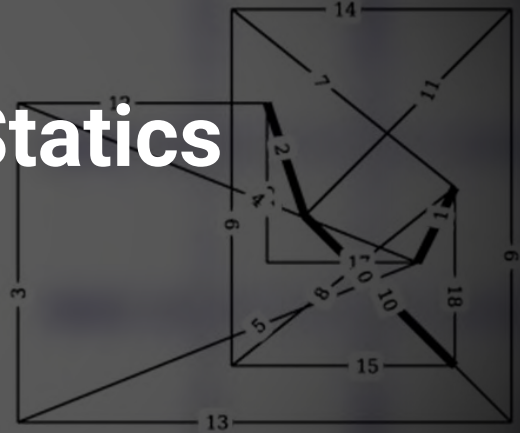
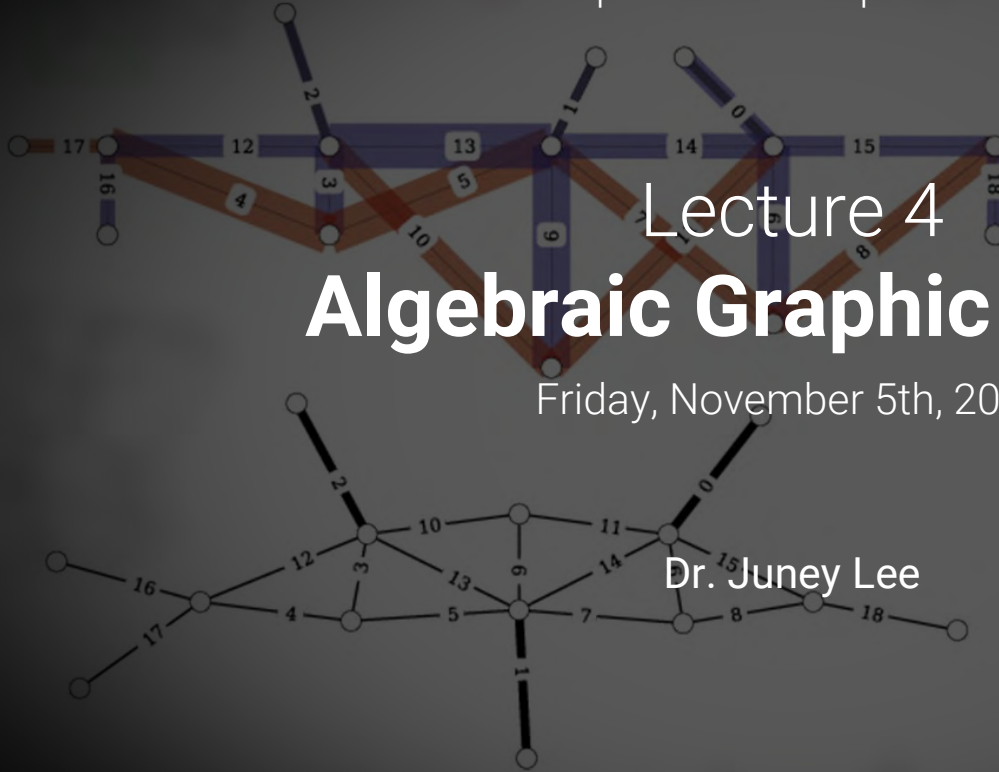


Lecture 4

Algebraic Graphic Statics

Friday, November 5th, 2021

Dr. Juney Lee



$$P = 24,000 \text{ lb}$$

$$= \underline{\underline{24^k}}$$

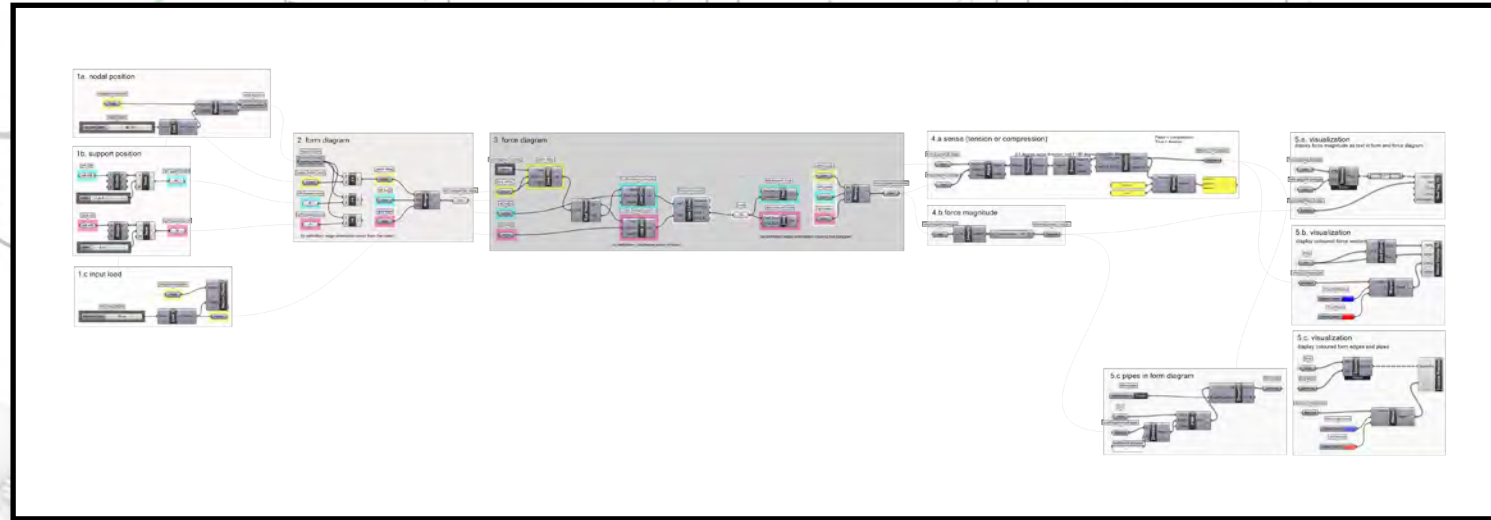

form diagram

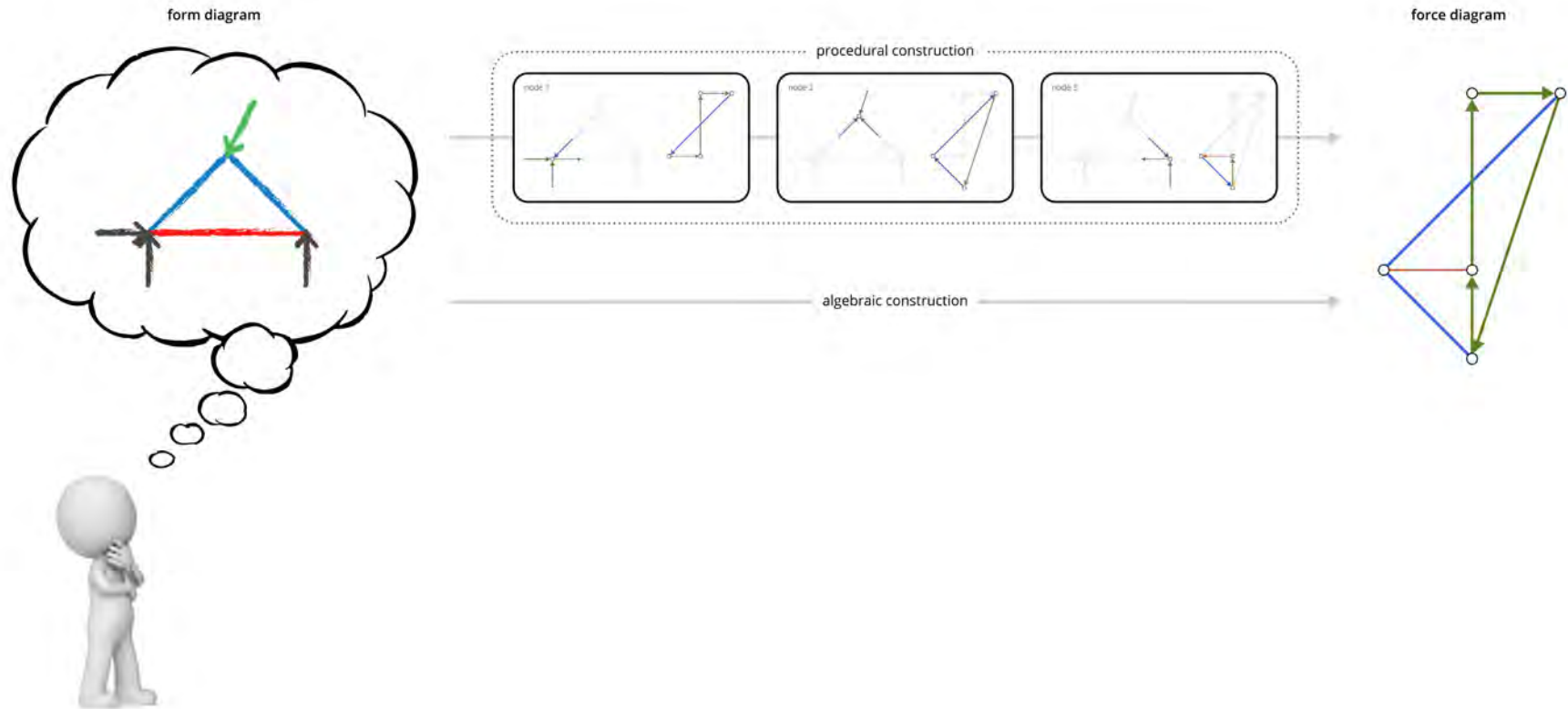


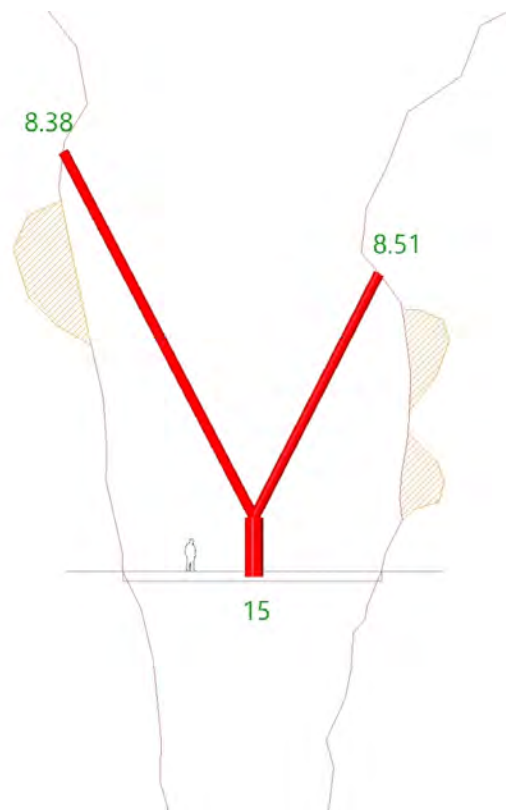
form diagram

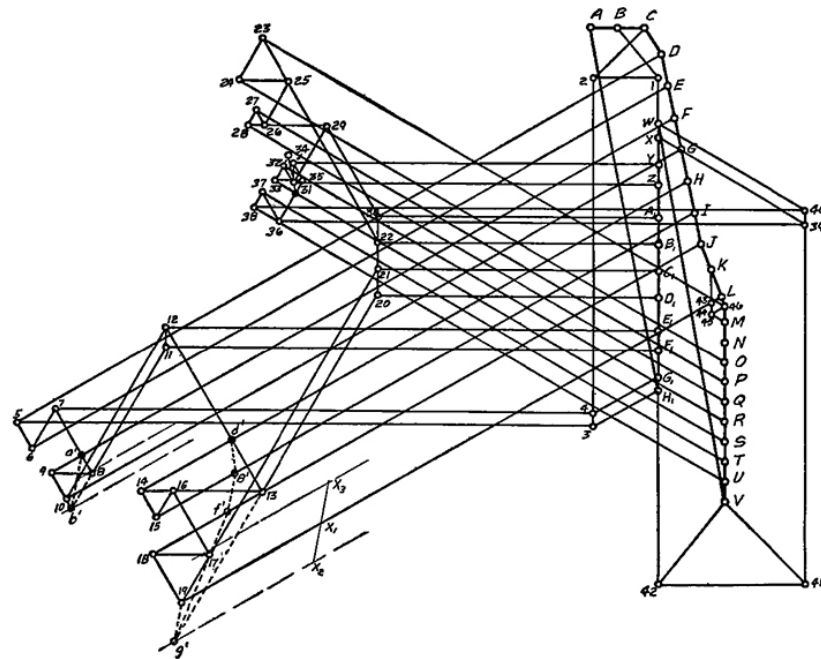
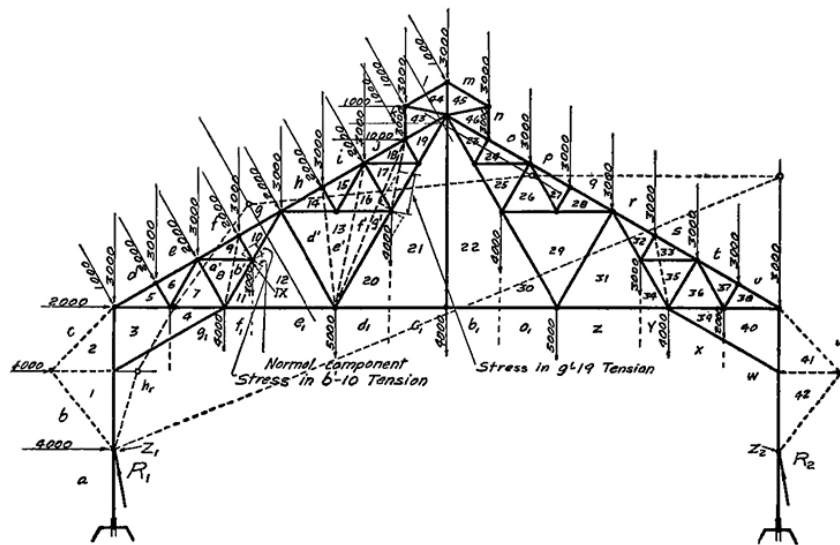
procedural construction

