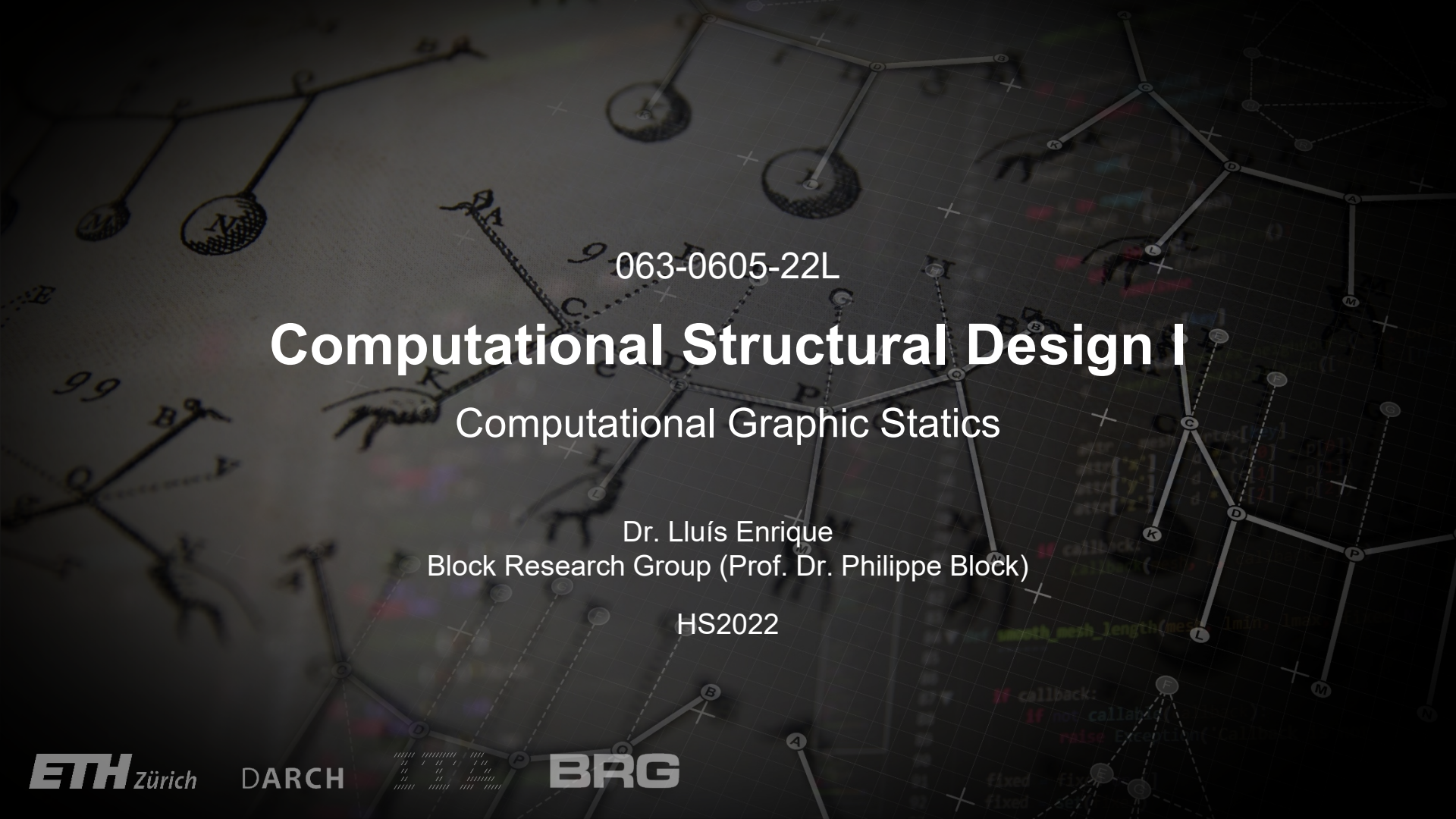


063-0605-22L

Computational Structural Design I

Computational Graphic Statics

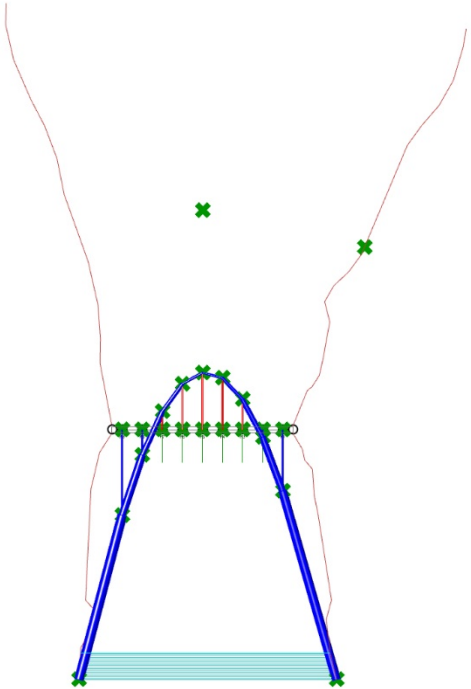
Dr. Lluís Enrique
Block Research Group (Prof. Dr. Philippe Block)

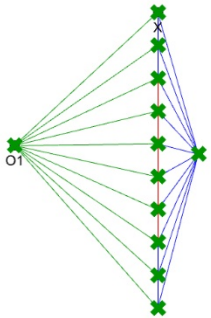
HS2022

ETH Zürich DARCH BRG

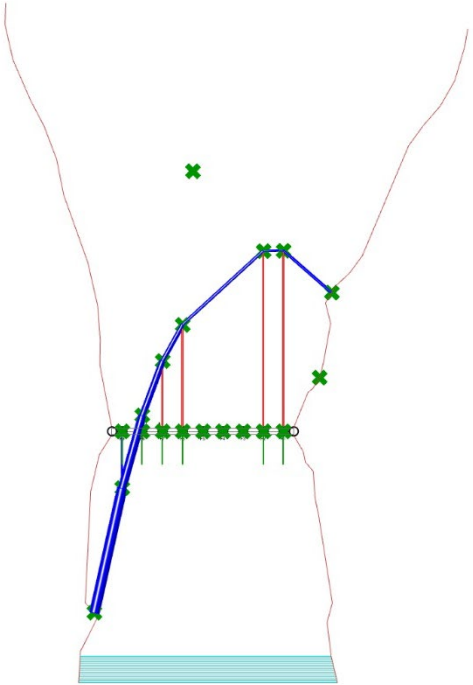
Week 9 Friday, November 18th

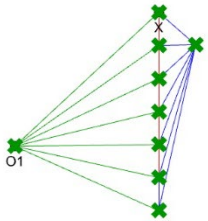
9:45 – 10:30	Recap of last week’s exercise **Lecture: Algebraic Graphic statics**
10:30 – 10:45	Break
10:45 – 11:30	**Tutorial: Interactive Graphic Statics**
11:30 – 11:45	Break
11:45 – 12:30	**Tutorial: Interactive Graphic Statics**



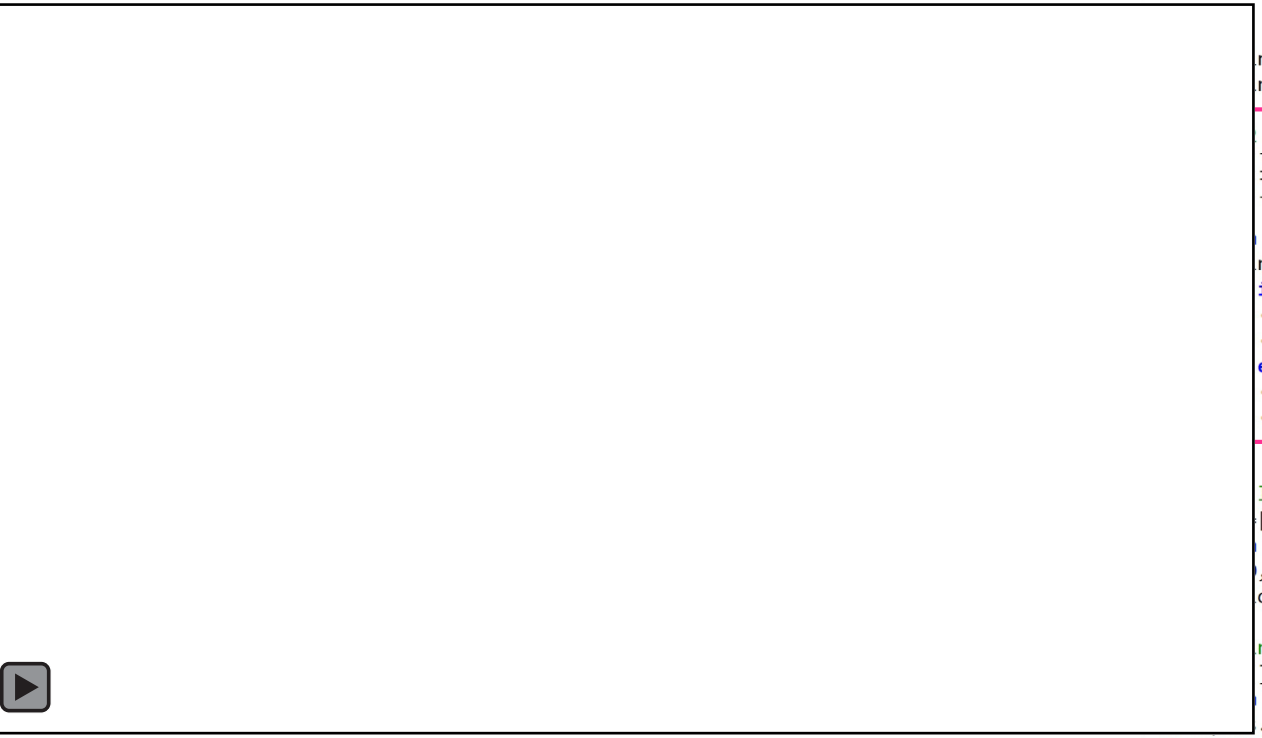


```python
import rhinoscriptsyntax as rs

#1.a INPUT LOADS

#Task 1
#input
Lp_anc=[p1,p2,p3,p4,p5,p6,p7,p8,p9]
Ln_mag=[n1,n2,n3,n4,n5,n6,n7,n8,n9]

#vector loads
Lv_load=[]
for i in range (0,len(Lp_anc)):
    v=[0,Ln_mag[i],0]
    Lv_load.append(v)

#LOA (Line Of Action)
Ll_LOA=[]
for i in range (0,len(Lp_anc)):
    p=rs.CopyObject(Lp_anc[i])
    rs.MoveObject(p,[0,-10,0])
    LOA=rs.AddLine(Lp_anc[i],p)
    Ll_LOA.append(LOA)
```

