DOI: 10.35789/fib.PROC.0055.2021.CDSymp.P054

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

Ricardo Maia Avelino, Juney Lee, Tom Van Mele, Philippe Block
maia@arch.ethz.ch, juney.lee@arch.ethz.ch, van.mele@arch.ethz.ch, 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 formulates the geometrical relationship between the graph representations of the reciprocal form and force diagrams in graphic statics using linear algebra. AGS and its extensions enable automatic construction of force diagrams from given form diagrams, and allow a few basic modifications of the force diagram from which the form diagram is updated. This paper builds on the previous work of AGS by implementing a real-time, bi-directional workflow allowing users to impose various constraints, and perform geometrical modifications in either the form or force diagram from which the other is automatically updated by using an iterative geometric solver. The presented implementation of interactive AGS provides a robust computational back-end to harness the advantages of traditional graphic statics for geometry-based teaching and design of structures.

1 Introduction

Recent research has demonstrated how the principles of graphic statics can be combined with computational tools to create interactive drawings that provide visual feedback to the user in real time [1-3]. Such interactive implementations of graphic statics have not only introduced new and effective methods of teaching structural design [4], but also enabled advanced research [5, 6]. Despite its numerous 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, 8]. 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 formulating 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]. "Bi-directional" AGS extended the method, allowing geometric transformations of a force diagram that result 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, formulated and incorporated in an integrated computational pipeline using the COMPAS 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 explorations in a smooth and intuitive manner through controlled modifications, while minimising the need for manual construction of form and force diagrams.

2 State of the art

The fundamentals of AGS are clearly laid out in [9]. The main procedure in AGS identifies the degrees-of-freedom (DOF) of a system *k*, which is defined by the number of independent edges, which reflect the degree of indeterminacy of the form diagram. The loads in the independent edges can be chosen freely by the user such that the forces in all other edges of the form diagram are back-calculated, from which the geometry of the reciprocal force diagram can be computed (Fig. 1a). However, as stated in [9], the inverse problem of computing the form diagram from a given force diagram was not introduced. This problem requires a constrained optimisation procedure that is not simply the reverse implementation of form-to-force construction.

Vedad et al. [10] presented a pipeline to address this inverse problem for some cases (Fig. 1b). It allows controlled manipulations of the force diagram, from which a new form diagram is iteratively computed with a root-finding procedure based on the Newton method. Although this method addressed the inverse problem for some cases, fully bi-directional modifications of both form and force diagrams are still not possible. The method in [10] requires the user to provide the exact geometry of a valid force diagram, which is not straightforward or apparent in most cases since uninformed movements of vertices may result in an invalid equilibrium configuration. Moreover, constraints can only be applied to the form diagram, which inhibits force-driven explorations.

The goal of this paper is to develop a workflow that enables a fully bi-directional interaction between form and force diagrams. This workflow enables not only controlled modifications to both diagrams, but it also allows for automatic modifications inherited from constraints intuitively applied to them (Fig.1c).

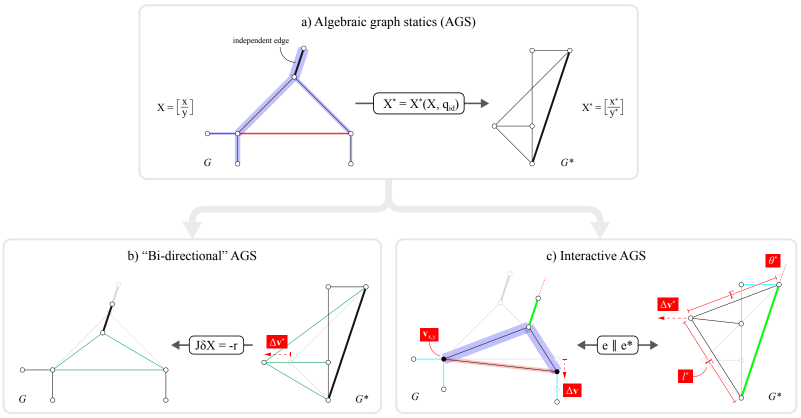


Fig. 1 Overview of the a) AGS procedure [9], b) "Bi-directional" AGS procedure [10]; and c) Interactive AGS, presented in this paper.