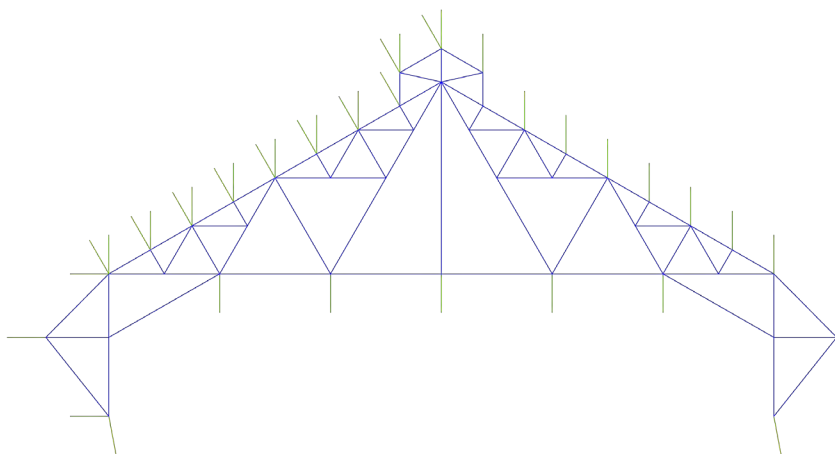
force diagram

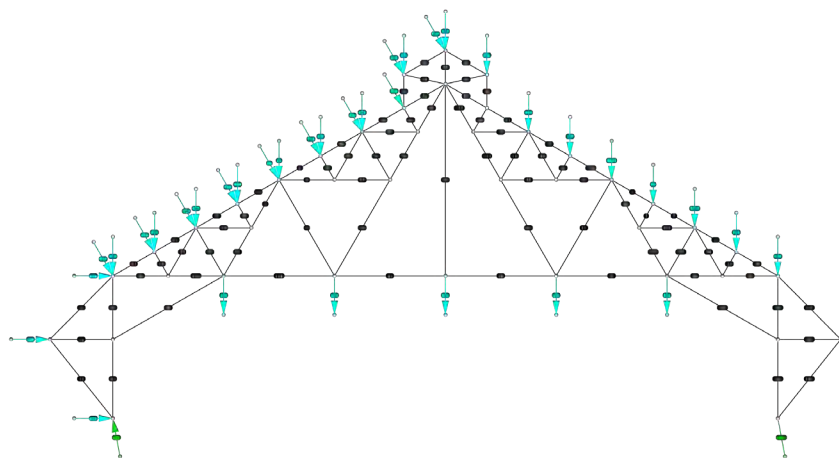


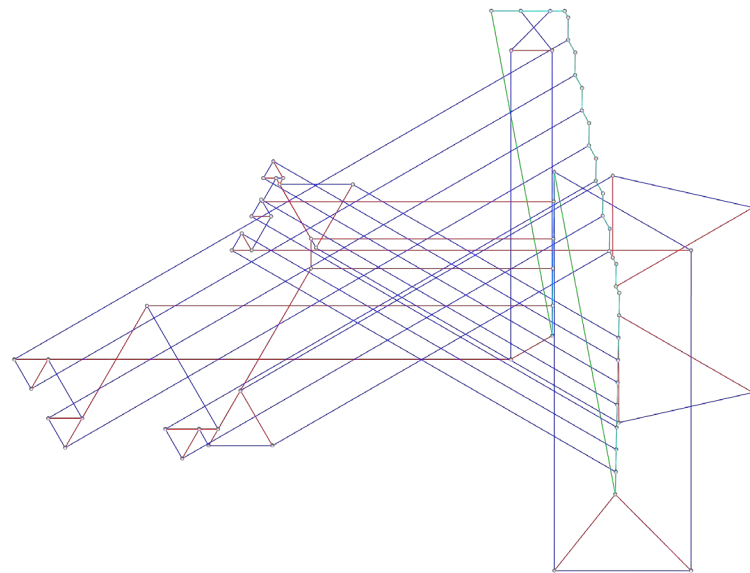
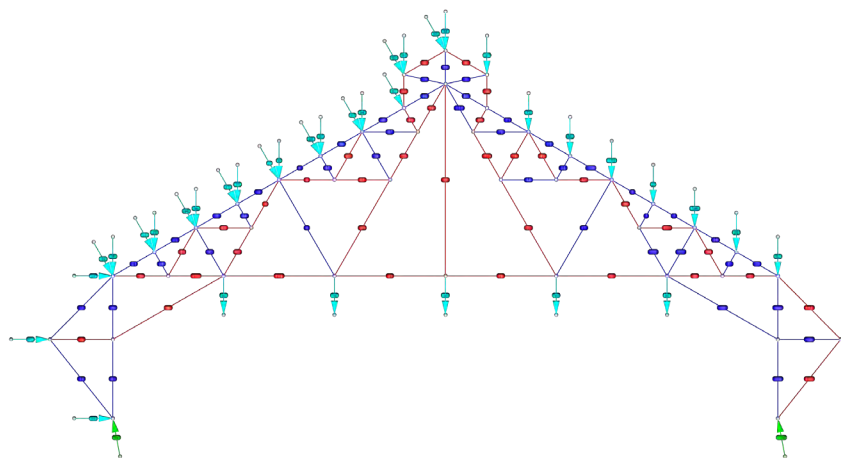


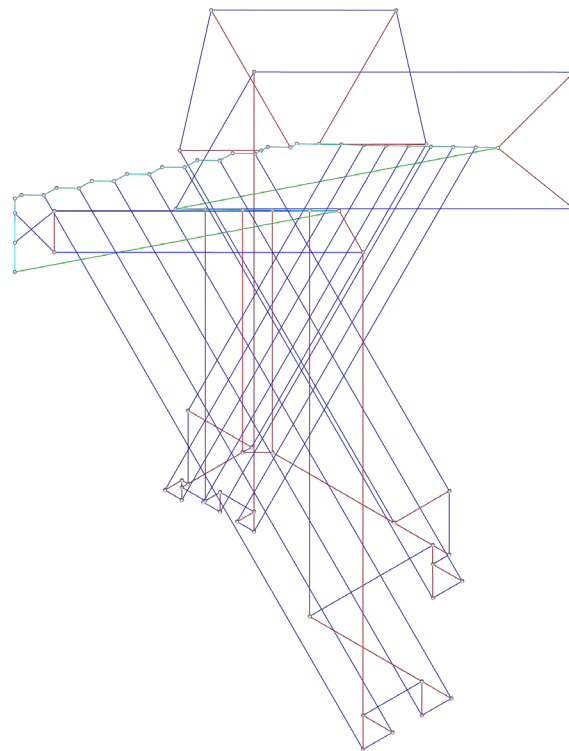
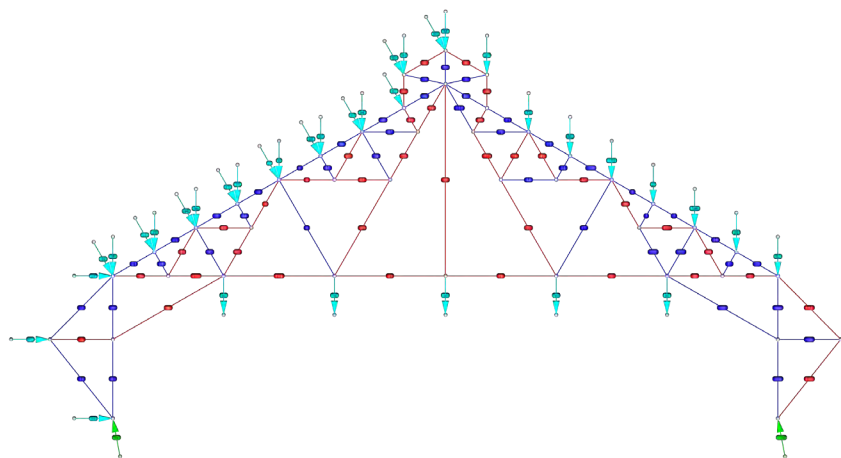


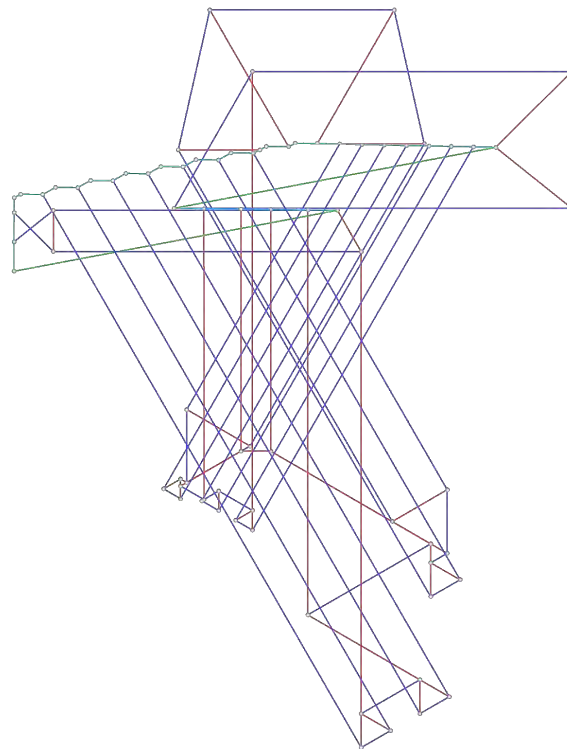
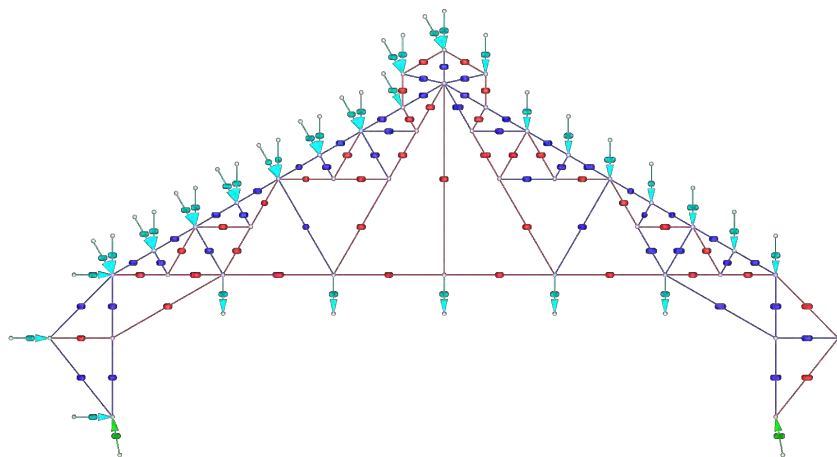