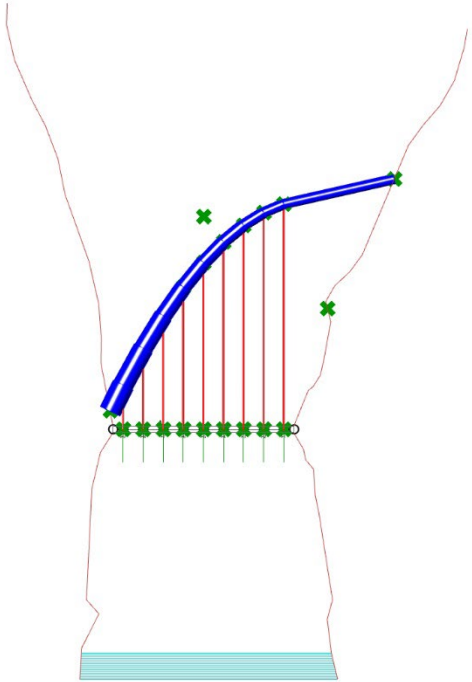


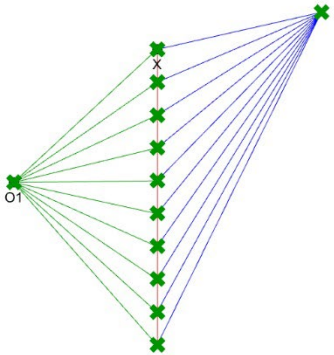



```
import rhinoscriptsyntax as rs

#1.a INPUT LOADS

#input
Lp_anc_in=[p1,p2,p3,p4,p5,p6,p7,p8,p9]
Ln_mag_in=[n1,n2,n3,n4,n5,n6,n7,n8,n9]

#Task 2
Lp_anc=[]
Ln_mag=[]
for i in range (0,len(Lp_anc_in)):
    if Ln_mag_in[i] != 0:
        Lp_anc.append(Lp_anc_in[i])
        Ln_mag.append(Ln_mag_in[i])


#vector loads
Lv_load=[]
for i in range (0,len(Lp_anc)):
    v=[0,Ln_mag[i],0]
    Lv_load.append(v)

#LOA (Line Of Action)
Ll_LOA=[]
for i in range (0,len(Lp_anc)):
    p=rs.CopyObject(Lp_anc[i])
    rs.MoveObject(p,[0,-10,0])
    LOA=rs.AddLine(Lp_anc[i],p)
    Ll_LOA.append(LOA)
```

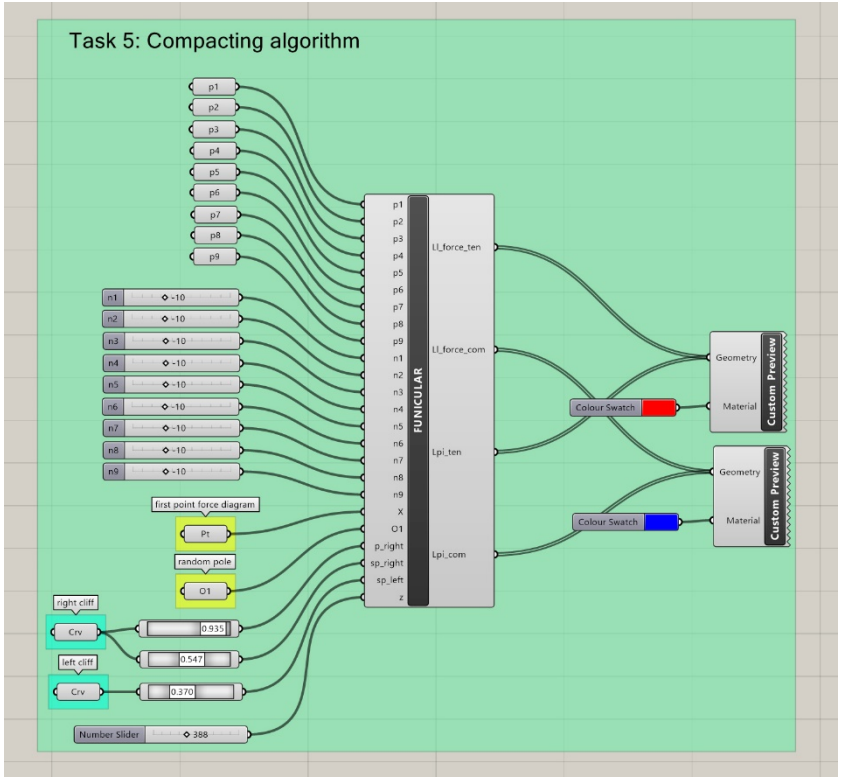


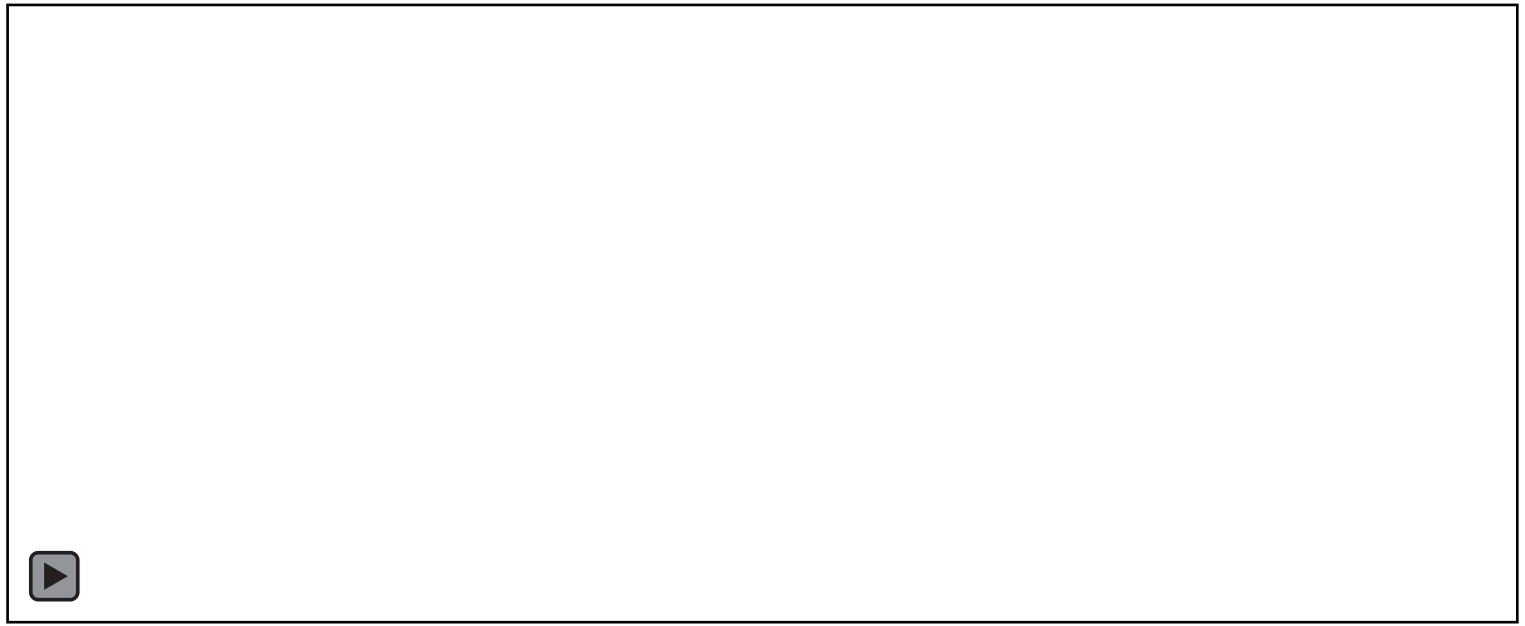
```
import rhinoscriptsyntax as rs

#1.a INPUT LOADS


n=[p1,p2,p3,p4,p5,p6,p7,p8,p9]
n=[n1,n2,n3,n4,n5,n6,n7,n8,n9]

 and 3
]
]

 range (0,len(Lp_anc_in)):
n_mag_in[i] != 0:
if M == i:
····Lp_anc.append(Lp_anc_in[i])
····Ln_mag.append(Ln_mag_in[i]+Q)
else:
····Lp_anc.append(Lp_anc_in[i])
····Ln_mag.append(Ln_mag_in[i])


loads
[]
 range (0,len(Lp_anc)):
),Ln_mag[i],0]
oad.append(v)

ne Of Action)
]
 range (0,len(Lp_anc)):
.CopyObject(Lp_anc[i])
····rs.MoveObject(p,[0,-10,0])
····LOA=rs.AddLine(Lp_anc[i],p)
····Ll_LOA.append(LOA)
```

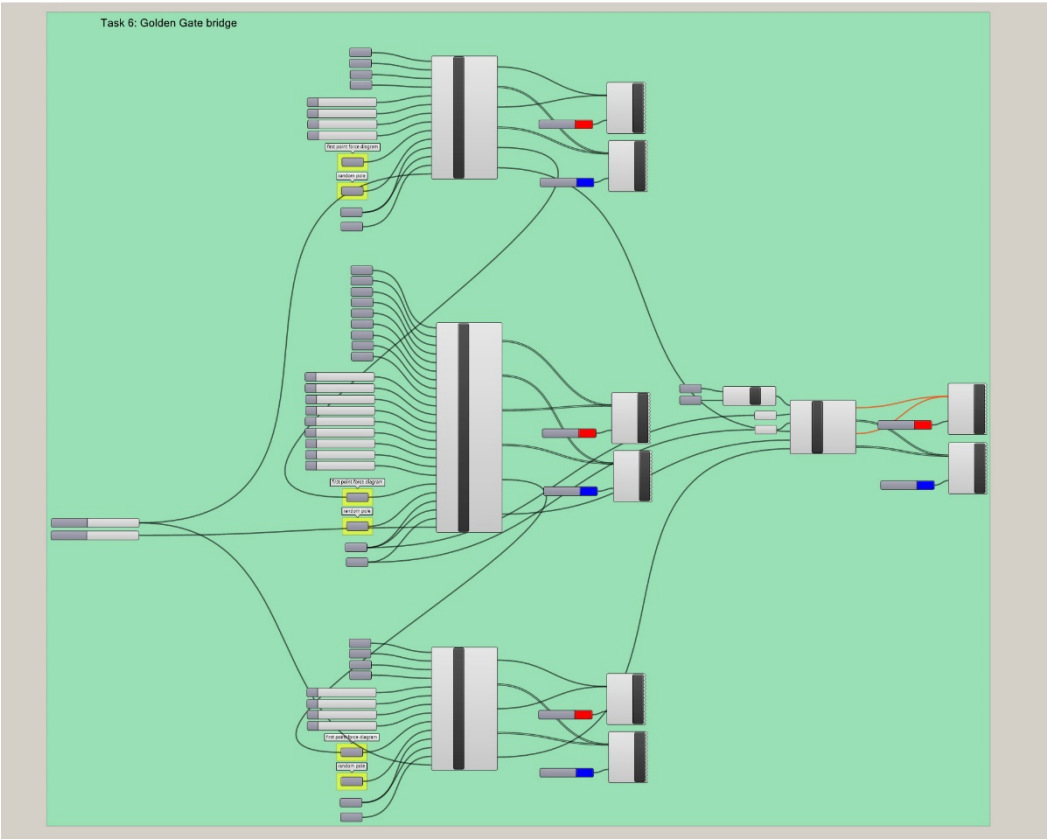




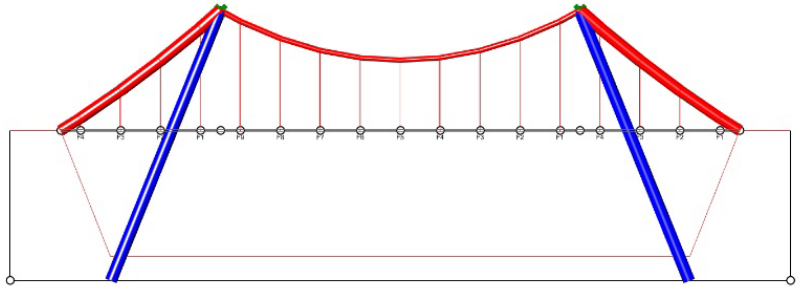

```python
import rhinoscriptsyntax as rs

#2. SUPPORTS AND REACTIONS

#creating lines of reaction forces in form diagram
l1=rs.AddLine(p_R,sp_left)
l2=rs.AddLine(p_R,sp_right)

#building reaction forces in force diagram
v1=rs.VectorCreate(Lp_loadline[-1],p_R)
l1_m=rs.MoveObject(l1,v1)

v2=rs.VectorCreate(Lp_loadline[0],p_R)
l2_m=rs.MoveObject(l2,v2)

O2=rs.LineLineIntersection(l1_m,l2_m)[0]

#Task 4
l_CS=rs.AddLine(sp_left,sp_right)
v=rs.VectorCreate(O2,sp_left)
l_CS_copy=rs.CopyObject(l_CS,v)

al=rs.AddLine(Lp_loadline[0],Lp_loadline[-1])
al_move=rs.MoveObject(al,[50,0,0])

O2=rs.LineLineIntersection(l_CS_copy, al_move)[0]
```