3 An interactive implementation of algebraic graphic statics

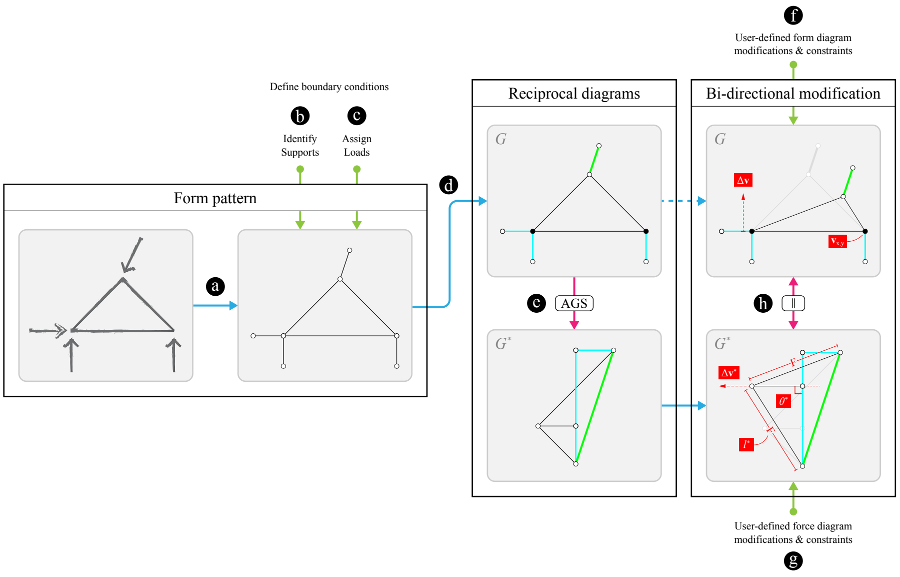
This section provides an overview of the workflow and computational setup of the interactive implementation of AGS.

3.1 Workflow

The workflow of interactive AGS (Fig. 2) can be summarised in the following steps.

a) Input pattern for the form diagram, including externally applied loads and reaction forces, as a network of line segments.
b) Identify the supports.
c) Check the DOF *k* and assign loads to at least *k* (independent) edges in the diagram. If more than *k* edges are selected, the additional loads are treated as "target loads" (see Section 3.2); as a convention, positive values indicate forces in tension, and negative in compression.
d) Create the form diagram *G*.

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



f
User-defined form diagram modifications & constraints
Define boundary conditions
b
Identify Supports
c
Assign Loads
Form pattern
a
d
Reciprocal diagrams
G
e
AGS
G*
Bi-directional modification
G
Δv
v_{x,y}
h
||
G*
Δv*
θ*
l*
User-defined force diagram modifications & constraints
g

Fig. 2 Overview of the interactive AGS workflow.

3.2 Constraints

During procedural constructions of form and force diagrams, various graphic statics constraints are intrinsically built into the drawings (i.e., definition of load lines, placement of the diagrams, etc). However, when the initial form diagram is generated from a network of line segments (Fig. 2d), these constraints need to be explicitly identified, defined and imposed onto the diagrams. Therefore, when the force diagram is obtained (Fig. 2e), the following constraints are automatically imposed (Fig. 3):

1. vertices selected as supports in the form diagram will have their x and y coordinates fixed;
2. the vertices in the form diagram with an externally applied load are constrained to remain on the line of action of the load;
3. edges representing the reaction forces have their orientations fixed in both diagrams; and,
4. edges representing the externally applied loads have their orientations fixed in both diagrams, and their lengths fixed in the force diagram.

