stresses after brick[i]
        ll = [int(ele.split(';')[0]) for ele in seq[1:]]
        nn = [int(ele.split(';')[1]) for ele in seq[1:]]
        ll.append(l_last),nn.append(n_last)
        # ll.append(1),nn.append(8)
        n0,n1,n2 = -2,-2,-2
        id0,id1,id2 = 0,0,0

        COres=np.array([[[0 if Sigres[brk][lay][idx]>0 else ele
                          for idx,ele in enumerate(Cstrres[brk][lay])]
                         for lay,Lay in enumerate(Cstrres[brk])]
                        for brk,Brk in enumerate(Cstrres)])
        TOres=np.array([[[0 if Sigres[brk][lay][idx]<0 else ele
                          for idx,ele in enumerate(Tstrres[brk][lay])]
                         for lay,Lay in enumerate(Tstrres[brk])]
                        for brk,Brk in enumerate(Tstrres)])
        UCres=np.array([[[Tucres[brk][lay][idx] if ele==0 else ele
                          for idx,ele in enumerate(Cucres[brk][lay])]
                         for lay,Lay in enumerate(Cucres[brk])]
                        for brk,Brk in enumerate(Cucres)])
        UC_maxi = np.array([[np.max(UCres[-1][:,idx])
                             for idx,ele in enumerate(UCres[-1][lst])]
                            for lst,Lst in enumerate(UCres[-1])][0])
        plotall = []
        for i in range(len(Sigres)):
        # for i in range(4):
            n0,n1,n2 = nn[i] if ll[i]==0 else n0, nn[i] if \
                ll[i]==1 else n1, nn[i] if ll[i]==2 else n2
            id0,id1,id2 = i0.index(n0+1) if n0>-1 else -1, i1.index(n1+1) if \
                n1>-1 else -1, i2.index(n2+1) if n2>-1 else -1
        #     print('n',n0+1,n1+1,n2+1)
```

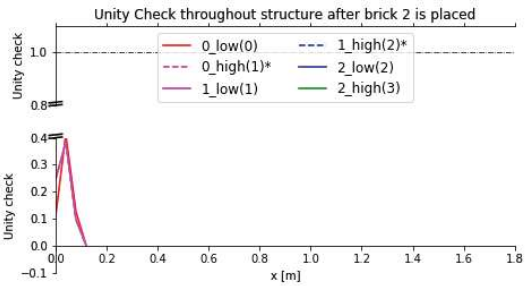
```
    plt.rcParams["figure.figsize"] = (16,4) # (w, h)
#     fig, (ax0,ax1,ax2) = plt.subplots(2,2,sharex=True,sharey=False)
    ax0 = plt.subplot(1,2,1)
    ax1 = plt.subplot(9,2,(2,6), sharex=ax0)
    ax2 = plt.subplot(9,2,(10,18), sharex=ax0)
    AX = [ax1,ax2]
    Yl = [[.8,1.1],[-.1,.4]]
#
    ax0.spines['left'].set_position('zero')
    ax0.spines['right'].set_color('none')
    ax0.yaxis.tick_left()

    ax0.spines['bottom'].set_position('zero')

    ax0.spines['top'].set_color('none')
    ax0.xaxis.tick_bottom()

    xx = x
#     print(Sigres[i][0][:id0+1])
    ax0.plot(xx[:id0+1],Sigres[i][0][:id0+1], label = '\u03C3%s' % (sub('0')),
             color='red')
    ax0.plot(xx[:id0+1],Sigres[i][1][:id0+1], label = '\u03C3%s' % (sub('1')),
             color='magenta')
    ax0.plot(xx[:id1+1],Sigres[i][2][:id1+1], label = '\u03C3%s' % (sub('2')),
             color='blue')
    ax0.plot(xx[:id2+1],Sigres[i][3][:id2+1], label = '\u03C3%s' % (sub('3')),
             color='green')

    alp = .1
    ax0.fill_between(xx[:id0+1],C0res[i][0][:id0+1], color='red',alpha=alp,
                     label = 'f%s' % (sub('k,0')))
    ax0.fill_between(xx[:id0+1],C0res[i][1][:id0+1], color='magenta',alpha=alp,
                     label = 'f%s' % (sub('k,1')))
    ax0.fill_between(xx[:id0+1],T0res[i][1][:id0+1], color='magenta',alpha=alp,
                     label = 'f%s' % (sub('kt,1')))
    ax0.fill_between(xx[:id1+1],Tstrres[i][2][:id1+1], color='blue',alpha=alp,
                     label = 'f%s' % (sub('kt,2')))
    ax0.fill_between(xx[:id2+1],Tstrres[i][3][:id2+1], color='green',alpha=alp,
                     label = 'f%s' % (sub('kt,3')))


    ax0.set_xlabel('x [m]')
    ax0.set_ylabel('\u03C3 (N/mmš)')
    ax0.title.set_text(
        'Normal stresses throughout structure after brick %s is placed (wythe %s, row %s)' % (i+1,ll[i],nn[i]))

    ax0.legend(loc=8, prop={"size":12},ncol=5)
    ax0.set_ylim([-3.5,2.5]), ax0.set_xlim([0,L/2])

    # # # # # # # # # # # # # #

    ax1.spines['bottom'].set_visible(False)
    ax2.spines['top'].set_color('none')

    ax2.spines['top'].set_visible(False)
    ax2.spines['bottom'].set_position('zero')

    ax2.xaxis.tick_bottom()
    ax1.xaxis.tick_top()
    ax1.tick_params(labeltop=False)
    ax2.spines['left'].set_position('zero')
    d,D = .014,.06
    rate = [.625,.375]

    for j in range(len(AX)):
        ax = AX[j]
        ax.spines['right'].set_color('none')
        ax.yaxis.tick_left()

        ax.plot(xx[:id0+1],Cucres[i][0][:id0+1], label = '0_low(0)',
                color='red')
        ax.plot(xx[:id0+1], UCres[i][1][:id0+1], label = '0_high(1)*',
                color='magenta',linestyle='dashed')
        ax.plot(xx[:id0+1], UCres[i][2][:id0+1], label = '1_low(1)',
                color='magenta')
        ax.plot(xx[:id1+1],Tucres[i][3][:id1+1], label = '1_high(2)*',
                color='blue',linestyle='dashed')
        ax.plot(xx[:id1+1],Tucres[i][4][:id1+1], label = '2_low(2)',
                color='blue')
        ax.plot(xx[:id2+1],Tucres[i][5][:id2+1], label = '2_high(3)',
                color='green')
        ax.hlines(1,0,L/2,color='k',linewidth=.75,linestyle='dashdot')

        ax.set_ylabel('Unity check')
```

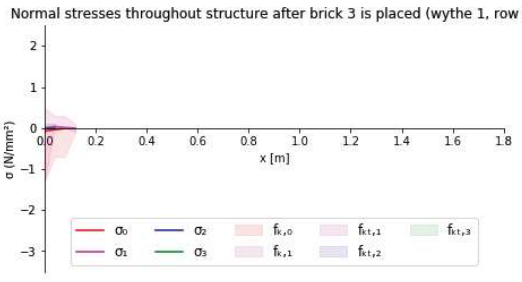
```
        ax.set_ylim(Yl[j])#, ax.set_xlim([0,L/2])

        dj,Dj = rate[j]*d,rate[j]*D
        kwargs = dict(transform=ax.transAxes, color='k', clip_on=False)
        ax.plot((-d, +d), (j-dj, j+dj), **kwargs)  # bottom-left diagonal
        ax.plot((-d, +d), (j-dj+Dj, j+dj+Dj), **kwargs)  # bottom-left diagonal