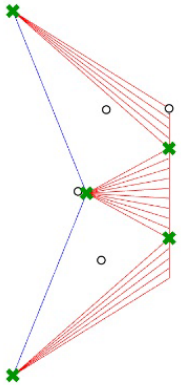

Task 5: Compacting algorithm
p1
p2
p3
p4
p5
p6
p7
p8
p9
n1 -10
n2 -10
n3 -10
n4 -10
n5 -10
n6 -10
n7 -10
n8 -10
n9 -10
first point force diagram
Pt
random pole
O1
right cliff
Crv
0.935
left cliff
Crv
0.547
0.370
Number Slider 388
FUNICULAR
p1
p2
p3
p4
p5
p6
p7
p8
p9
n1
n2
n3
n4
n5
n6
n7
n8
n9
X
O1
p_right
sp_right
sp_left
z
Ll_force_ten
Ll_force_com
Lpi_ten
Lpi_com
Colour Swatch
Colour Swatch
Geometry
Material
Custom Preview
Geometry
Material
Custom Preview

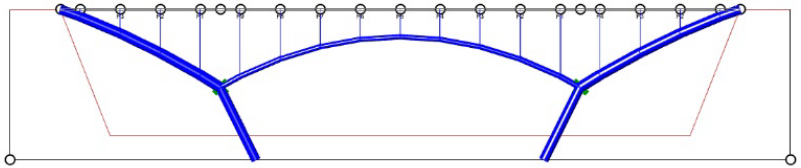


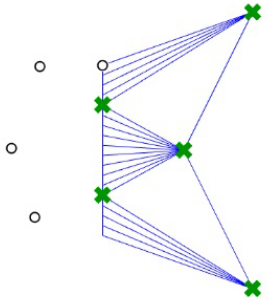


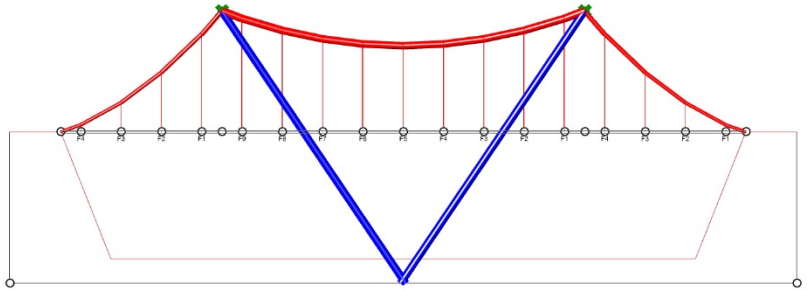

```python
import rhinoscriptsyntax as rs

l1_force=rs.AddLine(O2_2,O2_1)
l2_force=rs.AddLine(O2_3,O2_2)

v1=rs.VectorCreate(sp_right,O2_1)
al1_form=rs.CopyObject(l1_force,v1)
ip1=rs.LineLineIntersection(l_terrain,al1_form)[0]
l1_form=rs.AddLine(sp_right,ip1)

v2=rs.VectorCreate(sp_left,O2_2)
al2_form=rs.CopyObject(l2_force,v2)
ip2=rs.LineLineIntersection(l_terrain,al2_form)[0]
l2_form=rs.AddLine(sp_left,ip2)

Ll_force=[l1_force,l2_force]
Ll_form=[l1_form,l2_form]

#check angle
Langle=[]
for i in range (0,len(Ll_form)):
    angle=rs.Angle2(Ll_form[i],Ll_force[i])
    Langle.append(round(angle[0]))

#colors in force diagram and pipes in form diagram
x=0.03
Ll_force_ten=[]
Ll_force_com=[]
Lpi_ten=[]
Lpi_com=[]
for i in range (0,len(Ll_form)):
    pi=rs.AddPipe(Ll_form[i],0,x*rs.CurveLength(Ll_force[i]),0,1)
    if Langle[i] ==0:
        Ll_force_ten.append(Ll_force[i])
        Lpi_ten.append(pi[0])
    else:
        Ll_force_com.append(Ll_force[i])
        Lpi_com.append(pi[0])
```

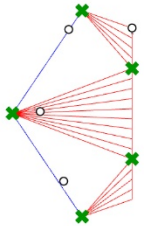


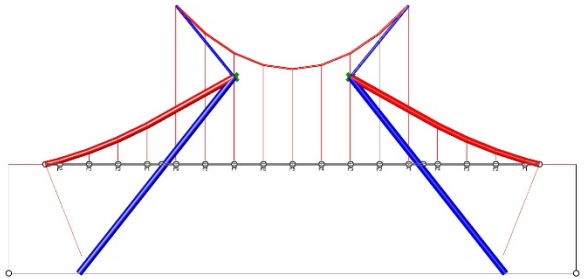


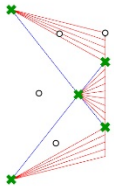


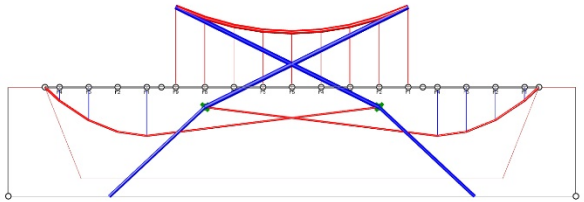


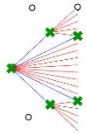


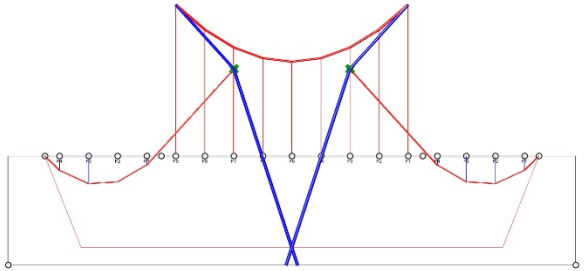


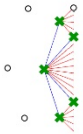


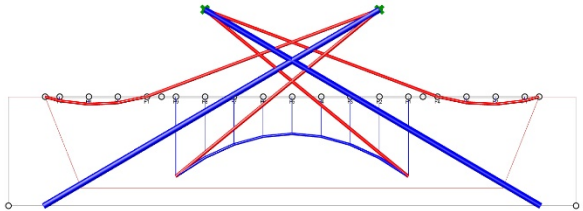


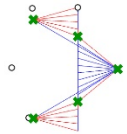


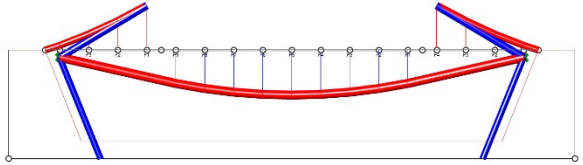


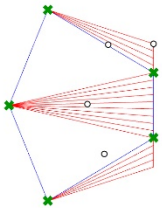


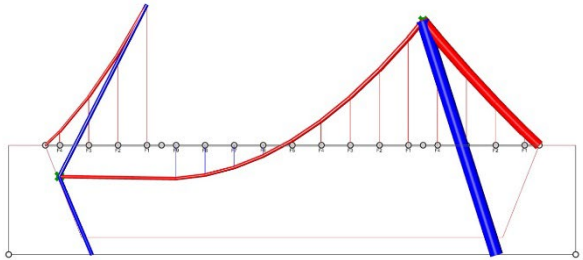


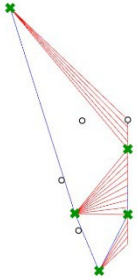


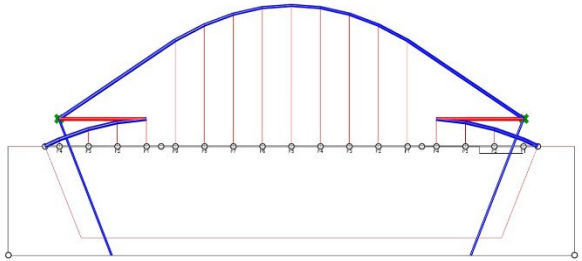


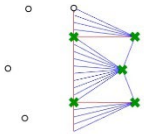


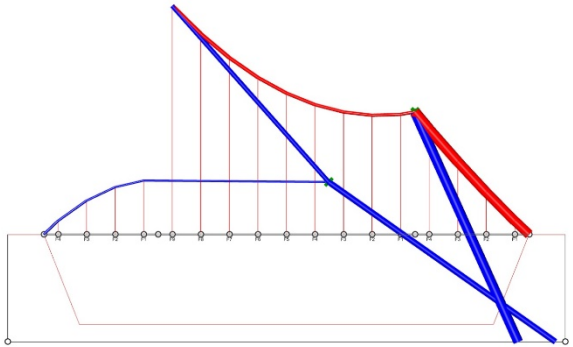


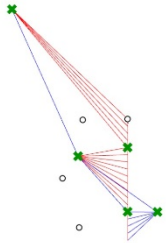






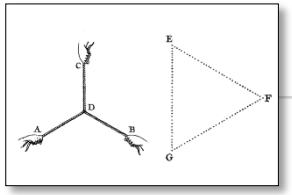




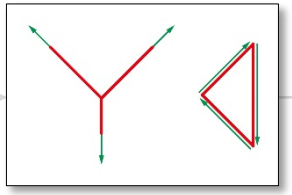


Schedule

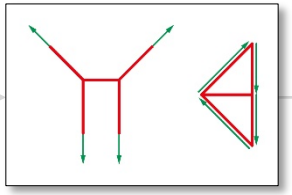
Typology	Topic	Module	Week		Topic	Lead
cables & arches	Introduction		Week 1 (23/9)	**Lecture**	**Introduction to graphic statics**	Dr. Lluis Enrique
				Tutorial	Introduction to Rhino and Grasshopper	
				Work session	graphic statics + Grasshopper	
	Procedural GS (Grasshopper)	I	Week 2 (30/9)		Quick recap of last week's exercise	Dr. Lluis Enrique
				Lecture	**Algorithmic design & thinking**	
				Tutorial	Single node bridge in Grasshopper	
			Week 3 (7/10)	Work session	Single node bridge in Grasshopper	
		II	Week 4 (14/10)		Quick recap of last week's exercise	Dr. Lluis Enrique
				Lecture	**Computational graphic statics**	
				Tutorial	Multi-node bridge in Grasshopper	
			Week 5 (21/10)	Work session	Multi-node bridge in Grasshopper	
Week 6: Seminar week						
cables & arches	Procedural GS (Python)	III	Week 7 (4/11)		Quick recap of last week's exercise	Dr. Lluis Enrique
				Lecture	**Programming**	
				Tutorial	Multi-node bridge in Python	
			Week 8 (11/11)	Work session	Multi-node bridge in Python	
trusses	AGS	IV	Week 9 (18/11)		Quick recap of last week's exercise	Dr. Lluis Enrique
				Lecture	**Algebraic Graph Statics (AGS)**	Dr. Lluis Enrique
				Tutorial	Interactive Graphic Statics (IGS)	Chaoyu Du
			Week 10 (25/11)	Work session	Truss analysis using IGS	
shells	TNA	V	Week 11 (02/12)		Quick recap of last week's exercise	Dr. Lluis Enrique
				Lecture	**Thrust Network Analysis (TNA)**	Dr. Lluis Enrique
				Tutorial	RhinoVault 2 (rV2)	Selina Bitting
			Week 12 (09/12)	Work session	Shell design using rV2	



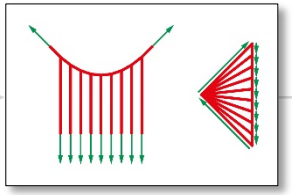
Introduction



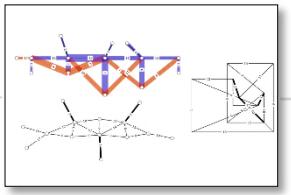
Module I
Procedural Graphic Statics I (single node)