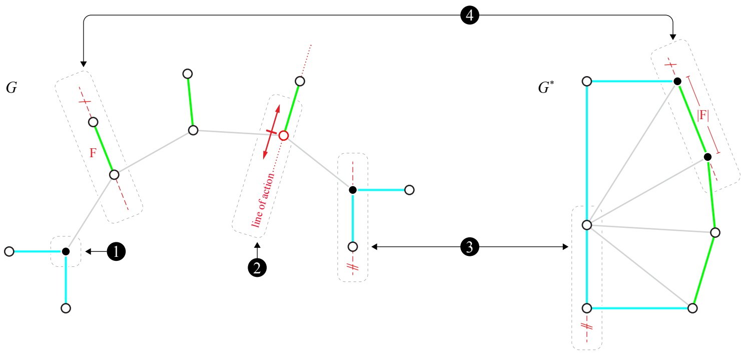


Fig. 3 Imposing default constraints from boundary conditions.

In addition to the above-listed default constraints, the following constraints can be applied to or removed from the form or force diagrams at any point during the workflow (Fig. 2f, 2g):

- vertex partial fixity in the x-direction, which restraints the vertex horizontally;
- vertex partial fixity in the y-direction, which restraints the vertex vertically;
- edge target orientation, which constrains the corresponding edges of both diagrams to align with such orientation;
- edge target force in the form diagram, which constrains the force of the edge in the form diagram and the length of the corresponding edge in the force diagram.

3.3 Solver

This section describes the solving strategies to compute the initial force diagram (Fig. 2e), and the parallelisation of form and force diagrams respecting their constraints (Fig. 2h).

3.3.1 Algebraic computation of force diagram

The coordinates of the initial force diagram ($\mathbf{x^*}$, $\mathbf{y^*}$) can be computed linearly from the force densities ($\mathbf{q}$) of the form diagram's edges and the geometry of the form diagram as in [9] with the following equations:

$$\begin{aligned}\mathbf{L}^*\mathbf{x}^* &= \mathbf{C}^*\mathbf{Qu},\\ \mathbf{L}^*\mathbf{y}^* &= \mathbf{C}^*\mathbf{Qv}\end{aligned} \quad \text{with:} \quad \mathbf{L}^* = \mathbf{C}^*\mathbf{C}^{*\boldsymbol{T}}\,, \tag{1}$$

where, $\mathbf{C}$ and $\mathbf{C^*}$, are the connectivity matrices for the form and force diagrams, respectively. $\mathbf{Q}$=diag($\mathbf{q}$) represents the matrix of force densities, and $\mathbf{u}$, $\mathbf{v}$ are the coordinate difference vectors of the form diagram's edges in x- and y-direction, respectively. The vector of force densities is computed from the geometry of the form diagram and the value of force densities in the independent edges. Although the force diagram can be computed with this procedure (Fig. 2e), only the forces assigned to the set of independent edges are considered, and additional constraints such as target forces to other edges cannot be imposed. The geometry of the form diagram is fixed from the start and no interactive modifications can be performed once the force diagram has been created.

3.3.2 Parallelisation of form and force diagrams

A parallelisation algorithm is developed to allow interactive modifications of form and force diagrams respecting their constraints (Fig. 2h). The process can be divided in two consecutive steps: (i) the force diagram is updated iteratively to reflect the constraints imposed; and (ii) form and force diagrams are updated by solving the least-squares intersection of lines. These two steps are executed interactively until form and force diagrams are reciprocal and the constraints are respected, or until the process hits a maximum number of iterations. The two steps are described in detail below:

i. Updating constrained force diagrams

To update the vertex coordinates of the force diagram, the following algorithm based on target lengths and orientation vectors is adapted from [14]. The algorithm will update the vertex coordinates of the force diagram taking into account only the edges that have constraints assigned to them. Let E_i^{constr} represent the group of edges $\mathbf{e}_{ij}^*$ connected to a vertex $\mathbf{v}_i^*$ of the force diagram with constraints assigned to them. Edge $\mathbf{e}_{ij}^* \in E_i^{constr}$ can be constrained to a target length l_{ij}^*, and/or to a target orientation vector $\hat{\boldsymbol{t}}_{ij}^*$. For each iteration, the coordinates of $\mathbf{v}_i^*$ are updated with barycentre $\mathbf{p}_i$, which is computed by the following equation:

