Computer-Aided Design 166 (2024) 103620

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Algebraic 3D Graphic Statics with Edge and Vertex Constraints: A Comprehensive Approach to Extend the Solution Space for Polyhedral Form-Finding

Yao Lu [a], Márton Hablicsek [a,b], Masoud Akbarzadeh [a,c,*]

[a] Polyhedral Structures Laboratory, Weitzman School of Design, University of Pennsylvania, Philadelphia, USA
[b] Department of Mathematics, Leiden University, Leiden, Netherlands
[c] General Robotic, Automation, Sensing and Perception (GRASP) Lab, School of Engineering and Applied Science, University of Pennsylvania, Philadelphia, USA

ARTICLE INFO

Keywords:
Algebraic three-dimensional graphic statics
Polyhedral reciprocal diagrams
Geometric degrees of freedom
Tension and compression combined
Vertex and edge constraints
Auxetic material

ABSTRACT

Built upon a previous algebraic framework for polyhedron-based 3D Graphic Statics (3DGS) that can numerically solve for a variety of dual diagrams given an input force or form diagram, this paper introduces an improved algebraic formulation that integrates edge lengths and vertex location constraints for better control over the results. Those constraints are realized by additional edge and vertex constraint equations to previously established closing equations. The entire system of equations can be solved using the Moore–Penrose inverse (MPI) method, and each solution represents a set of compatible edge lengths for the dual diagram to be constructed. The whole solution space of the equation system provides a wide range of dual diagrams, including forms with both tensile and compressive members, which can be easily explored and was not possible using iterative methods or previous algebraic formulations. This improved formulation has been computationally implemented and released as part of a plug-in software program within the environment of Rhino3D® and Grasshopper3D®, enriching the structural form-finding toolset for designers, engineers, researchers, and educators. The tool's performance and accuracy are demonstrated through a series of comparative studies with iterative methods. Various case studies are also presented to showcase the application of this method.

1. Introduction

The geometry-based structural design method based on 2D reciprocal diagrams, namely graphic statics (GS), has been studied and developed in the past centuries [1–7]. It can be used for not only structural analysis but also structural form-finding and has been adopted in various projects [8]. Graphic statics in three dimensions (3DGS) was initially explored by Rankine [2] and Maxwell [9] shortly after the introduction of the 2D method. Recently, this historical method in 3D has been revitalized by further investigations and the integration of computational techniques, enabling its application to the design of complex spatial structures. Amidst the different approaches of 3DGS using reciprocal diagrams, vector-based [10] and polyhedron-based [11,12] strategies have been actively developed in the past few years. In the force diagram of the vector-based approach, the equilibrium for each node is guaranteed by a closed cycle of force vectors, where each force vector is parallel to the corresponding edge in the form diagram, and the vector length indicates its force magnitude. In contrast, the polyhedron-based approach uses a closed polyhedron to represent equilibrium for each node, where each face is perpendicular to the corresponding edge in the form diagram, and its area shows the force magnitude.

Each of these approaches has advantages and disadvantages. As highlighted by Maxwell [3], the vector-based approach can describe the equilibrium of almost any 3D frame [13]. However, the form and force reciprocity is not explicit and intuitive, making its use challenging for designers without prior knowledge in this field. The polyhedron-based approach is limited to polyhedral geometries and can only represent a subset of the equilibrium states. Nevertheless, its polyhedral nature and inherent planarity could facilitate materializations in real-world practices using sheet materials or modular building blocks. This property can also be combined with various fabrication techniques such as concrete 3D printing, origami, and kirigami [14–18]. When materialized as spatial frame systems, the polyhedral form can also incorporate various design features to enrich the simple bar-node geometry [19–24]. In material science, polyhedron-based 3DGS can generate geometries for cellular solids with specific micro-architectures. Recent research suggests that subdividing the force diagram along specific

* Corresponding author at: Polyhedral Structures Laboratory, Weitzman School of Design, University of Pennsylvania, Philadelphia, USA.
E-mail addresses: yaolu61@upenn.edu (Y. Lu), m.hablicsek@math.leidenuniv.nl (M. Hablicsek), masouda@upenn.edu (M. Akbarzadeh).

https://doi.org/10.1016/j.cad.2023.103620
Received 28 March 2023; Received in revised form 26 June 2023; Accepted 11 September 2023
Available online 29 September 2023