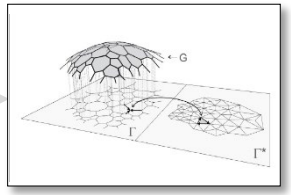
Module II
Procedural Graphic Statics II (multi node)



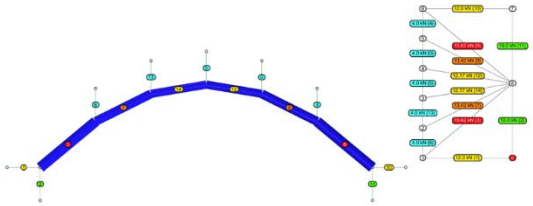
Module III
Programming Graphic Statics

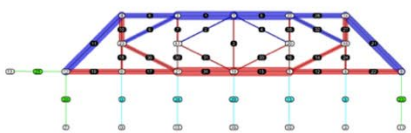


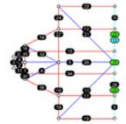
Module IV
Algebraic Graph Statics

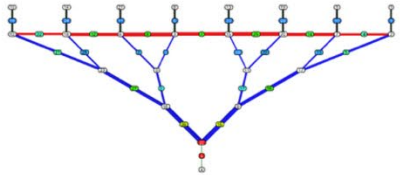


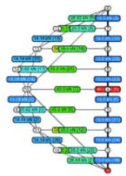
Module V
Thrust Network Analysis

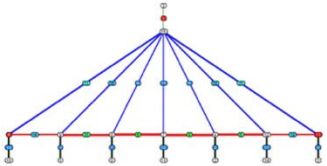


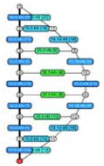


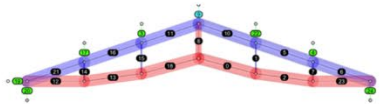


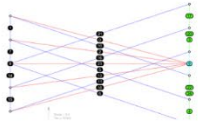


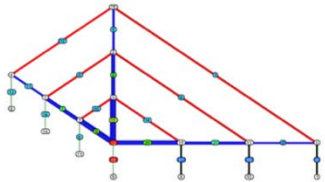


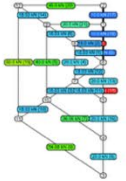


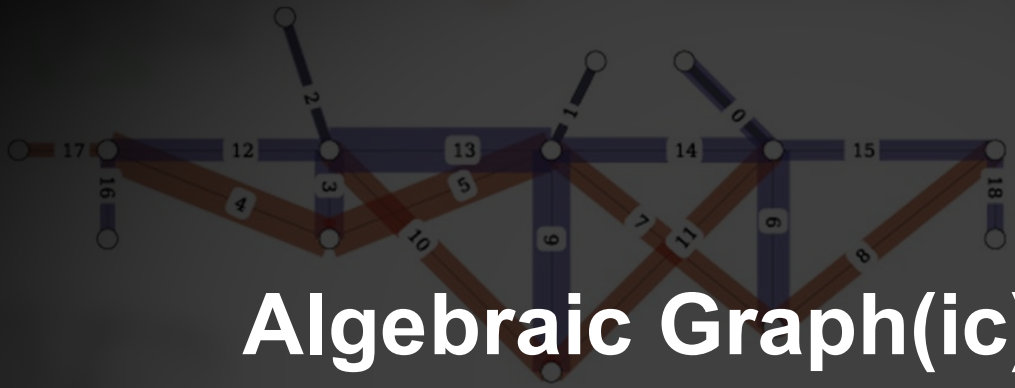






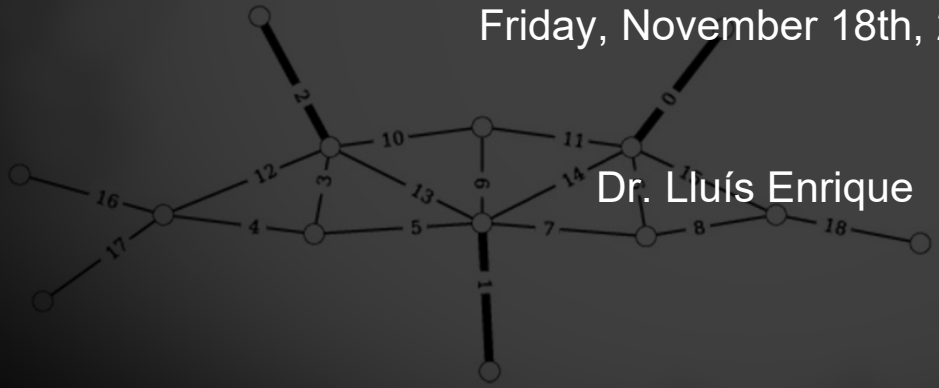






Algebraic Graph(ic) Statics

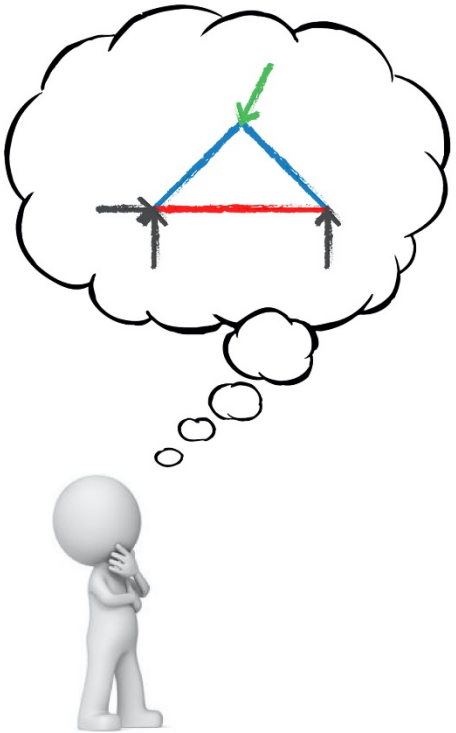
Friday, November 18th, 2022



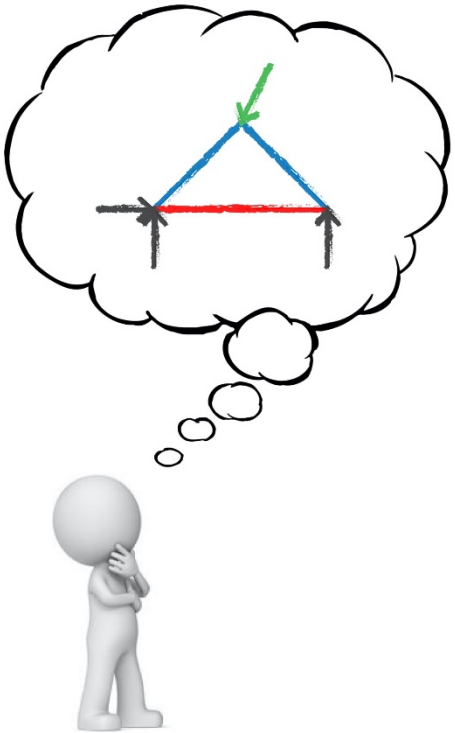
Dr. Lluís Enrique

ETH Zürich DARCH BRG

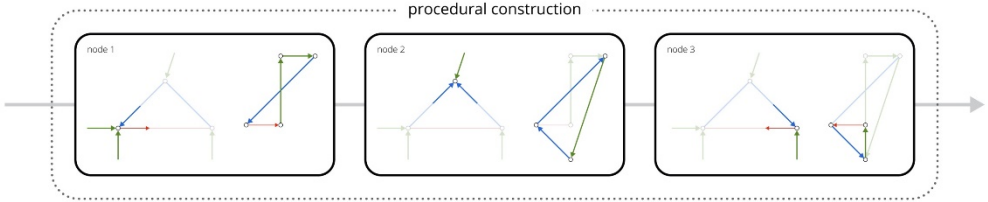
form diagram



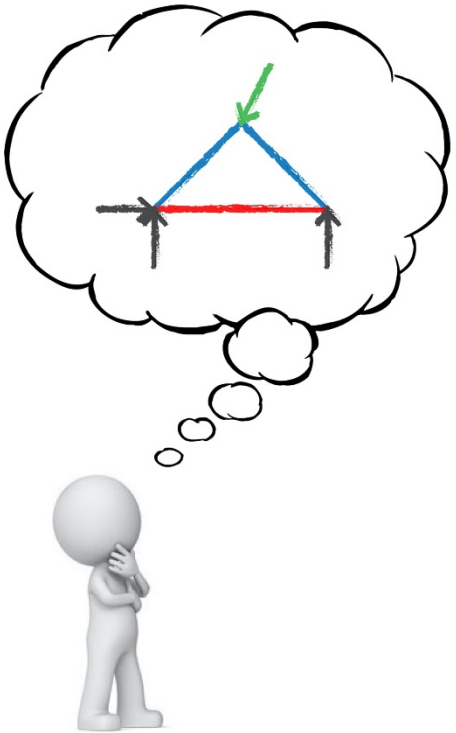
form diagram



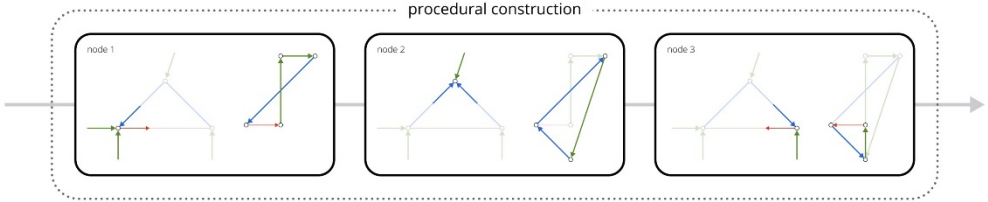
procedural construction



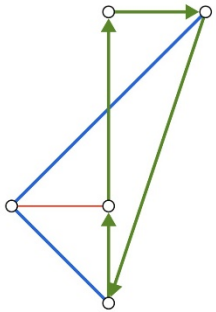
form diagram



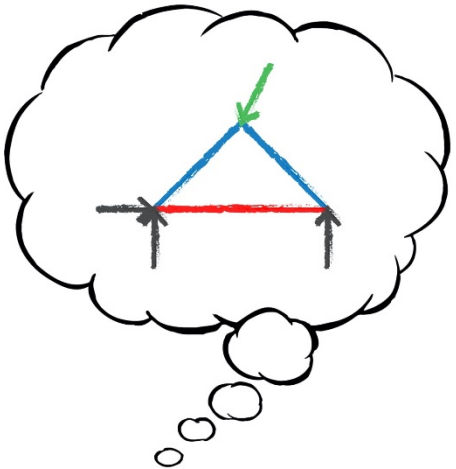
procedural construction



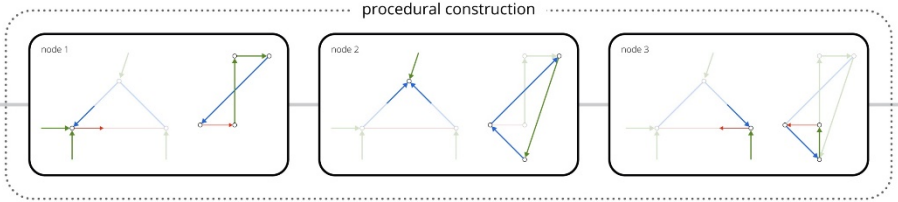
force diagram



form diagram

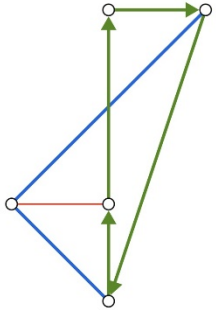


procedural construction



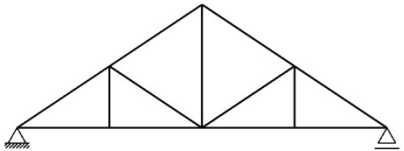
algebraic construction

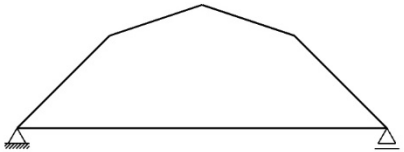
force diagram



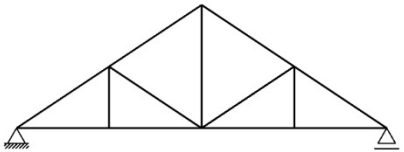
Differences between typology, topology and geometry