$$\mathbf{p}_i = \frac{\sum_{j\in E_i^{constr}}\left(\mathbf{v}_i^* + l_{ij}^* * \hat{\boldsymbol{t}}_{ij}^*\right)}{\mathrm{n}\left(E_i^{constr}\right)}\,, \tag{2}$$

where if edge $\mathbf{e}_{ij}^*$ is constrained exclusively to a target length (l_{ij}^*), the target vector $\hat{\boldsymbol{t}}_{ij}^*$ is the unit vector pointing from $\mathbf{v}_i^*$ to $\mathbf{v}_j^*$, and, if $\mathbf{e}_{ij}^*$ is constrained only to a target vector ($\hat{\boldsymbol{t}}_{ij}^*$), l_{ij}^* is the edge's length.

Once the force diagram is updated, respecting the constraints, form and force diagrams are not guaranteed to be reciprocal anymore (i.e., corresponding edges may not be parallel). An additional procedure is needed to parallelise the corresponding edges of the two diagrams by updating their vertex coordinates.

ii. Least-squares parallelisation of form and force diagrams

Form and force diagrams are updated by solving a least-squares problem for the intersection of lines [15]. This procedure is applied sequentially first to the form and then to the force diagram. The procedure applies similarly to both diagrams. When it is applied to the form diagram, the orientation of the edges of the force diagram are considered; and, conversely, when it is applied to the force diagram, the directions of the edges of the form are used. It is thus sufficient to explain the iterative process by only considering, for example, the form diagram, which is done next:

In that case, the ideal position x_i, y_i of vertex $\mathbf{v}_i$ of the form diagram connected by m_i edges to each neighbour vertex $\mathbf{v}_j$ with coordinates x_j, y_j, can be found by solving the following system:

$$\mathbf{A}_i \begin{bmatrix} x_i \\ y_i \end{bmatrix} = \mathbf{b}_i,$$

with:

$$\mathbf{A}_i = \sum_{j=1}^{m_i} \left(\mathbf{I} - \hat{\boldsymbol{t}}_{ij}^{*} \hat{\boldsymbol{t}}_{ij}^{*\,T} \right), \qquad \mathbf{b}_i = \sum_{j=1}^{m_i} \left(\mathbf{I} - \hat{\boldsymbol{t}}_{ij}^{*} \hat{\boldsymbol{t}}_{ij}^{*\,T} \right) \begin{bmatrix} x_j \\ y_j \end{bmatrix}, \tag{3}$$

in which $\hat{\boldsymbol{t}}_{ij}^{*}$ is the orientation of the edge's corresponding dual edge in the force diagram.

4 Applications

This section presents graphic-statics applications of the proposed implementation of interactive AGS through a series of examples.

4.1 A “bad” input: form finding of an arch

The first example deals with the well known problem of form finding of an arch subjected to an equally distributed vertical load. When constructed procedurally, the geometry of the arch can be found step by step. In an AGS-based workflow, the user needs to provide the geometry of the “correct” arch from the beginning, which is not always easy or convenient to construct without the force diagram. This example shows how given a “bad” starting form diagram geometry, imposing proper constraints can result in correct form and force diagrams with the desired boundary conditions.

The initial form diagram *G* represents a semi-circular arch (Fig. 4a). Following the convention of AGS [9], the external forces are input as edges to the form diagram (applied loads in green, reaction forces in cyan). The two internal vertices in the extremity of the arch are defined as supports (shown in black). This system has DOF of only one, which means that a desired force magnitude can be assigned to only one independent edge. A force magnitude of -1.0 is assigned to the independent edge (highlighted in bold in Fig. 4a) resulting in the initial force diagram *G**. As expected, the magnitudes of the other applied loads are not 1.0, which signifies that the semi-circular arch does not have the correct geometry for an equally distributed loading case.