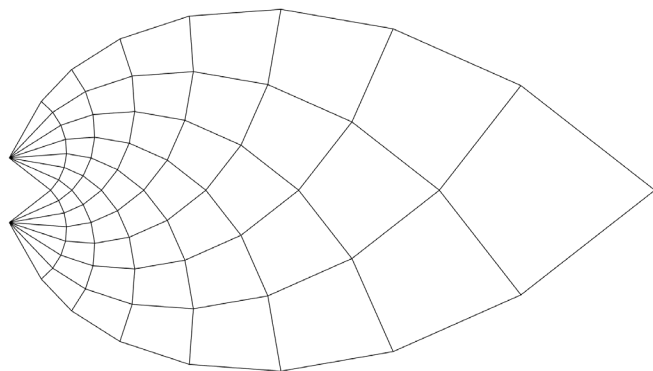


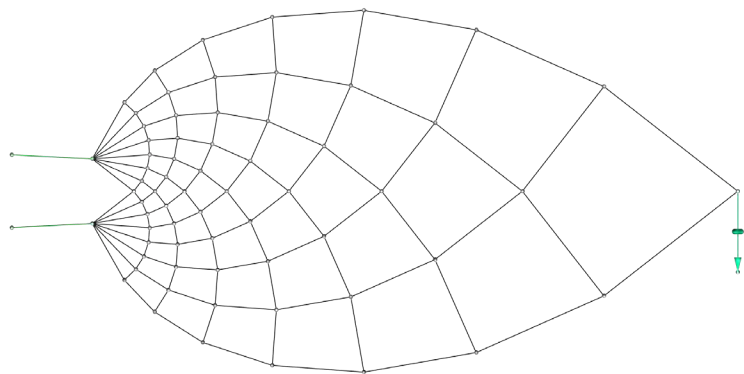


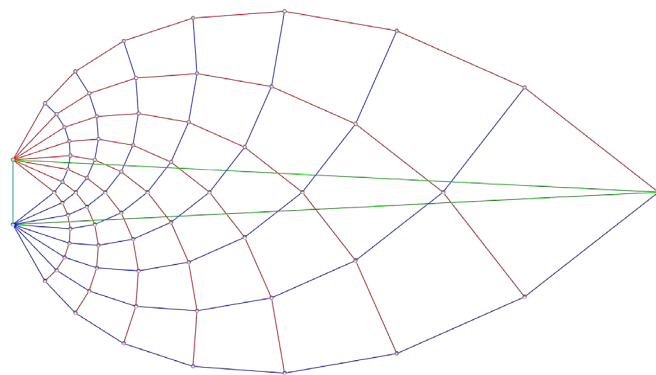
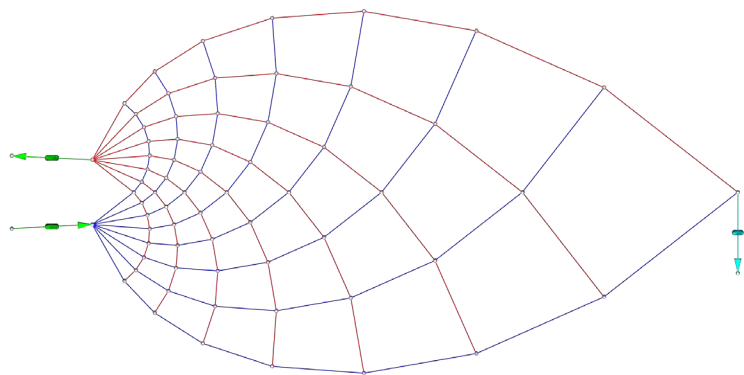


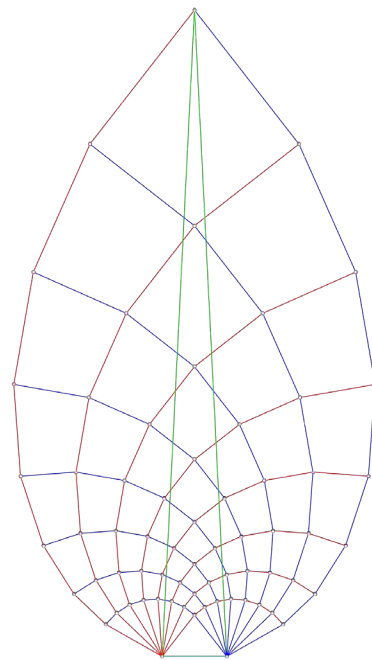
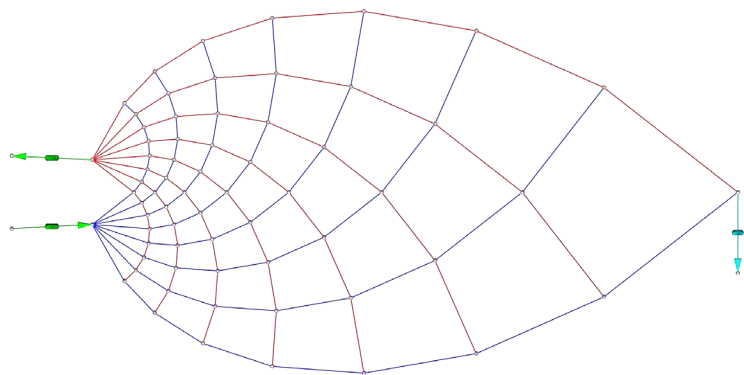


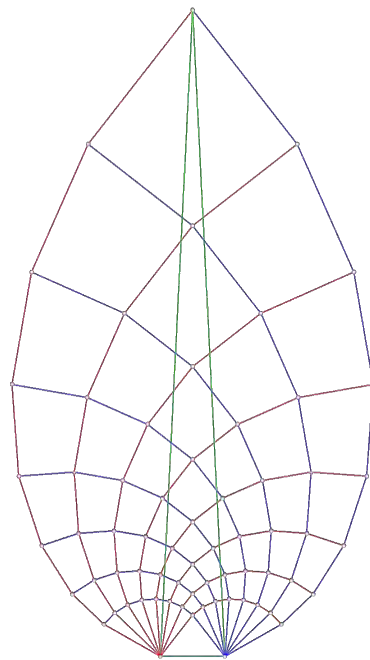
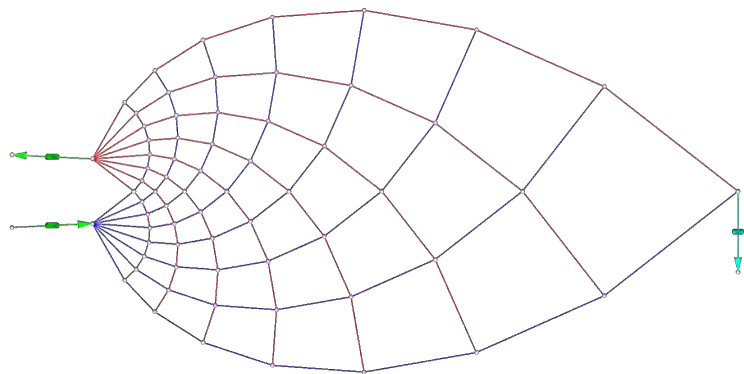


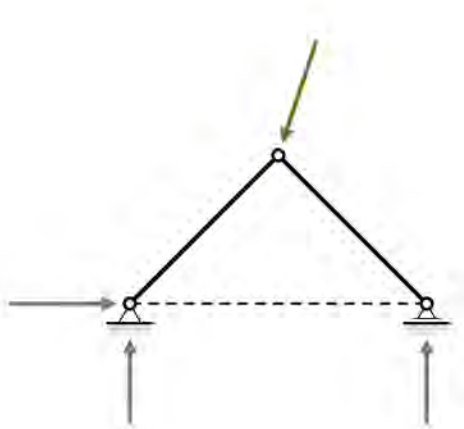




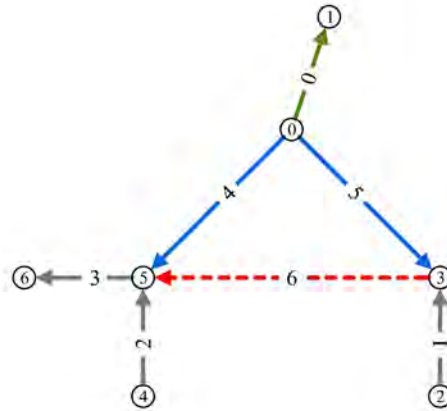




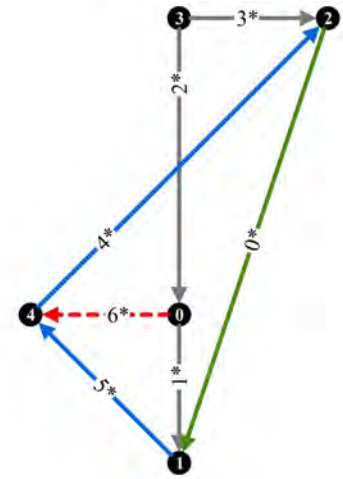




Form diagram



Form diagram as a graph



Force diagram

Get topology and connectivity

Solve via equilibrium matrix

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

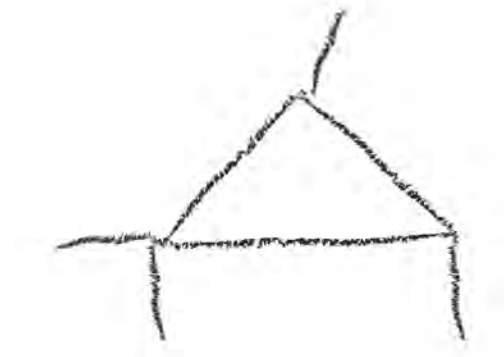
Directed force graph

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

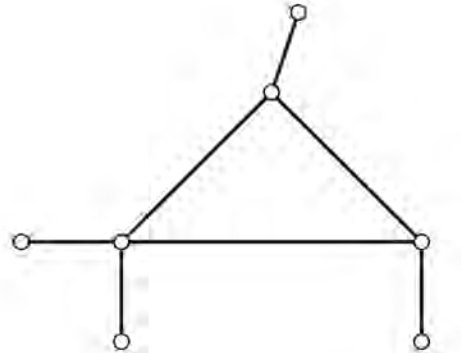
Directed force graph

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

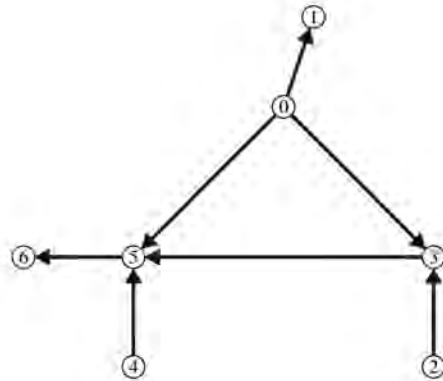
Directed force graph

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

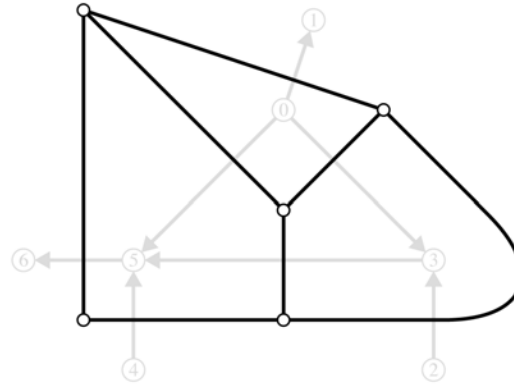
Directed force graph

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](#)

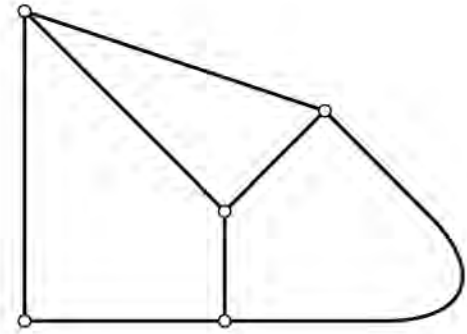
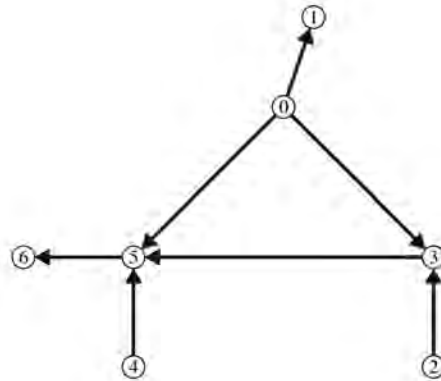
Directed force graph

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

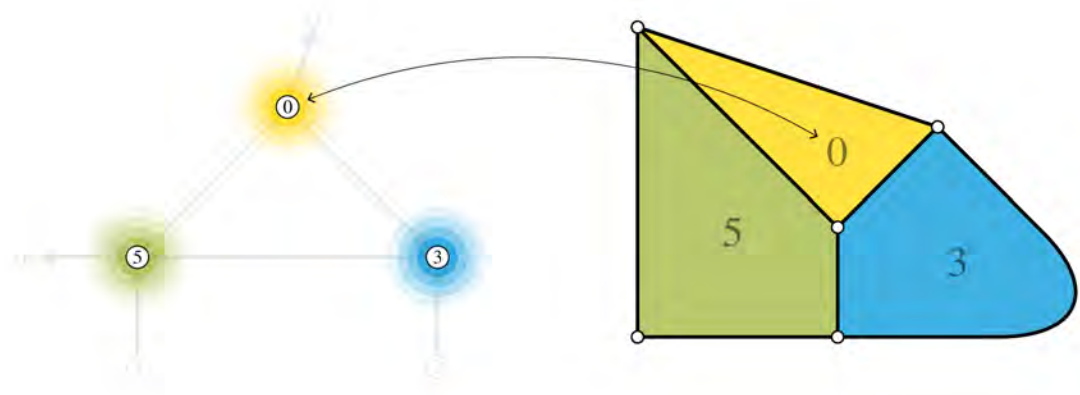
Directed force graph

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

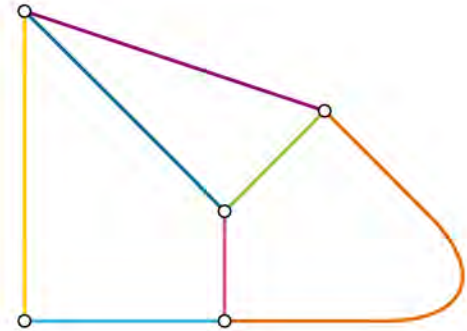
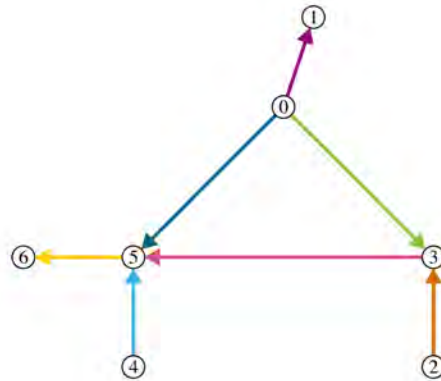
Directed force graph