    ax1.title.set_text(
        'Unity Check throughout structure after brick %s is placed' % (i+1))
    ax2.set_xlabel('x [m]')
    ax1.legend(loc=9, prop={"size":12},ncol=2)
    if i == len(Sigres)-1:
        try:
            del textp
        except:
            print()
        annot_max(xx,UC_maxi, ax=ax1,textp=[0.015,0.9])
        xmax = xx[np.argmax(UC_maxi)]
        ax0.vlines(xmax,-3.5,2.5,linestyle=(0,(2,12)))
#         print(id0,id1,id2)
    plt.show()
```

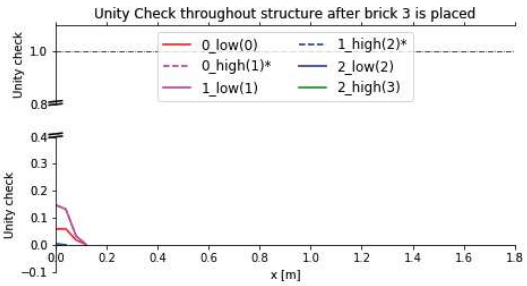


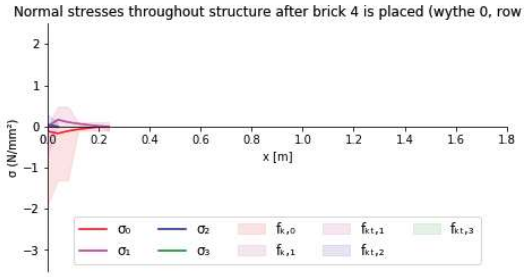


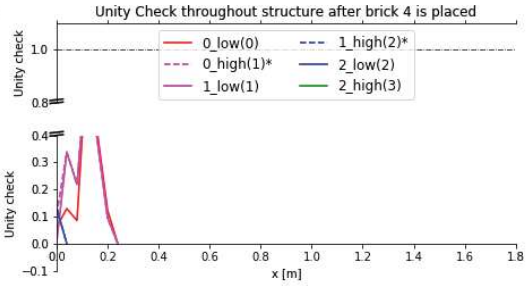


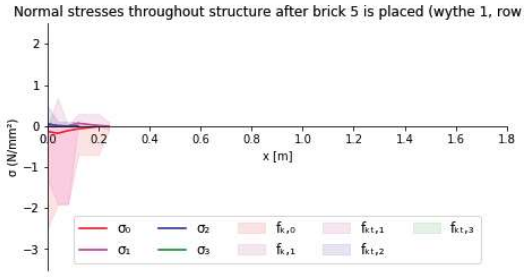


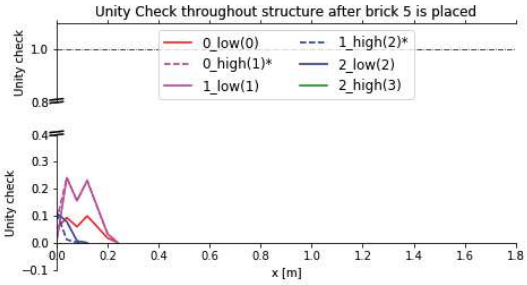


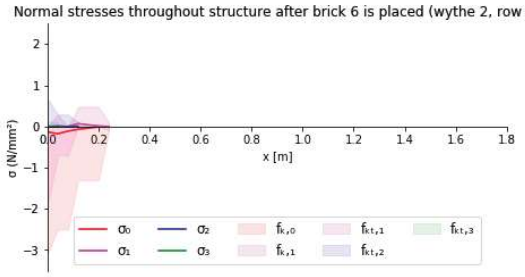


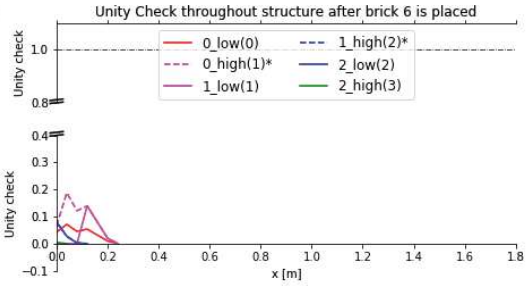

Normal stresses throughout structure after brick 4 is placed (wythe 0, row 2)
σ (N/mm²)
x [m]
σ0
σ1
σ2
σ3
fk,0
fk,1
fkt,1
fkt,2
fkt,3

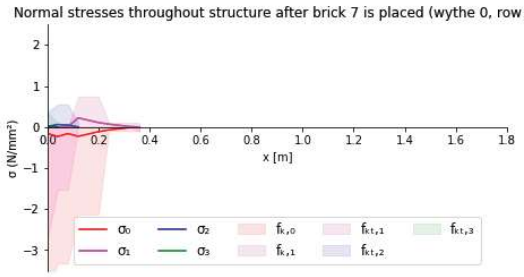

Unity Check throughout structure after brick 4 is placed
Unity check
x [m]
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

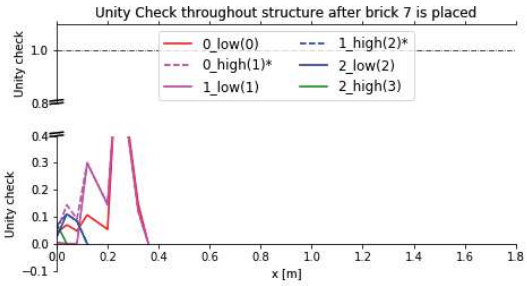

Normal stresses throughout structure after brick 5 is placed (wythe 1, row 1)
σ (N/mm²)
x [m]
σ0
σ1
σ2
σ3
fk,0
fk,1
fkt,1
fkt,2
fkt,3

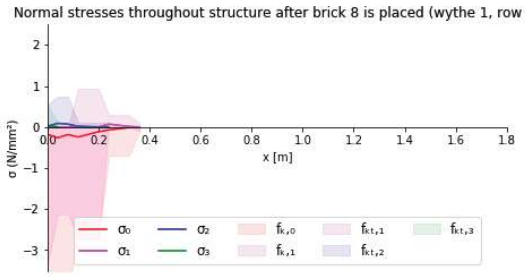

Unity Check throughout structure after brick 5 is placed
Unity check
x [m]
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

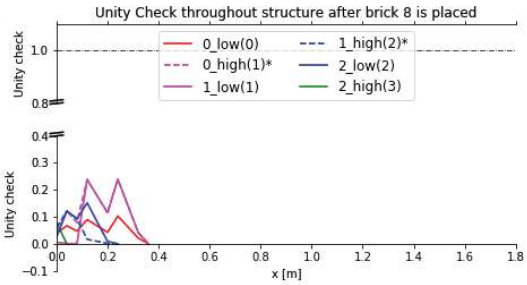

Normal stresses throughout structure after brick 6 is placed (wythe 2, row 0)
σ (N/mm²)
x [m]
σ0
σ1
σ2
σ3
fk,0
fk,1
fkt,1
fkt,2
fkt,3

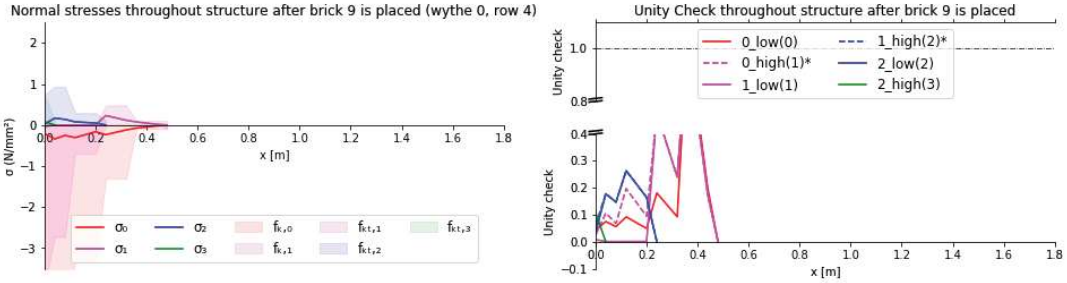

Unity Check throughout structure after brick 6 is placed
Unity check
x [m]
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

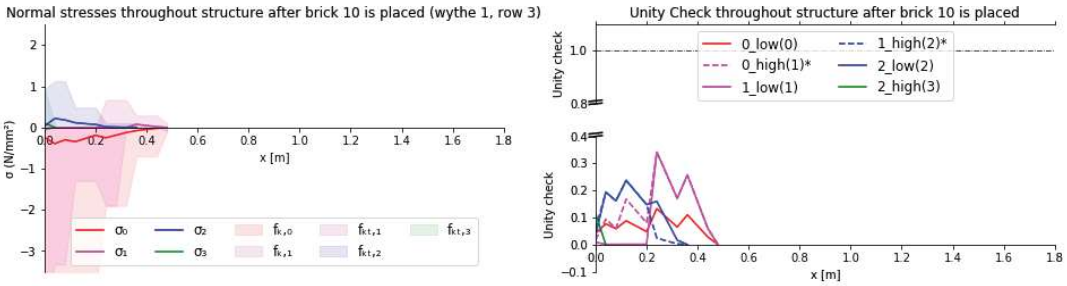

Normal stresses throughout structure after brick 7 is placed (wythe 0, row 3)
σ (N/mm²)
x [m]
σ0
σ1
σ2
σ3
fk,0
fk,1
fkt,1
fkt,2
fkt,3

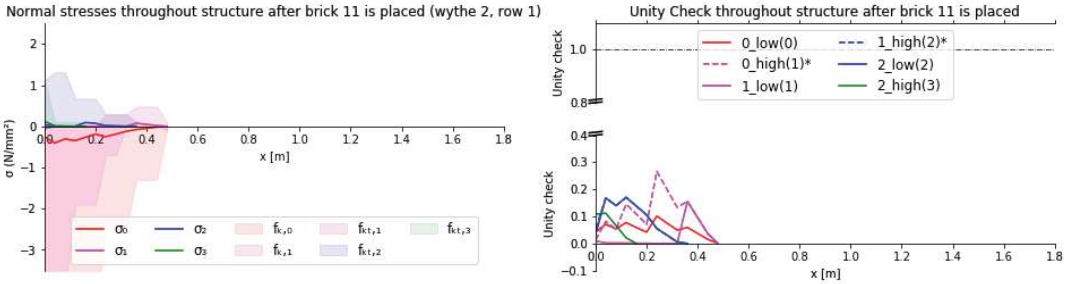

Unity Check throughout structure after brick 7 is placed
Unity check
x [m]
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

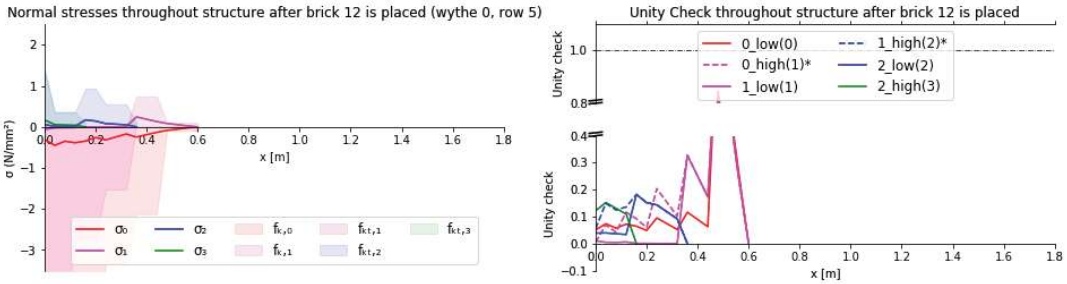

Normal stresses throughout structure after brick 8 is placed (wythe 1, row 2)
σ (N/mm²)
x [m]
σ0
σ1
σ2
σ3
fk,0
fk,1
fkt,1
fkt,2
fkt,3

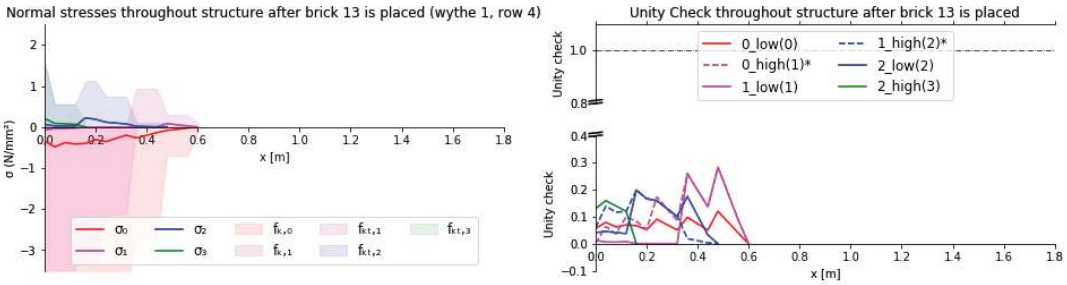

Unity Check throughout structure after brick 8 is placed
Unity check
x [m]
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

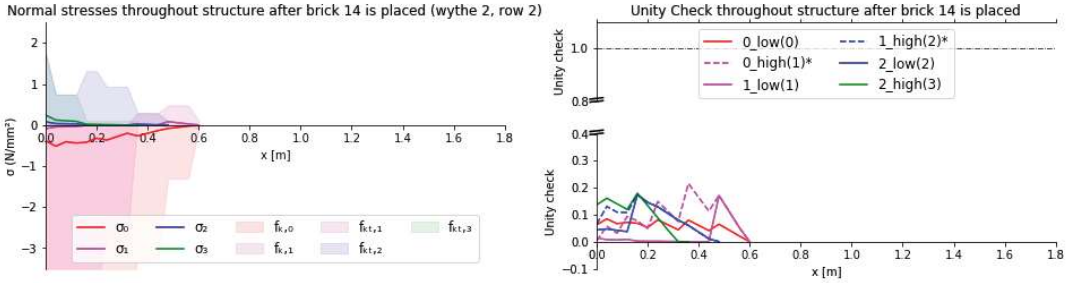

Normal stresses throughout structure after brick 9 is placed (wythe 0, row 4)
Unity Check throughout structure after brick 9 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

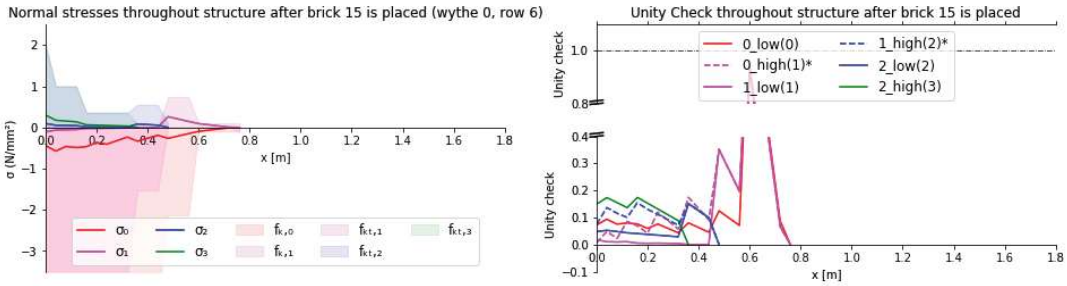

Normal stresses throughout structure after brick 10 is placed (wythe 1, row 3)
Unity Check throughout structure after brick 10 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

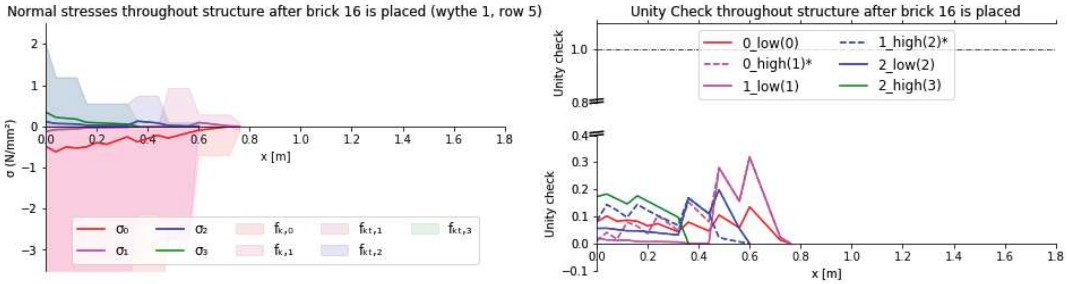

Normal stresses throughout structure after brick 11 is placed (wythe 2, row 1)
Unity Check throughout structure after brick 11 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

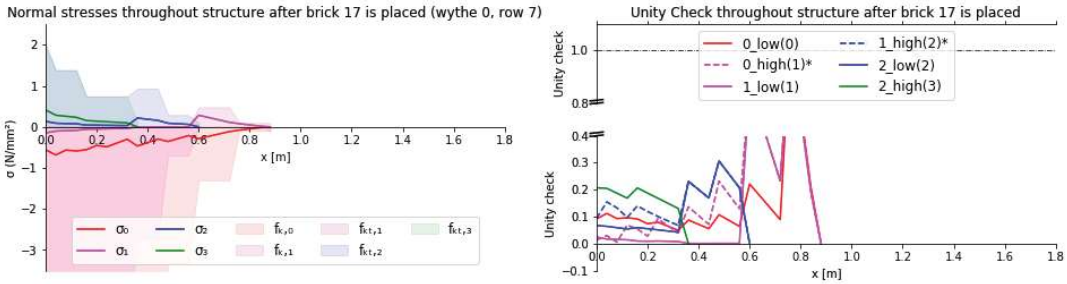

Normal stresses throughout structure after brick 12 is placed (wythe 0, row 5)
Unity Check throughout structure after brick 12 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

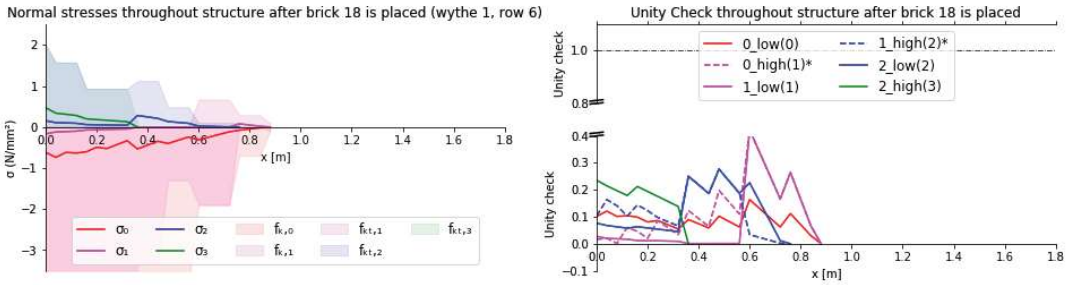

Normal stresses throughout structure after brick 13 is placed (wythe 1, row 4)
Unity Check throughout structure after brick 13 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

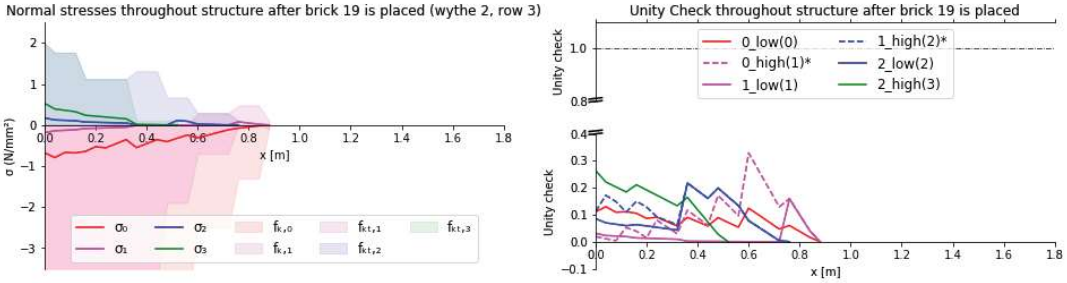

Normal stresses throughout structure after brick 14 is placed (wythe 2, row 2)
Unity Check throughout structure after brick 14 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

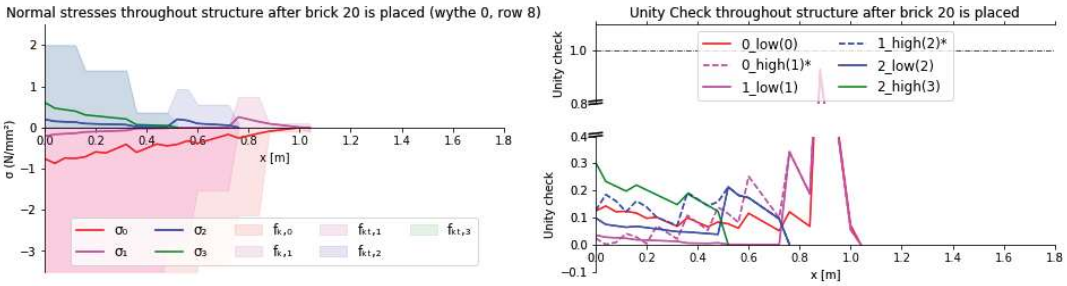

Normal stresses throughout structure after brick 15 is placed (wythe 0, row 6)
Unity Check throughout structure after brick 15 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

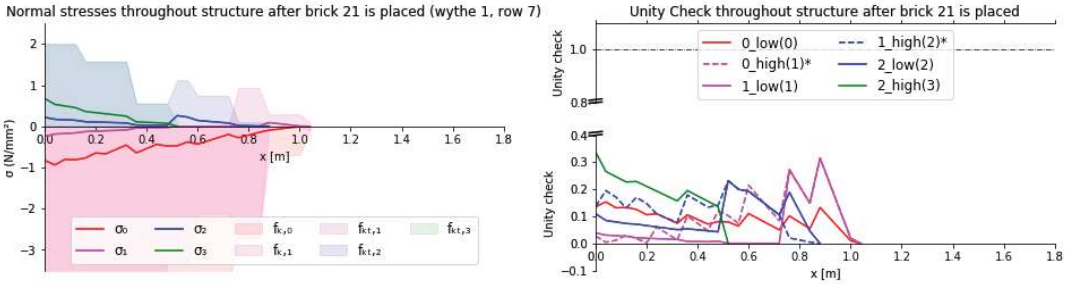

Normal stresses throughout structure after brick 16 is placed (wythe 1, row 5)
Unity Check throughout structure after brick 16 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

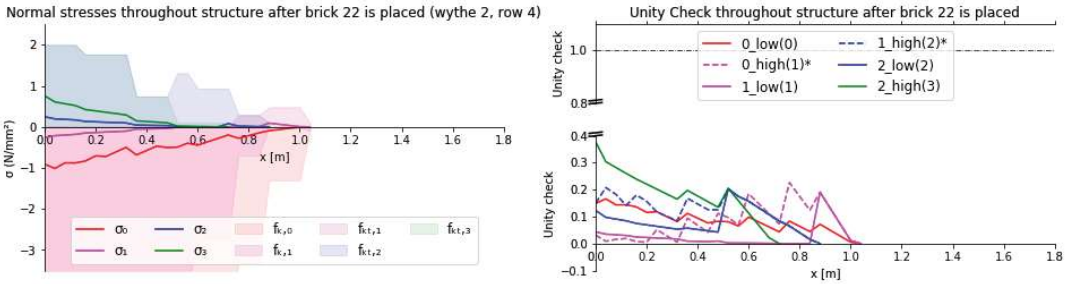

Normal stresses throughout structure after brick 17 is placed (wythe 0, row 7)
Unity Check throughout structure after brick 17 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

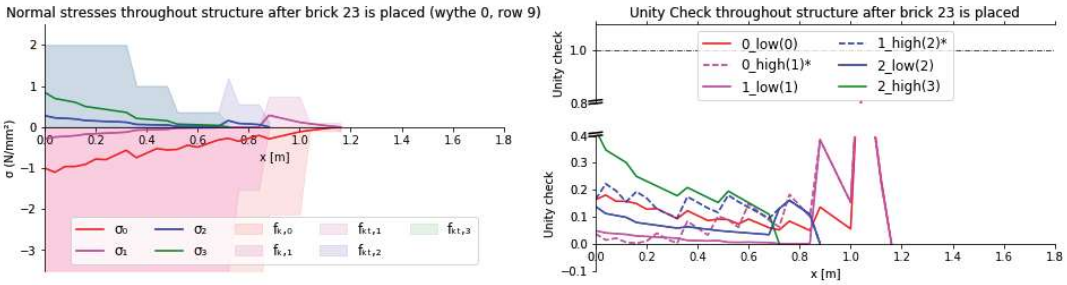

Normal stresses throughout structure after brick 18 is placed (wythe 1, row 6)
Unity Check throughout structure after brick 18 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

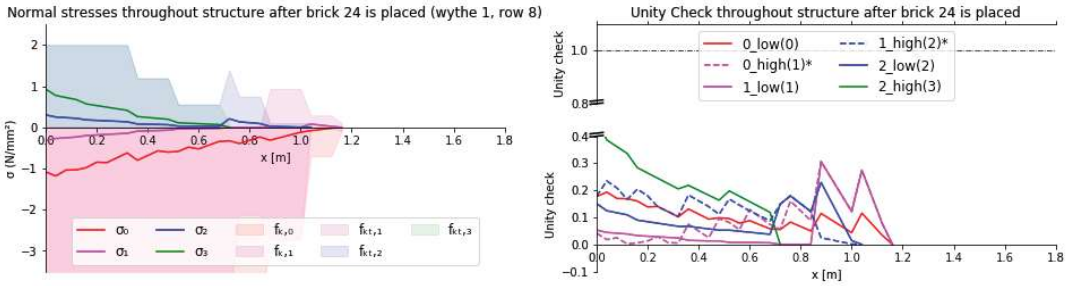

Normal stresses throughout structure after brick 19 is placed (wythe 2, row 3)
Unity Check throughout structure after brick 19 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)


Normal stresses throughout structure after brick 20 is placed (wythe 0, row 8)
Unity Check throughout structure after brick 20 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

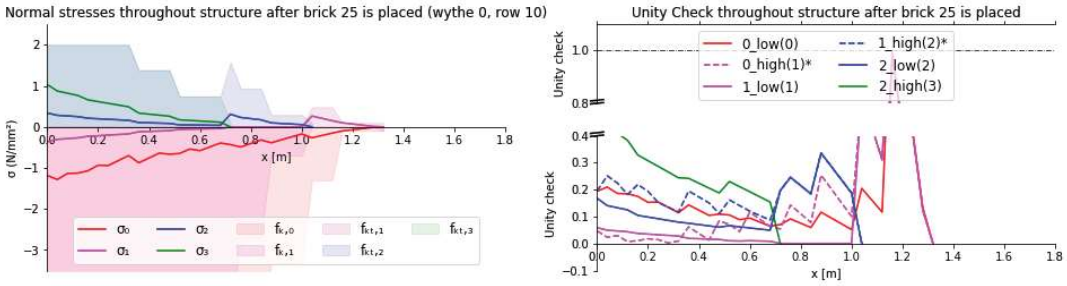

Normal stresses throughout structure after brick 21 is placed (wythe 1, row 7)
Unity Check throughout structure after brick 21 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)


Normal stresses throughout structure after brick 22 is placed (wythe 2, row 4)
Unity Check throughout structure after brick 22 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

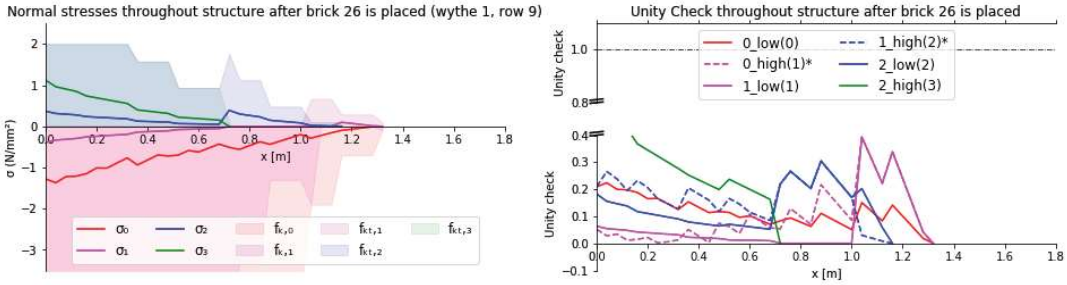

Normal stresses throughout structure after brick 23 is placed (wythe 0, row 9)
Unity Check throughout structure after brick 23 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)


Normal stresses throughout structure after brick 24 is placed (wythe 1, row 8)
σ (N/mm²)
x [m]
σ0 σ2 fk,0 fkt,1 fkt,3
σ1 σ3 fk,1 fkt,2

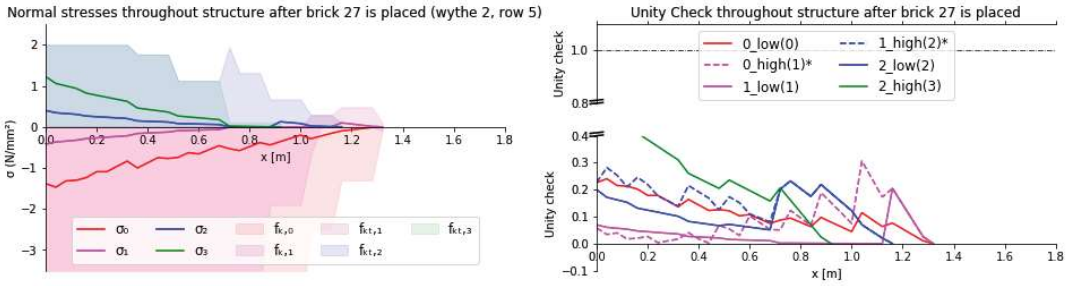
Unity Check throughout structure after brick 24 is placed
Unity check
x [m]
0_low(0) 1_high(2)*
0_high(1)* 2_low(2)
1_low(1) 2_high(3)


Normal stresses throughout structure after brick 25 is placed (wythe 0, row 10)
σ (N/mm²)
x [m]
σ0 σ2 fk,0 fkt,1 fkt,3
σ1 σ3 fk,1 fkt,2

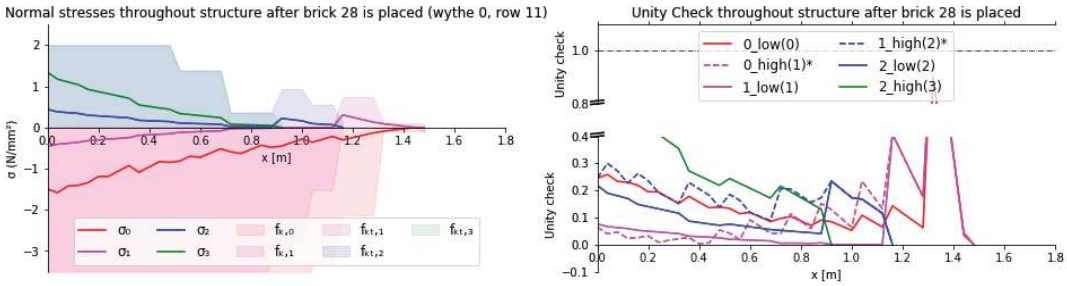
Unity Check throughout structure after brick 25 is placed
Unity check
x [m]
0_low(0) 1_high(2)*
0_high(1)* 2_low(2)
1_low(1) 2_high(3)


Normal stresses throughout structure after brick 26 is placed (wythe 1, row 9)
σ (N/mm²)
x [m]
σ0 σ2 fk,0 fkt,1 fkt,3
σ1 σ3 fk,1 fkt,2

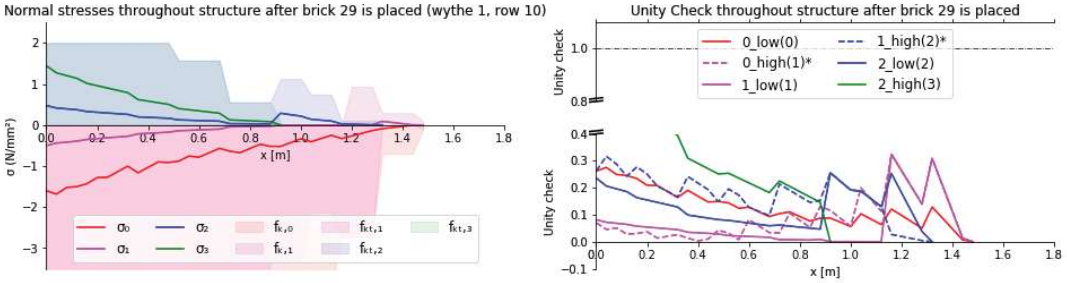
Unity Check throughout structure after brick 26 is placed
Unity check
x [m]
0_low(0) 1_high(2)*
0_high(1)* 2_low(2)
1_low(1) 2_high(3)

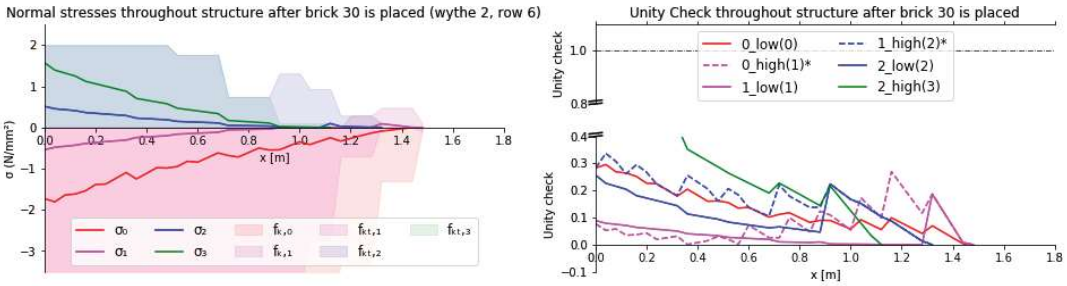

Normal stresses throughout structure after brick 27 is placed (wythe 2, row 5)
σ (N/mm²)
x [m]
σ0 σ2 fk,0 fkt,1 fkt,3
σ1 σ3 fk,1 fkt,2

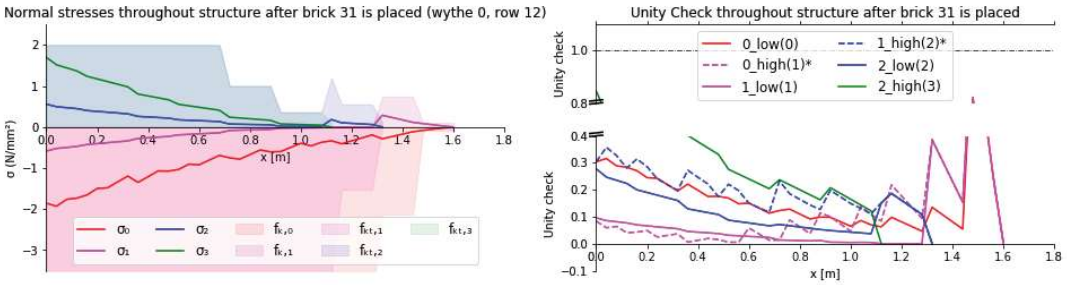
Unity Check throughout structure after brick 27 is placed
Unity check
x [m]
0_low(0) 1_high(2)*
0_high(1)* 2_low(2)
1_low(1) 2_high(3)

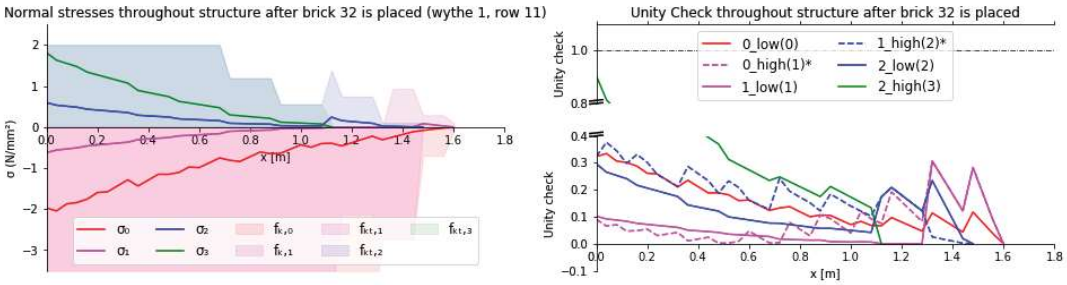

Normal stresses throughout structure after brick 28 is placed (wythe 0, row 11)
σ (N/mm²)
x [m]
σ0 σ2 fk,0 fkt,1 fkt,3
σ1 σ3 fk,1 fkt,2

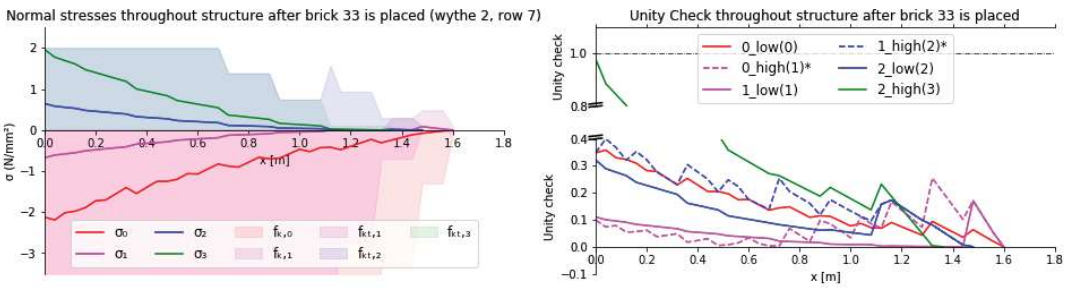
Unity Check throughout structure after brick 28 is placed
Unity check
x [m]
0_low(0) 1_high(2)*
0_high(1)* 2_low(2)
1_low(1) 2_high(3)

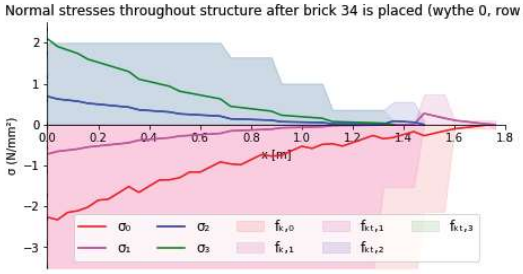

Normal stresses throughout structure after brick 29 is placed (wythe 1, row 10)
Unity Check throughout structure after brick 29 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

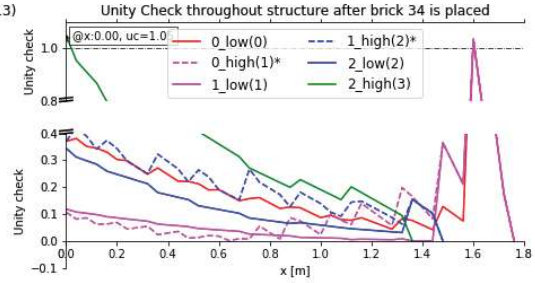

Normal stresses throughout structure after brick 30 is placed (wythe 2, row 6)
Unity Check throughout structure after brick 30 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)


Normal stresses throughout structure after brick 31 is placed (wythe 0, row 12)
Unity Check throughout structure after brick 31 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)


Normal stresses throughout structure after brick 32 is placed (wythe 1, row 11)
Unity Check throughout structure after brick 32 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)


Normal stresses throughout structure after brick 33 is placed (wythe 2, row 7)
Unity Check throughout structure after brick 33 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)





In [10]:
```
#
get = 28
Cfan = np.array(Cstrres)[:,:,get]
Tfan = np.array(Tstrres)[:,:,get]
Sigan = np.array(Sigres)[:,:,get]

start = [min([idx if ele==wth else len(ll) for idx,ele in enumerate(ll)])
         for wth in [0,1,2]]
start.append(start[-1])