Fig. 1. Selected projects designed with polyhedron-based 3DGS. (a) A three-meter span funicular glass bridge constructed with 1 cm float glass using a modular assembly technique [16]. (b) A shellular geometry made of one-millimeter stainless steel sheets with the help of kirigami techniques [15]. (c) A funicular bridge model made of 1 mm paper board [18]. (d) A spatial table made of prefabricated concrete parts [22]. (e) A spatial timber framework prototype that uses kerf bending and zipper [23]. (f) A cellular structure is turned into shellular structure by subdividing the force diagram [25]. Images (a)–(e) courtesy of PSL, University of Pennsylvania. Image (f) courtesy of AML^3, McGill University.

axes can transform cellular structures that are prone to buckling into shellular funicular structures that exhibit high structural efficiency and low structural density [25]. Some design and research projects that use polyhedron-based 3DGS are selected and shown in Fig. 1. Lee et al. [26] proposed a novel framework under the polyhedron-based approach that releases certain geometrical constraints brought by the polyhedral reciprocity. It is based on the concept of disjointed force polyhedrons, which allows the adjacent cells in a force diagram to have the same contact face areas and orientations, but different shapes and vertex locations. Lee's framework allows for more flexible control of force magnitudes of the applied loads as well as the internal force members and then uses those inputs to inform a wider range of 3D frames beyond just polyhedral ones, such as tensegrity structures and non-polyhedral spatial frames. However, it does not hold the reciprocity of the form and force diagrams and therefore increases computation complexities.

The following sections of this paper focus on the polyhedron-based approach formulated by Akbarzadeh [11], and the term 3DGS is used to refer specifically to this formulation unless stated otherwise.

It is important to note that a force diagram can correspond to multiple form diagrams with the same topology and edge directions but different edge lengths, and vice versa. This is called geometric degrees of freedom (GDoF), where the topology and edge directions are preserved while edge lengths are allowed to vary. Manipulating GDoF will not break the reciprocal relationship and will lead to various design possibilities. Most implementations of 3DGS rely on procedural techniques or iterative algorithms based on local nodal operations to solve for solutions one at a time. Procedural techniques [11] can be tedious and labor intensive, and they cannot be generalized to a wide range of inputs. Nodal operation-based iterative algorithms [26–28] are advantageous in using a unified solver with control over edge lengths, vertex locations, and face areas under one framework. However, it requires an input angle tolerance as the criterion to terminate the iterations, meaning that the output will always have some level of deviation. The smaller the tolerance is, the longer the runtime will be, making it computationally intensive for complex geometries. Additionally, it cannot construct reciprocal diagrams with non-convex polyhedrons that represent tension–compression combined systems, which limits the flexibility of exploring the full solution space.

The algebraic formulation for the reciprocal diagrams provides a different perspective on the implementation of 3DGS. It extracts the mathematical essence of the reciprocal relationship between force and form diagrams and can leverage the existing numerical computation libraries to efficiently access the entire solution space, unleashing the potential of 3DGS in form-finding. The robustness and efficiency of those libraries allow the manipulation of reciprocal diagrams with greater flexibility, shorter time, and higher accuracy compared to the alternatives. For instance, as has been shown in the previous research [29,30], the algebraic relationship between the form and force allows exploring multiple equilibrium configurations for a single force diagram, which was not easy to compute using other approaches.

1.1. Related work

1.1.1. Algebraic 2D graphic statics

Micheletti [31] described the algebraic duality between the reciprocal diagrams of self-stressed frameworks following Cremona's construction. The term algebraic graphic statics was then brought up by Van Mele and Block [32] as linear equations based on the algebraic reciprocal constraints. As an analysis tool, it takes a group of connected lines representing the form diagram as input and calculates the internal force distribution. The edge lengths of the force diagram can be derived by enforcing the closed edge vectors of its polygons. Therefore all internal and external force magnitudes of the form diagram can be obtained. This was later extended to a bi-directional approach by Alic and Åkesson [33], where both force and form diagrams can be manipulated and the reciprocal diagrams can be updated instantly, making it an effective interactive form-finding tool in 2D.