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

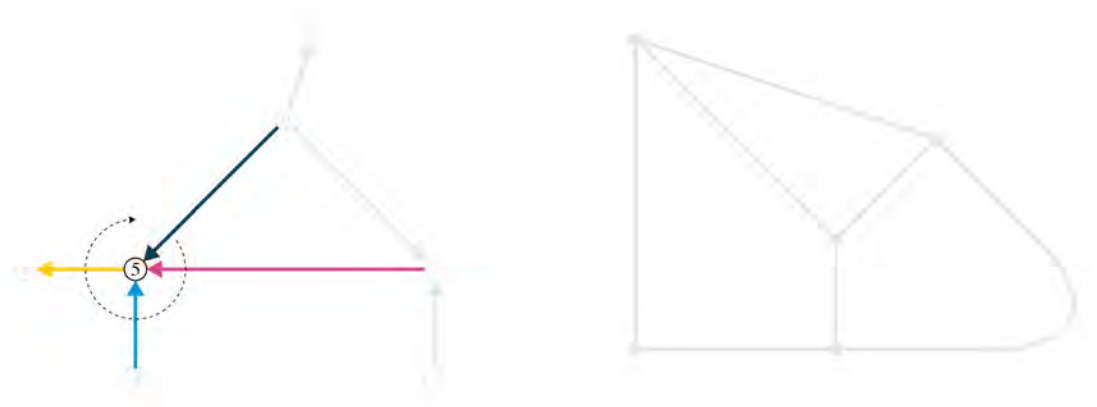
Directed force graph

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

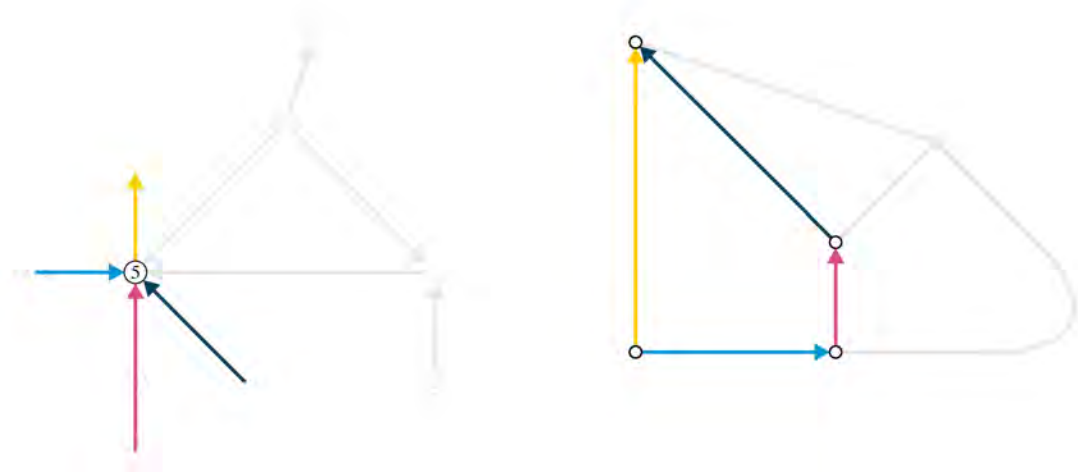
Directed force graph

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

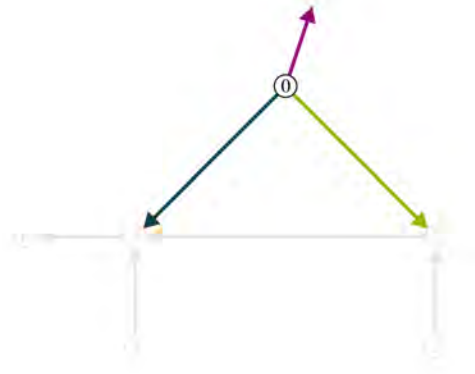
[Directed force graph](#)

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

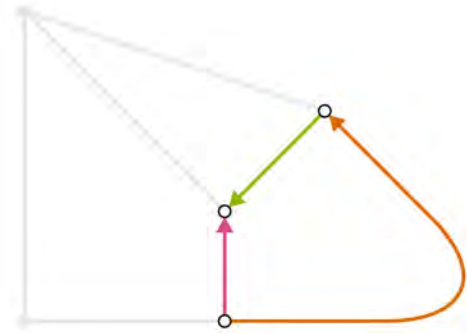
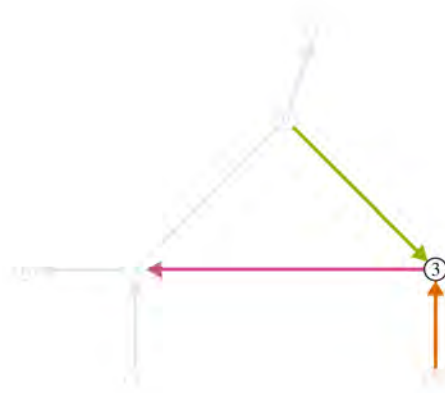
Directed force graph

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

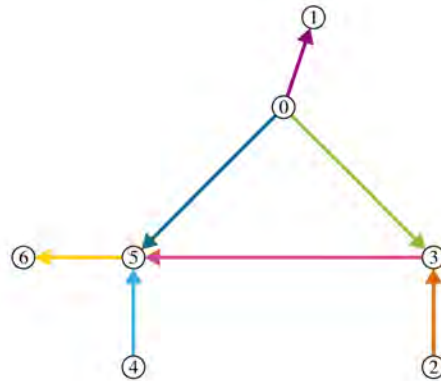
[Directed force graph](#)

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

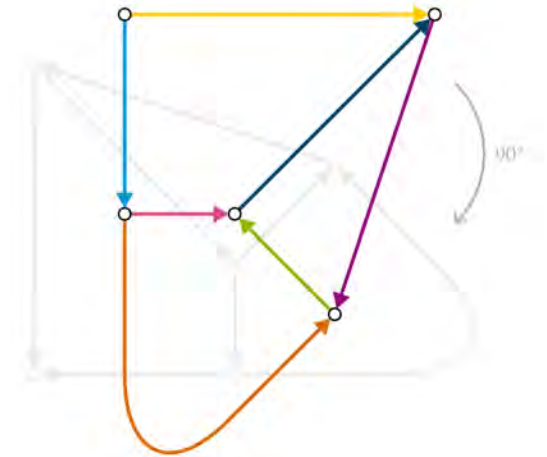
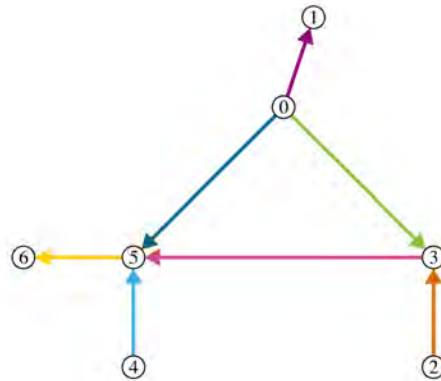
Directed force graph

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

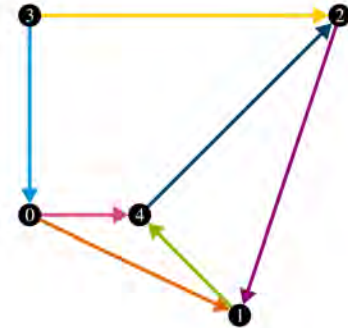
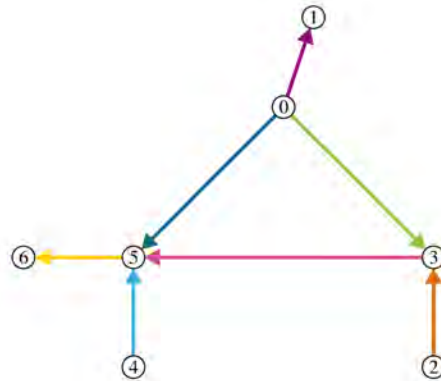
Directed force graph

3. Assign forces

4. Solve equilibrium matrix

5. Update force graph

6. Reciprocal form & force graphs



1. Form graph

Connectivity (edges & vertices)

Directed form graph

2. Force graph

Dual graph

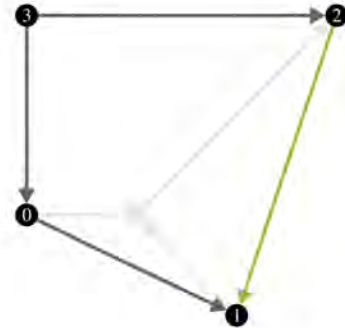
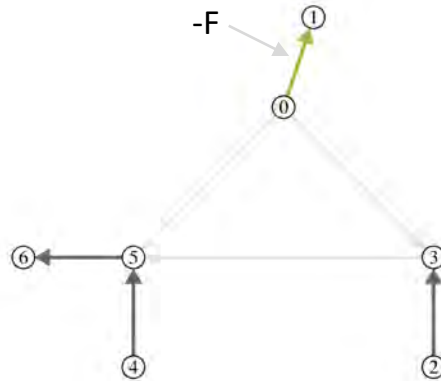
Directed force graph

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

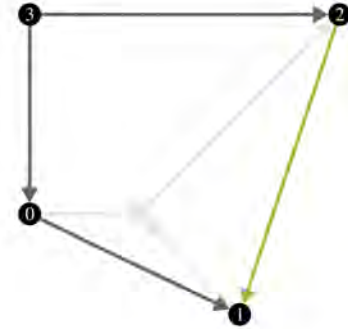
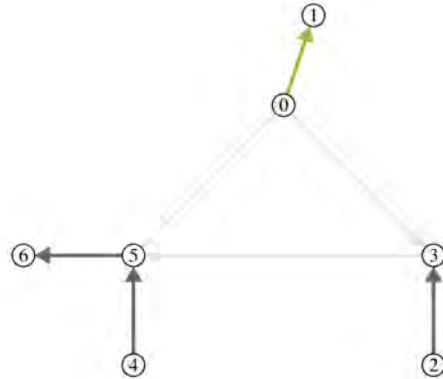
4. Solve equilibrium matrix

5. Update force graph

6. Reciprocal form & force graphs

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

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



Closed polygon constraint

$$\begin{cases} C_i u^* = 0 \\ C_i v^* = 0 \end{cases}$$

Parallel constraint

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

$$\begin{cases} C_i Uq = 0 \\ C_i Vq = 0 \end{cases}$$

equilibrium matrix

$$Aq = 0, \quad A = \begin{bmatrix} C_i U \\ C_i V \end{bmatrix}$$

1. Form graph

Connectivity (edges & vertices)

Directed form graph

2. Force graph

Dual graph

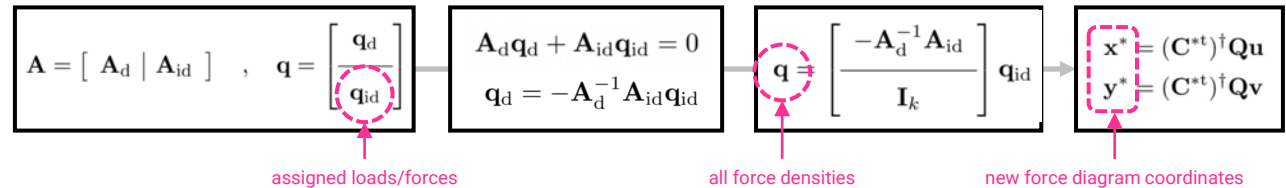
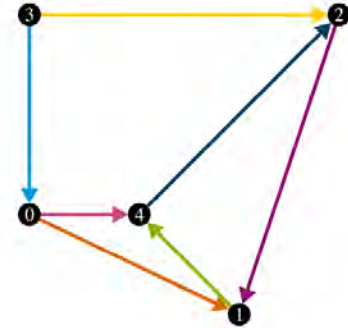
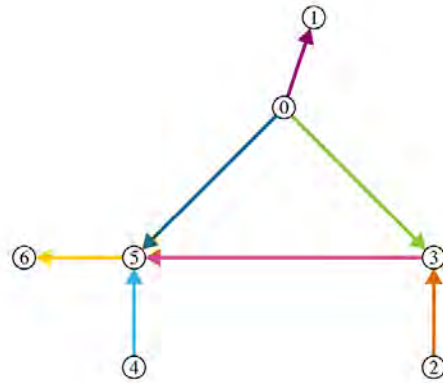
Directed force graph