Inter = np.array([[0 if lst<start[idx] else ele
                   for idx,ele in enumerate(inter[:,get])]
                  for lst,Lst in enumerate(range(len(seq)))])

Sigani = np.array([np.zeros(len(seq)),Sigan[:,0],Sigan[:,1]+Inter[:,1],Sigan[:,1],
                   Sigan[:,2]+Inter[:,2],Sigan[:,2],Sigan[:,3],np.zeros(len(seq))])
Cfani = np.array([Cfan[:,0],Cfan[:,0],Cfan[:,1],Cfan[:,1],Cfan[:,2],Cfan[:,3]])
Tfani = np.array([Tfan[:,0],Tfan[:,0],Tfan[:,1],Tfan[:,1],Tfan[:,2],Tfan[:,3]])
Cfani[2:4,:2],Tfani[2:4,:2] = np.nan,np.nan
Cfani[4:7,:6],Tfani[4:7,:6] = np.nan,np.nan

print(Cfani)
print(Tfani)
print(Sigani)
```

```
[[-0.1        -0.1        -0.1        -0.1        -0.1        -0.1
  -0.1        -0.1        -0.1        -0.1        -0.1        -0.1
  -0.1        -0.1        -0.1        -0.1        -0.1        -0.1
  -0.1        -0.1        -0.1        -0.1        -0.1        -0.7006358
  -1.30127159 -1.90190739 -2.50254319 -3.32841741 -3.92905321 -4.52968901
  -5.35556323 -5.95619902 -6.10635797 -6.10635797]
 [-0.1        -0.1        -0.1        -0.1        -0.1        -0.1
  -0.1        -0.1        -0.1        -0.1        -0.1        -0.1
  -0.1        -0.1        -0.1        -0.1        -0.1        -0.1
  -0.1        -0.1        -0.1        -0.1        -0.1        -0.7006358
  -1.30127159 -1.90190739 -2.50254319 -3.32841741 -3.92905321 -4.52968901
  -5.35556323 -5.95619902 -6.10635797 -6.10635797]
 [        nan         nan -0.1        -0.1        -0.1        -0.1
  -0.1        -0.1        -0.1        -0.1        -0.1        -0.1
  -0.1        -0.1        -0.1        -0.1        -0.1        -0.1
  -0.1        -0.1        -0.1        -0.1        -0.1        -0.7006358
  -1.30127159 -1.90190739 -0.7006358  -1.52651002 -2.12714582 -2.72778161
  -3.55365583 -4.15429163 -4.75492743 -5.58080165]
 [        nan         nan -0.1        -0.1        -0.1        -0.1
  -0.1        -0.1        -0.1        -0.1        -0.1        -0.1
  -0.1        -0.1        -0.1        -0.1        -0.1        -0.1
  -0.1        -0.1        -0.1        -0.1        -0.1        -0.7006358
  -1.30127159 -1.90190739 -0.7006358  -1.52651002 -2.12714582 -2.72778161
  -3.55365583 -4.15429163 -4.75492743 -5.58080165]
 [        nan         nan         nan         nan         nan         nan
  -0.1        -0.1        -0.1        -0.1        -0.1        -0.1
  -0.1        -0.1        -0.1        -0.1        -0.1        -0.1
  -0.1        -0.1        -0.1        -0.1        -0.1        -0.1
  -0.1        -0.1        -0.7006358  -1.52651002 -2.12714582 -2.72778161
  -3.55365583 -4.15429163 -4.75492743 -0.92587422]
 [        nan         nan         nan         nan         nan         nan
  -0.1        -0.1        -0.1        -0.1        -0.1        -0.1
  -0.1        -0.1        -0.1        -0.1        -0.1        -0.1
  -0.1        -0.1        -0.1        -0.1        -0.1        -0.1
```

```
  -0.1        -0.1        -0.1        -0.1        -0.1        -0.1
  -0.1        -0.1        -0.1        -0.92587422]]
[[0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1
  0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1
  0.1     0.1     0.1     0.1     0.1     0.29    0.48    0.67    0.86
  1.12125 1.31125 1.50125 1.7625  1.9525  2.      2.     ]
 [0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1
  0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1
  0.1     0.1     0.1     0.1     0.1     0.29    0.48    0.67    0.86
  1.12125 1.31125 1.50125 1.7625  1.9525  2.      2.     ]
 [    nan     nan 0.1     0.1     0.1     0.1     0.1     0.1     0.1
  0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1
  0.1     0.1     0.1     0.1     0.1     0.29    0.48    0.67    0.29
  0.55125 0.74125 0.93125 1.1925  1.3825  1.5725  1.83375]
 [    nan     nan 0.1     0.1     0.1     0.1     0.1     0.1     0.1
  0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1
  0.1     0.1     0.1     0.1     0.1     0.29    0.48    0.67    0.29
  0.55125 0.74125 0.93125 1.1925  1.3825  1.5725  1.83375]
 [    nan     nan     nan     nan     nan     nan 0.1     0.1     0.1
  0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1
  0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.29
  0.55125 0.74125 0.93125 1.1925  1.3825  1.5725  0.36125]
 [    nan     nan     nan     nan     nan     nan 0.1     0.1     0.1
  0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1
  0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1
  0.1     0.1     0.1     0.1     0.1     0.1     0.36125]]
[[ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00 -2.37071738e-02 -2.37071738e-02
  -1.51863356e-01 -1.58455012e-01 -1.58455012e-01 -2.17113514e-01
  -2.50139704e-01 -2.50139704e-01 -3.35502123e-01 -3.95281863e-01
  -4.18935571e-01 -4.68881929e-01]
 [ 0.00000000e+00  0.00000000e+00  1.48887260e-01  1.48887260e-01
   1.48887260e-01  1.48887260e-01  1.48887260e-01  1.48887260e-01
   1.48887260e-01  1.48887260e-01  1.48887260e-01  1.48887260e-01
   1.48887260e-01  1.48887260e-01  1.48887260e-01  1.48887260e-01
   1.48887260e-01  1.48887260e-01  1.48887260e-01  1.48887260e-01
   1.48887260e-01  1.48887260e-01  1.71268984e-01  1.71268984e-01
   2.98099716e-01  1.48887260e-01  1.48887260e-01  1.48555897e-01
   1.48230705e-01  1.48230705e-01  1.47899343e-01  1.47574165e-01
   1.39541349e-01  1.22745290e-01]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  2.23817241e-02  2.23817241e-02
   1.49212456e-01  2.77555756e-17  2.77555756e-17 -3.31362426e-04
  -6.56554315e-04 -6.56554315e-04 -9.87916741e-04 -1.31309461e-03
  -9.34591112e-03 -2.61419693e-02]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  2.47730000e-01  2.47730000e-01  2.47730000e-01
   2.47730000e-01  2.47730000e-01  2.47730000e-01  2.47730000e-01
   2.47730000e-01  2.47730000e-01  2.47730000e-01  2.47730000e-01
   2.47730000e-01  2.47730000e-01  2.47730000e-01  2.47730000e-01
   2.47730000e-01  2.47730000e-01  2.47730000e-01  2.47730000e-01
   2.47730000e-01  2.53671263e-01  2.53671263e-01  3.11667041e-01
   3.44042847e-01  3.44042847e-01  4.28742541e-01  4.87871925e-01
   2.47730000e-01  2.64084241e-01]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  5.94126308e-03  5.94126308e-03  6.39370409e-02
   9.63128469e-02  9.63128469e-02  1.81012541e-01  2.40141926e-01
   0.00000000e+00  1.63542416e-02]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
```

```
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   2.32089651e-02  7.27135065e-02]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00]]
```

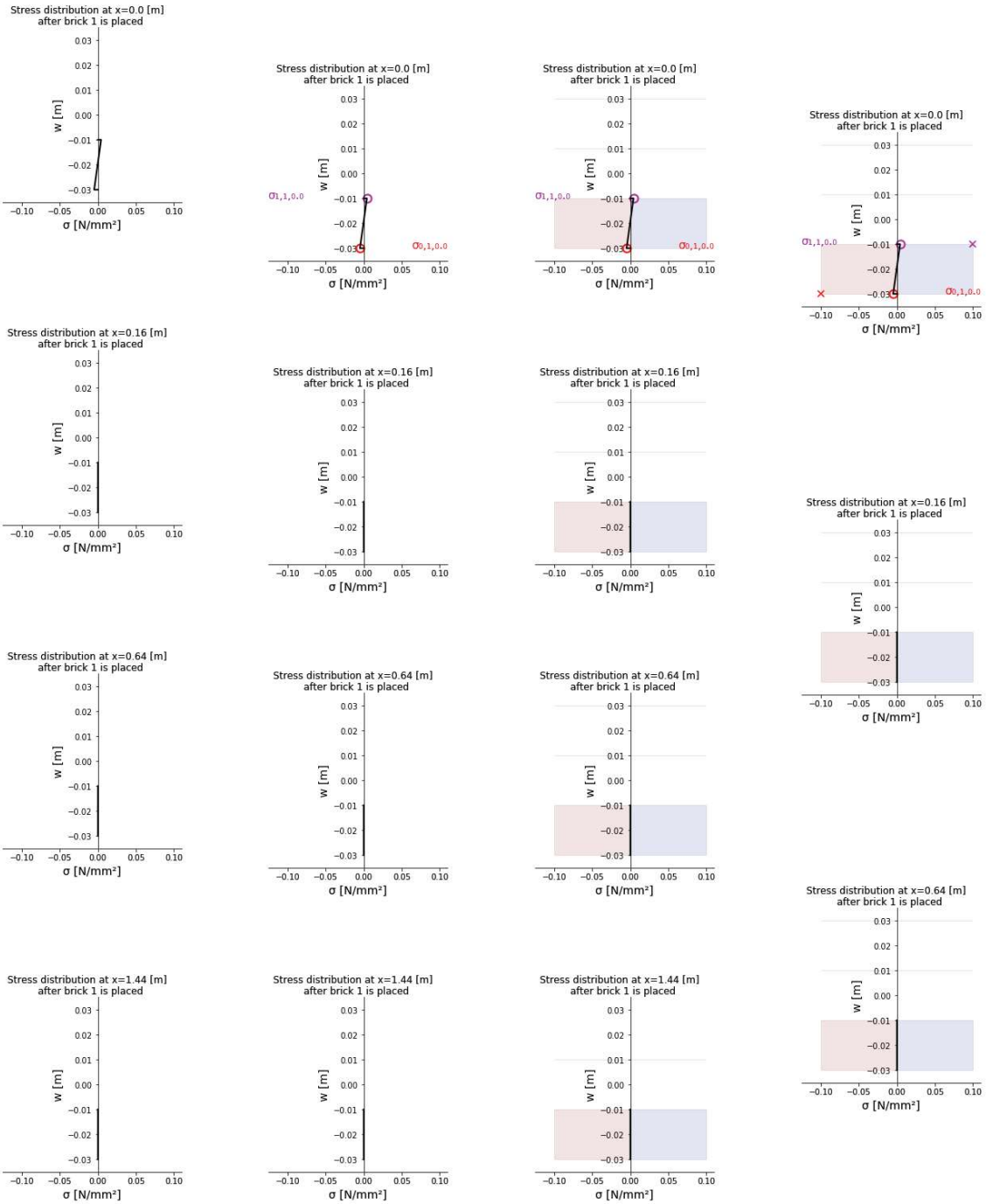
In [11]:
```
#
len(Cfani)
phasedC = Cfani
phasedT = Tfani
phasedS = Sigani
np.savetxt('C:/Users/Joris/Documents/Master Thesis quick access/PhasedC.txt',
           phasedC, delimiter = ';', fmt = '%s')
np.savetxt('C:/Users/Joris/Documents/Master Thesis quick access/PhasedT.txt',
           phasedT, delimiter = ';', fmt = '%s')
np.savetxt('C:/Users/Joris/Documents/Master Thesis quick access/PhasedS.txt',
           phasedS, delimiter = ';', fmt = '%s')
```

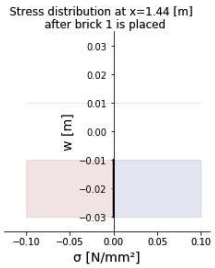
In [12]:
```
# posi stands for the x-linspace, plcd stands for bricks placed:
# posi=0 is at support, plcd=0 is 1 brick placed
xshift=4
runn = []

[runn.append([[0,0,0],[0,xshift*1**2,0],[0,xshift*2**2,0],
              [0,xshift*3**2,0]][ele]) for ele in range(4)]
[runn.append([[0,0,1],[0,xshift*1**2,1],[0,xshift*2**2,1],
              [0,xshift*3**2,1]][ele]) for ele in range(4)]
[runn.append([[0,0,1],[0,xshift*1**2,1],[0,xshift*2**2,1],
              [0,xshift*3**2,1]][ele]) for ele in range(4)]
[runn.append([[0,0,1],[0,xshift*1**2,1],[0,xshift*2**2,1],
              [0,xshift*3**2,1]][ele]) for ele in range(4)]
manu = [0,1,2,3,4,5,6,7,8,12,16,20,24,28,30]
if len(seq)>8:
    manu = np.linspace(0,8,9).tolist()
else:
    manu = np.linspace(0,len(seq)-1,len(seq)).tolist()
manu.extend([ele for ele in np.arange(0,100,4) if ele>manu[-1] and ele<len(seq)-3])
manu.extend([len(seq)-3,len(seq)-2])
manu = [int(ele) for ele in manu]
# manu = [1,2,3,4,5,6,7,8,12,16,20,24,28,30]
manul = len(manu)
for i in range(len(manu)*4):
    runn.append([manu[int(i/4)], xshift*int((i%4)**2), 1]) # plcd, posi, labelled
print(runn)
print()

for plc in range(len(runn)):
    posi = runn[plc][1]
    plcd = runn[plc][0]
    lbl = runn[plc][2]
    hh = [[-1.5*t,-1.5*t,-.5*t,-.5*t,.5*t,.5*t,1.5*t,1.5*t]
          for ele in range(len(Sigani[0]))]
    hh = [-1.5*t,-1.5*t,-.5*t,-.5*t,.5*t,.5*t,1.5*t,1.5*t]

    Cfan = np.array(Cstrres)[:,:,posi]
    Tfan = np.array(Tstrres)[:,:,posi]
    Sigan = np.array(Sigres)[:,:,posi]

    length = [sum([1 if ele==wth else 0 for idx,ele in enumerate(ll[0:plcd+1])])-1
              for wth in [0,1,2]]
    start = [y0[length[0]]<x[posi], y1[length[1]]<x[posi], y2[length[2]]<x[posi]]
    start = [True if length[idx]==-1 else ele for idx,ele in enumerate(start)]
    start.append(start[-1])

#     Inter = np.array([[0 if lst<start[idx] else ele
#                        for idx,ele in enumerate(inter[:,posi+1])]
#                       for lst,Lst in enumerate(range(len(seq)))])
    Inter = np.array([[0 if start[idx] else ele
                       for idx,ele in enumerate(inter[:,posi+1])]
                      for lst,Lst in enumerate(range(len(seq)))])

    Sigani = np.array([np.zeros(len(seq)),#0
                       Sigan[:,0],#1
                       Sigan[:,1]+Inter[:,1],#2
```

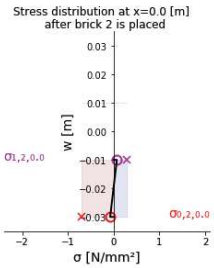
```
                    Sigan[:,1],#3
                    Sigan[:,2]+Inter[:,2],#4
                    Sigan[:,2],#5
                    Sigan[:,3],#6
                    np.zeros(len(seq))])#7
    Cfani = np.array([Cfan[:,0],Cfan[:,0],Cfan[:,1],Cfan[:,1],Cfan[:,2],Cfan[:,3]])
    Tfani = np.array([Tfan[:,0],Tfan[:,0],Tfan[:,1],Tfan[:,1],Tfan[:,2],Tfan[:,3]])
    Sigani[5] = [0 if Inter[:,2][idx]==0 else ele
                 for idx,ele in enumerate(Sigani[5])]
    Sigani[3] = [0 if Inter[:,1][idx]==0 else ele
                 for idx,ele in enumerate(Sigani[3])]

    for i in range(len(Sigani)-2-1,0,-1): # i is per interface
        i +=2
        for j in range(len(Sigani[i])): # j is per plcd
            if Sigani[i-1][j]==0 and Sigani[i-1][j-1]!=0:
                Sigani[i-1][j]+=10**-6
            if i>1 and i<7 and Sigani[i][j]==0:
                Cfani[i-1][j] = np.nan
                Tfani[i-1][j] = np.nan
            if Sigani[i-1][j]==0:# and Sigani[i-1][j-1]==0:
                Sigani[i][j] = np.nan
    if plcd==2:
        Cfani[2:4,plcd] = fm
        Tfani[2:4,plcd] = fti

    fig = plt.figure()
    fig.set_figwidth(4),fig.set_figheight(4)
    ax = fig.add_subplot(1, 1, 1)
    if plc>=4*4 or len(runn)/4==len(manu):
        ax.set_xlim(-2.4,2.1)
    else:
        ax.set_xlim(-.1257,.11)
    ax.set_ylim(-1.75*t,1.75*t)
    ax.spines['left'].set_position('zero')
    ax.spines['right'].set_color('none')
    ax.yaxis.tick_left()
    ax.spines['bottom'].set_position(('axes',0))
    ax.spines['top'].set_color('none')
    ax.xaxis.tick_bottom()
    ax.set_xlabel('\u03C3 [N/mmš]', fontsize=14)
    ax.set_ylabel('w [m]', fontsize=14)
    fig.suptitle('Stress distribution at x=%s [m] \n after brick %s is placed' % (x[posi],plcd+1),fontsize=12)

    if lbl==1:
        lyrs = np.count_nonzero([ele for ele in Sigani[:,plcd]
                                 if (not np.isnan(ele))])
        if lyrs==2:
            sig = [1,2,4,6]
        elif lyrs==4:
            sig = [1,3,4,6]
        else:
            sig = [1,3,5,6]

        clr = ['r','m','b','g']
        sig = [ele for ele in sig if (Sigani[:,plcd][ele]!=0)*(
            not np.isnan(Sigani[:,plcd][ele]))]
        for i in range(len(sig)):
            text= "\u03C3%s" % (sub('%s,%s,%s' % (i,plcd+1,x[posi]))) #text
            labpo = [-np.sign(Sigani[sig[i],plcd])/2+0.5,hh[sig[i]]/sum(
                np.abs(ax.get_ylim()))+0.5]
            ax.annotate(text, xy=(labpo[0],labpo[1]), xycoords='axes fraction',
                        color=clr[i], ha=['left','right'][int(labpo[0])],
                        fontsize=14)
            plt.plot(Sigani[sig[i],plcd],hh[sig[i]], 'o', ms=10, mfc='none',
                     mew=2,  mec=clr[i])
            if plc>=3*4 or len(runn)/4==len(manu):
                if Sigani[sig[i],plcd]<0:
                    plt.plot(Cfani[sig[i]-1,plcd],hh[sig[i]], 'x', ms=8, mfc='none',
                             mew=1.5,  mec=clr[i])
                if Sigani[sig[i],plcd]<0 and Cfani[sig[i]-1,plcd]<-2.4:
                    plt.arrow(-2.4+.4, hh[sig[i]], -.4, 0, length_includes_head=True,
                              shape='full',width=.0004, head_width = .0015,
                              head_length = .15, ec = 'none', fc = clr[i])
                if Sigani[sig[i],plcd]>=0:
                    plt.plot(Tfani[sig[i]-1,plcd],hh[sig[i]], 'x', ms=8, mfc='none',
                             mew=1.5,  mec=clr[i])