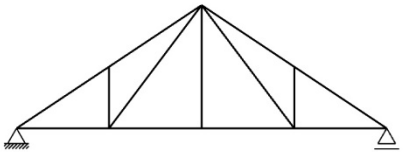
Typology



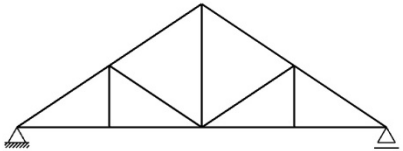


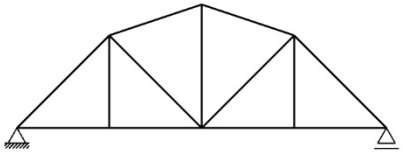
Topology



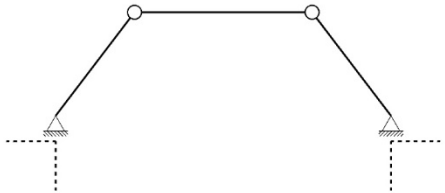


Geometry





Cables / Arches



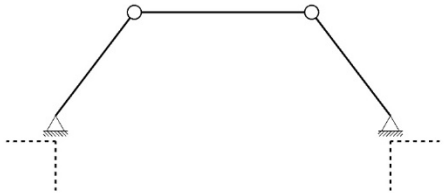
b (bars): 3
r (reactions): 4
n (nodes): 4

b + r < 2*n

3 + 4 < 2*4

Unstable!

Cables / Arches



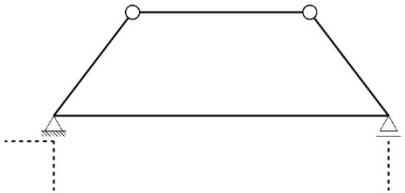
b (bars): 3
r (reactions): 4
n (nodes): 4

b + r < 2*n

3 + 4 < 2*4

Unstable!

Arch-cables



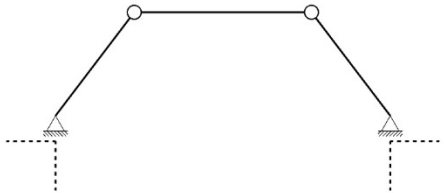
b (bars): 4
r (reactions): 3
n (nodes): 4

b + r < 2*n

4 + 3 < 2*4

Unstable!

Cables / Arches



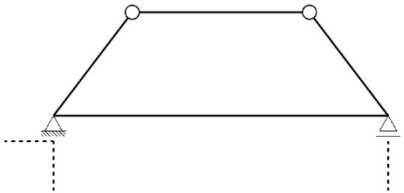
b (bars): 3
r (reactions): 4
n (nodes): 4

b + r < 2*n

3 + 4 < 2*4

Unstable!

Arch-cables



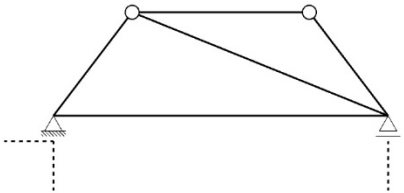
b (bars): 4
r (reactions): 3
n (nodes): 4

b + r < 2*n

4 + 3 < 2*4

Unstable!

Trusses

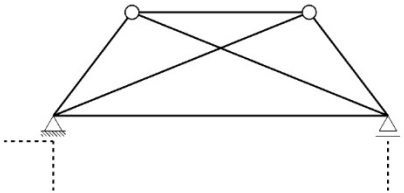


b (bars): 5
r (reactions): 3
n (nodes): 4

b + r = 2*n

5 + 3 = 2*4

Stable!
and
statically determinate!



b (bars): 6
r (reactions): 3
n (nodes): 4

b + r > 2*n

6 + 3 > 2*4

Stable!
but
statically indeterminate!

1. Form graph

Connectivity (edges & vertices)

Directed form graph

2. Force graph

Dual graph

Directed force graph

3. Assign forces

4. Solve equilibrium matrix

5. Update force graph

6. Reciprocal form & force graphs

1. Form graph

Connectivity (edges & vertices)

Directed form graph

2. Force graph

Dual graph

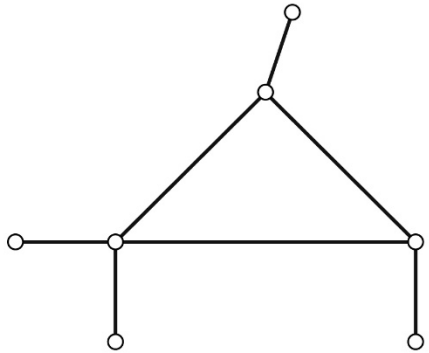
Directed force graph

3. Assign forces

4. Solve equilibrium matrix

5. Update force graph

6. Reciprocal form & force graphs



1. Form graph

Connectivity (edges & vertices)

Directed form graph

2. Force graph

Dual graph

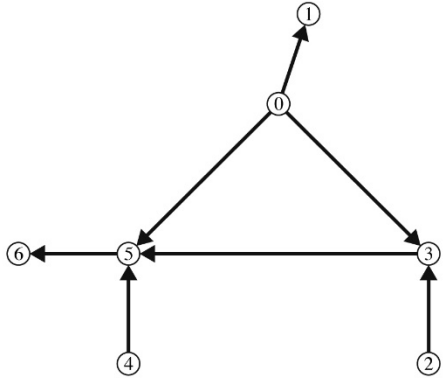
Directed force graph

3. Assign forces

4. Solve equilibrium matrix

5. Update force graph

6. Reciprocal form & force graphs



1. Form graph

Connectivity (edges & vertices)

Directed form graph

2. Force graph

Dual graph

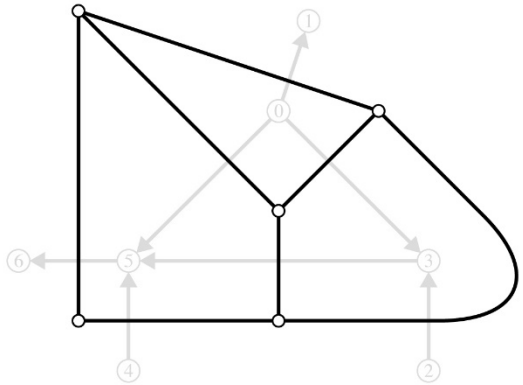
Directed force graph

3. Assign forces

4. Solve equilibrium matrix

5. Update force graph

6. Reciprocal form & force graphs



1. Form graph

Connectivity (edges & vertices)

Directed form graph

2. Force graph

Dual graph

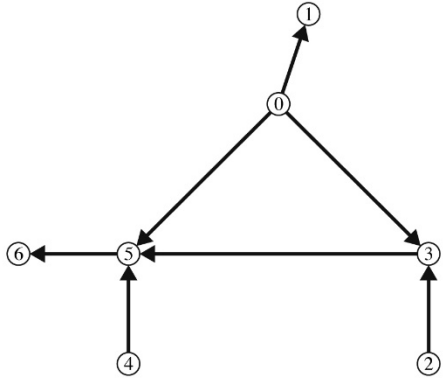
Directed force graph

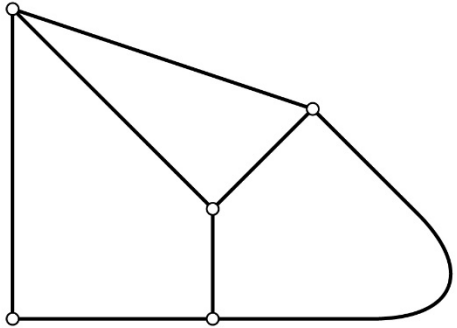
3. Assign forces

4. Solve equilibrium matrix

5. Update force graph

6. Reciprocal form & force graphs





1. Form graph

Connectivity (edges & vertices)

Directed form graph

2. Force graph

Dual graph

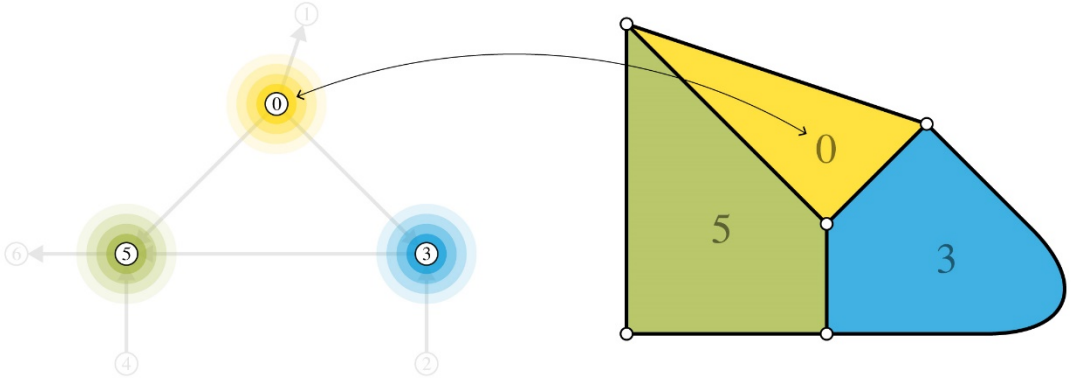
Directed force graph

3. Assign forces

4. Solve equilibrium matrix

5. Update force graph

6. Reciprocal form & force graphs



1. Form graph

Connectivity (edges & vertices)

Directed form graph

2. Force graph

Dual graph

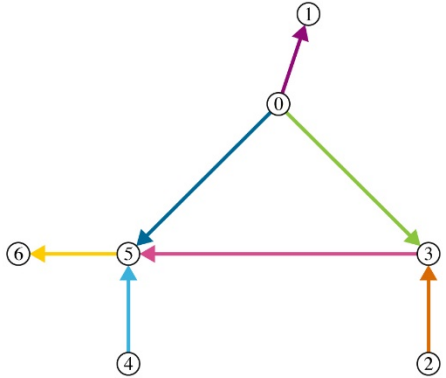
Directed force graph

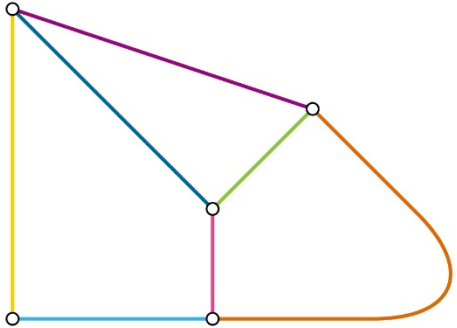
3. Assign forces

4. Solve equilibrium matrix

5. Update force graph

6. Reciprocal form & force graphs





1. Form graph

Connectivity (edges & vertices)

Directed form graph

2. Force graph

Dual graph

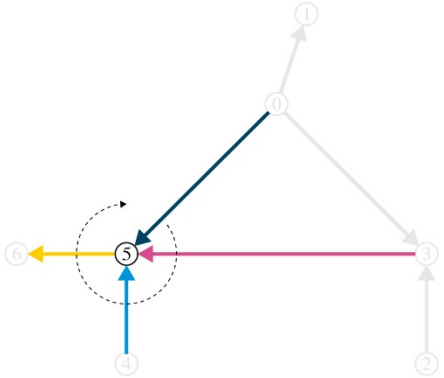
Directed force graph

3. Assign forces

4. Solve equilibrium matrix

5. Update force graph

6. Reciprocal form & force graphs



1. Form graph

Connectivity (edges & vertices)