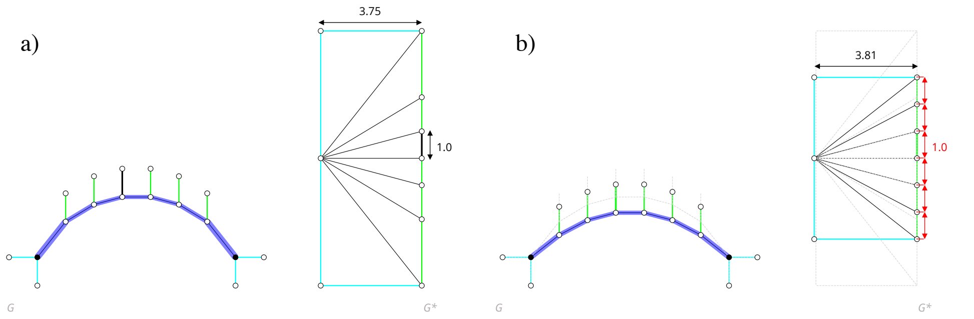


Fig. 4 a) Initial form *G* and force *G** diagrams of a semi-circular arch, where a force magnitude of -1.0 is assigned to the highlighted independent edge; b) updated form *G* and force *G** diagrams after imposing target force magnitudes to the applied loads, resulting in a parabolic arch subjected to equally distributed vertical load.

As introduced in Section 3.2, target force magnitudes can be imposed on the form diagram, which, as consequence, will be reflected as target 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 equilibrium is performed updating both form and force diagrams. The resulting form and force diagrams in Fig. 4b now shows 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.

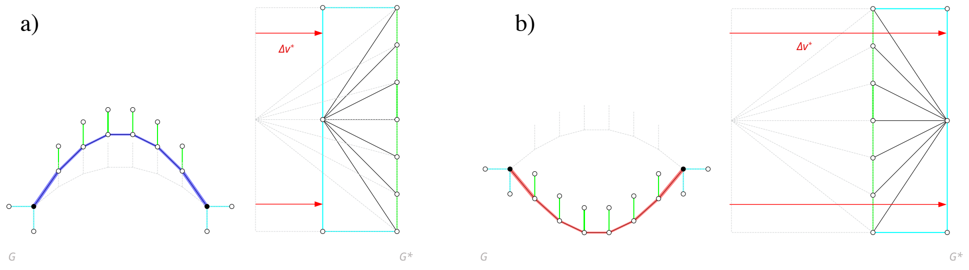


Fig. 5 a) Moving the three vertices on the left side of the force diagram *G** to the right, which results in reduced internal forces and therefore a taller arch in form diagram *G*; b) further movement of three vertices until the forces flip from compression to tension, with form diagram *G* becoming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

Fig. 6 shows two examples of modifications in the vertices of the form diagram. In Fig. 6a, an internal vertex of the arch is moved up and its *y* 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 translation 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 uniform loads applied to it, but with uneven vertical force reactions due to the different support heights.

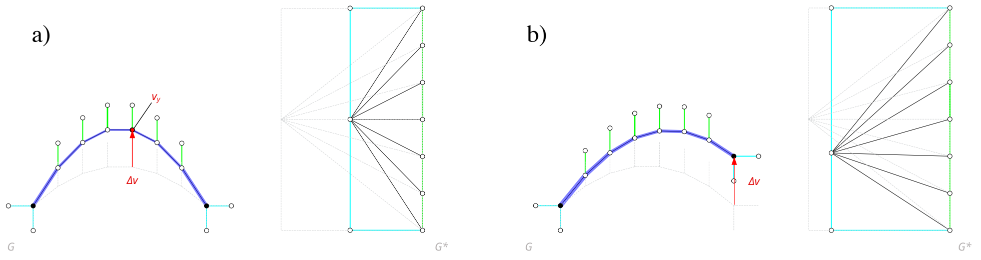


Fig. 6 a) Update in form *G* and force *G** diagrams generated by moving, and constraining an internal vertex of the arch, controlling its structural height; b) update in form *G* and force diagrams *G** generated by moving one of the supports.