#     print ('Sig:',Sigani[:,plcd])
#     print('Cf',Cfani[:,plcd])
#     print('Tf',Tfani[:,plcd])
    ax.plot(Sigani[:,plcd],hh,color='k',lw=2)
    x_fill = np.zeros(len(hh))
    if plc>=2*4 or len(runn)/4==len(manu):
```

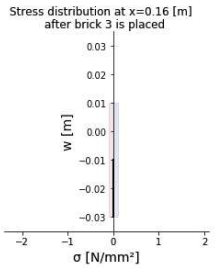
```
            ax.fill_betweenx(hh[1:7],Cfani[:,plcd],x_fill[1:7],color='maroon',alpha=.1)
            ax.fill_betweenx(hh[1:7],Tfani[:,plcd],x_fill[1:7],color='navy',alpha=.1)
        #     if plc==len(runn)-2:
        #         fig.set_figwidth(12),fig.set_figheight(4)
        #         saveplt = ax
        plt.show();
```

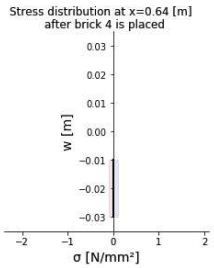
```
[[0, 0, 0], [0, 4, 0], [0, 16, 0], [0, 36, 0], [0, 0, 1], [0, 4, 1], [0, 16, 1], [0, 36, 1], [0, 0, 1], [0, 4, 1],
 [0, 16, 1], [0, 36, 1], [0, 0, 1], [0, 4, 1], [0, 16, 1], [0, 36, 1], [0, 0, 1], [0, 4, 1], [0, 16, 1], [0, 36, 1],
 [1, 0, 1], [1, 4, 1], [1, 16, 1], [1, 36, 1], [2, 0, 1], [2, 4, 1], [2, 16, 1], [2, 36, 1], [3, 0, 1], [3, 4, 1],
 [3, 16, 1], [3, 36, 1], [4, 0, 1], [4, 4, 1], [4, 16, 1], [4, 36, 1], [5, 0, 1], [5, 4, 1], [5, 16, 1], [5, 36, 1],
 [6, 0, 1], [6, 4, 1], [6, 16, 1], [6, 36, 1], [7, 0, 1], [7, 4, 1], [7, 16, 1], [7, 36, 1], [8, 0, 1], [8, 4, 1],
 [8, 16, 1], [8, 36, 1], [12, 0, 1], [12, 4, 1], [12, 16, 1], [12, 36, 1], [16, 0, 1], [16, 4, 1], [16, 16, 1],
 [16, 36, 1], [20, 0, 1], [20, 4, 1], [20, 16, 1], [20, 36, 1], [24, 0, 1], [24, 4, 1], [24, 16, 1], [24, 36, 1],
 [28, 0, 1], [28, 4, 1], [28, 16, 1], [28, 36, 1], [31, 0, 1], [31, 4, 1], [31, 16, 1], [31, 36, 1], [32, 0, 1],
 [32, 4, 1], [32, 16, 1], [32, 36, 1]]
```

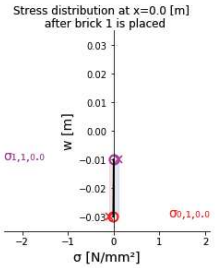


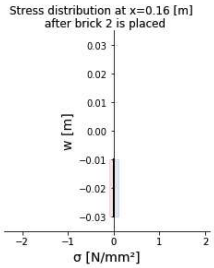

Stress distribution at x=1.44 [m]
after brick 1 is placed
w [m]
σ [N/mm²]

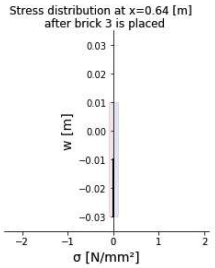

Stress distribution at x=0.0 [m]
after brick 2 is placed
σ1,2,0.0
σ0,2,0.0
w [m]
σ [N/mm²]

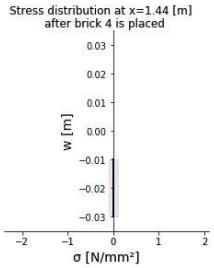

Stress distribution at x=0.16 [m]
after brick 3 is placed
w [m]
σ [N/mm²]

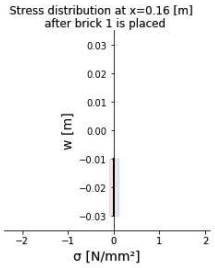

Stress distribution at x=0.64 [m]
after brick 4 is placed
w [m]
σ [N/mm²]

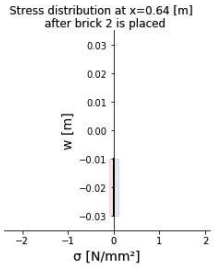

Stress distribution at x=0.0 [m]
after brick 1 is placed
σ1,1,0.0
σ0,1,0.0
w [m]
σ [N/mm²]

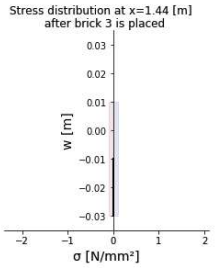

Stress distribution at x=0.16 [m]
after brick 2 is placed
w [m]
σ [N/mm²]

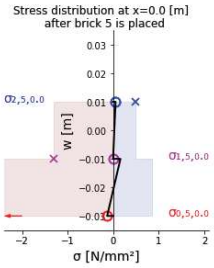

Stress distribution at x=0.64 [m]
after brick 3 is placed
w [m]
σ [N/mm²]

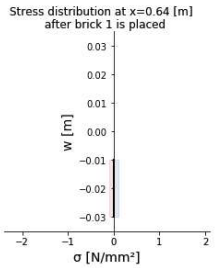

Stress distribution at x=1.44 [m]
after brick 4 is placed
w [m]
σ [N/mm²]

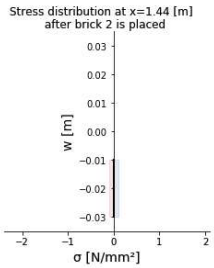

Stress distribution at x=0.16 [m]
after brick 1 is placed
w [m]
σ [N/mm²]

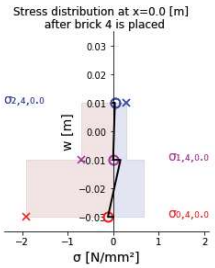

Stress distribution at x=0.64 [m]
after brick 2 is placed
w [m]
σ [N/mm²]

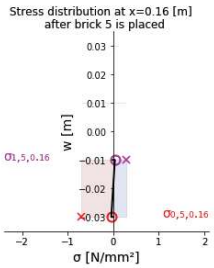

Stress distribution at x=1.44 [m]
after brick 3 is placed
w [m]
σ [N/mm²]

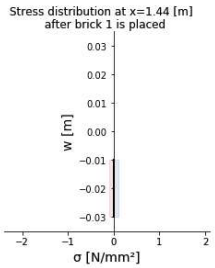

Stress distribution at x=0.0 [m]
after brick 5 is placed
σ2,5,0.0
σ1,5,0.0
σ0,5,0.0
w [m]
σ [N/mm²]

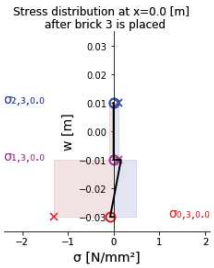

Stress distribution at x=0.64 [m]
after brick 1 is placed
w [m]
σ [N/mm²]

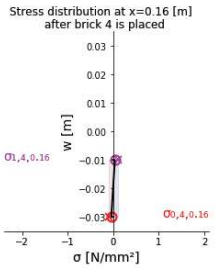

Stress distribution at x=1.44 [m]
after brick 2 is placed
w [m]
σ [N/mm²]

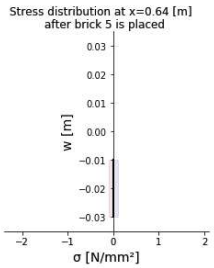

Stress distribution at x=0.0 [m]
after brick 4 is placed
σ2,4,0.0
σ1,4,0.0
σ0,4,0.0
w [m]
σ [N/mm²]

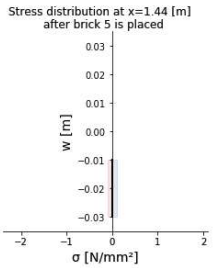

Stress distribution at x=0.16 [m]
after brick 5 is placed
σ1,5,0.16
σ0,5,0.16
w [m]
σ [N/mm²]

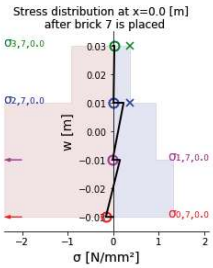

Stress distribution at x=1.44 [m]
after brick 1 is placed
w [m]
σ [N/mm²]

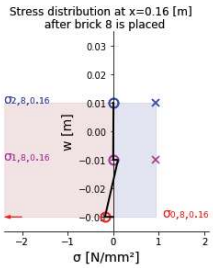

Stress distribution at x=0.0 [m]
after brick 3 is placed
σ2,3,0.0
σ1,3,0.0
σ0,3,0.0
w [m]
σ [N/mm²]

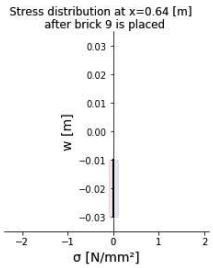

Stress distribution at x=0.16 [m]
after brick 4 is placed
σ1,4,0.16
σ0,4,0.16
w [m]
σ [N/mm²]

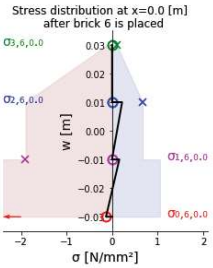

Stress distribution at x=0.64 [m]
after brick 5 is placed
w [m]
σ [N/mm²]

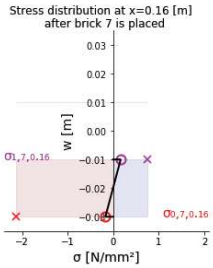

Stress distribution at x=1.44 [m]
after brick 5 is placed
w [m]
σ [N/mm²]

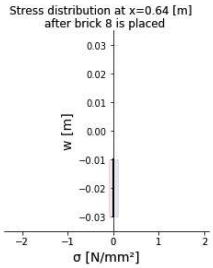

Stress distribution at x=0.0 [m]
after brick 7 is placed
σ3,7,0.0
σ2,7,0.0
σ1,7,0.0
σ0,7,0.0
w [m]
σ [N/mm²]

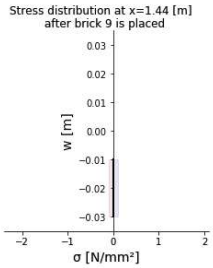

Stress distribution at x=0.16 [m]
after brick 8 is placed
σ2,8,0.16
σ1,8,0.16
σ0,8,0.16
w [m]
σ [N/mm²]

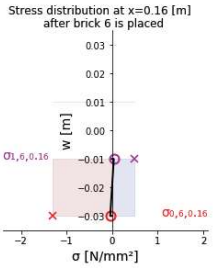

Stress distribution at x=0.64 [m]
after brick 9 is placed
w [m]
σ [N/mm²]

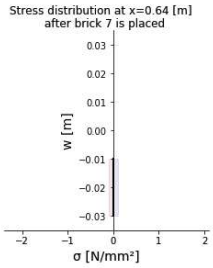

Stress distribution at x=0.0 [m]
after brick 6 is placed
σ3,6,0.0
σ2,6,0.0
σ1,6,0.0
σ0,6,0.0
w [m]
σ [N/mm²]

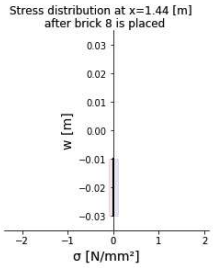

Stress distribution at x=0.16 [m]
after brick 7 is placed
σ1,7,0.16
σ0,7,0.16
w [m]
σ [N/mm²]

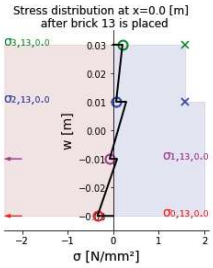

Stress distribution at x=0.64 [m]
after brick 8 is placed
w [m]
σ [N/mm²]

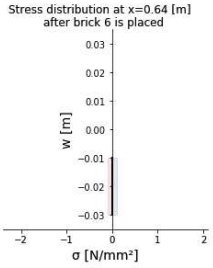

Stress distribution at x=1.44 [m]
after brick 9 is placed
w [m]
σ [N/mm²]

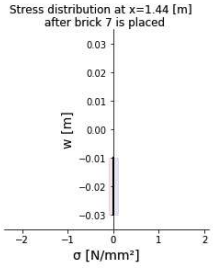

Stress distribution at x=0.16 [m]
after brick 6 is placed
σ1,6,0.16
σ0,6,0.16
w [m]
σ [N/mm²]

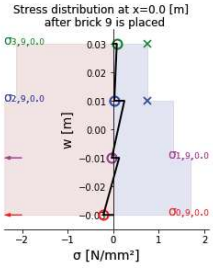

Stress distribution at x=0.64 [m]
after brick 7 is placed
w [m]
σ [N/mm²]

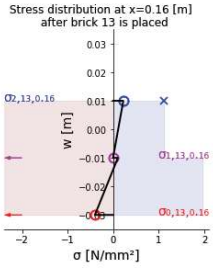

Stress distribution at x=1.44 [m]
after brick 8 is placed
w [m]
σ [N/mm²]

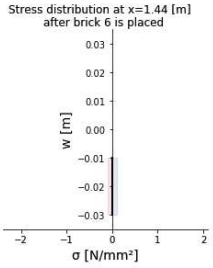

Stress distribution at x=0.0 [m]
after brick 13 is placed
σ3,13,0.0
σ2,13,0.0
σ1,13,0.0
σ0,13,0.0
w [m]
σ [N/mm²]

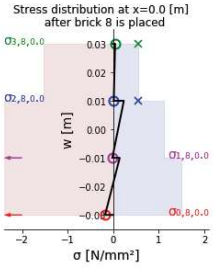

Stress distribution at x=0.64 [m]
after brick 6 is placed
w [m]
σ [N/mm²]

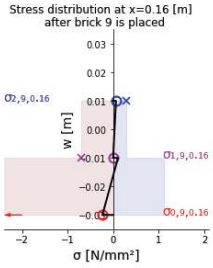

Stress distribution at x=1.44 [m]
after brick 7 is placed
w [m]
σ [N/mm²]

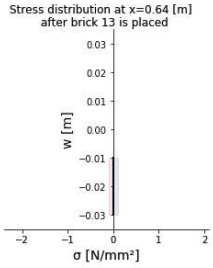

Stress distribution at x=0.0 [m]
after brick 9 is placed
σ3,9,0.0
σ2,9,0.0
σ1,9,0.0
σ0,9,0.0
w [m]
σ [N/mm²]

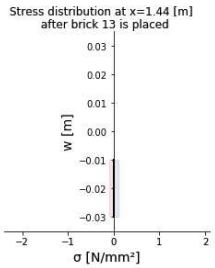

Stress distribution at x=0.16 [m]
after brick 13 is placed
σ2,13,0.16
σ1,13,0.16
σ0,13,0.16
w [m]
σ [N/mm²]

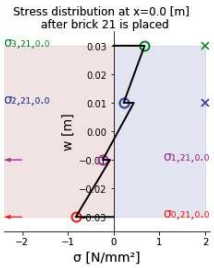

Stress distribution at x=1.44 [m]
after brick 6 is placed
w [m]
σ [N/mm²]

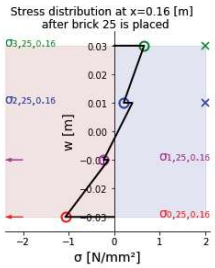

Stress distribution at x=0.0 [m]
after brick 8 is placed
σ3,8,0.0
σ2,8,0.0
σ1,8,0.0
σ0,8,0.0
w [m]
σ [N/mm²]

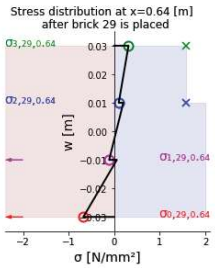

Stress distribution at x=0.16 [m]
after brick 9 is placed
σ2,9,0.16
σ1,9,0.16
σ0,9,0.16
w [m]
σ [N/mm²]

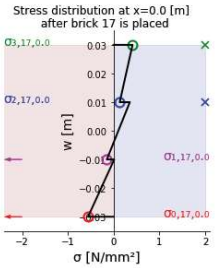

Stress distribution at x=0.64 [m]
after brick 13 is placed
w [m]
σ [N/mm²]

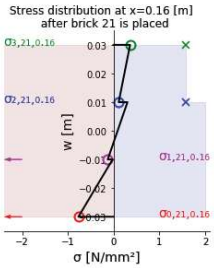

Stress distribution at x=1.44 [m]
after brick 13 is placed
w [m]
σ [N/mm²]

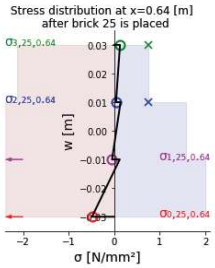

Stress distribution at x=0.0 [m]
after brick 21 is placed
σ3,21,0.0
σ2,21,0.0
σ1,21,0.0
σ0,21,0.0
w [m]
σ [N/mm²]

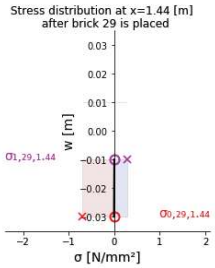

Stress distribution at x=0.16 [m]
after brick 25 is placed
σ3,25,0.16
σ2,25,0.16
σ1,25,0.16
σ0,25,0.16
w [m]
σ [N/mm²]

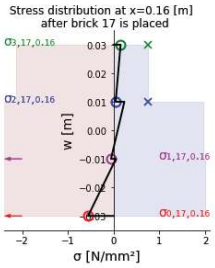

Stress distribution at x=0.64 [m]
after brick 29 is placed
σ3,29,0.64
σ2,29,0.64
σ1,29,0.64
σ0,29,0.64
w [m]
σ [N/mm²]

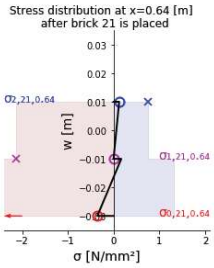

Stress distribution at x=0.0 [m]
after brick 17 is placed
σ3,17,0.0
σ2,17,0.0
σ1,17,0.0
σ0,17,0.0
w [m]
σ [N/mm²]

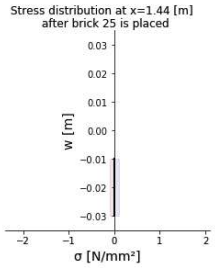

Stress distribution at x=0.16 [m]
after brick 21 is placed
σ3,21,0.16
σ2,21,0.16
σ1,21,0.16
σ0,21,0.16
w [m]
σ [N/mm²]

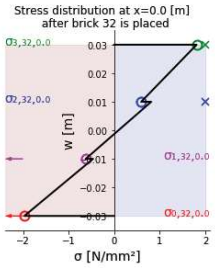

Stress distribution at x=0.64 [m]
after brick 25 is placed
σ3,25,0.64
σ2,25,0.64
σ1,25,0.64
σ0,25,0.64
w [m]
σ [N/mm²]

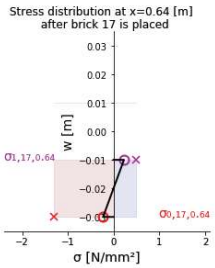

Stress distribution at x=1.44 [m]
after brick 29 is placed
σ1,29,1.44
σ0,29,1.44
w [m]
σ [N/mm²]

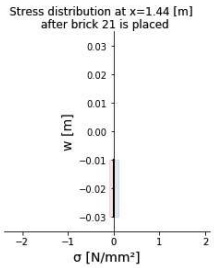

Stress distribution at x=0.16 [m]
after brick 17 is placed
σ3,17,0.16
σ2,17,0.16
σ1,17,0.16
σ0,17,0.16
w [m]
σ [N/mm²]

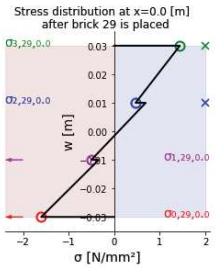

Stress distribution at x=0.64 [m]
after brick 21 is placed
σ2,21,0.64
σ1,21,0.64
σ0,21,0.64
w [m]
σ [N/mm²]

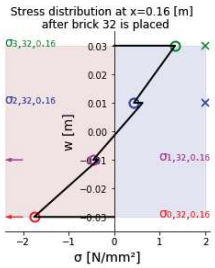

Stress distribution at x=1.44 [m]
after brick 25 is placed
w [m]
σ [N/mm²]

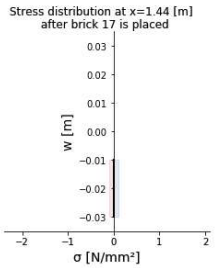

Stress distribution at x=0.0 [m]
after brick 32 is placed
σ3,32,0.0
σ2,32,0.0
σ1,32,0.0
σ0,32,0.0
w [m]
σ [N/mm²]

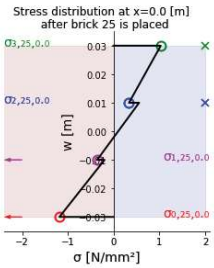

Stress distribution at x=0.64 [m]
after brick 17 is placed
σ1,17,0.64
σ0,17,0.64
w [m]
σ [N/mm²]

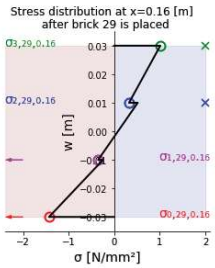

Stress distribution at x=1.44 [m]
after brick 21 is placed
w [m]
σ [N/mm²]

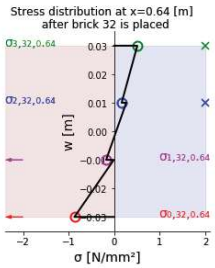

Stress distribution at x=0.0 [m]
after brick 29 is placed
σ3,29,0.0
σ2,29,0.0
σ1,29,0.0
σ0,29,0.0
w [m]
σ [N/mm²]

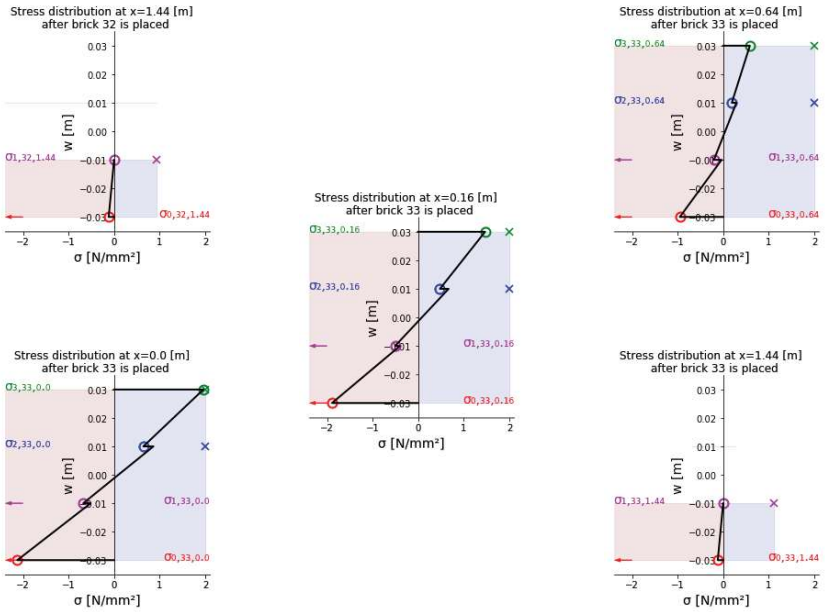

Stress distribution at x=0.16 [m]
after brick 32 is placed
σ3,32,0.16
σ2,32,0.16
σ1,32,0.16
σ0,32,0.16
w [m]
σ [N/mm²]


Stress distribution at x=1.44 [m]
after brick 17 is placed
w [m]
σ [N/mm²]


Stress distribution at x=0.0 [m]
after brick 25 is placed
σ3,25,0.0
σ2,25,0.0
σ1,25,0.0
σ0,25,0.0
w [m]
σ [N/mm²]

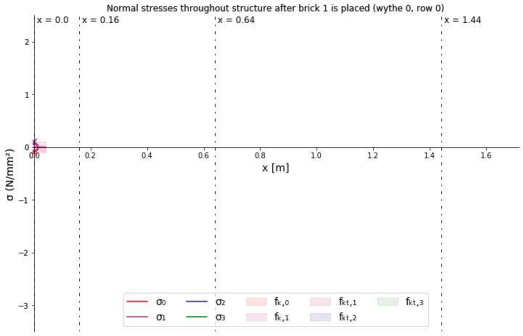

Stress distribution at x=0.16 [m]
after brick 29 is placed
σ3,29,0.16
σ2,29,0.16
σ1,29,0.16
σ0,29,0.16
w [m]
σ [N/mm²]

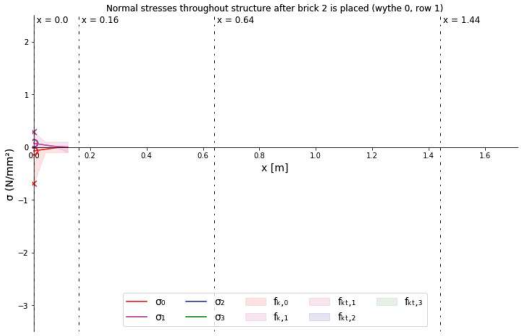

Stress distribution at x=0.64 [m]
after brick 32 is placed
σ3,32,0.64
σ2,32,0.64
σ1,32,0.64
σ0,32,0.64
w [m]
σ [N/mm²]

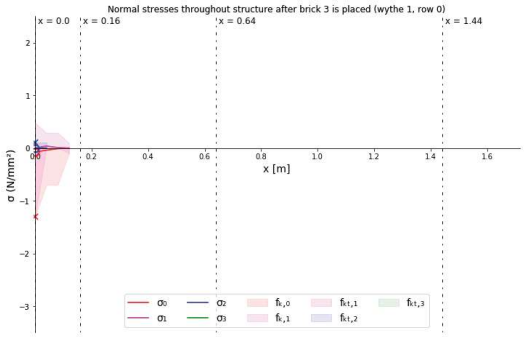


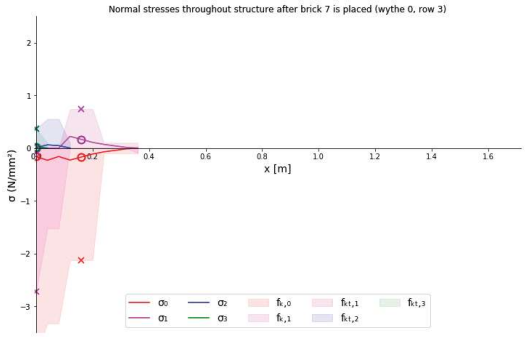
In [13]:
```
#
n0,n1,n2 = -2,-2,-2
id0,id1,id2 = 0,0,0