Directed form graph

2. Force graph

Dual graph

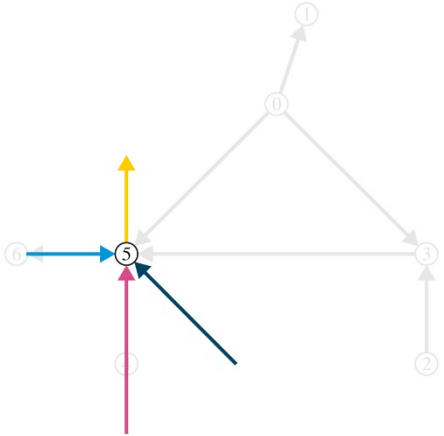
Directed force graph

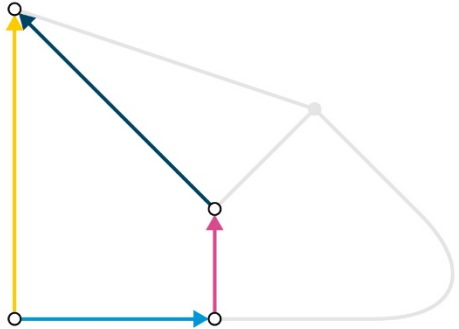
3. Assign forces

4. Solve equilibrium matrix

5. Update force graph

6. Reciprocal form & force graphs





1. Form graph

Connectivity (edges & vertices)

Directed form graph

2. Force graph

Dual graph

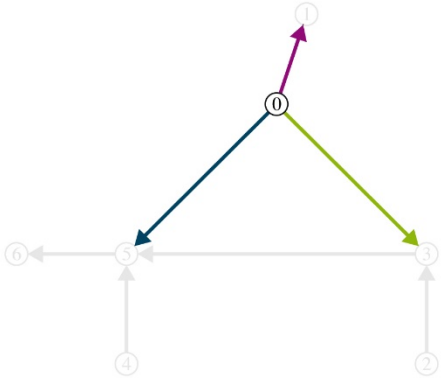
Directed force graph

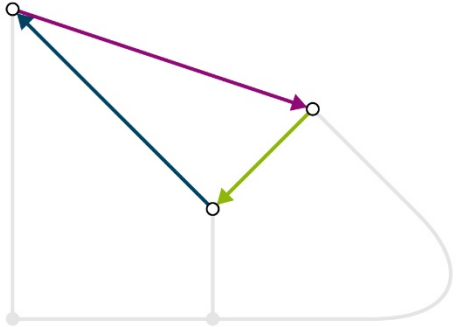
3. Assign forces

4. Solve equilibrium matrix

5. Update force graph

6. Reciprocal form & force graphs





1. Form graph

Connectivity (edges & vertices)

Directed form graph

2. Force graph

Dual graph

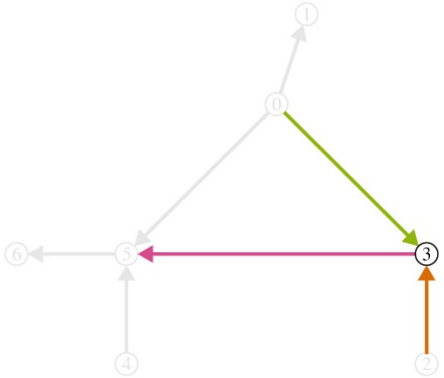
Directed force graph

3. Assign forces

4. Solve equilibrium matrix

5. Update force graph

6. Reciprocal form & force graphs



1. Form graph

Connectivity (edges & vertices)

Directed form graph

2. Force graph

Dual graph

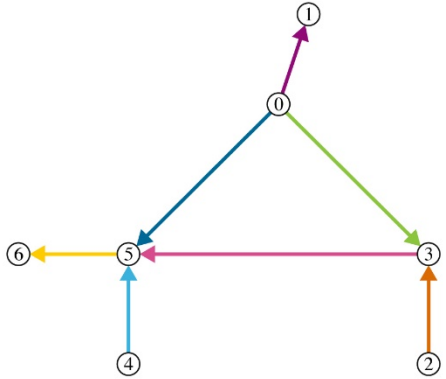
Directed force graph

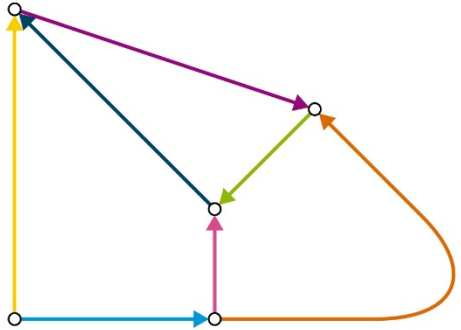
3. Assign forces

4. Solve equilibrium matrix

5. Update force graph

6. Reciprocal form & force graphs





1. Form graph

Connectivity (edges & vertices)

Directed form graph

2. Force graph

Dual graph

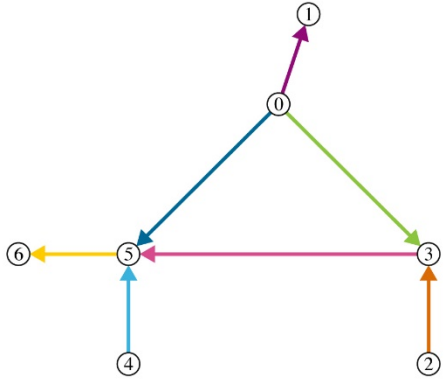
Directed force graph

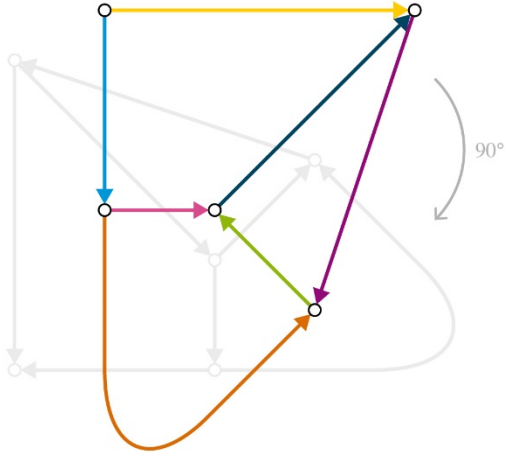
3. Assign forces

4. Solve equilibrium matrix

5. Update force graph

6. Reciprocal form & force graphs





1. Form graph

Connectivity (edges & vertices)

Directed form graph

2. Force graph

Dual graph

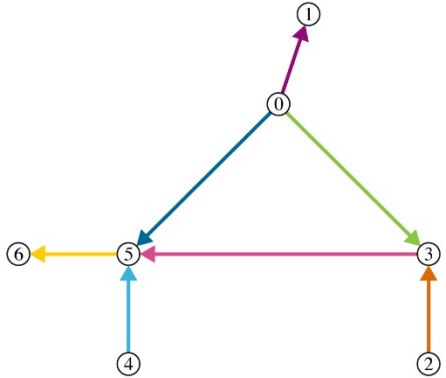
Directed force graph

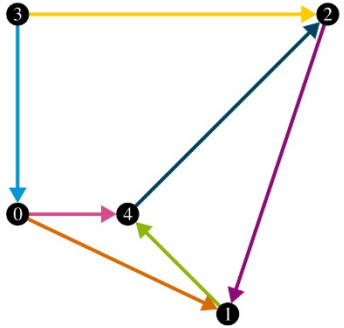
3. Assign forces

4. Solve equilibrium matrix

5. Update force graph

6. Reciprocal form & force graphs





1. Form graph

Connectivity (edges & vertices)

Directed form graph

2. Force graph

Dual graph

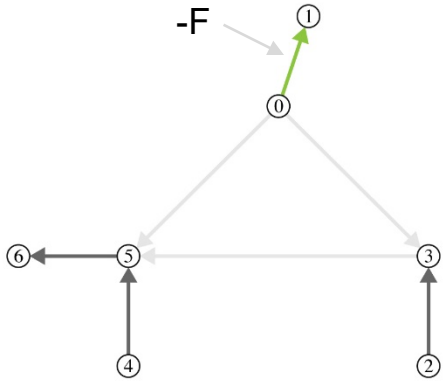
Directed force graph

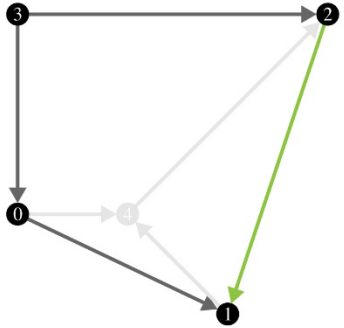
3. Assign forces

4. Solve equilibrium matrix

5. Update force graph

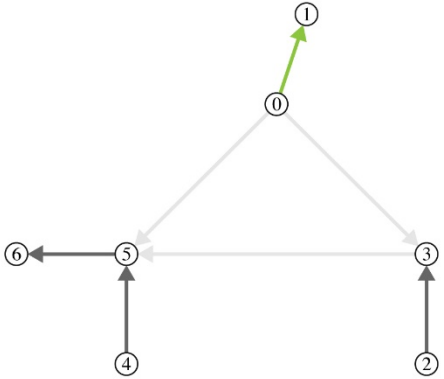
6. Reciprocal form & force graphs

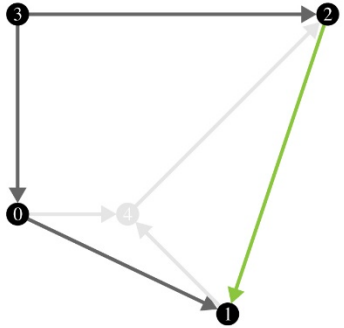




$$C_{ij} = \begin{cases} +1 \text{ if vertex } i \text{ is the head of edge } j \\ -1 \text{ if vertex } i \text{ is the tail of edge } j \\ 0 \text{ otherwise} \end{cases}$$

$$C^*_{ij} = \begin{cases} +1 \text{ if the face cycle traverses edge } j \text{ in the same direction as its orientation} \\ -1 \text{ if in the opposite direction} \\ 0 \text{ otherwise} \end{cases}$$





Closed polygon constraint $\begin{cases} \mathbf{C_i u^* = 0} \\ \mathbf{C_i v^* = 0} \end{cases}$

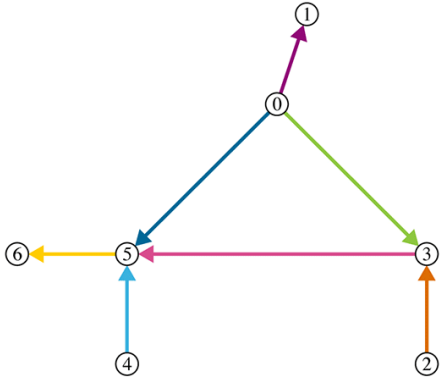
Parallel constraint $\begin{cases} \mathbf{u^* = Qu} \\ \mathbf{v^* = Qv} \end{cases}$

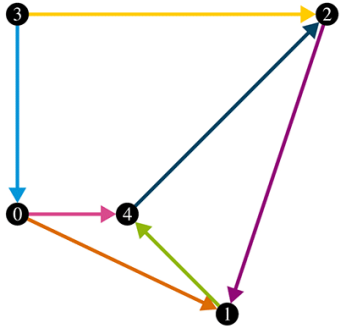
$$\begin{cases} \mathbf{C_i U q = 0} \\ \mathbf{C_i V q = 0} \end{cases}$$

equilibrium matrix

$$\mathbf{Aq = 0} \quad , \quad \mathbf{A} = \left[\frac{\mathbf{C_i U}}{\mathbf{C_i V}} \right]$$

1. **Form graph**
   - Connectivity (edges & vertices)
   - Directed form graph
2. **Force graph**
   - Dual graph
   - Directed force graph
3. **Assign forces**
4. **Solve equilibrium matrix**
5. **Update force graph**
6. Reciprocal form & force graphs