4.3 Constraint-based form finding: Constant edge forces

The last example discusses the case of the truss depicted in Figure 7a. Supports are assigned to the two extremes of the truss, and the load is equally distributed to the bottom chord of the truss, having an individual force magnitude of +1.0. The initial form and force diagrams are computed in Figure 7a. In this example, a new geometry of the truss will be computed such that the compression forces in its arching edges become constant. A target force magnitude of -2.5 is imposed to these edges as well as the vertex fixities along the bottom chord of the truss, constraining them to remain horizontal. The parallelisation algorithm results in the truss depicted in Figure 7b, which has constant compression force along the arching edges of the truss and non-constant tensile forces in the horizontal bottom chord. As a consequence, the struts connecting the arching top edges and bottom horizontal chords of the truss are no longer vertical.

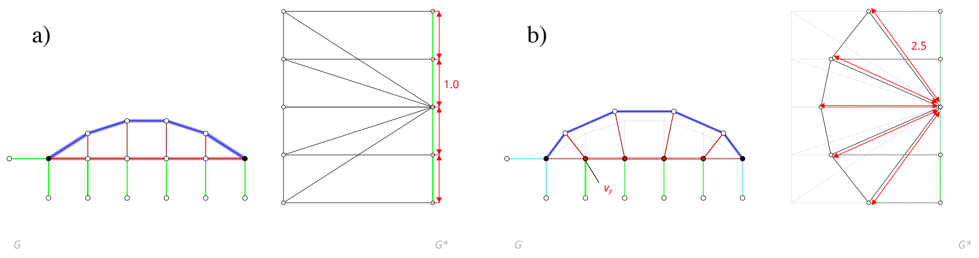


Fig. 7 a) Form *G* and force *G** diagrams obtained for the initial truss assuming with equally distributed of loads of 1.0; b) updated form *G* and force *G** diagrams after imposing constant-force of 2.5 to the top arching edges, and constraining the bottom chord vertically.

The constant-force truss problem can be discretised further, as depicted in Figure 8a. Further modification is applied to the vertices along the bottom chords of the truss, which are constrained to follow a sinusoidal curve. The new geometry of the truss can be found with the presented implementation while always maintaining the constant-force constraint in the top arching edges of the truss and the same loading case. Different variations of the constant-force truss obtaining by accentuating the curve on the bottom chord are depicted in Figures 8b-d.

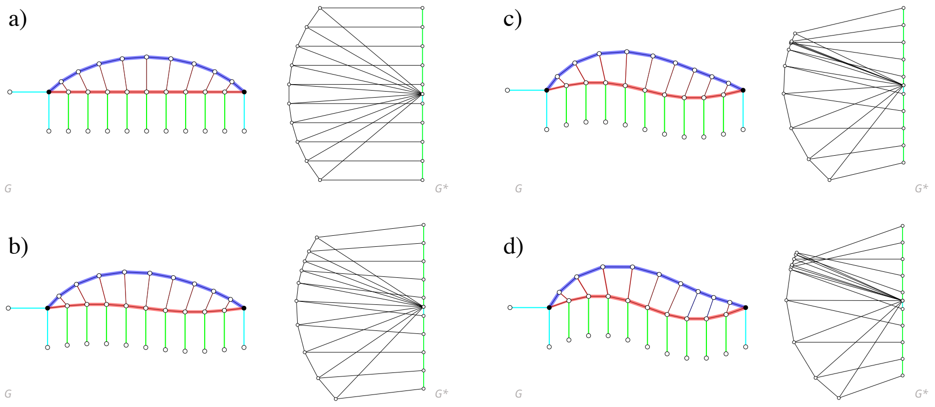


Fig. 8 Variations of the constant-force problem for sinusoidal bottom chord inputs.