C0res=np.array([[[0 if Sigres[brk][lay][idx]>0 else ele
                  for idx,ele in enumerate(Cstrres[brk][lay])]
                 for lay,Lay in enumerate(Cstrres[brk])]
                for brk,Brk in enumerate(Cstrres)])
T0res=np.array([[[0 if Sigres[brk][lay][idx]<0 else ele
                  for idx,ele in enumerate(Tstrres[brk][lay])]
                 for lay,Lay in enumerate(Tstrres[brk])]
                for brk,Brk in enumerate(Tstrres)])

for i in range(len(manu)):
    posi = np.array(runn)[:4,1]
    i,add=manu[i],i
    n0,n1,n2 = [sum([1 if ele==wth else 0 for idx,ele in enumerate(ll[0:i+1])])-1
                for wth in [0,1,2]]
    id0,id1 = i0.index(n0+1) if n0>-1 else -1, i1.index(n1+1) if n1>-1 else -1
    id2 = i2.index(n2+1) if n2>-1 else -1

    fig = plt.figure()
    fig.set_figwidth(12),fig.set_figheight(8)
    ax = fig.add_subplot(1, 1, 1)
#
    ax.spines['left'].set_position('zero')
    ax.spines['right'].set_color('none')
    ax.yaxis.tick_left()

    ax.spines['bottom'].set_position('zero')

    ax.spines['top'].set_color('none')
    ax.xaxis.tick_bottom()

    xx = x

    ax.plot(xx[:id0+1],Sigres[i][0][:id0+1], label = '\u03C3%s' % (sub('0')),
            color='red')
    ax.plot(xx[:id0+1],Sigres[i][1][:id0+1], label = '\u03C3%s' % (sub('1')),
            color='magenta')
    ax.plot(xx[:id1+1],Sigres[i][2][:id1+1], label = '\u03C3%s' % (sub('2')),
            color='blue')
    ax.plot(xx[:id2+1],Sigres[i][3][:id2+1], label = '\u03C3%s' % (sub('3')),
            color='green')

    alp = .1
    ax.fill_between(xx[:id0+1],C0res[i][0][:id0+1], color='red',alpha=alp,
                    label = 'f%s' % (sub('k,0')))
    ax.fill_between(xx[:id0+1],C0res[i][1][:id0+1], color='magenta',alpha=alp,
                    label = 'f%s' % (sub('k,1')))
    ax.fill_between(xx[:id0+1],T0res[i][1][:id0+1], color='magenta',alpha=alp,
                    label = 'f%s' % (sub('kt,1')))
    ax.fill_between(xx[:id1+1],Tstrres[i][2][:id1+1], color='blue',alpha=alp,
                    label = 'f%s' % (sub('kt,2')))
```

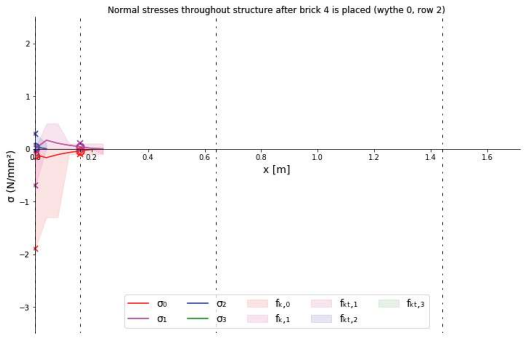
```
    ax.fill_between(xx[:id2+1],Tstrres[i][3][:id2+1], color='green',alpha=alp,
                    label = 'f%s' % (sub('kt,3')))

    ax.set_xlabel('x [m]', fontsize=14)
    ax.set_ylabel('\u03C3 (N/mmš)', fontsize=14)
    ax.title.set_text('Normal stresses throughout structure after brick %s is placed (wythe %s, row %s)
    ' % (i+1,ll[i],nn[i]))

    ax.legend(loc=8, fontsize=14,ncol=5)
    ax.set_ylim([-3.5,2.5]), ax.set_xlim([-0.005,1.715])

    hd_wd = .008*sum(np.abs(ax.get_xlim()))
    hd_ln = 8*hd_wd
    for ar in range(len(posi)):

        if np.sign(Sigres[i][0][posi[ar]]) != 0:
            ax.plot(xx[posi[ar]], Sigres[i][0][posi[ar]], 'o', ms=10, mfc='none',
                    mew=2,  mec='r')
            ax.plot(xx[posi[ar]], COres[i][0][posi[ar]], 'x', ms=8, mfc='none',
                    mew=1.5,  mec='r')
# #
        if np.sign(Sigres[i][1][posi[ar]]) != 0:
            ax.plot(xx[posi[ar]],Sigres[i][1][posi[ar]], 'o', ms=10, mfc='none',
                    mew=2,  mec='m')
            if Sigres[i][1][posi[ar]]<0:
                    ax.plot(xx[posi[ar]], COres[i][1][posi[ar]], 'x', ms=8,
                            mfc='none', mew=1.5,  mec='m')
            if Sigres[i][1][posi[ar]]>=0:
                    ax.plot(xx[posi[ar]], TOres[i][1][posi[ar]], 'x', ms=8,
                            mfc='none', mew=1.5,  mec='m')
# #
        if np.sign(Sigres[i][2][posi[ar]]) != 0:
            ax.plot(xx[posi[ar]],Sigres[i][2][posi[ar]], 'o', ms=10, mfc='none',
                    mew=2,  mec='b')
            ax.plot(xx[posi[ar]], Tstrres[i][2][posi[ar]], 'x', ms=8, mfc='none',
                    mew=1.5,  mec='b')
# #
        if np.sign(Sigres[i][3][posi[ar]]) != 0:
            ax.plot(xx[posi[ar]],Sigres[i][3][posi[ar]], 'o', ms=10, mfc='none',
                    mew=2,  mec='g')
            ax.plot(xx[posi[ar]], Tstrres[i][3][posi[ar]], 'x', ms=8, mfc='none',
                    mew=1.5,  mec='g')

        if i<manu[5] or i==manu[-1]:
                    ax.plot([xx[posi[ar]],xx[posi[ar]]],ax.get_ylim(),
                            ls=(0,(2,7,4,7)),c='k',lw=1)
        if i<manu[3] or i==manu[-1]:
                    ax.annotate(" x = %s" % (x[posi[ar]]),
                                xy=(xx[posi[ar]],ax.get_ylim()[1]),  color='k',
                                fontsize=12, va='top')

    plt.show()
```

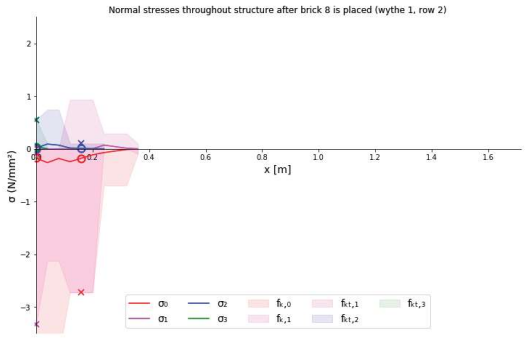


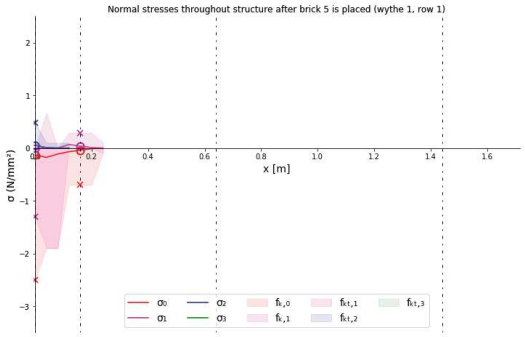


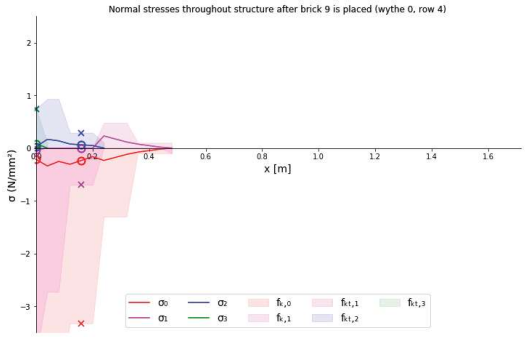

Normal stresses throughout structure after brick 3 is placed (wythe 1, row 0)
x = 0.0
x = 0.16
x = 0.64
x = 1.44
x [m]
σ (N/mm²)

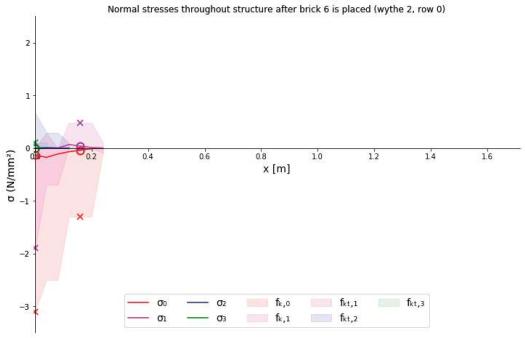

Normal stresses throughout structure after brick 7 is placed (wythe 0, row 3)
x [m]
σ (N/mm²)

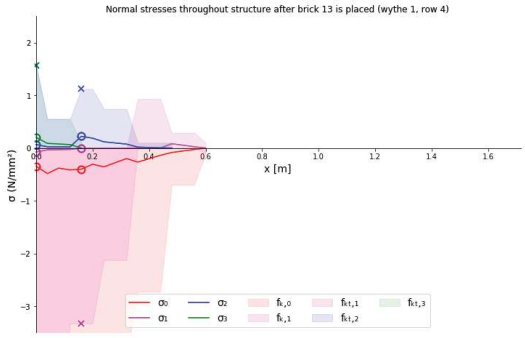

Normal stresses throughout structure after brick 4 is placed (wythe 0, row 2)
x [m]
σ (N/mm²)

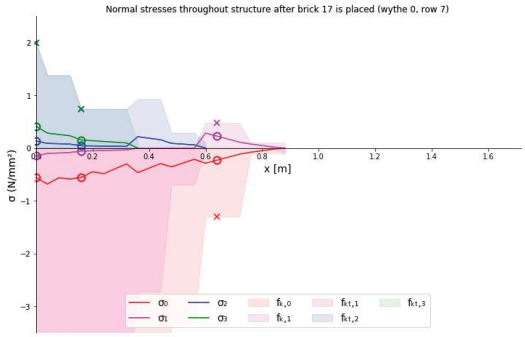

Normal stresses throughout structure after brick 8 is placed (wythe 1, row 2)
x [m]
σ (N/mm²)

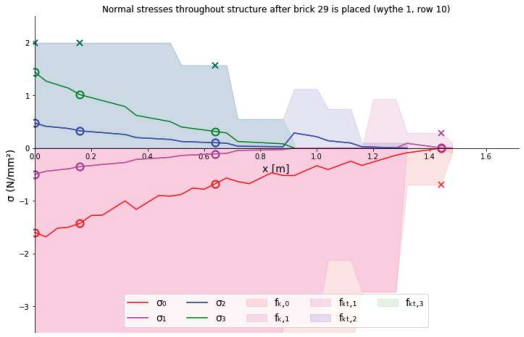

Normal stresses throughout structure after brick 5 is placed (wythe 1, row 1)
x [m]
σ (N/mm²)

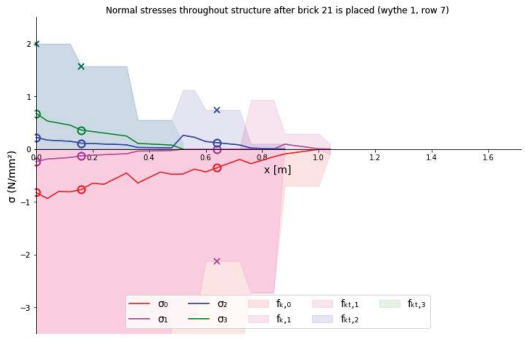

Normal stresses throughout structure after brick 9 is placed (wythe 0, row 4)
x [m]
σ (N/mm²)

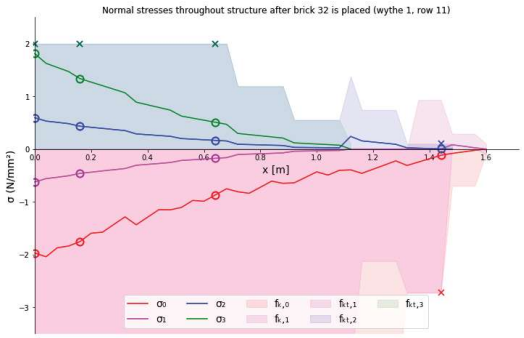

Normal stresses throughout structure after brick 6 is placed (wythe 2, row 0)
x [m]
σ (N/mm²)

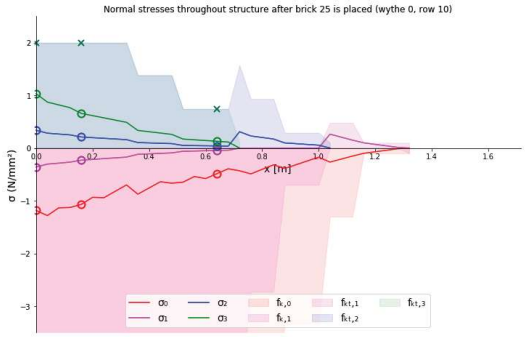

Normal stresses throughout structure after brick 13 is placed (wythe 1, row 4)
x [m]
σ (N/mm²)

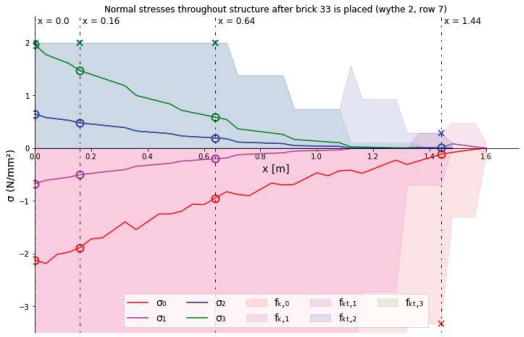










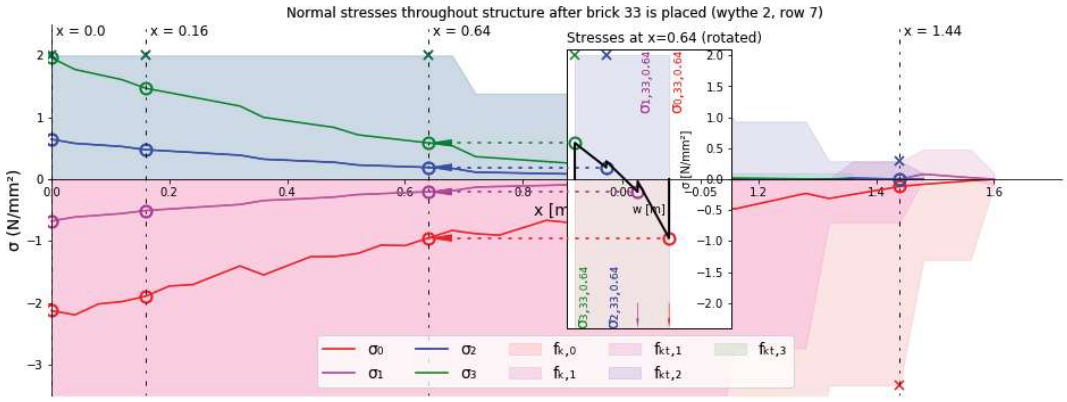