$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_\mathrm{d} \mid \mathbf{A}_\mathrm{id} \end{bmatrix} \quad , \quad \mathbf{q} = \begin{bmatrix} \mathbf{q}_\mathrm{d} \\ \hline \mathbf{q}_\mathrm{id} \end{bmatrix}$$

assigned loads/forces

$$\mathbf{A}_\mathrm{d}\mathbf{q}_\mathrm{d} + \mathbf{A}_\mathrm{id}\mathbf{q}_\mathrm{id} = 0$$
$$\mathbf{q}_\mathrm{d} = -\mathbf{A}_\mathrm{d}^{-1}\mathbf{A}_\mathrm{id}\mathbf{q}_\mathrm{id}$$

$$\mathbf{q} = \begin{bmatrix} -\mathbf{A}_\mathrm{d}^{-1}\mathbf{A}_\mathrm{id} \\ \hline \mathbf{I}_k \end{bmatrix} \mathbf{q}_\mathrm{id}$$

all force densities

$$\mathbf{x}^* = (\mathbf{C}^{*\mathrm{t}})^\dagger \mathbf{Q}\mathbf{u}$$
$$\mathbf{y}^* = (\mathbf{C}^{*\mathrm{t}})^\dagger \mathbf{Q}\mathbf{v}$$

new force diagram coordinates

1. Form graph

Connectivity (edges & vertices)

Directed form graph

2. Force graph

Dual graph

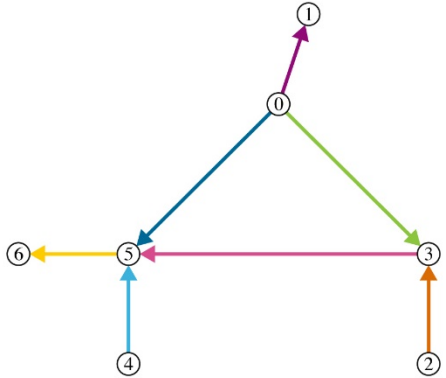
Directed force graph

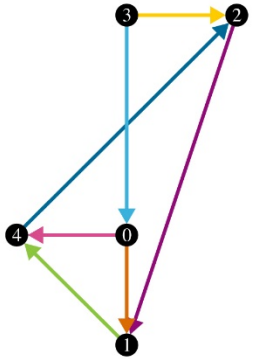
3. Assign forces

4. Solve equilibrium matrix

5. Update force graph

6. Reciprocal form & force graphs





1. Form graph

Connectivity (edges & vertices)

Directed form graph

2. Force graph

Dual graph

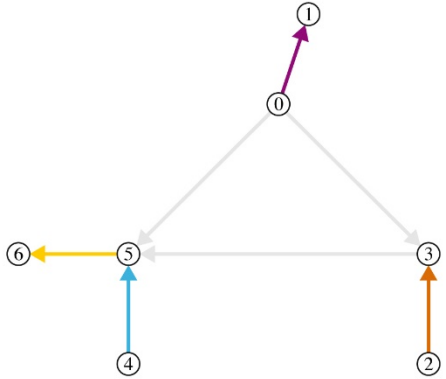
Directed force graph

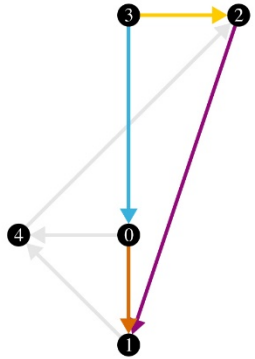
3. Assign forces

4. Solve equilibrium matrix

5. Update force graph

6. Reciprocal form & force graphs





1. Form graph

Connectivity (edges & vertices)

Directed form graph

2. Force graph

Dual graph

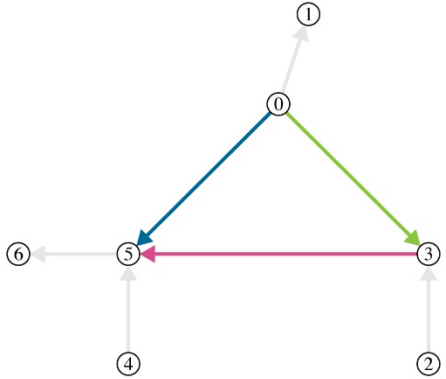
Directed force graph

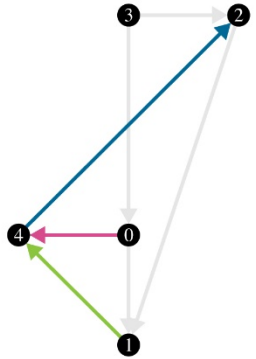
3. Assign forces

4. Solve equilibrium matrix

5. Update force graph

6. Reciprocal form & force graphs





1. Form graph

Connectivity (edges & vertices)

Directed form graph

2. Force graph

Dual graph

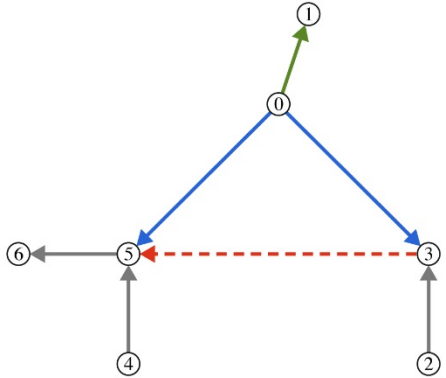
Directed force graph

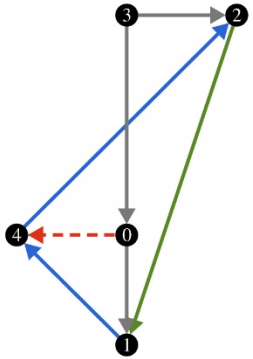
3. Assign forces

4. Solve equilibrium matrix

5. Update force graph

6. Reciprocal form & force graphs





1. Form graph

Connectivity (edges & vertices)

Directed form graph

2. Force graph

Dual graph

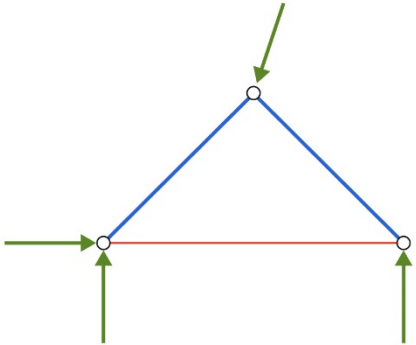
Directed force graph

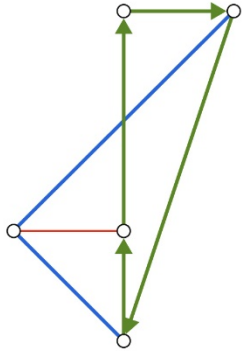
3. Assign forces

4. Solve equilibrium matrix

5. Update force graph

6. Reciprocal form & force graphs





1. Form graph

Connectivity (edges & vertices)

Directed form graph

2. Force graph

Dual graph

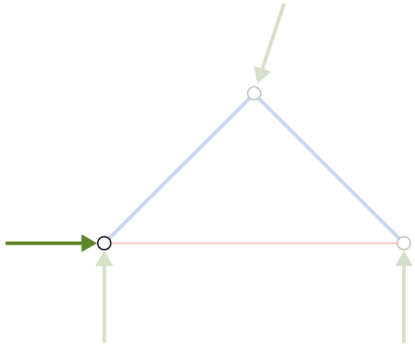
Directed force graph

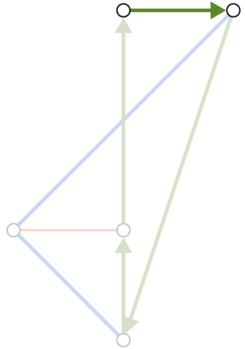
3. Assign forces

4. Solve equilibrium matrix

5. Update force graph

6. Reciprocal form & force graphs





1. Form graph

Connectivity (edges & vertices)

Directed form graph

2. Force graph

Dual graph

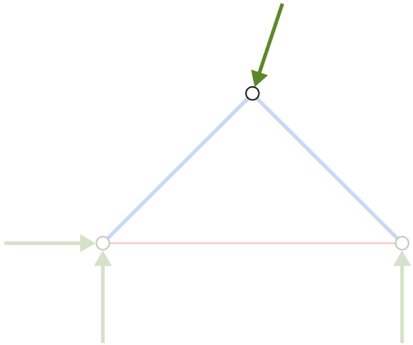
Directed force graph

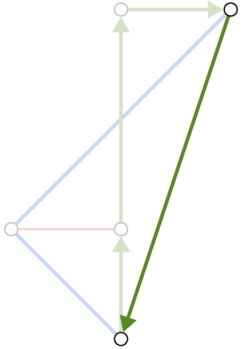
3. Assign forces

4. Solve equilibrium matrix

5. Update force graph

6. Reciprocal form & force graphs





1. Form graph

Connectivity (edges & vertices)