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

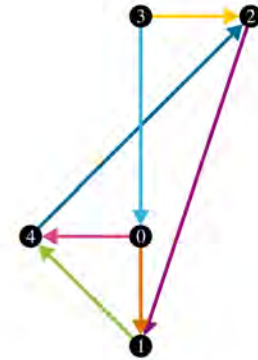
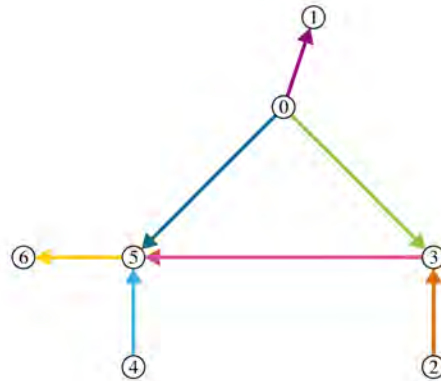
Directed force graph

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

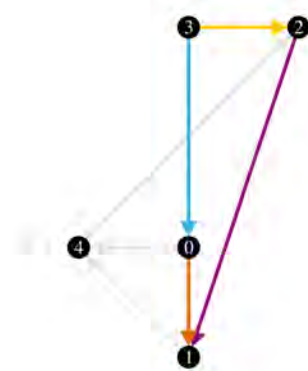
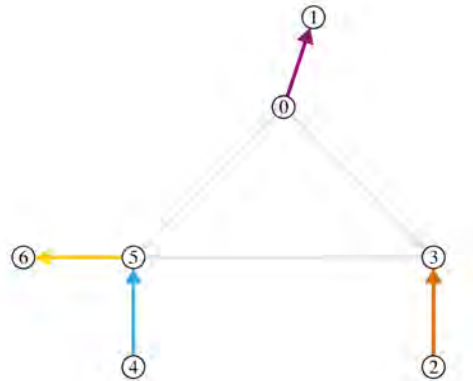
Directed force graph

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

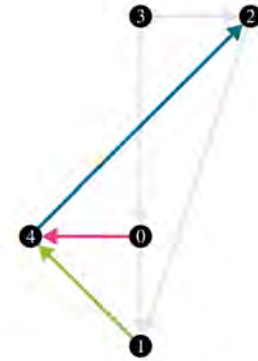
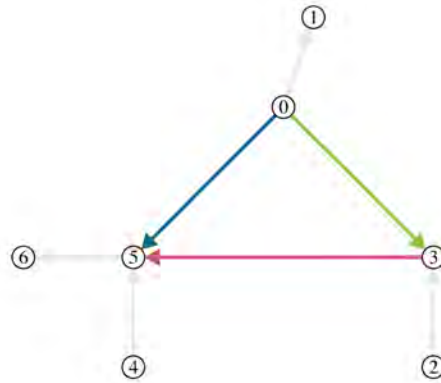
Directed force graph

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

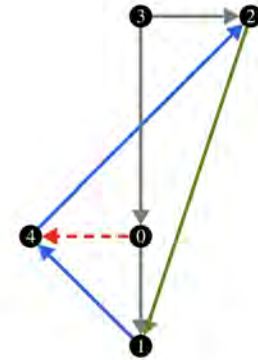
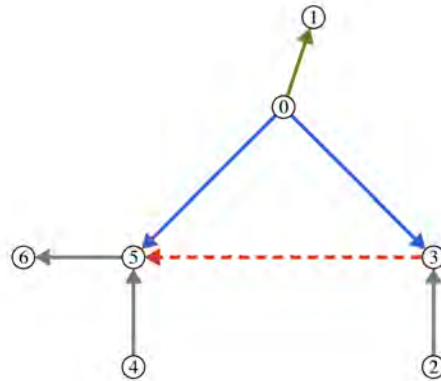
Directed force graph

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

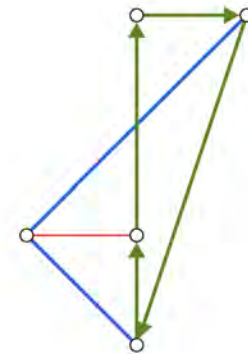
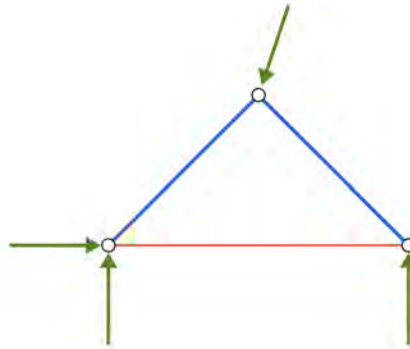
Directed force graph

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

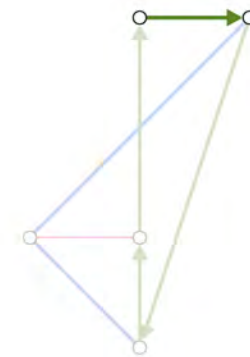
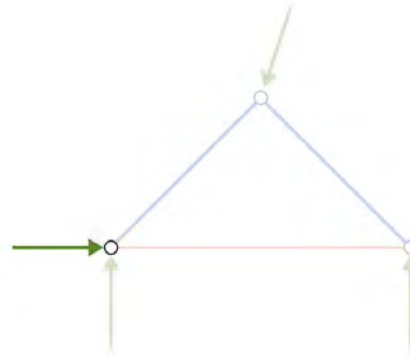
Directed force graph

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

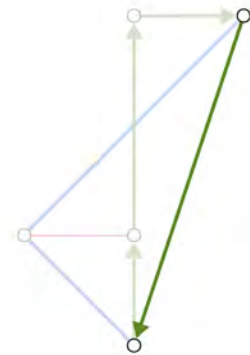
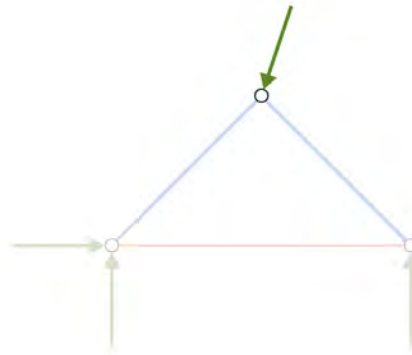
Directed force graph

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

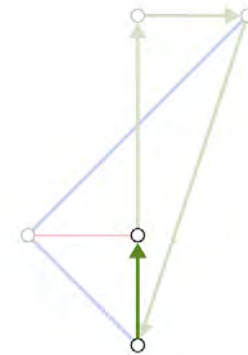
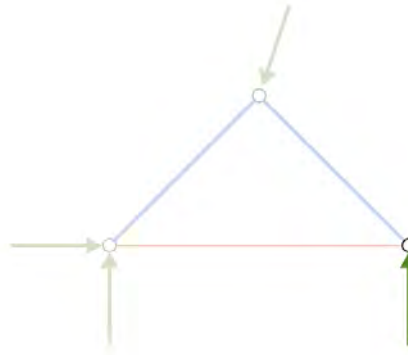
Directed force graph

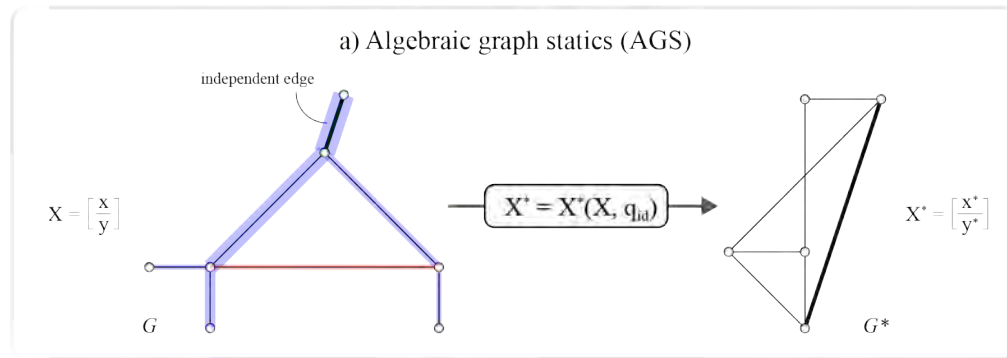
3. Assign forces

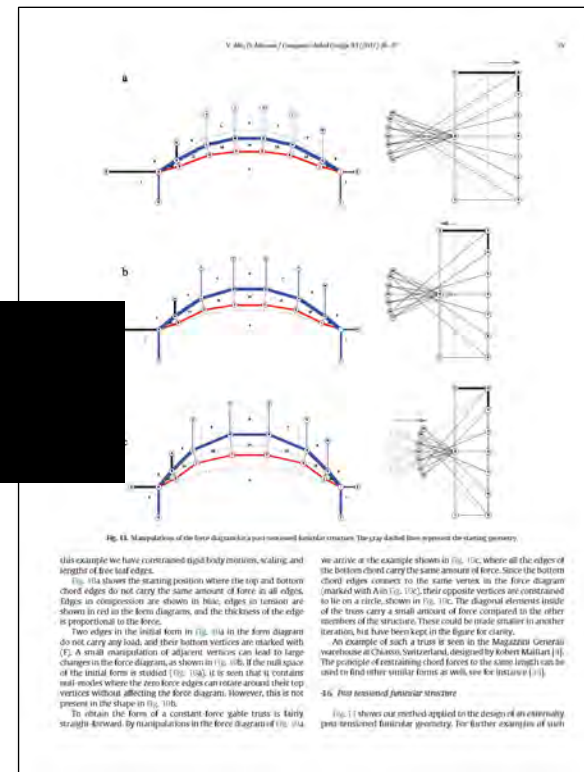
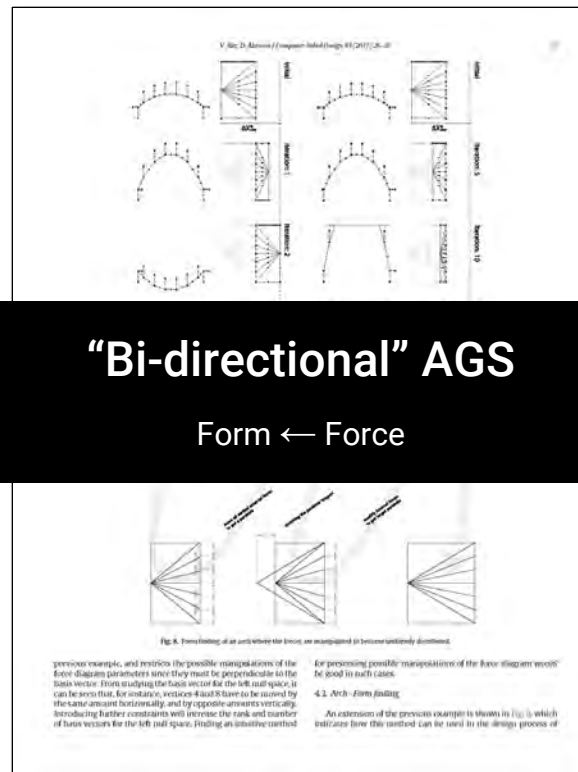
4. Solve equilibrium matrix

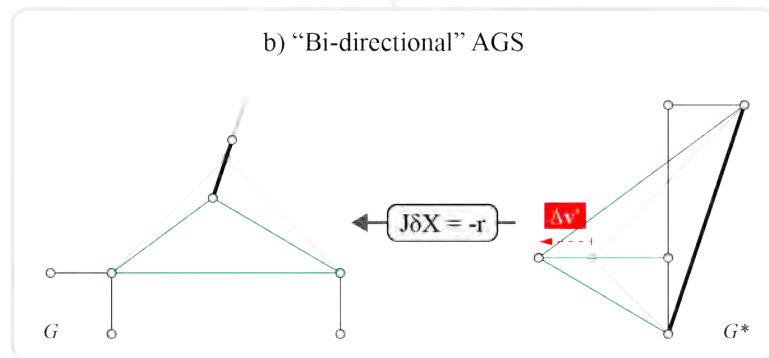
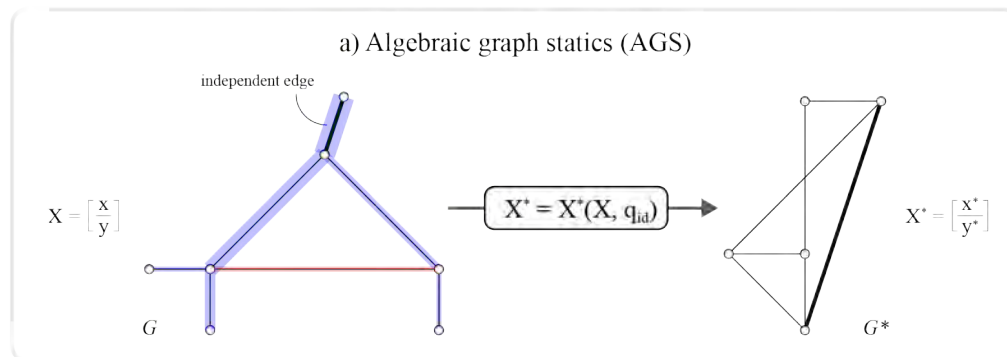
5. Update force graph

6. Reciprocal form & force graphs











Computational Journal of Structures 2021
 Computational 10 (1) (2021)
 Switzerland, September 16-18 2021

An interactive implementation of algebraic graphic statics for geometry-based teaching and design of structures

Ricardo Avelino, Junyi Lee, Tom Van Mele, Philipp Block
 ETH Zurich, Institute of Technology in Architecture, B204 Research Group
 Stefano-Franscini-Platz 1, HIB E 45, 8093 Zurich, Switzerland
 (maia, junyi.lee, van.mele, block)@arch.ethz.ch

Abstract

This paper presents an interactive implementation of graphic statics, which can be integrated into a CAD environment. Graphic statics is a well-known design and analysis method for two-dimensional discrete structures that relies on geometrical rather than analytical representation of the relation between the structure's geometry and the equilibrium of its internal forces. The method was formalised in the 19th century, but slowly disappeared from structural engineering practice over the 20th century. Recent developments have introduced Algebraic Graph Statics (AGS), which formalises the relationship between the graph representations of the reciprocal form and force diagrams in geometry-based design. AGS and its extensions enable automatic construction of force diagrams from diagrams, and allow a few basic modifications of the force diagram from which the diagram is updated. This paper builds on the previous work of AGS by implementing a two-dimensional workflow allowing users to impose various constraints, and perform geometric transformations in either the form or force diagram from which the other is automatically updated iteratively. The presented implementation of interactive AGS provides a natural feedback and to harness the advantages of traditional graphic statics for geometry-based design of structures.