```
In [14]: #
         from matplotlib.transforms import Affine2D
         import mpl_toolkits.axisartist.floating_axes as floating_axes
         posi = np.array(runn)[:4,1]
         fig  = plt.figure(figsize=(16,6))
         ax1 = fig.add_subplot(1,1,1)

         if True:
             i = manu[-1]
             n0,n1,n2 = [sum([1 if ele==wth else 0 for idx,ele in enumerate(ll[0:i+1])])-1
                         for wth in [0,1,2]]
             id0,id1 = i0.index(n0+1) if n0>-1 else -1, i1.index(n1+1) if n1>-1 else -1
             id2 = i2.index(n2+1) if n2>-1 else -1
             ax1.spines['left'].set_position('zero')
             ax1.spines['right'].set_color('none')
             ax1.yaxis.tick_left()

             ax1.spines['bottom'].set_position('zero')

             ax1.spines['top'].set_color('none')
             ax1.xaxis.tick_bottom()

             xx = x
```

```
    ax1.plot(xx[:id0+1],Sigres[i][0][:id0+1], label = '\u03C3%s' % (sub('0')),
             color='red')
    ax1.plot(xx[:id0+1],Sigres[i][1][:id0+1], label = '\u03C3%s' % (sub('1')),
             color='magenta')
    ax1.plot(xx[:id1+1],Sigres[i][2][:id1+1], label = '\u03C3%s' % (sub('2')),
             color='blue')
    ax1.plot(xx[:id2+1],Sigres[i][3][:id2+1], label = '\u03C3%s' % (sub('3')),
             color='green')

    alp = .1
    ax1.fill_between(xx[:id0+1],C0res[i][0][:id0+1], color='red',alpha=alp,
                     label = 'f%s' % (sub('k,0')))
    ax1.fill_between(xx[:id0+1],C0res[i][1][:id0+1], color='magenta',alpha=alp,
                     label = 'f%s' % (sub('k,1')))
    ax1.fill_between(xx[:id0+1],T0res[i][1][:id0+1], color='magenta',alpha=alp,
                     label = 'f%s' % (sub('kt,1')))
    ax1.fill_between(xx[:id1+1],Tstrres[i][2][:id1+1], color='blue',alpha=alp,
                     label = 'f%s' % (sub('kt,2')))
    ax1.fill_between(xx[:id2+1],Tstrres[i][3][:id2+1], color='green',alpha=alp,
                     label = 'f%s' % (sub('kt,3')))

    ax1.set_xlabel('x [m]', fontsize=14)
    ax1.set_ylabel('\u03C3 (N/mmš)', fontsize=14)
    ax1.title.set_text('Normal stresses throughout structure after brick %s is placed (wythe %s, row %s)
    ' % (i+1,ll[i],nn[i]))

    ax1.legend(loc=8, fontsize=14,ncol=5)
    ax1.set_ylim([-3.5,2.5]), ax1.set_xlim([-0.005,1.715])

    hd_wd = .008*sum(np.abs(ax1.get_xlim()))
    hd_ln = 8*hd_wd
    for ar in range(len(posi)):
        if np.sign(Sigres[i][0][posi[ar]]) != 0:
            ax1.plot(xx[posi[ar]], Sigres[i][0][posi[ar]], 'o', ms=10, mfc='none',
                     mew=2,  mec='r')
            ax1.plot(xx[posi[ar]], C0res[i][0][posi[ar]], 'x', ms=8, mfc='none',
                     mew=1.5,  mec='r')

        if np.sign(Sigres[i][1][posi[ar]]) != 0:
            ax1.plot(xx[posi[ar]],Sigres[i][1][posi[ar]], 'o', ms=10, mfc='none',
                     mew=2,  mec='m')
            if Sigres[i][1][posi[ar]]<0:
                      ax1.plot(xx[posi[ar]], C0res[i][1][posi[ar]], 'x', ms=8,
                               mfc='none', mew=1.5,  mec='m')
            if Sigres[i][1][posi[ar]]>=0:
                      ax1.plot(xx[posi[ar]], T0res[i][1][posi[ar]], 'x', ms=8,
                               mfc='none', mew=1.5,  mec='m')

        if np.sign(Sigres[i][2][posi[ar]]) != 0:
            ax1.plot(xx[posi[ar]],Sigres[i][2][posi[ar]], 'o', ms=10, mfc='none',
                     mew=2,  mec='b')
            ax1.plot(xx[posi[ar]], Tstrres[i][2][posi[ar]], 'x', ms=8, mfc='none',
                     mew=1.5,  mec='b')

        if np.sign(Sigres[i][3][posi[ar]]) != 0:
            ax1.plot(xx[posi[ar]],Sigres[i][3][posi[ar]], 'o', ms=10, mfc='none',
                     mew=2,  mec='g')
            ax1.plot(xx[posi[ar]], Tstrres[i][3][posi[ar]], 'x', ms=8, mfc='none',
                     mew=1.5,  mec='g')

        if i<manu[5] or i==manu[-1]:
                      ax1.plot([xx[posi[ar]],xx[posi[ar]]],ax1.get_ylim(),
                               ls=(0,(2,7,4,7)),c='k',lw=1)
        if i<manu[3] or i==manu[-1]:
                      ax1.annotate(" x = %s" % (xx[posi[ar]]),
                                   xy=(xx[posi[ar]],ax1.get_ylim()[1]),
                                   color='k', fontsize=12, va='top')
xstart = 6
subwidth = xstart + 7

tr_rot = Affine2D().scale(.1,100).rotate_deg(90)

ax2 = fig.add_axes([xx[posi[2]+xstart] / sum(np.abs(ax1.get_xlim())) *.775+.125,
       (-2.4-ax1.get_ylim()[0]) / sum(np.abs(ax1.get_ylim())) *.755+.125,
       (xx[posi[2]+subwidth]-xx[posi[2]+xstart]) / sum(np.abs(ax1.get_xlim())) *.775,
       (2.1--2.4) / sum(np.abs(ax1.get_ylim())) *.755 ])
if True:
    plc = len(runn)-2
    posi = runn[plc][1]
    plcd = runn[plc][0]
    lbl = runn[plc][2]
    hh = [[-1.5*t,-1.5*t,-.5*t,-.5*t,.5*t,.5*t,1.5*t,1.5*t]
          for ele in range(len(Sigani[0]))]
```

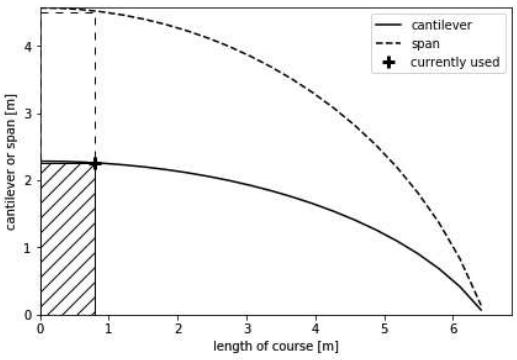
```
    hh = [-1.5*t,-1.5*t,-.5*t,-.5*t,.5*t,.5*t,1.5*t,1.5*t]

    Cfan = np.array(Cstrres)[:,:,posi]
    Tfan = np.array(Tstrres)[:,:,posi]
    Sigan = np.array(Sigres)[:,:,posi]

    length = [sum([1 if ele==wth else 0 for idx,ele in enumerate(ll[0:plcd+1])])-1
              for wth in [0,1,2]]
    start = [y0[length[0]]<xx[posi], y1[length[1]]<xx[posi], y2[length[2]]<xx[posi]]
    start = [True if length[idx]==-1 else ele for idx,ele in enumerate(start)]
    start.append(start[-1])

    Inter = np.array([[0 if start[idx] else ele
                       for idx,ele in enumerate(inter[:,posi+1])]
                      for lst,Lst in enumerate(range(len(seq)))])

    Sigani = np.array([np.zeros(len(seq)),#0
                       Sigan[:,0],#1
                       Sigan[:,1]+Inter[:,1],#2
                       Sigan[:,1],#3
                       Sigan[:,2]+Inter[:,2],#4
                       Sigan[:,2],#5
                       Sigan[:,3],#6
                       np.zeros(len(seq))])#7
    Cfani = np.array([Cfan[:,0],Cfan[:,0],Cfan[:,1],Cfan[:,1],Cfan[:,2],Cfan[:,3]])
    Tfani = np.array([Tfan[:,0],Tfan[:,0],Tfan[:,1],Tfan[:,1],Tfan[:,2],Tfan[:,3]])

    Sigani[5] = [0 if Inter[:,2][idx]==0 else ele for idx,ele
                 in enumerate(Sigani[5])]
    Sigani[3] = [0 if Inter[:,1][idx]==0 else ele for idx,ele
                 in enumerate(Sigani[3])]

    for i in range(len(Sigani)-2): # i is per interface
        i +=2
        for j in range(len(Sigani[i])): # j is per plcd
            if i>1 and i<7 and Sigani[i][j]==0:
                Cfani[i-1][j] = np.nan
                Tfani[i-1][j] = np.nan
            if Sigani[i-1][j]==0:
                Sigani[i][j] = np.nan

    ax2.set_ylim(-2.4,2.1)
    ax2.set_xlim(1.75*t,-1.75*t-.5*(3.5*t))
    ax2.spines['bottom'].set_position('zero')
#     ax2.spines['top'].set_color('none')
    ax2.yaxis.tick_left()
    ax2.spines['left'].set_position(('axes',1))
#     ax2.spines['right'].set_color('none')
    ax2.patch.set_edgecolor('k'),ax2.patch.set_linewidth('1')
    ax2.xaxis.tick_bottom()
    ax2.set_ylabel('\u03C3 [N/mmš]', fontsize=10,ha='left')
    ax2.set_xlabel('w [m]', fontsize=10)
    ax2.set_title('Stresses at x=%s (rotated)' % (x[posi]),loc='left')

    if lbl==1:
        lyrs = np.count_nonzero([ele for ele in Sigani[:,plcd]
                                 if (not np.isnan(ele))])
        if lyrs==2:
            sig = [1,2,4,6]
        elif lyrs==4:
            sig = [1,3,4,6]
        else:
            sig = [1,3,5,6]

        clr = ['r','m','b','g']
        sig = [ele for ele in sig if (Sigani[:,plcd][ele]!=0)*(
            not np.isnan(Sigani[:,plcd][ele]))]
        for i in range(len(sig)):
            text= "\u03C3%s" % (sub('%s,%s,%s' % (i,plcd+1,xx[posi]))) #text
            labpo = [-np.sign(Sigani[sig[i],plcd])/2+0.5,
                     -(hh[sig[i]]-ax2.get_xlim()[0])/sum(np.abs(ax2.get_xlim()))]
            ax2.annotate(text, xy=(labpo[1],labpo[0]),
                         xycoords='axes fraction', color=clr[i], fontsize=14,
                         va=['bottom','top'][int(labpo[0])],
                         rotation=90)
            ax2.plot(hh[sig[i]],Sigani[sig[i],plcd], 'o', ms=10, mfc='none',
                     mew=2,  mec=clr[i])
            if plc>=3*4:
                if Sigani[sig[i],plcd]<0:
                       ax2.plot(hh[sig[i]],Cfani[sig[i]-1,plcd], 'x', ms=8,
                                mfc='none', mew=1.5,  mec=clr[i])
                if Sigani[sig[i],plcd]<0 and Cfani[sig[i]-1,plcd]<-2.4:
                       ax2.arrow(hh[sig[i]], -2.4+.4, 0, -.4,
                                 length_includes_head=True, shape='full',
```

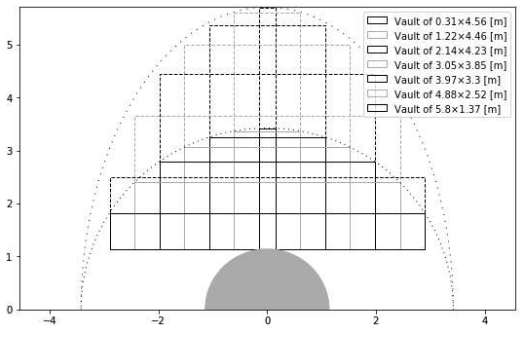
```
                                    width=.0004, head_width = .0015,
                                    head_length = .15, ec = 'none', fc = clr[i])
                        if Sigani[sig[i],plcd]>=0:
                               ax2.plot(hh[sig[i]],Tfani[sig[i]-1,plcd], 'x', ms=8,
                                        mfc='none', mew=1.5,  mec=clr[i])
                    ax2.plot([ax2.get_xlim()[0],hh[sig[i]]],
                             [Sigani[sig[i],plcd],Sigani[sig[i],plcd]],
                             c=clr[i],ls=(1,(2,4)))
                    ax1.plot([xx[posi],xx[posi+xstart]],
                             [Sigani[sig[i],plcd],Sigani[sig[i],plcd]],
                             c=clr[i],ls=(1,(2,4)))
                    ax1.arrow(xx[posi+1],
                              Sigani[sig[i],plcd],
                               -xx[posi+1]+xx[posi],
                               0,
                               length_includes_head=True,
                               shape='full',
                               head_width = .12,
                               head_length = xx[posi+1]-xx[posi],
                               ec = 'none',
                               fc = clr[i])
                ax2.plot(hh,Sigani[:,plcd],color='k',lw=2)
                x_fill = np.zeros(len(hh))

                ax2.fill_between(hh[1:7],Cfani[:,plcd],color='maroon',alpha=.1)
                ax2.fill_between(hh[1:7],Tfani[:,plcd],color='navy',alpha=.1)

        plt.show()
```




```
In [15]: def annot_max1(x,y, ax=None):
             xmax = x[np.argmax(y)]
             ymax = y.max()
             text= "bricks placed={:}, uc={:.2f}".format(int(xmax), ymax)
             if not ax:
                 ax=plt.gca()
             bbox_props = dict(boxstyle="square,pad=0.3", fc="w", ec="k", lw=0.72)
             arrowprops=dict(arrowstyle="->",connectionstyle="angle,angleA=0,angleB=60")
             kw = dict(xycoords='data',textcoords="axes fraction",
                       arrowprops=arrowprops, bbox=bbox_props, ha="right", va="top")
             ax.annotate(text, xy=(xmax, ymax), xytext=(0.82,0.96), **kw)

In [16]: # Maximum dimensions of vault
         %matplotlib inline
         %matplotlib inline
         from matplotlib.patches import Rectangle
         from matplotlib.patches import Circle
         from matplotlib.patches import Ellipse

         Ro = 3123 #mm
         Ri = 1092.7
         Ro = 3425
         Ri = 1140
         Ro,Ri = Ro/1000,Ri/1000
         # a2+b2=c2

         plt.rcParams["figure.figsize"] = (2*(Ro),2*(Ro-Ri)) # (w, h)
         fig, ax = plt.subplots()

         course = np.linspace(0,Ro+1,30)
         course = [ele for ele in course if abs(ele)<=Ro]
```

```
cant = [np.sqrt(Ro**2 - ele**2)-Ri for ele in course]
course = [ele for idx, ele in enumerate(course) if cant[idx]>=0]
cant = [ele for ele in cant if ele>=0]
span = [2*ele for ele in cant]
cours = [2*ele for ele in course]

plt.plot(cours,cant,color='k',label='cantilever')
plt.plot(cours,span,linestyle='dashed',color='k',label='span')
width = .8
height= 4.5/2
plt.plot(width,height,'+',color='k',ms=10,label='currently used',mew=3)
ax.add_patch(Rectangle((0,0),width,height,fill=None,hatch='//'))
ax.add_patch(Rectangle((0,height),width,height,fill=None, linestyle=(0,(6,6))))

plt.ylim([0,2*(Ro-Ri)]), plt.xlim([0,2*(Ro)])

plt.xlabel('length of course [m]')
plt.ylabel('cantilever or span [m]')
plt.legend(prop={"size":10})
plt.show()

    #Other graph
x_0 = [-ele for idx,ele in enumerate(course) if idx % 3 ==1]
x_1 = [ele for idx,ele in enumerate(course) if idx % 3 ==1]
y_0 = [Ri for idx,ele in enumerate(cant) if idx % 3 ==1]
y_1 = [ele+Ri for idx,ele in enumerate(cant) if idx % 3 ==1]
y_2 = [ele+Ri for idx,ele in enumerate(span) if idx % 3 ==1]
d_x = [x_1[idx]-x_0[idx] for idx,ele in enumerate(x_0)]
d_y = [y_1[idx]-y_0[idx] for idx,ele in enumerate(x_0)]

plt.rcParams["figure.figsize"] = (2*(Ro+Ri),2*Ro-Ri) # (w, h)
fig, ax = plt.subplots()
lne = ['dashed','solid']

for i in range(len(x_0)):
    g = float((i % 2)/3*2)
    ax.add_patch(Rectangle((x_0[i], y_1[i]), (x_1[i]-x_0[i]), (y_1[i]-y_0[i]), fill=None,color=str(g),
    linestyle = 'dashed'))
    ax.add_patch(Rectangle((x_0[i], y_0[i]), (x_1[i]-x_0[i]), (y_1[i]-y_0[i]), fill=None,color=str(g),
                           label = 'Vault of %s%s [m]' % (np.round(d_x[i],2),np.round(2*d_y[i],2))))

ax.add_patch(Circle((0,0), Ro,fill=None,linestyle=(0,(1,4))))
ax.add_patch(Ellipse((0,0), width=2*Ro,height=(4*Ro-2*Ri),fill=None,linestyle=(0,(1,6))))
ax.add_patch(Circle((0,0), Ri,color='darkgray'))

plt.ylim([0,2*Ro-Ri]), plt.xlim([-Ro-Ri,Ro+Ri])
plt.legend(prop={"size":10})
plt.show()
```

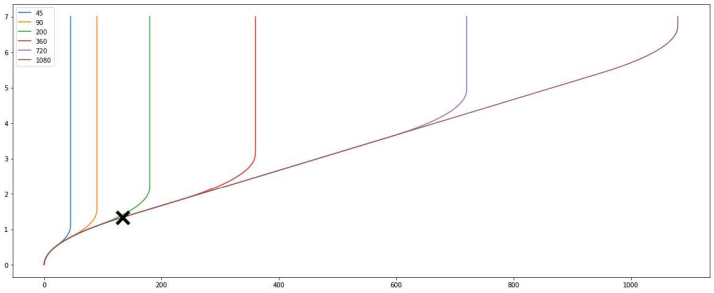




In [17]:
```
from scipy.misc import derivative as diff
H1 = (8/3)
xq = np.linspace(0,7,101)
def func(xq):
    afcd = (
        (H1>(2*4/3))*(
            (xq<(4/3))*(xq>=0) *
            (.5*150*xq**2)
            + (xq>=(4/3))*(xq<=(H1-4/3))  *
            (200*(xq-4/3)+0.5*150*(4/3)**2)
            + (xq>(H1-4/3))*(xq<=H1) *
            (150*H1*xq -.5*150*xq**2 - .5*150*H1**2 +200*H1 - 800/3)
            + (xq>H1) *
```

```
                    (200*(H1-4/3))
                )
                + (H1<=(2*4/3))*(
                    (xq<=(H1/2))*(xq>=0) *
                    (.5*150*xq**2)
                    + (xq>(H1/2))*(xq<=H1) *
                    (150*H1*xq-.5*150*xq**2-150/4*H1**2)
                    + (xq>H1) *
                    (150/4*H1**2)
                )
            )
            return afcd

        # def dif(xq):
        #     if H1>(2*4/3):
        #         afcd2 = (xq<(4/3)) * (150*xq) + (xq>=(4/3))*(xq<=(H1-4/3)) * (200) + (xq>(H1-4/3))*(xq<=H1) *
        #                 (200 - 150*(xq-H1+4/3)) + (xq>H1) * 0
        #     return afcd2
        # print(func(xq))
        print('4/3',func(4/3))
        # print('d4/3',diff(func,4/3))
        T = [(np.sqrt(30)/5),(2*np.sqrt(15)/5),(2*np.sqrt(30)/5),(47/15),(74/15),(101/15)]
        Deg = [45,90,200,360,720,1080]
        plt.figure(figsize=(20,8))
        for i in range(len(T)):
            H1 = T[i]
            print('H1',func(H1),H1)
            plt.plot(func(xq),xq,label=Deg[i]);
        plt.plot((400/3),(4/3),"x",ms=20,mec='k',mew=5)
        plt.legend();
        # plt.plot(dif(xq),xq);
```

```
4/3 133.33333333333331
H1 45.00000000000002 1.0954451150103321
H1 89.99999999999996 1.5491933384829668
H1 180.00000000000009 2.1908902300206643
H1 359.99999999999994 3.1333333333333333
H1 720.0 4.933333333333334
H1 1080.0 6.733333333333333
```

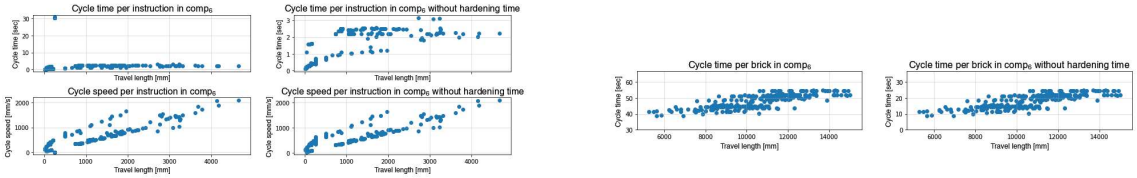


In [18]:
```
        travel = [15696,10667,2534,168,1451,16061,41464,18238,1451,1415,11049,252,1451,1666,1451...
        # travel[0] = 46624
        times = [1414,2191,30425,123,408,2261,2014,1223,417,351,2191,176,422,1616,434,2191,30455,428,411,2495,2004,11...
        speed = [1110,487,8,137,355,710,2058,1491,348,403,504,143,344,103,335,479,8,491,353,671,1841,1649,8,300,1463,...
        times = [ele/1000 for ele in times]
        travel = [ele/10 for ele in travel]
        print(len(travel),len(times),len(speed))

        comper = "6"

        ### Per Instruction
        plt.rcParams["figure.figsize"] = (15,4.5) # (w, h)
        params = {'mathtext.default': 'regular' }
        plt.rcParams.update(params)
        fig,((ax0,ax1),(ax2,ax3)) = plt.subplots(2,2,sharex=True)
        # fig = plt.figure()
        # ax0 = fig.add_subplot(2,2,1)
        # ax1 = fig.add_subplot(2,2,2,sharex=ax0)
        # ax2 = fig.add_subplot(2,2,3,sharex=ax0)
        # ax3 = fig.add_subplot(2,2,1)

        ax0.set_title("Cycle time per instruction in $comp_%s$" % (comper),fontsize=16,fontname="Helvetica")
        ax0.set_xlabel("Travel length [mm]",fontsize=12,fontname="Helvetica")
        ax0.set_ylabel("Cycle time [sec]",fontsize=12,fontname="Helvetica")
```

```
ax0.grid(which='major',lw=.5,alpha=0.75)
ax0.scatter(travel,times)
# ax0.set_xlim(xlims)
# xlims = ax0.set_xlim()

timesmin = [ele-30 if ele>30 else ele for ele in times]

ax1.set_title("Cycle time per instruction in $comp_%s$ without hardening time" % (comper),fontsize=16,
fontname="Helvetica")
ax1.set_xlabel("Travel length [mm]",fontsize=12,fontname="Helvetica")
ax1.set_ylabel("Cycle time [sec]",fontsize=12,fontname="Helvetica")
ax1.grid(which='major',lw=.5,alpha=0.75)
ax1.scatter(travel,timesmin)

ax2.set_title("Cycle speed per instruction in $comp_%s$" % (comper),fontsize=16,fontname="Helvetica")
ax2.set_xlabel("Travel length [mm]",fontsize=12,fontname="Helvetica")
ax2.set_ylabel("Cycle speed [mm/s]",fontsize=12,fontname="Helvetica")
ax2.grid(which='major',lw=.5,alpha=0.75)
ax2.scatter(travel,speed)

speedmin = [travel[ind]/timesmin[ind] for ind,ele in enumerate(travel)]

ax3.set_title("Cycle speed per instruction in $comp_%s$ without hardening time" % (comper),fontsize=16,
fontname="Helvetica")
ax3.set_xlabel("Travel length [mm]",fontsize=12,fontname="Helvetica")
ax3.set_ylabel("Cycle speed [mm/s]",fontsize=12,fontname="Helvetica")
ax3.grid(which='major',lw=.5,alpha=0.75)
ax3.scatter(travel,speedmin)
# sns.regplot(np.array(travel), np.array(speedmin),ci=99.9,ax=ax3)
# b3,a3 = np.polynomial.polynomial.polyfit(travel,speedmin,deg=1)
# b30,b31 = b3+300, b3-300
# upper,downer = [a3*ele+b30 for ele in travel], [a3*ele+b31 for ele in travel]
# upper,downer = sum([speedmin[ind]<upper[ind] for ind,ele in enumerate(upper)]),sum([speedmin[ind]>downer[ind]
#         for ind,ele in enumerate(downer)])
# print(upper/len(travel),downer/len(travel))
# ax3.plot(travel, [a3*ele+b30 for ele in travel], ls='dotted', c='#1f77b4',alpha=0.5)
# ax3.plot(travel, [a3*ele+b31 for ele in travel], ls='dotted', c='#1f77b4',alpha=0.5)
plt.subplots_adjust(hspace=0.45)