Directed form graph

2. Force graph

Dual graph

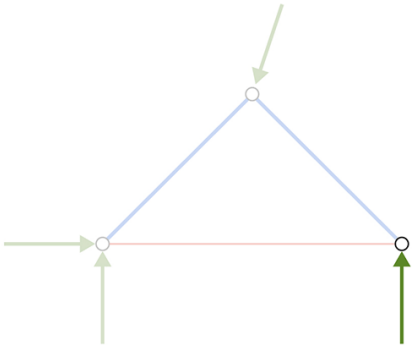
Directed force graph

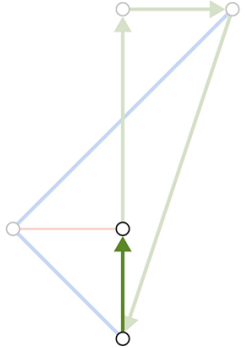
3. Assign forces

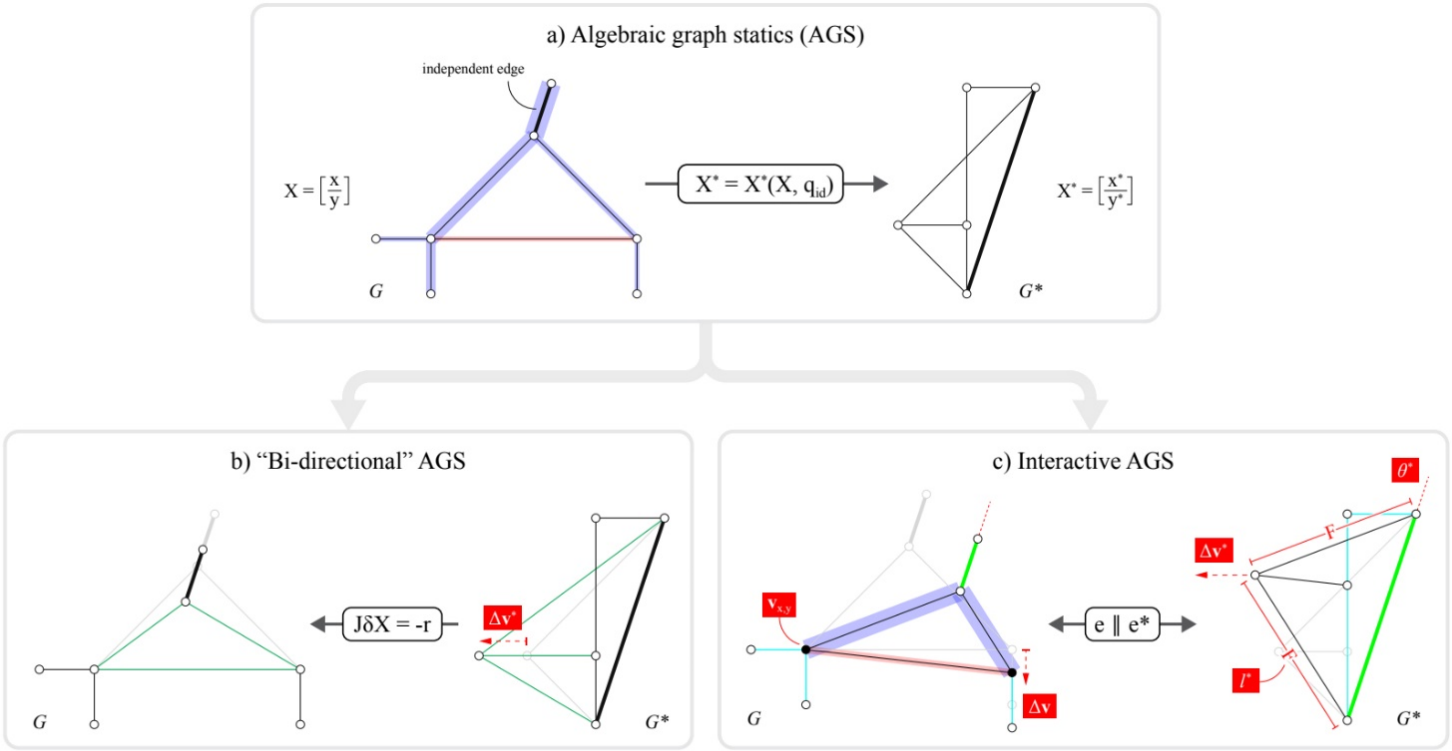
4. Solve equilibrium matrix

5. Update force graph

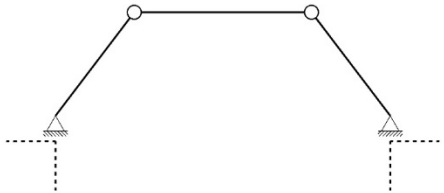
6. Reciprocal form & force graphs






a) Algebraic graph statics (AGS)
independent edge
X = [x; y]
X* = X*(X, q_id)
X* = [x*; y*]
G
G*
b) "Bi-directional" AGS
JδX = -r
Δv*
G
G*
c) Interactive AGS
v_x,y
e ∥ e*
Δv
Δv*
θ*
l*
G
G*

Cables / Arches



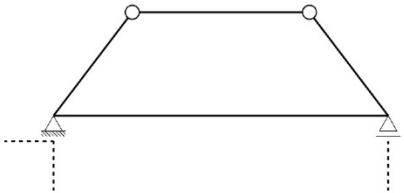
b (bars): 3
r (reactions): 4
n (nodes): 4

b + r < 2*n

3 + 4 < 2*4

Unstable!

Arch-cables



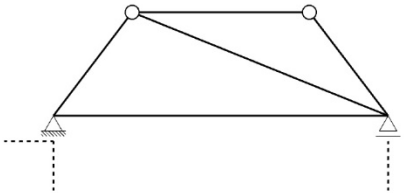
b (bars): 4
r (reactions): 3
n (nodes): 4

b + r < 2*n

4 + 3 < 2*4

Unstable!

Trusses

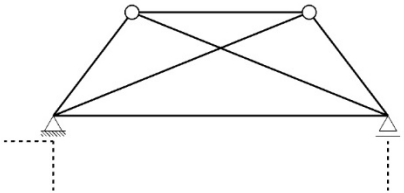


b (bars): 5
r (reactions): 3
n (nodes): 4

b + r = 2*n

5 + 3 = 2*4

Stable!
and
statically determinate!



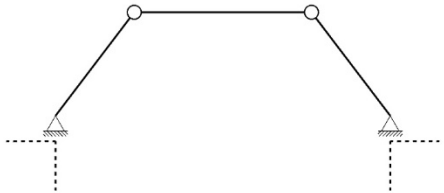
b (bars): 6
r (reactions): 3
n (nodes): 4

b + r > 2*n

6 + 3 > 2*4

Stable!
but
statically indeterminate!

Cables / Arches



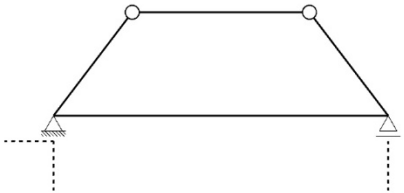
b (bars): 3
r (reactions): 4
n (nodes): 4

b + r < 2*n

3 + 4 < 2*4

Unstable!

Arch-cables



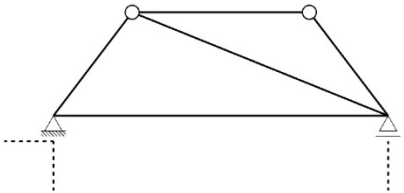
b (bars): 4
r (reactions): 3
n (nodes): 4

b + r < 2*n

4 + 3 < 2*4

Unstable!

Trusses

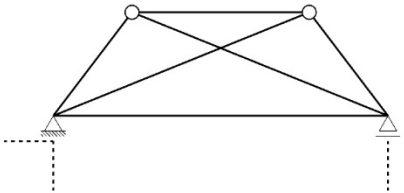


b (bars): 5
r (reactions): 3
n (nodes): 4

b + r = 2*n

5 + 3 = 2*4

Stable!
and
statically determinate!

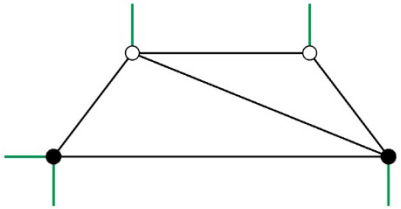


b (bars): 6
r (reactions): 3
n (nodes): 4

b + r > 2*n

6 + 3 > 2*4

Stable!
but
statically indeterminate!



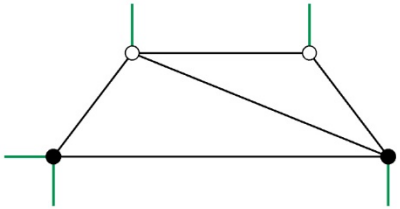
b (bars): 5
e (external forces): 5
n (nodes): 4

DOF = b + e - (2*n)

5 + 5 - (2*4) = 2

We must define the two external loads!

We have the freedom to define the two external loads!



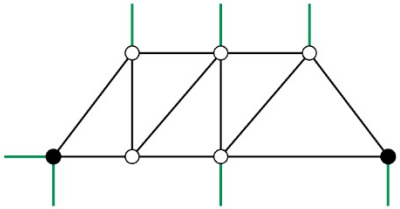
b (bars): 5
e (external forces): 5
n (nodes): 4

DOF = b + e - (2*n)

5 + 5 - (2*4) = 2

We must define the two external loads!

We have the freedom to define the two external loads!



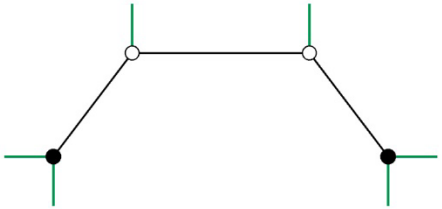
b (bars): 11
e (external forces): 7
n (nodes): 7

DOF = b + e - (2*n)

11 + 7 - (2*7) = 4

We must define the four external loads!

We have the freedom to define the four external loads!



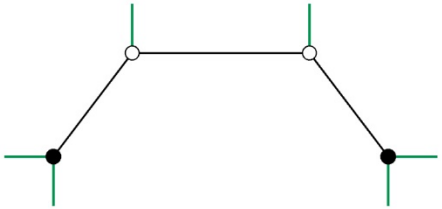
b (bars): 3
e (external forces): 6
n (nodes): 4

DOF = b + e - (2*n)

3 + 6 - (2*4) = 1

We must define one of the external loads!

We have the freedom to define one of the external loads but not the other one!



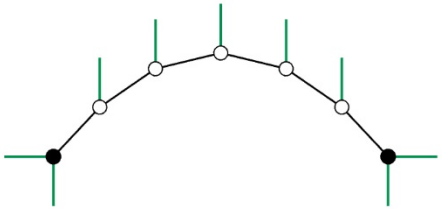
b (bars): 3
e (external forces): 6
n (nodes): 4

DOF = b + e - (2*n)

3 + 6 - (2*4) = 1

We must define one of the external loads!

We have the freedom to define one of the external loads but not the other one!



b (bars): 6
e (external forces): 9
n (nodes): 7

DOF = b + e - (2*n)

6 + 9 - (2*7) = 1

We must define one of the external loads!

We have the freedom to define only one of the external loads but not the other ones!

linear programming
(Grasshopper)

1. variable(s)

parameters

2. Loops

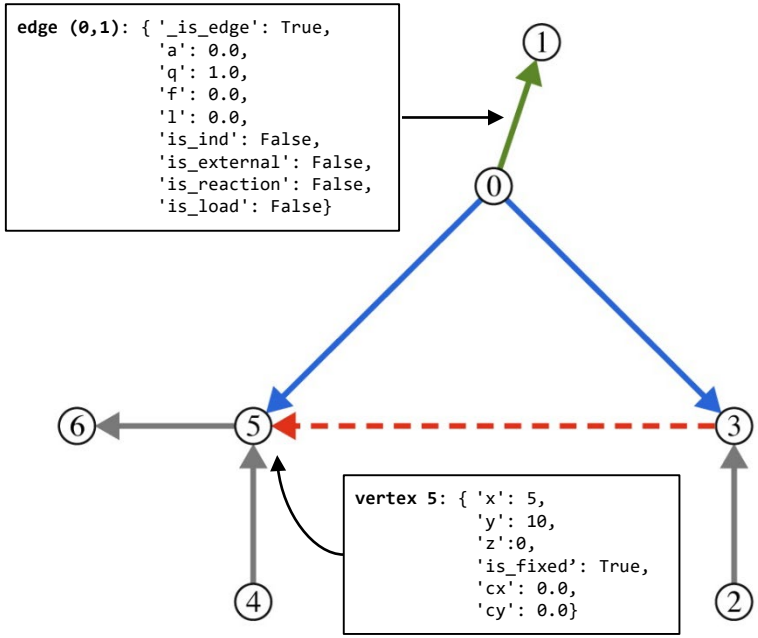
3. Conditional

4. Functions

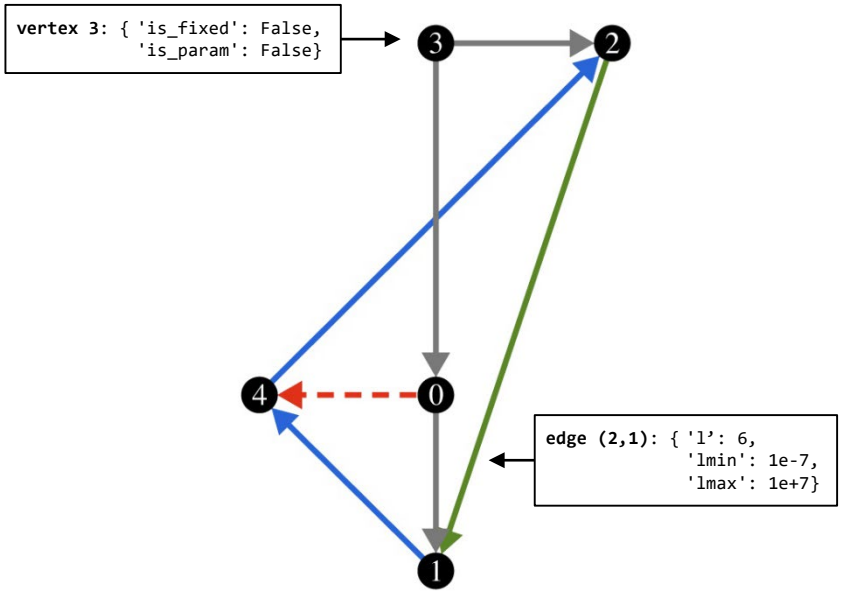
components

5. Object

form data



force data



COMPAS

Open-source, Python-based framework for computational research and collaboration in architecture, engineering and digital fabrication