1 Introduction

Recent research has demonstrated how the principles of graphic statics can be combined with digital tools to create interactive drawings that provide visual feedback to the user in real-time. Such interactive implementations of graphic statics have not only introduced new and effective teaching tools for structural design [1], but also enabled advanced research [2, 3]. Despite its increasing benefits, interactive graphic statics drawings still have some major drawbacks. The tedious and time-consuming process of constructing drawings in a procedural manner requires previous knowledge and experience with graphic statics [7, 4]. More importantly, each drawing is representative of just one instance of a structure, meaning that topological changes to the design require a complete redraw of the form and force diagrams. Algebraic Graph Statics (AGS) introduced an algebraic method of formalising the reciprocal relationship between the form and force diagrams, which enables automatic construction of force diagrams from graph representations of form diagrams given by the user [9]. “Bio-inspired” AGS extended the method, allowing geometric transformations of a force diagram that results in an automatic reconfiguration of the corresponding form diagram [10]. Other methods for generating reciprocal diagrams have been presented using Airy stress functions [11] and projective geometry [12], but these limit to self-stressed structures and add 3D polyhedral geometries into the workflow.

This paper presents a computational implementation and extension of previous research in AGS in an interactive design workflow. In order to create a fluid user experience while maintaining a robust back-end of solvers, various rules and constraints of graphic statics construction are explicitly defined, formalised and incorporated in an integrated computational pipeline using the COM4S framework [13]. The examples presented in this paper demonstrate how the proposed implementation can be used to maximise the inherent benefits of graphic-statics-based structural design explanations in a smooth and intuitive manner through controlled modifications, while minimising the need for external construction of form and force diagrams.

Conceptual Design of Structures 2021

- Compute reciprocal force diagram G^* and inherit default constraints from form diagram boundary conditions.
- Interactively modify and add constraints to the form diagram.
- Interactively modify and add constraints to the force diagram.
- Solve equilibrium by parallelisation.

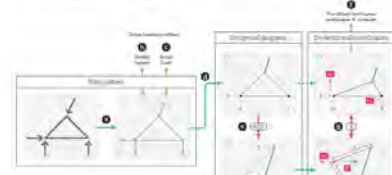


Fig. 3 Imposing default constraints from boundary conditions

- fix the vertices in the form diagram with an externally applied load are constrained to remain on the line of action of the load,
- edges representing the reaction forces have their orientations fixed in both diagrams, and,
- edges representing the externally applied loads have their orientations fixed in both diagrams, and their lengths fixed in the force diagram.

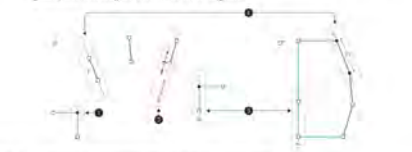


Fig. 3 Imposing default constraints from boundary conditions

Conceptual Design of Structures 2021

As introduced in Section 3.2, larger force magnitudes can be imposed on the form diagram, which, in consequence, will be reflected as larger lengths in the force diagram. Therefore, in order to assign equally distributed vertical loads to the arch, a target force -1.0 is applied to the loaded edges in the form diagram, or equivalently, target lengths of 1.0 to the loaded edges in the force diagram (Fig. 4b). With the default constraints from the boundary conditions already imposed, the dual algorithm is performed updating both form and force diagrams. The resulting form and force diagrams in Fig. 4b now show the “correct”, parabolic arch subjected to equally distributed vertical loads. Fig. 4b corresponds to one of the possible parabolic arch solutions, which depend on the magnitude of the unconstrained horizontal reactions, which in this case is equal 3.81 after the interactive parallelisation. Controlling this horizontal magnitude to alter the arch height will be discussed in the next example.

4.2 Form and force diagram modifications

The second example shows how the geometry of the arch from Fig. 4b can be modified through controlled translations of the vertices of both diagrams. After the transformations, the new diagrams are then parallelised (Section 3.3.2). Fig. 5 shows two possible manipulations on the force diagram. In Fig. 5a, the three vertices on the left side of the force diagram are dragged to decrease the magnitude of the internal forces, resulting in a taller arch. In Fig. 5b, the vertices are moved further to the right, such that the form diagram results in a geometry that corresponds to a funicular cable in tension.

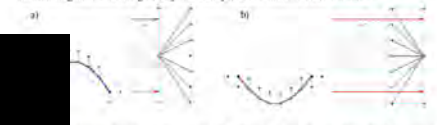


Fig. 5 Moving the three vertices on the left side of the force diagram G^* to the right, which results in reduced internal forces and therefore a taller arch in form diagram G ; b) further movement of three vertices until the forces flip from compression to tension, with form diagram G becoming a funicular cable.

two examples of modifications in the vertices of the form diagram: in Fig. 6a, an internal arch is moved up and its coordinate is fixed, which constrains the arch to pass through this point. The target force magnitude constraints still apply, i.e., the loading case is equally distributed. After the foundation of the internal vertex in the form diagram, both diagrams are updated (Fig. 6a). Similarly, in Fig. 6b, the right support of the arch is moved up. After this modification, both diagrams are updated while respecting all imposed constraints, resulting in an arch with random loads applied to it, but with uneven vertical force reactions due to the different support heights.

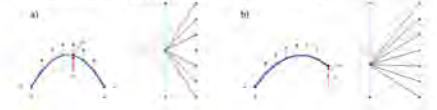
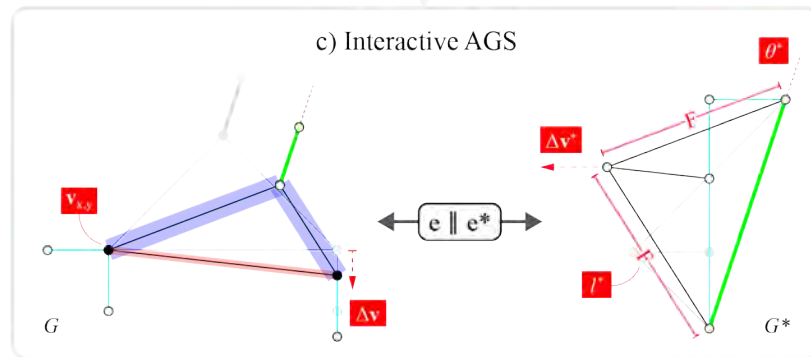
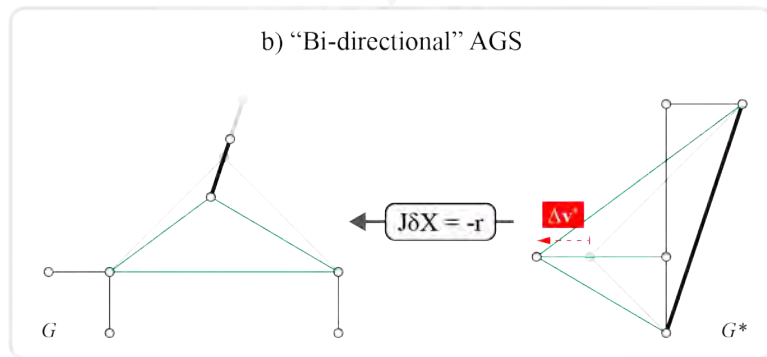
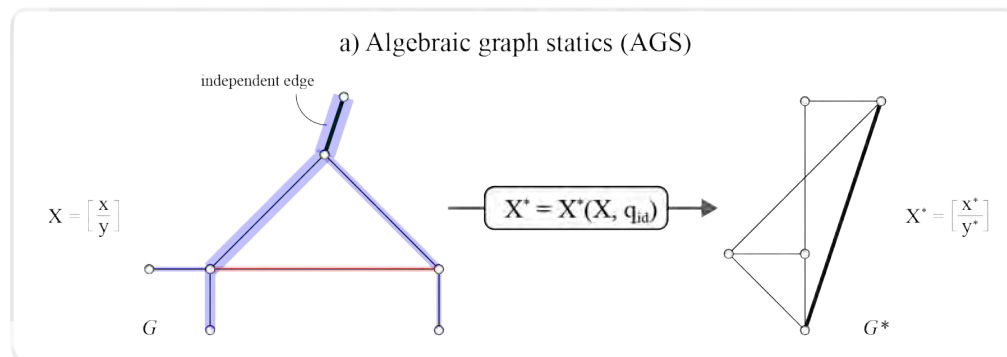
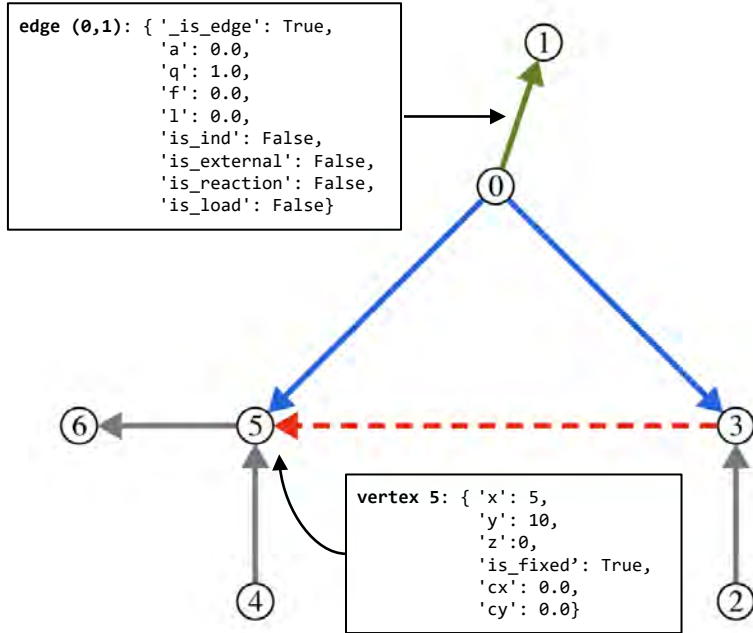


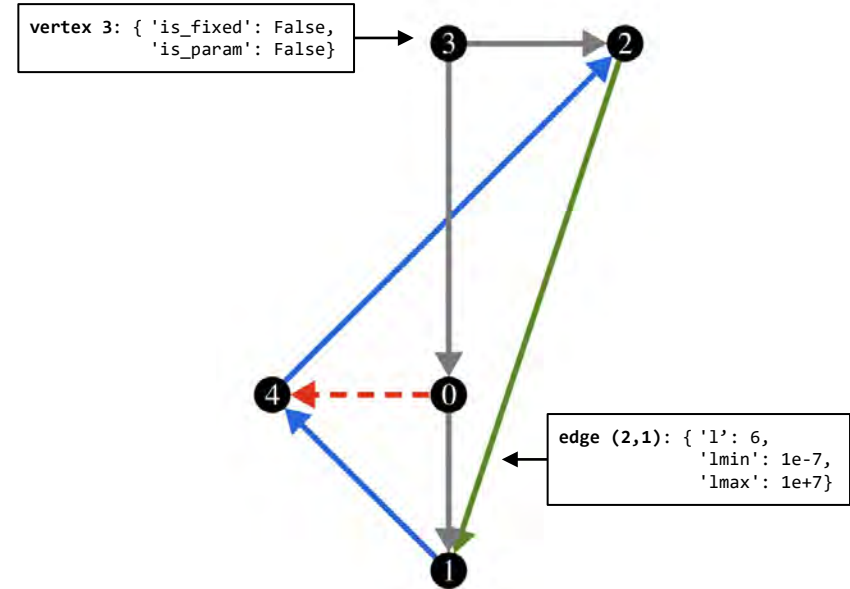
Fig. 6 a) Update in form G and force G^* diagram generated by moving, and constraining an internal vertex of the arch, controlling its structural height; b) update in form G and force diagram G^* generated by moving one of the supports.