#### Per brick
# travel0 = [57764,46934,50007,53566,57722,62457,67763,73616,67952,58728,53866,57012,61007,65541,69943,75678,7...
# times0 = [41464,42016,41820,41676,41237,41058,40624,40580,40924,41282,41357,41466,41753,42197,41661,42614,43...
# bri_ins0 = [10,14,14,14,14,14,14,14,14,14,14,14,14,14,14,14,14,10,18,24,24,24,24,24,24,24,24,24,24,14,14,...
# travel1 = [49619,55060,56783,58894,60672,63523,66880,70619,65125,54613,60847,62892,65320,68098,70926,74409,7...
# times1 = [41518,41798,41738,41627,41269,41078,40574,40468,40764,41103,41327,41436,41682,42068,41663,42570,43...
# bri_ins1 = [10,14,14,14,14,14,14,14,14,14,14,14,14,14,14,14,14,10,18,24,24,24,24,24,24,24,24,24,24,14,14,...
# travel2 = [63831,46128,48767,57125,57624,56944,61778,73027,67610,59040,61402,65765,74517,69024,69950,75236,7...
# times2 = [38311,41047,41050,42782,42726,40463,40484,39937,40172,41376,41448,41527,43287,43079,40835,40847,41...
# bri_ins2 = [10,14,14,14,14,14,14,14,14,14,14,14,14,14,14,14,14,10,18,24,24,24,24,24,24,24,24,24,24,14,14,...
# travel3 = [56839,46613,49330,53002,57231,61936,67074,72606,67296,59995,54976,57935,61646,65899,70641,75864,7...
# times3 = [39926,40849,40734,40590,40388,40143,39844,39603,40011,43540,43538,43477,43348,43241,42876,42265,42...
# travel4 = [61686,53530,56147,59740,63913,68582,73702,79232,74154,64961,59545,62460,66142,70382,75123,80356,7...
# times4 = [310782,312422,312318,312184,311990,311752,311460,311224,311641,312881,312852,312785,312651,312540,...
# travel5 = [70294,57708,60291,63802,67954,72704,71767,70305,66683,69576,73154,71924,78787,73961,82426,80715,8...
# times5 = [40077,41184,41074,40974,40868,40663,43463,43625,44122,43979,43842,38581,43330,46239,47497,46004,47...
travel6 = [75098,100549,101178,102773,104949,107657,110855,114504,108526,106988,97732,98774,100932,103650,106...
times6 = [41062,43799,43772,43759,43754,43755,43763,43775,43742,44686,44664,44658,44664,44679,44700,44743,447...
# travel7 = [58029,47434,49753,53464,57756,62556,67827,73524,67778,59267,54663,57543,61249,65415,70187,75490,7...
# times7 = [40663,41980,40941,40819,40639,40423,40134,39917,40349,42833,42724,42672,42560,41843,41521,40863,41...
times = [ele/1000 for ele in times6]
travel = [ele/10 for ele in travel6]
# bri_ins = bri_ins1
speed = [travel[ind]/times[ind] for ind,ele in enumerate(travel)]

# fig,((ax0,ax1),(ax2,ax3)) = plt.subplots(2,2,sharex="row")
plt.rcParams["figure.figsize"] = (15,1.8) # (w, h)
fig,(ax0,ax1) = plt.subplots(1,2,sharex="row")

ax0.set_title("Cycle time per brick in $comp_%s$" % (comper),fontsize=16,fontname="Helvetica")
ax0.set_xlabel("Travel length [mm]",fontsize=12,fontname="Helvetica")
ax0.set_ylabel("Cycle time [sec]",fontsize=12,fontname="Helvetica")
ax0.grid(which='major',lw=.5,alpha=0.75)
ax0.scatter(travel,times)
ax0.set_ylim(30,max(times)+12)

timesmin = [ele-30 if ele>30 else ele for ele in times]
# colo10 = ['#79a3c7' for ind,ele in enumerate(bri_ins) if bri_ins[ind]==10]
# colo12 = ['#6394bd' for ind,ele in enumerate(bri_ins) if bri_ins[ind]==12]
# colo14 = ['#4d85b4' for ind,ele in enumerate(bri_ins) if bri_ins[ind]==14]
# colo16 = ['#3776ab' for ind,ele in enumerate(bri_ins) if bri_ins[ind]==16]
# colo18 = ['#306898' for ind,ele in enumerate(bri_ins) if bri_ins[ind]==18]
# colo20 = ['#2a5b85' for ind,ele in enumerate(bri_ins) if bri_ins[ind]==20]
# colo22 = ['#244e72' for ind,ele in enumerate(bri_ins) if bri_ins[ind]==22]
# colo24 = ['#1e415f' for ind,ele in enumerate(bri_ins) if bri_ins[ind]==24]
```

```
# colo26 = ['#18344c' for ind,ele in enumerate(bri_ins) if bri_ins[ind]==26]
# colo28 = ['#122739' for ind,ele in enumerate(bri_ins) if bri_ins[ind]==28]
# time10 = [ele for ind,ele in enumerate(timesmin) if bri_ins[ind]==10]
# time12 = [ele for ind,ele in enumerate(timesmin) if bri_ins[ind]==12]
# time14 = [ele for ind,ele in enumerate(timesmin) if bri_ins[ind]==14]
# time16 = [ele for ind,ele in enumerate(timesmin) if bri_ins[ind]==16]
# time18 = [ele for ind,ele in enumerate(timesmin) if bri_ins[ind]==18]
# time20 = [ele for ind,ele in enumerate(timesmin) if bri_ins[ind]==20]
# time22 = [ele for ind,ele in enumerate(timesmin) if bri_ins[ind]==22]
# time24 = [ele for ind,ele in enumerate(timesmin) if bri_ins[ind]==24]
# time26 = [ele for ind,ele in enumerate(timesmin) if bri_ins[ind]==26]
# time28 = [ele for ind,ele in enumerate(timesmin) if bri_ins[ind]==28]
# dist10 = [ele for ind,ele in enumerate(travel) if bri_ins[ind]==10]
# dist12 = [ele for ind,ele in enumerate(travel) if bri_ins[ind]==12]
# dist14 = [ele for ind,ele in enumerate(travel) if bri_ins[ind]==14]
# dist16 = [ele for ind,ele in enumerate(travel) if bri_ins[ind]==16]
# dist18 = [ele for ind,ele in enumerate(travel) if bri_ins[ind]==18]
# dist20 = [ele for ind,ele in enumerate(travel) if bri_ins[ind]==20]
# dist22 = [ele for ind,ele in enumerate(travel) if bri_ins[ind]==22]
# dist24 = [ele for ind,ele in enumerate(travel) if bri_ins[ind]==24]
# dist26 = [ele for ind,ele in enumerate(travel) if bri_ins[ind]==26]
# dist28 = [ele for ind,ele in enumerate(travel) if bri_ins[ind]==28]

ax1.set_title("Cycle time per brick in $comp_%s$ without hardening time" % (comper),fontsize=16,
fontname="Helvetica")
ax1.set_xlabel("Travel length [mm]",fontsize=12,fontname="Helvetica")
ax1.set_ylabel("Cycle time [sec]",fontsize=12,fontname="Helvetica")
ax1.grid(which='major',lw=.5,alpha=0.75)
# ax1.scatter(dist10,time10, color=colo10,label='10')
# # ax1.scatter(dist12,time12, color=colo12,label='12')
# ax1.scatter(dist14,time14, color=colo14,label='14')
# ax1.scatter(dist16,time16, color=colo16,label='16')
# ax1.scatter(dist18,time18, color=colo18,label='18')
# ax1.scatter(dist20,time20, color=colo20,label='20')
# ax1.scatter(dist22,time22, color=colo22,label='22')
# ax1.scatter(dist24,time24, color=colo24,label='24')
# # ax1.scatter(dist26,time26, color=colo26,label='26')
# ax1.scatter(dist28,time28, color=colo28,label='28')
ax1.scatter(travel,timesmin)
# ax1.legend(ncol=4, framealpha=0.4, handletextpad=0.2, columnspacing=1.2)
# ax1.legend(['8 instructions','12 instructions','14 instructions','16 instructions','18 instructions',
#         '20 instructions','22 instructions','24 instructions','26 instructions','28 instructions'])
ax1.set_ylim(0,max(timesmin)+12)

# ## Per Station
# stat = ['Pallet','Adhesive','Vault']
# stattim = [ 1220,5626,10470]
# statdis = [1390628,1061459,1400550]

# stattimmin = [stattim[0],stattim[1],stattim[2]-30*len(timesmin)]
# stattimplu = [ele - stattimmin[ind] for ind,ele in enumerate(stattim)]

# ax2.set_title("Total time for all instructions per station in $comp_%s$" % (comper),fontsize=16,
#         fontname="Helvetica")
# ax2.set_xlabel("Station",fontsize=12,fontname="Helvetica")
# ax2.set_ylabel("Time [sec]",fontsize=12,fontname="Helvetica")
# ax2.bar(stat,stattimmin,width=0.6,label="Without hardening time")
# ax2.bar(stat,stattimplu,width=0.6,bottom=stattimmin, color="cornflowerblue",label="Hardening time")
# ax2.legend()
# ax3.set_title("Total distance for all instructions per station in $comp_%s$" % (comper),fontsize=16,
#         fontname="Helvetica")
# ax3.set_xlabel("Station",fontsize=12,fontname="Helvetica")
# ax3.set_ylabel("Distance [mm]",fontsize=12,fontname="Helvetica")
# ax3.bar(stat,statdis,width=0.6)

plt.subplots_adjust(hspace=0.5)
plt.show();
```

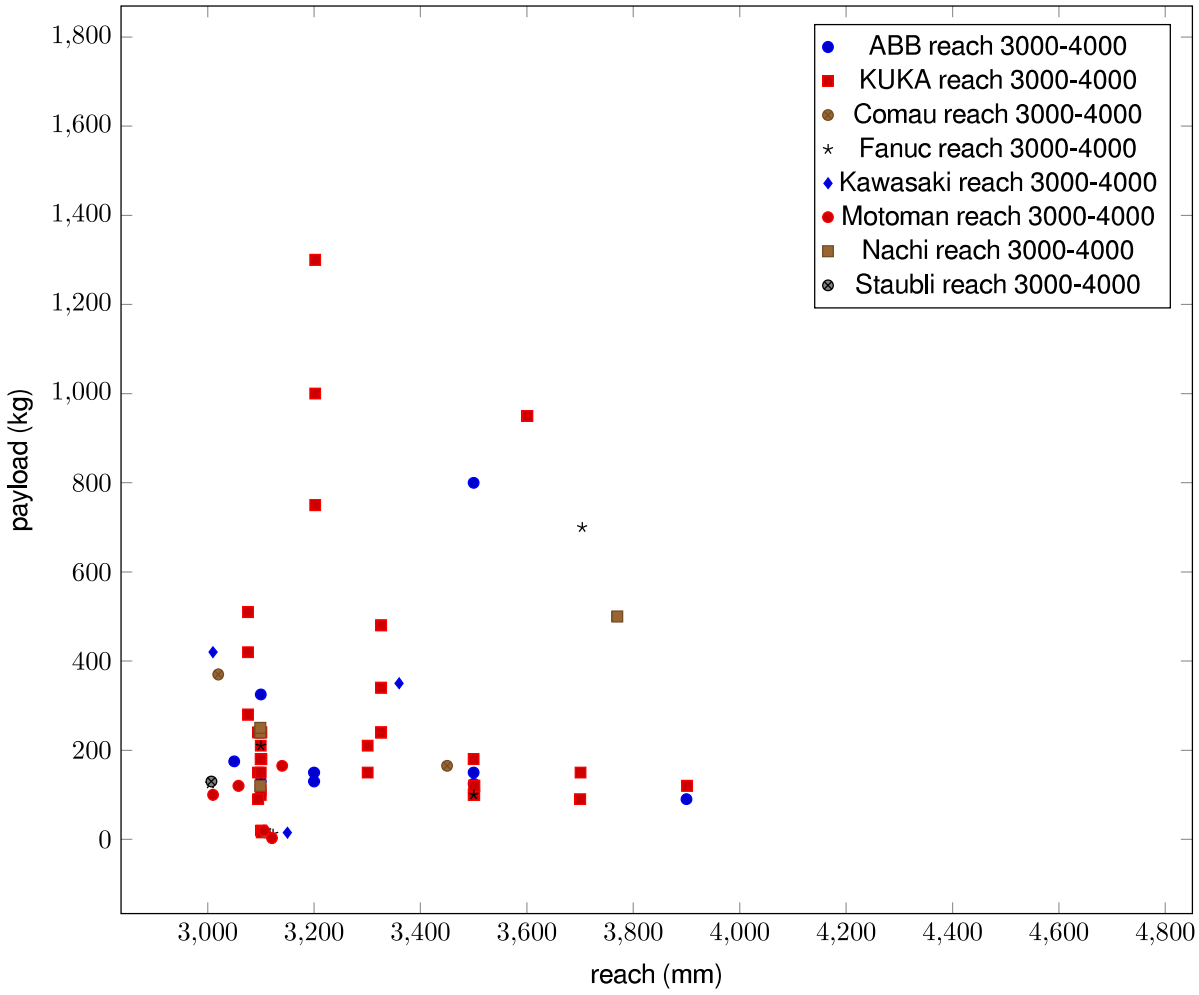
```
548 548 548
```

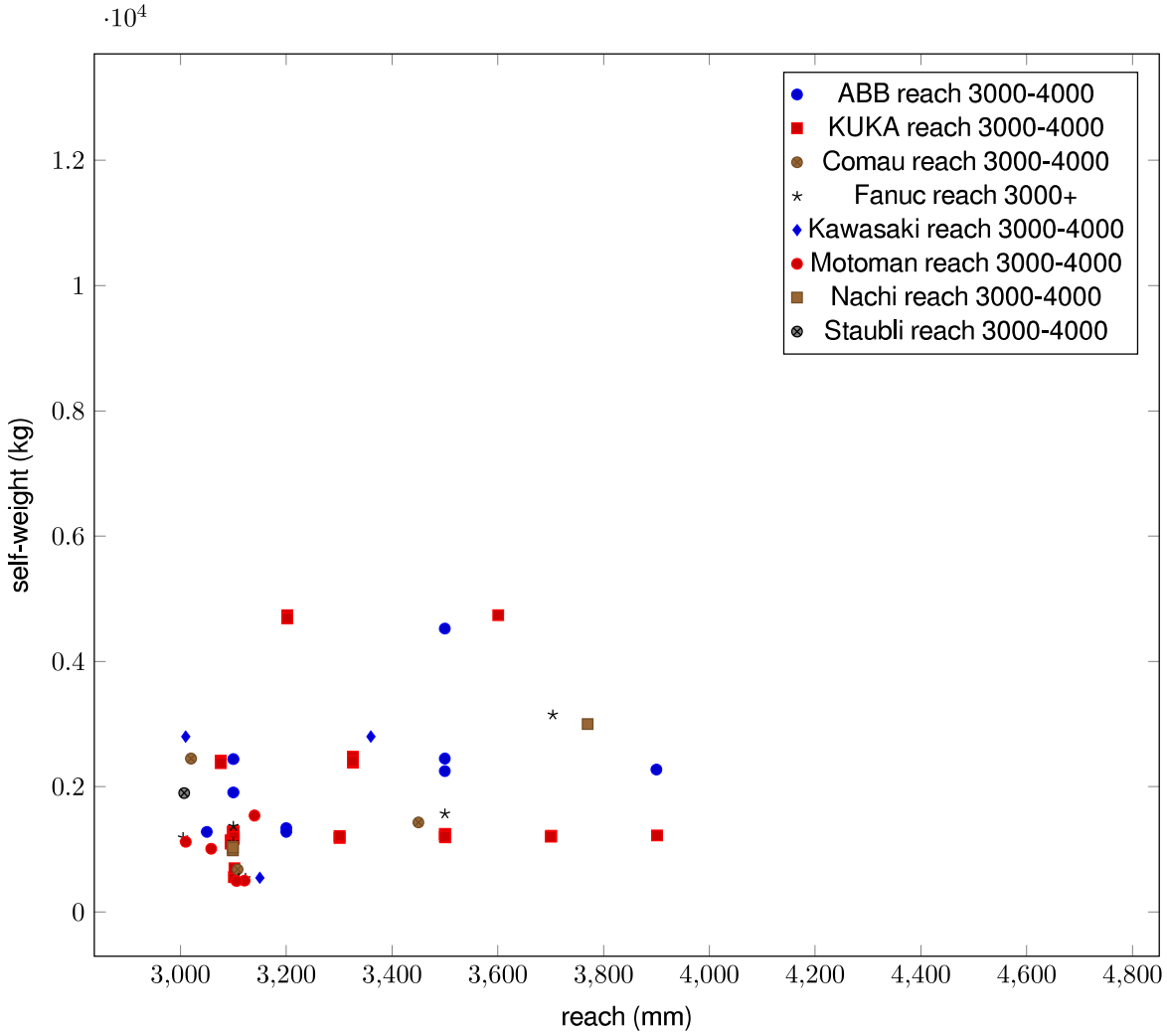


E

Robotic arms from the RoboDK library

This chapter provides a broader view of the robots that could have been selected and the overall programming from the works done in RoboDK.




$\cdot 10^4$
1.2
1
0.8
0.6
0.4
0.2
0
self-weight (kg)
3,000
3,200
3,400
3,600
3,800
4,000
4,200
4,400
4,600
4,800
reach (mm)
ABB reach 3000-4000
KUKA reach 3000-4000
Comau reach 3000-4000
Fanuc reach 3000+
Kawasaki reach 3000-4000
Motoman reach 3000-4000
Nachi reach 3000-4000
Staubli reach 3000-4000

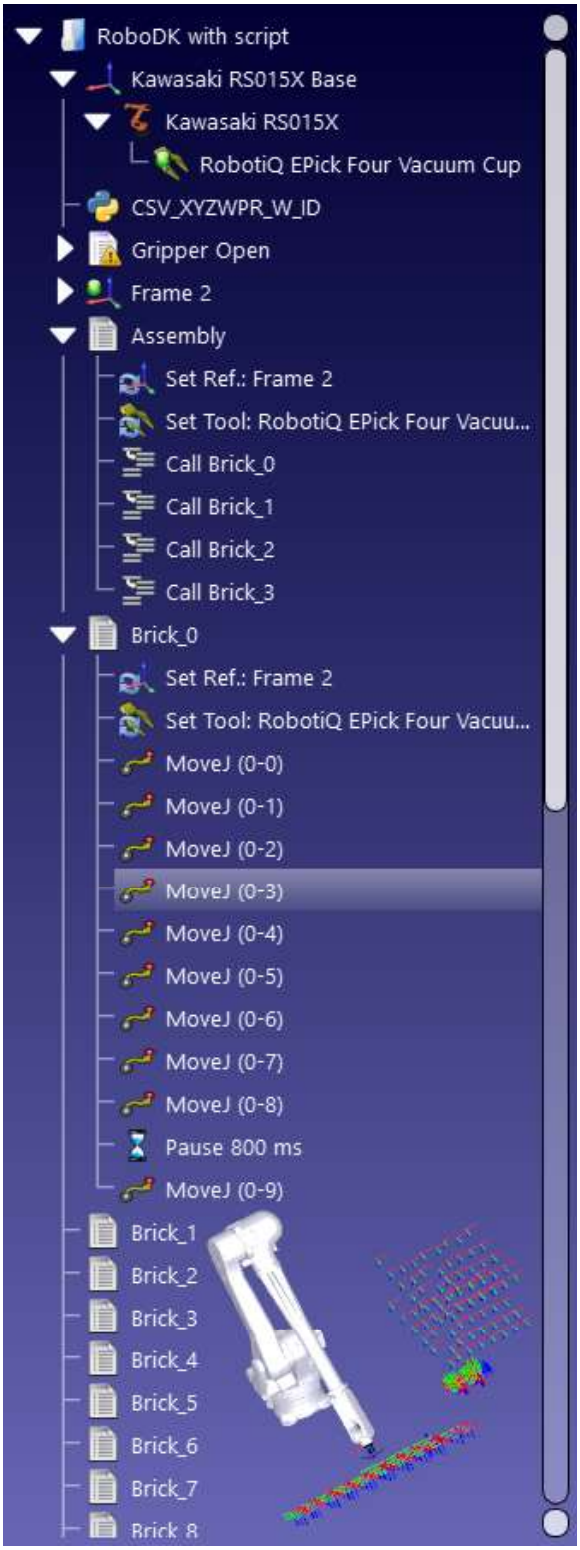


Figure E.1: Work process in RoboDK (as GUI).

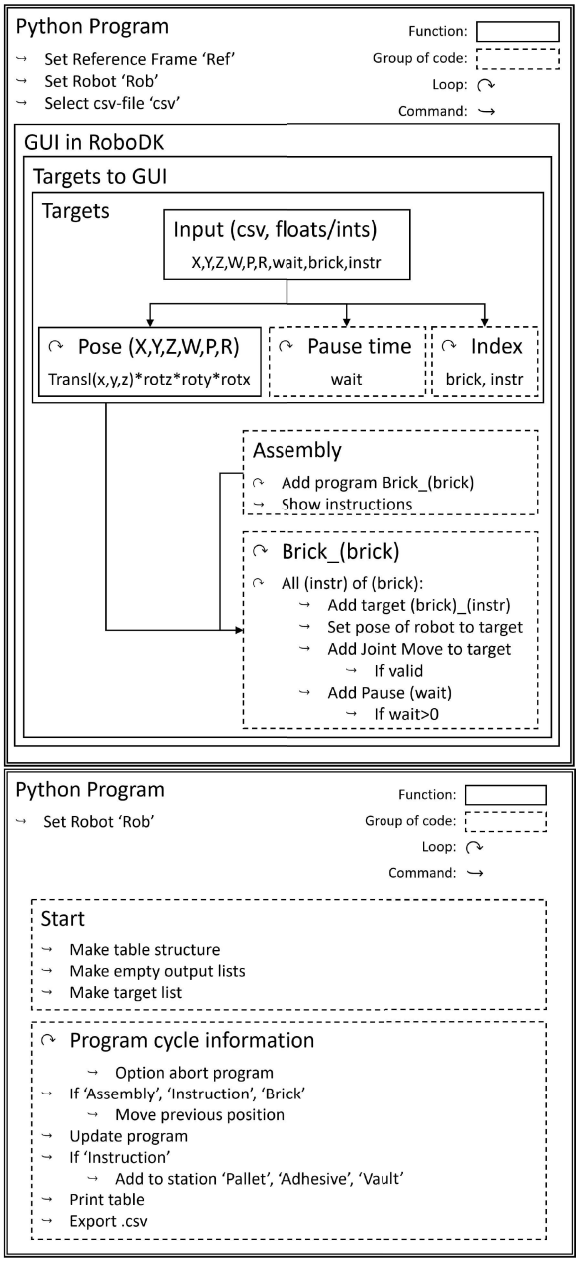


Figure E.2: Workflows in Python.

In [1]:
```
print("This is python file CSV_XYZWPR_W_ID_2.py")
```