5 Conclusions

This paper presented a fully bi-directional implementation of AGS by enabling users to impose various constraints to both form and force diagrams in an intuitive and interactive manner. With the introduction of an iterative geometric solver that simultaneously updates form and force diagrams, a flexible and robust application of computational graphic statics is made possible in a design-oriented workflow.

This implementation is a step forward in establishing an interactive tool incorporated in a CAD environment with a graphical user interface that can leverage the power of graphic statics for geometry-based teaching and designing of structures. This open-source tool is currently in development and will be made available soon to the scientific community. Future work will focus on incorporating various geometric objects as constraints, as well as enforcing adaptive target force magnitudes.

References

[1] "Active Statics". 2013. Last access 08.04.2021. http://acg.media.mit.edu/people/simong/statics

[2] Block, P., Ciblac, T., & Ochsendorf, J. 2006. "Real-time limit analysis of vaulted masonry buildings." *Computers and Structures*, *84*(29–30):1841–1852. doi:10.1016/j.compstruc.2006.08.002

[3] "eQUILIBRIUM". 2011. Last access 08.04.2021. http://block.arch.ethz.ch/equilibrium

[4] Enrique, L., Tanadini, D., Block, P., & Schwartz, J. 2018. "Design-oriented approach to teach structures in architecture based on graphic statics." In: *Proceedings of IASS Annual Symposia*, Boston, July.

[5] Van Mele, T., Lachauer, L., Rippmann, M., & Block, P. 2012. "Geometry-based understanding of structures." *Journal of the International Association for Shell and Spatial Structures*, *53*(174):285–295.

[6] Beghini, L. L., Carrion, J., Beghini, A., Mazurek, A., & Baker, W. F. 2014. "Structural optimization using graphic statics." *Structural and Multidisciplinary Optimization*, *49*(3):351–366. doi:10.1007/s00158-013-1002-x

[7] Wolfe, W.S. 1921. *Graphical analysis: a handbook on graphic statics*. New Cork: McGraw-Hill.

[8] Allen, E. and Zalewski, W. 2009. *Form and forces: designing efficient, expressive structures*. John Wiley & Sons.

[9] Van Mele, T., & Block, P. 2014. "Algebraic graph statics." *CAD Computer Aided Design*, *53*:104–116. doi:10.1016/j.cad.2014.04.004

[10] Alic, V., & Åkesson, D. 2017. "Bi-directional algebraic graphic statics." *CAD Computer Aided Design*, *93*:26–37. doi:10.1016/j.cad.2017.08.003

[11] McRobie, A., Baker, W., Mitchell, T., & Konstantatou, M. 2016. Mechanisms and states of self-stress of planar trusses using graphic statics, Part II: Applications and extensions. *International Journal of Space Structures, 31(2-4)*:102-111. doi:10.1177/0266351116660791

[12] Konstantatou, M., D'Acunto, P., & McRobie, A. 2018. Polarities in structural analysis and design: N-dimensional graphic statics and structural transformations. *Int J Solids Struct, 152*:272–293. doi:10.1016/j.ijsolstr.2018.07.003

[13] Van Mele, T. et al. 2021. "COMPAS: A framework for computational research in architecture and structures." doi:10.5281/zenodo.2594510

[14] Rippmann, M., Lachauer, L., & Block, P. 2012. Interactive Vault Design. *International Journal of Space Structures*, *27*(4):219–230. doi:10.1260/0266-3511.27.4.219

[15] Traa, J. 2013. "Least Square solution to the intersection of Lines." University of Illinois at Urbana-Champaign. https://docplayer.net/21072949-Least-squares-intersection-of-lines.html