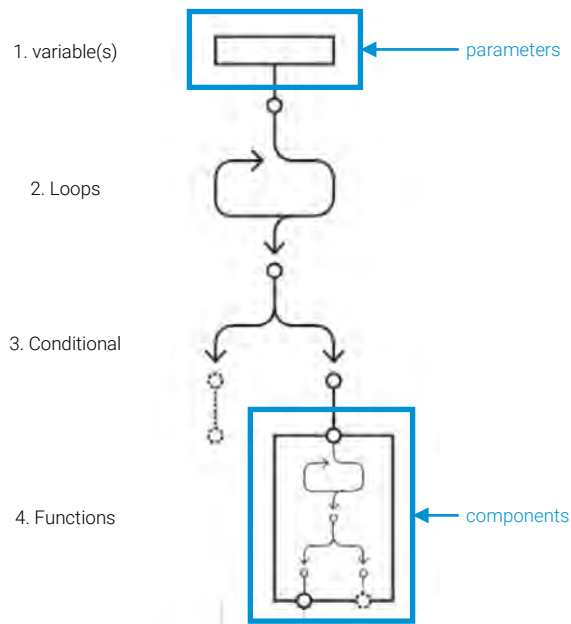
form diagram data



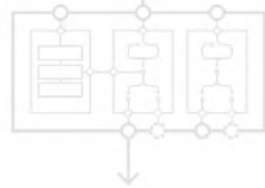
force diagram data



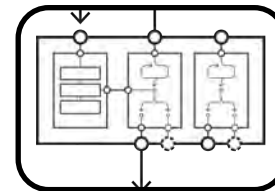
**linear
programming**
(Grasshopper)



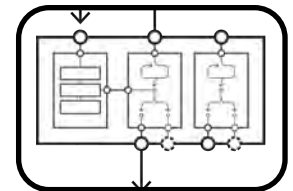
5. Object



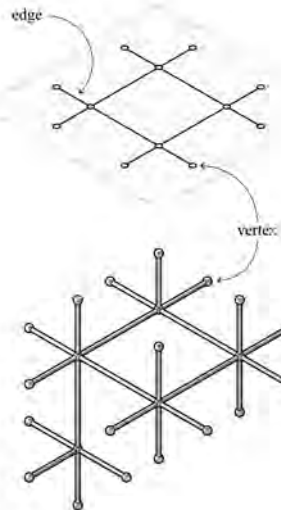
Object



Object

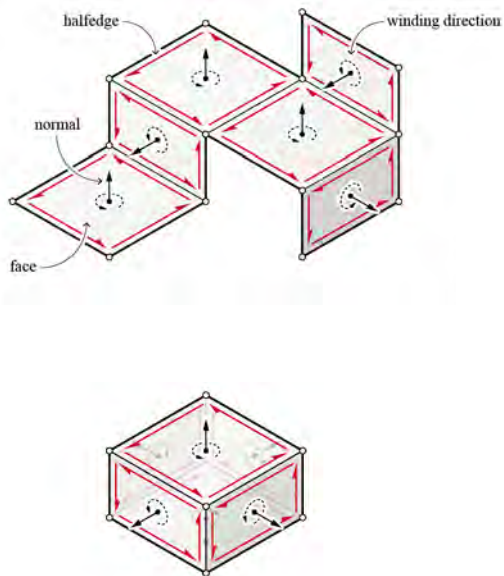


**object-oriented
programming**



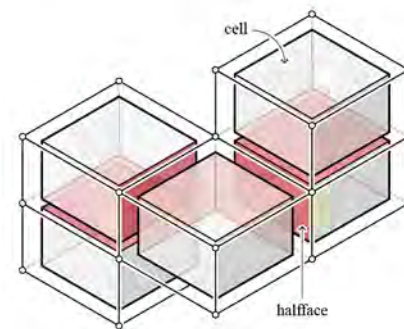
network

network of vertices



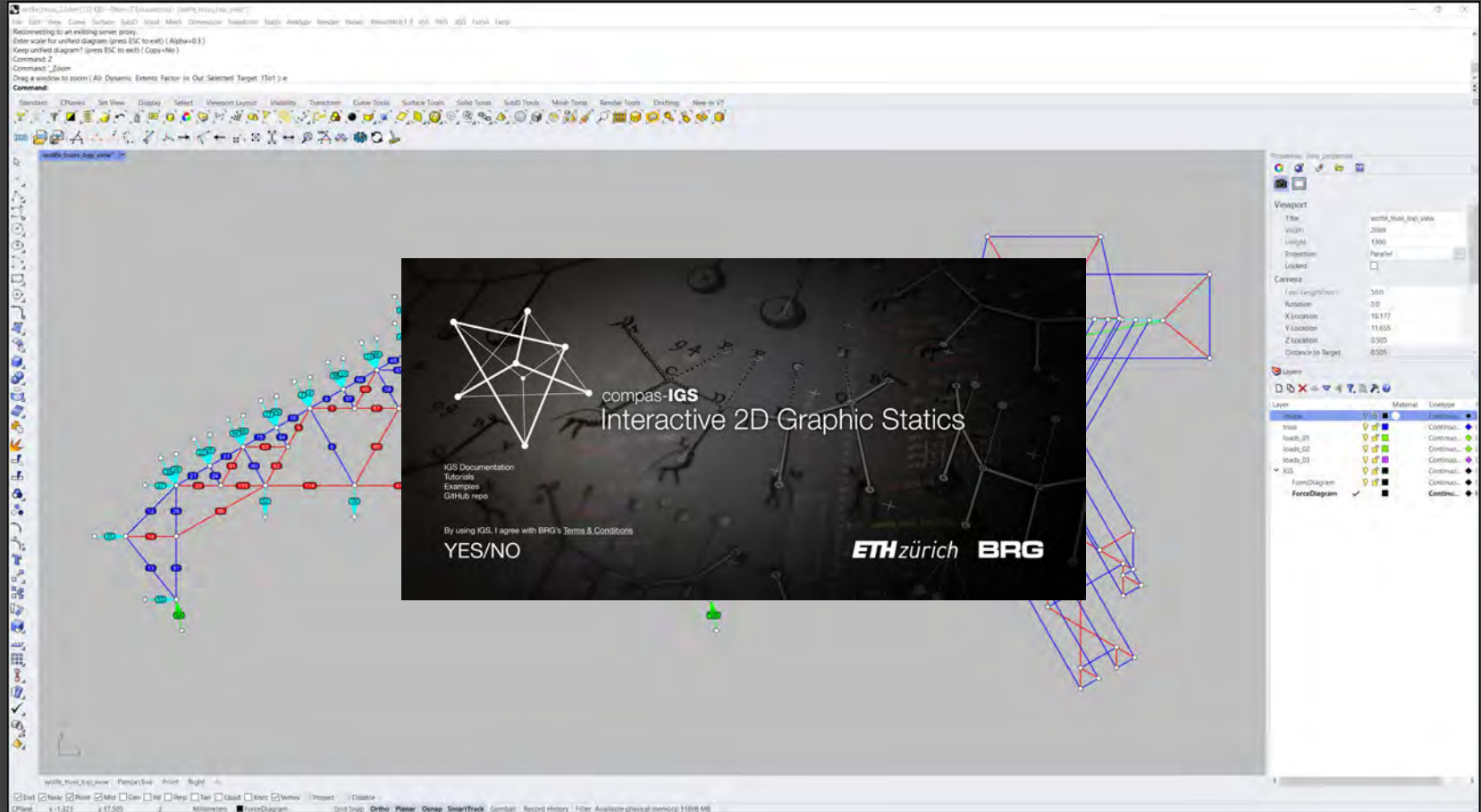
mesh

network of faces



volmesh

network of cells





COMPAS

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

```
def smooth_mesh_length(mesh, lmin, lmax, fixed=None, callback=None):
    """Smooth the mesh length.

    Parameters
    ----------
    mesh : trimesh.Trimesh
        The mesh to smooth.
    lmin : float
        The minimum length of the edges.
    lmax : float
        The maximum length of the edges.
    fixed : list of str
        The names of the edges that are fixed.
    callback : callable
        A function that is called for each edge. It should return a
        boolean value indicating whether the edge should be smoothed.

    Returns
    -------
    mesh : trimesh.Trimesh
        The smoothed mesh.
    """
    # Check if callback is callable
    if callback:
        if not callable(callback):
            raise Exception('Callback is not callable')

    # Get the edges of the mesh
    edges = mesh.edges

    # Iterate over the edges
    for key in range(len(edges)):
        # Get the edge key
        key_xyz = (key // 3, key // 3 % 3, key % 3)

        # Get the edge length
        l = mesh.length(edges[key])

        # Check if the edge is fixed
        if key in fixed:
            continue

        # Check if the edge should be smoothed
        if callback:
            callback(mesh, key, callback_args)

        # Smooth the edge
        p = key_xyz[key]
        nbs = mesh.vertex_neighbours(key, order=True)
        c = center_of_mass_polygon([key_xyz[nbr] for nbr in nbs])

        # Update the mesh vertex
        mesh.vertices[key] = p[0], p[1], p[2]
        attr['y'] += d * (c[1] - p[1])
        attr['z'] += d * (c[2] - p[2])

    # Return the smoothed mesh
    return mesh
```




The COMPAS Home Page features a dark header with the COMPAS logo and navigation links: Home, Store, Plugins, Extensions, Tutorials. Below the header is a 'Getting Started' button and a 'Tutorial' link. The main content area includes a diagram showing the COMPAS core and its connections to various platforms like ROS, SPPL, and PHIT. Below the diagram are three sections: DRY(O), Share your Work, and Collaborate.

COMPAS

Getting Started Tutorial

core

DRY(O)

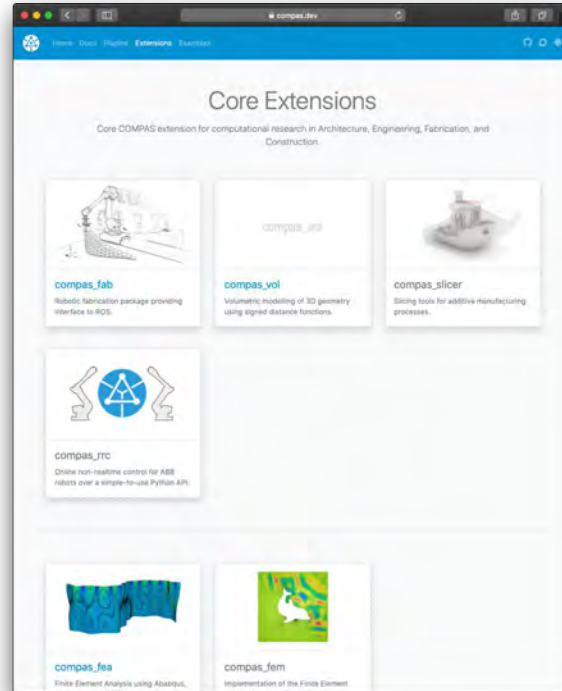
Don't repeat yourself (or others). COMPAS provides a base framework and easy access to peer-reviewed computational research, libraries, and tools, such that you don't have to reinvent the wheel over and over again.

Share your Work

COMPAS is open source framework with a permissive license, research that is based on COMPAS or that is compatible with it can be easily shared and reused, across platforms and software ecosystems.

Collaborate

AEC research and development is highly multidisciplinary. COMPAS simplifies collaboration between individuals and teams with various academic backgrounds.



The COMPAS Core Extensions Page features a white header with the COMPAS logo and navigation links: Home, Store, Plugins, Extensions, Tutorials. Below the header is the title 'Core Extensions' and a subtitle 'Core COMPAS extension for computational research in Architecture, Engineering, Fabrication, and Construction.' The main content area displays six extension cards: compas_fab, compas_vol, compas_slicer, compas_irc, compas_fea, and compas_fem.