```
This is python file CSV_XYZWPR_W_ID_2.py
```

In [2]:
```
# This macro can load CSV files from Denso programs in RoboDK.
# Supported types of files are:
#  1-Tool data : Tool.csv
#  2-Work object data: Work.csv
#  3-Target data: P_Var.csv
# This macro can also filter a given targets file

# Type help("robolink") or help("robodk") for more information
# Press F5 to run the script
# Visit: http://www.robodk.com/doc/PythonAPI/
# For RoboDK API documentation

from robolink import *    # API to communicate with RoboDK
from robodk import *      # basic matrix operations

# Start communication with RoboDK
RDK = Robolink()

# Ask the user to select the robot (ignores the popup if only
ROBOT = RDK.ItemUserPick('Select a robot', ITEM_TYPE_ROBOT)

# Check if the user selected a robot
if not ROBOT.Valid():
    quit()

# Automatically retrieve active reference and tool

#Remove old stuff
##fram = RDK.Item('Targets', ITEM_TYPE_FRAME)# Get the first program called program_name
##if fram.Valid():
##        fram.Delete()
##fram = RDK.Item('Objects', ITEM_TYPE_FRAME)# Get the first program called program_name
##if fram.Valid():
##        fram.Delete()

FRAME = ROBOT.getLink(ITEM_TYPE_FRAME)
##print(FRAME)
##FRAME = Rhino.RDK.AddFrame('Targets','Rhino')
##FRAMO = Rhino.RDK.AddFrame('Objects','Rhino')
TOOL = ROBOT.getLink(ITEM_TYPE_TOOL)

##FRAME = RDK.ItemUserPick('Select a reference frame', ITEM_TYPE_FRAME)
##TOOL = RDK.ItemUserPick('Select a tool', ITEM_TYPE_TOOL)


if not FRAME.Valid() or not TOOL.Valid():# or not FRAMO.Valid():
    raise Exception("Select appropriate FRAME and TOOL references")

#       103 7101
redoit = mbox("Is this a continuation? \n Type No if this is first iteration \n Check row 49",entry='No')
if redoit == 'No':
    start=0
else:
    start = int(redoit)

# Function to convert XYZWPR to a pose
# Important! Specify the order of rotation
def xyzwpr_to_pose(xyzwpr):
    x,y,z,rx,ry,rz = xyzwpr
    return transl(x,y,z)*rotz(rz*pi/180)*roty(ry*pi/180)*rotx(rx*pi/180)
    #return transl(x,y,z)*rotx(rx*pi/180)*roty(ry*pi/180)*rotz(rz*pi/180)
    #return KUKA_2_Pose(xyzwpr)

# csv_file = 'C:/Users/Albert/Desktop/Var_P.csv'
csv_file = getOpenFile(RDK.getParam(r'C:/Users/Joris/Documents/Master Thesis quick access/RoboDK'))

# Specify file codec
codec = 'utf-8' #'ISO-8859-1'

# Load P_Var.CSV data as a list of poses, including links to reference and tool frames
def load_targets(strfile):
    csvdata = LoadList(strfile, ',', codec) # X, Y, Z, W, P, R, wait, brick, instruction, # instr. per brick
    poses = []
    waits = []
    idxs = []
    for i in range(0, len(csvdata)):
        x,y,z,rx,ry,rz = csvdata[i][0:6]
        poses.append(xyzwpr_to_pose([x,y,z,rx,ry,rz]))
        waits.append(csvdata[i][6])
        #idxs.append(csvdata[i][6])
        idxs.append(csvdata[i][7:10])
```

```
    return poses, idxs, waits

# Load and display Targets from P_Var.CSV in RoboDK
def load_targets_GUI(strfile,start):
    #input
    poses, idxs, waits = load_targets(strfile) # Values from csv file

    program_name = 'Assembly' # Name program GH_RDK
    program_name = program_name.replace('-','_').replace(' ','_')
    print(program_name)
    #Remove old stuff
    program = RDK.Item(program_name, ITEM_TYPE_PROGRAM)# Get the first program called program_name
    if start==0:
        if program.Valid():
            program.Delete()

        #Add & set
        program = RDK.AddProgram(program_name, ROBOT)   # Add program and relate to used robot (optional)
        #program.setFrame(FRAME)    #Obsolete. reference frame of robot
    program.setPoseFrame(FRAME) # Sets reference frame of robot
    program.setPoseTool(TOOL)   # Set the robot tool pose (TCP) with respect to the robot flange
        #program.setTool(TOOL)  #Obsolete. robot tool pose to robot flange
    program.ShowInstructions(show=False)
    RDK.setSimulationSpeed(1)

    ls,js,la,ja = 100,200,150,150 #mm/s,deg/s,mm/sš,deg/sš
    program.setSpeed(speed_linear=ls,speed_joints=js,accel_linear=la,accel_joints=ja)
    js1,js2,js3,js4,js5,js6 = 180,180,200,410,360,610
##    program.setSpeedJoints(js1,js2,js3,js4,js5,js6)

    #instructions
    #1
    program.MoveJ(ROBOT.JointsHome())

    #2-...
    proglist,statlist,instruclist = [],[],[]

##    for i in range(4):
##        stat_name = ['Pallet','Adhesive','Vault','Inbetween'][i] # Name program GH_RDK
##        stat_name = stat_name.replace('-','_').replace(' ','_')
##        #Remove old stuff
##        stat = RDK.Item(stat_name, ITEM_TYPE_PROGRAM)# Get the first program called program_name
##        if stat.Valid():
##                stat.Delete()
##        #Add & set
##        stat = RDK.AddProgram(stat_name, ROBOT)   # Add program and relate to used robot (optional)
##        stat.setPoseFrame(FRAME) # Sets reference frame of robot
##        stat.setPoseTool(TOOL)   # Set the robot tool pose (TCP) with respect to the robot flange
##        statlist.append(stat)
    if start==0:
        redo=int(idxs[-1][0])+1
    else:
        redo=0
    for i in range(redo): #Create each program call, len() won't work, -1 is last brick, 0 is brick number
        if (i+1)/10 == int((i+1)/10):
            print(i+1,int(idxs[-1][0])+1)


##        program.RunInstruction('Brick_%s' % (i))    #Assemble all program calls
        #Remove old stuff
##        brick_name = 'Bricks_P_%i' % (i)
##        bric = RDK.Item(brick_name, ITEM_TYPE_OBJECT)# Get the first program called program_name
##        if bric.Valid():
##                bric.Delete()
##        RDK.AddFile(r'C:/Users/Joris/Documents/Master Thesis quick access/RoboDK/Objects/Bricks_P_%i.stp' % (i))

        prog_name = 'Brick_%s' % (i) # Name program GH_RDK
        prog_name = prog_name.replace('-','_').replace(' ','_')
        #Remove old stuff
        prog = RDK.Item(prog_name, ITEM_TYPE_PROGRAM)# Get the first program called program_name
        if prog.Valid():
                prog.Delete()
        #Add & set
        prog = RDK.AddProgram(prog_name, ROBOT)   # Add program and relate to used robot (optional)
        prog.setPoseFrame(FRAME) # Sets reference frame of robot
        prog.setPoseTool(TOOL)   # Set the robot tool pose (TCP) with respect to the robot flange
        prog.ShowInstructions(show=False)
        prog.setSpeed(speed_linear=ls,speed_joints=js,accel_linear=la,accel_joints=ja)
##        prog.setSpeedJoints(js1,js2,js3,js4,js5,js6)
        proglist.append(prog)

    print('Brick_i done')

    for j in range(start,len(idxs)):
        bri, instr = int(idxs[j][0]), int(idxs[j][1])

##        proglist[bri].RunInstruction('Instruction_%s_%s' % (bri,instr))    #Assemble all program calls
```

```
        if int(j/11)==j/11:
            print(j,len(idxs)-1)
            breakable = mbox("Are you sure you want to break the program?",b1 = 'Yes!', b2 = 'No!',t=.5)
            if breakable == True:
                print('Write down:' +j)
                break
            instruc_name = 'Instruction_%s_%s' % (bri,instr) # Name program GH_RDK
            instruc_name = instruc_name.replace('-','_').replace(' ','_')
            #Remove old stuff
            instruc = RDK.Item(instruc_name, ITEM_TYPE_PROGRAM)# Get the first program called program_name
            if instruc.Valid():
                    instruc.Delete()
            #Add & set
            instruc = RDK.AddProgram(instruc_name, ROBOT)   # Add program and relate to used robot (optional)
            instruc.setPoseFrame(FRAME) # Sets reference frame of robot
            instruc.setPoseTool(TOOL)   # Set the robot tool pose (TCP) with respect to the robot flange
            instruc.ShowInstructions(show=False)
            instruc.setSpeed(speed_linear=ls,speed_joints=js,accel_linear=la,accel_joints=ja)
##        instruc.setSpeedJoints(js1,js2,js3,js4,js5,js6)

        for pose, idx, wait in zip(poses, idxs, waits):
##              bri, instr = int(idxs[i][0]), int(idxs[i][1])
            if idx[0:2]==[bri,instr]:
                name = '%i-%i' % (bri,instr)    # Sets name of target to (GH_RDK-0)
                #Remove old stuff
                target = RDK.Item(name, ITEM_TYPE_TARGET)   # Get the first target named 'name'
                if target.Valid():
                    target.Delete()
                #Add target
                target = RDK.AddTarget(name, FRAME, ROBOT)  # Add target to reference frame and to robot
                target.setPose(pose) # Pose relative to robot reference frame

                try:
    ##                  if idx[1]==3 or idx[1]==idx[2]-3:
    ##                      program.MoveJ(target)
    ##                  else:
    ##                      program.MoveL(target)

                    instruc.MoveJ(target)
                    if wait>0:
                        instruc.Pause(wait)
                except:
                    if int(j/11)==j/11:
                        print('Warning: %s can not be reached. It will not be added to the program' % name)
                try:
                    proglist[bri].MoveJ(target)
                    if wait>0:
                        proglist[bri].Pause(wait)
                except:
                    print('Warning: %s can not be reached. It will not be added to the program' % name)
                try:
                    program.MoveJ(target)
                    if wait>0:
                        program.Pause(wait)

    ##                  if idx[1]==1:
    ##                      AttachClosest(keyword='Bricks_P_%i' % (idx[0]),tolerance_mm=15)
    ##                  if idx[1]==idx[2]-2:
    ##                      DetachAll(parent=FRAMO)

                except:
                    print('Warning: %s can not be reached. It will not be added to the program' % name)

        if int(j/11)==j/11:
            del instruc
##        statlist[].RunInstruction('Instruction_%s_%s' % (bri,instr))    #Assemble all program calls

    program.MoveJ(ROBOT.JointsHome())
    program.InstructionListJoints(flags=4,save_to_file=
    r"C:\Users\Joris\Documents\Master Thesis quick access\RoboDK\jointlist.csv")
    #


def load_targets_move(strfile):
    poses, idxs = load_targets(strfile)

    ROBOT.setFrame(FRAME)
    ROBOT.setTool(TOOL)

    ROBOT.MoveJ(ROBOT.JointsHome())

    for pose, idx in zip(poses, idxs):
        try:
            ROBOT.MoveJ(pose)
        except:
            RDK.ShowMessage('Target %i can not be reached' % idx, False)
    ROBOT.MoveJ(ROBOT.JointsHome())
```

```
        # Force just moving the robot after double clicking
        #load_targets_move(csv_file)
        #quit()

        # Recommended mode of operation:
        # 1-Double click the python file creates a program in RoboDK station
        # 2-Generate program generates the program directly

        MAKE_GUI_PROGRAM = False

        ROBOT.setFrame(FRAME)
        ROBOT.setTool(TOOL)


        if RDK.RunMode() == RUNMODE_SIMULATE:
            MAKE_GUI_PROGRAM = True
            # MAKE_GUI_PROGRAM = mbox('Do you want to create a new program? If not, the robot will just move along
             #         the tagets', 'Yes', 'No')
        else:
            # if we run in program generation mode just move the robot
            MAKE_GUI_PROGRAM = False



        if MAKE_GUI_PROGRAM:
            RDK.Render(False) # Faster if we turn render off
            load_targets_GUI(csv_file,start)
        else:
            load_targets_move(csv_file)

        RDK.ShowMessage('Program has run')


        ---------------------------------------------------------------------------

        ModuleNotFoundError                       Traceback (most recent call last)

        <ipython-input-2-01b1dc9e8f0f> in <module>()
         11 # For RoboDK API documentation
         12
    ---> 13 from robolink import *    # API to communicate with RoboDK
         14 from robodk import *      # basic matrix operations
         15


        ModuleNotFoundError: No module named 'robolink'


In [3]: print("This is python file Cycle Time Asembly, Brick & Instructions.py")

This is python file Cycle Time Asembly, Brick & Instructions.py


In [4]: # This example shows how to quickly calculate the cycle time of all programs in the RoboDK station
        #
        # Important notes and tips for accurate cycle time calculation:
        # https://robodk.com/doc/en/General.html#CycleTime

        # Start the RoboDK API
        from robolink import *    # RoboDK API
        from robodk import *
        import numpy as np
        RDK = Robolink()
        ROBOT = RDK.ItemUserPick('Select a robot', ITEM_TYPE_ROBOT)

        Comp = 5

        check_comp = mbox("Have you updated Comp? \n Please check \n Really awful if you hadn't",b1 = 'Yes!',
        b2 = 'No, thanks!')
        if check_comp == False:
            print('Change comp!')
            quit()

        export = [["Program name","Have all targets been reached?","Travel length [mm]","Cycle Time [s]",
        "Cycle Speed [mm/s]"]]
        writeline = "Program name\tHave all targets been reached?\tTravel length\tCycle Time\tCycle Speed"
        msg_html = "<table border=1><tr><td>"+writeline.replace('\t','</td><td>')+"</td></tr>"
        Pallt,Adhet,Vault = [],[],[]
        Pallx,Adhex,Vaulx = [],[],[]
        exprt = [['Pallt','Pallx','Adhet','Adhex','Vault','Vaulx']]
        i=0
        tot = len(RDK.ItemList(ITEM_TYPE_PROGRAM))
        targ = RDK.ItemList(ITEM_TYPE_TARGET)
        targ = [[int(ta) for ta in ele.Name().split('-')] for ele in targ]
```

```
# Ask the user to select a program
#program = RDK.ItemUserPick('Select a program', ITEM_TYPE_PROGRAM)
for program in RDK.ItemList(ITEM_TYPE_PROGRAM):
    if int(i/5)==i/5 or i==1:
        breakable = mbox("Are you sure you want to break the program?",b1 = 'Yes!', b2 = 'No!',t=.5)
        if breakable == True:
            print('Note down:' +i)
            break
        i+=1
        print(i,tot)
    else:
        i+=1
    if program == RDK.Item('Gripper Open'):
        continue
    elif program == RDK.Item('Assembly') or program == RDK.Item('Brick_0')
    or program == RDK.Item('Instruction_0_0'):
        ROBOT.MoveJ(ROBOT.JointsHome())
        target = program
        ROBOT.WaitFinished()
    elif program.Name().split('_')[0]=='Instruction' and int(program.Name().split('_')[2])!=0:
        name = '%i-%i' % (int(program.Name().split('_')[1]),int(program.Name().split('_')[2])-1)
        target = RDK.Item(name, ITEM_TYPE_TARGET)
        ROBOT.MoveJ(target)
        ROBOT.WaitFinished()
    else:
        name = int(program.Name().split('_')[1])-1
        name = '%i-%i' % (int(program.Name().split('_')[1])-1,max([int(ele[1]) for ele in targ
        if int(ele[0])==name]))
        target = RDK.Item(name, ITEM_TYPE_TARGET)
        ROBOT.MoveJ(target)
        ROBOT.WaitFinished()

##    if program == RDK.Item('Gripper Open'):
##        continue
##    elif program == RDK.Item('Assembly') or program == RDK.Item('Brick_0')
##    or program == RDK.Item('Instruction_0_0'):
##        ROBOT.MoveJ(ROBOT.JointsHome())
##        ROBOT.WaitFinished()
##    else:
##        name = '%i-%i' % (int(program.split('_')[1]),int(program.split('_')[2])-1)
##        target = RDK.Item(name, ITEM_TYPE_TARGET)
##        ROBOT.MoveJ(target)
##
####        start_inst = previ.Instruction(previ.InstructionCount()-1)[0]
####        if 'Instruction' in start_inst:
####            RDK.RunCode(start_inst)
####        else:
####            RDK.RunCode(previ.Name())
##        ROBOT.WaitFinished()
####    target = RDK.ItemUserPick(itemtype_or_list=ITEM_TYPE_TARGET)
####    ROBOT.MoveJ(target)
##    previ = program
##    # Retrieve the robot linked to the selected program
##    #robot = program.getLink(ITEM_TYPE_ROBOT)
##
##    # Output the linear speed, joint speed and time (separated by tabs)

    result = program.Update()
    instructions, time, travel, ok, error = result
    speed = travel/time
    if program in [RDK.Item('Brick_%s' %(i)) for i in range(3)]:
        speed = travel/(time-30)
    if program == RDK.Item('Assembly'):
        speed = travel/(time-3*30)
    if ok==1:
        ok = 'Yes'
    elif ok==0:
        ok = 'No'
    if 'Instruction' in program.Name():
        nm,br,ins = program.Name().split('_')
        sum_ins = len([ele for ele in RDK.ItemList(ITEM_TYPE_PROGRAM,True) if 'Instruction_%s' % (br) in ele])
        if int(ins)<3:
            Pallt = (time)
            Pallx = (travel)
            exprt.append([Pallt,Pallx,0,0,0,0])
        elif int(ins)>sum_ins-4:
            Vault = (time)
            Vaulx = (travel)
            exprt.append([0,0,0,0,Vault,Vaulx])
        elif int(ins)>2 and int(ins)<sum_ins-3:
            Adhet = (time)
            Adhex = (travel)
            exprt.append([0,0,Adhet,Adhex,0,0])

    # Print the information
    newline = "%s\t %s \t %.1f mm \t %.1f s \t %.1f mm/s" % (program.Name(), str(ok), travel, time, speed)
```

```
        export.append([program.Name(),str(ok),travel, time, speed])

        msg_html = msg_html + '<tr><td>' + newline.replace('\t','</td><td>') + '</td></tr>'


    try:
        export = [[str(ele).replace('.',',') for ele in export[idx] ] for idx,Ele in enumerate(export)]
        exprt = [[str(ele).replace('.',',') for ele in exprt[idx] ] for idx,Ele in enumerate(exprt)]
        np.savetxt(r"C:\Users\Joris\Documents\Master Thesis quick access\RoboDK\CycleTime.csv",
        export,delimiter ="; ",fmt ='% s')
        np.savetxt(r"C:\Users\Joris\Documents\Master Thesis quick access\RoboDK\CycleTime2.csv",
        exprt,delimiter ="; ",fmt ='% s')
        np.savetxt(r"C:\Users\Joris\Documents\Master Thesis quick access\RoboDK\CycleTime_%s.csv" % (Comp),
        export,delimiter ="; ",fmt ='% s')
        np.savetxt(r"C:\Users\Joris\Documents\Master Thesis quick access\RoboDK\CycleTime2_%s.csv" % (Comp),
        exprt,delimiter ="; ",fmt ='% s')
        RDK.ShowMessage('CSV created/updated')
    except:
        pass
    msg_html = msg_html + '</table>'

    program_name = 'Assembly' # Name program GH_RDK
    program_name = program_name.replace('-','_').replace(' ','_')
    program = RDK.Item(program_name, ITEM_TYPE_PROGRAM)
    program.InstructionListJoints(flags=4,save_to_file=r"C:\Users\Joris\Documents\Master Thesis quick access
    \RoboDK\jointlist.csv")

    RDK.ShowMessage(msg_html)


    ---------------------------------------------------------------------------

    ModuleNotFoundError                       Traceback (most recent call last)

    <ipython-input-4-d47a08201e1a> in <module>()
      5
      6 # Start the RoboDK API
----> 7 from robolink import *    # RoboDK API
      8 from robodk import *
      9 import numpy as np


    ModuleNotFoundError: No module named 'robolink'
```