Core Extensions

Core COMPAS extension for computational research in Architecture, Engineering, Fabrication, and Construction.

compas_fab

Robotic fabrication package providing interface to ROS.

compas_vol

Volumetric modeling of 3D geometry using signed distance functions.

compas_slicer

Slicing tools for additive manufacturing processes.

compas_irc

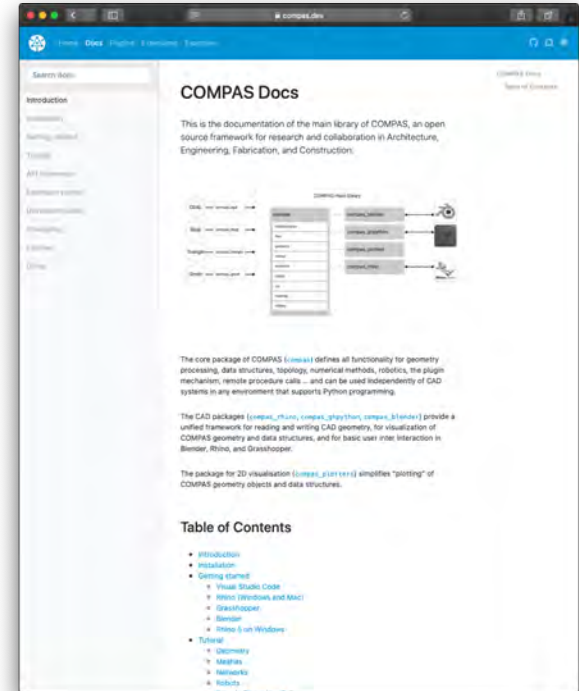
Online non-realtime control for ABB robots over a simple-to-use Python API.

compas_fea

Finite Element Analysis using Abaqus.

compas_fem

Implementation of the Finite Element Method.



The COMPAS Docs Page features a white header with the COMPAS logo and navigation links: Home, Store, Plugins, Extensions, Tutorials. Below the header is the title 'COMPAS Docs' and a subtitle 'This is the documentation of the main library of COMPAS, an open source framework for research and collaboration in Architecture, Engineering, Fabrication, and Construction.' The main content area includes a 'Table of Contents' section with links to Introduction, Installation, Getting started, and Tutorial.

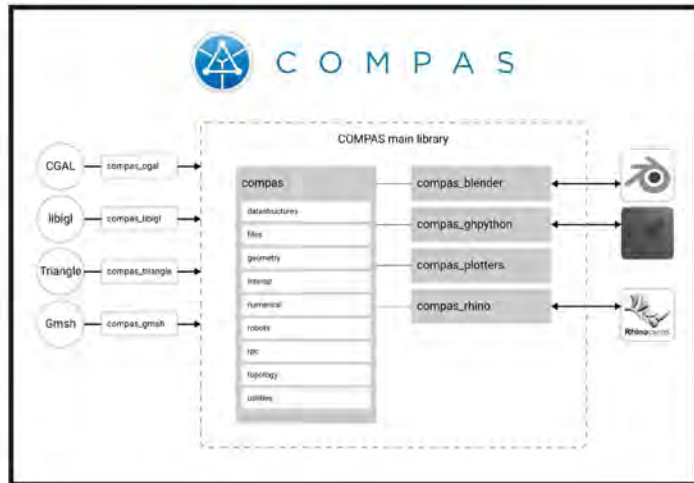
COMPAS Docs

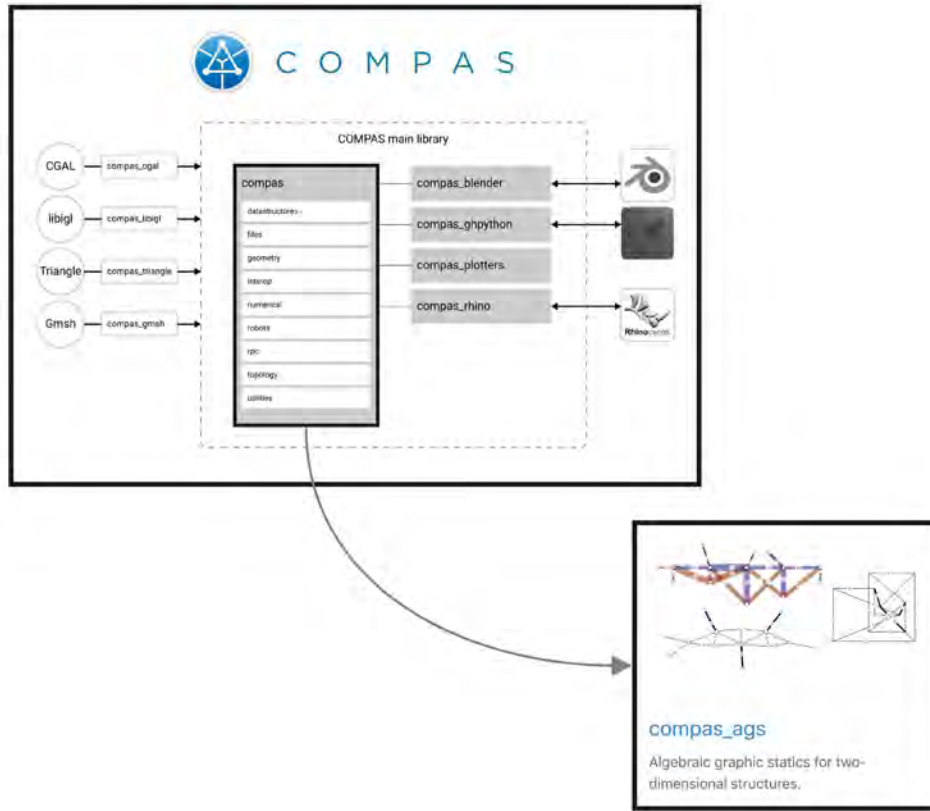
This is the documentation of the main library of COMPAS, an open source framework for research and collaboration in Architecture, Engineering, Fabrication, and Construction.

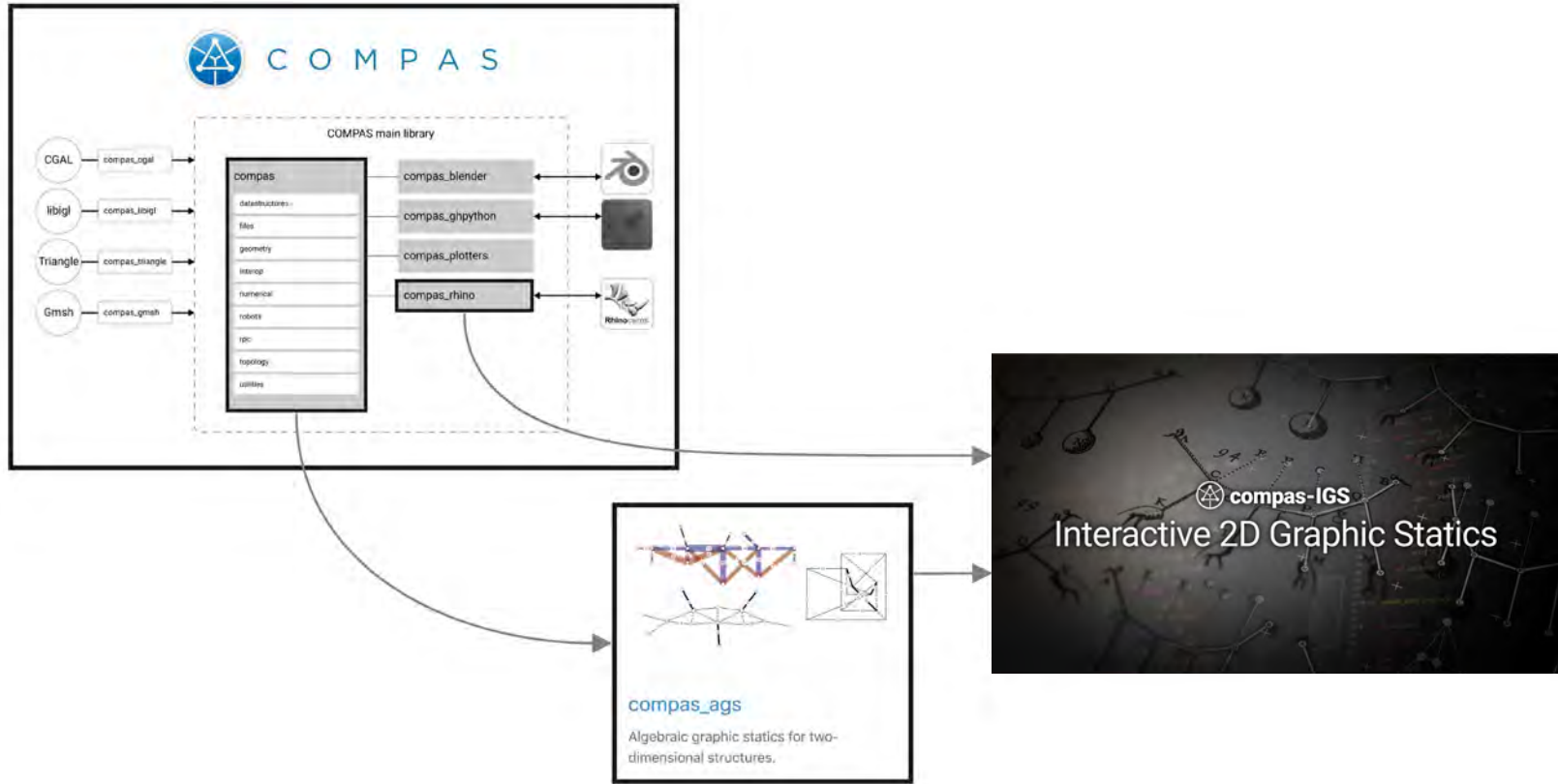
Table of Contents

- Introduction
- Installation
- Getting started
 - Visual Studio Code
 - Linux (Ubuntu and Mac)
 - Grasshopper
 - Blender
 - Linux 32-bit Windows
- Tutorial
 - Geometry
 - Materials
 - Networks
 - Robotics

<https://compas.dev>







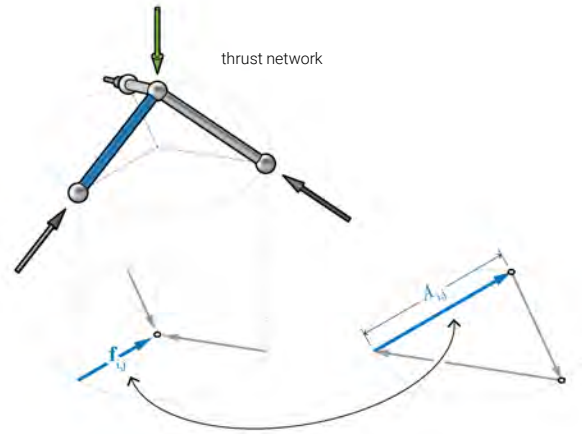
2D graphic statics



form diagram

force diagram

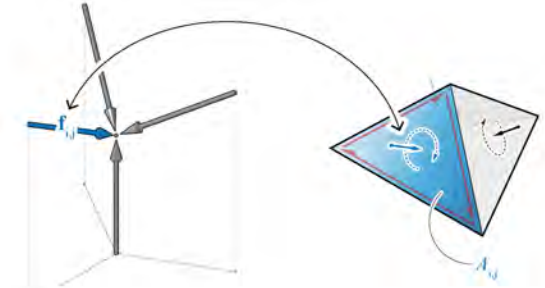
2.5D graphic statics



form diagram

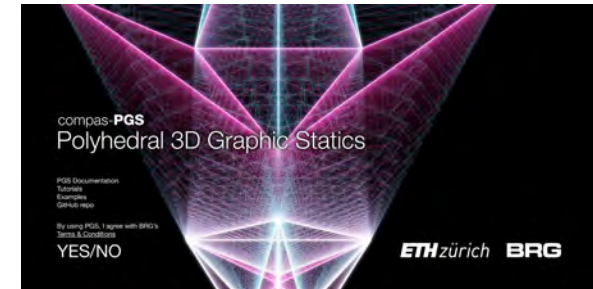
force diagram

3D graphic statics

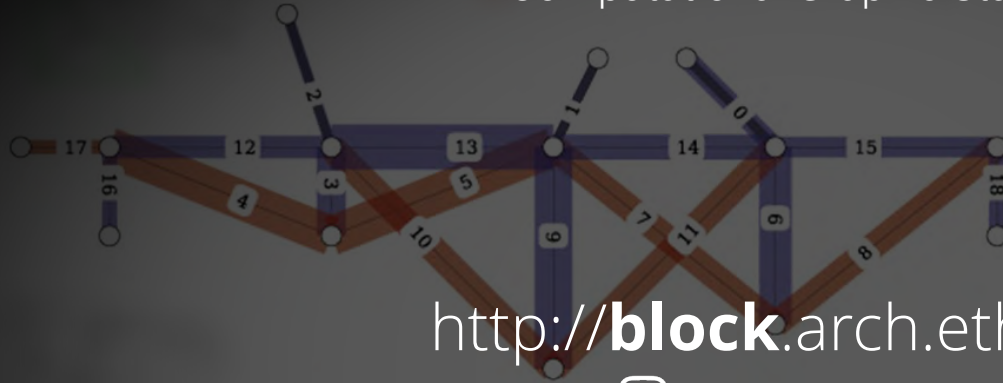


form diagram

force diagram



063-0605-00L : Computational Structural Design 1
Computational Graphic Statics



<http://block.arch.ethz.ch>

 @blockresearchgroup