1.1.2. Algebraic 3D graphic statics

Hablicsek et al. [29] established an algebraic framework for 3DGS where the reciprocal polyhedral diagrams are described through a linear system of equations that ensures the closed coplanar edge vectors of the polyhedron faces. This method can use either a force diagram or a form diagram as input and efficiently compute all possible dual diagrams using closed-form solutions with high accuracy. While this method also allows user inputs to control edge lengths and thus provides flexibility in exploring the solution space [34], the vertex locations and edge lengths cannot be precisely specified by users as with the iterative method [28]. Akbarzadeh and Hablicsek [35] then proposed a quadratic formulation capable of accommodating assigned area and edge lengths. This is particularly useful as it allows the manipulation of force diagrams with desired face areas, through which a variety of equilibrium states with both tension and compression can be explored. Nonetheless, this method is time-consuming due to its quadratic nature and still, it does not provide precise control over the location of vertices.

1.2. Problem statement and objectives

As previously mentioned, while 3DGS has shown significant potential for real-world applications through a variety of materialization strategies, existing implementations have not fully leveraged its capabilities. The iterative method used in PolyFrame can only handle compressive form-finding and is computationally expensive for complex geometries. The algebraic formulation of 3DGS enables extensive exploration of the solution space, but it does not provide the same level of precise control over the vertex locations and edge lengths of the reciprocal diagrams as the iterative method does.

To address the above-mentioned issues, this paper aims to provide an enhanced algebraic formulation of 3DGS with vertex and edge constraints. This improved formulation has the ability to precisely control vertex locations and edge lengths while preserving the benefits of high efficiency, flexibility, and accuracy.

1.3. Paper outline

Section 2 provides a detailed account of how the constrained form diagrams can be derived given an input force diagram, starting with a recap of the previous algebraic formulation in Section 2.1, which is then extended to a new construction method that requires fewer equations. In Section 2.2, additional edge constraint equations for the form diagram are explained, followed by the introduction of algebraic vertex constraints in Section 2.3, where points, lines, and planes can be taken as constraint geometries. Sections 2.4 and 2.5 delve into the solution space of the equation system. Section 2.6 explains how the methodology can be applied to the force diagram itself to constrain it in the same way as the form diagram. Section 3 provides details on the implementation of the methodology as a software tool. It includes a brief performance study and presents the differences from the iterative methods. Section 4 shows various use cases and scales of the methodology, demonstrating its versatility and applicability. Section 5 discusses the limitations of the proposed methodology and outlines possible future improvements to further enhance its performance and applicability.

1.4. Nomenclatures

The algebraic elements of this paper are denoted as follows; matrices are denoted by bold capital letters (e.g. $\mathbf{A}$); vectors are denoted by lowercase, bold letters (e.g., $\mathbf{v}$), except the user input vectors which are represented by Greek letters (e.g., ξ); the topological data of the primal diagram are described by italic letters (e.g., f); and the data corresponding to the dual and reciprocal diagram are represented by italic letters with a † sign (e.g., $f^\dagger$). Table 1 provides a comprehensive list of all major notations used in this paper.

Table 1
A list of all notations used in this paper.

Topology	*Description*
Γ	Force diagram
$\Gamma^\dagger$	Form diagram
v	# of vertices in Γ
e	# of edges in Γ
f	# of faces in Γ
c	# of cells in Γ
e_{int}	# of internal edges in Γ
f_{int}	# of internal face in Γ
$v^\dagger$	# of vertices in $\Gamma^\dagger$
$e^\dagger$	# of edges in $\Gamma^\dagger$
$f^\dagger$	# of faces in $\Gamma^\dagger$
$c^\dagger$	# of cells in $\Gamma^\dagger$
$e^\dagger_{int}$	# of internal edges in $\Gamma^\dagger$
$f^\dagger_{int}$	# of internal faces in $\Gamma^\dagger$
Matrices	
$C_{e_{int}\times f_{int}}$	Internal edge-face connectivity matrix of Γ
$\mathbf{A}$	Closing matrix
$\mathbf{B}$	Edge constraint matrix
$\mathbf{D}$	Vertex constraint matrix
$\mathbf{M}$	Constrained closing matrix
$\mathbf{M}^+$	Moore–Penrose inverse of $\mathbf{M}$
$\mathbf{N}_x$	Diagonal matrix of n_x
$\mathbf{N}_y$	Diagonal matrix of n_y
$\mathbf{N}_z$	Diagonal matrix of n_z
$\mathbf{N}$	Column-wise stacking of n_x, n_y, n_z
$\mathbf{P}_i$	Edge-face connectivity diagonal matrix for $f_i^\dagger$
$\mathbf{Q}_i$	Diagonal matrix that filters a path of edges
Vectors	
$\hat{\mathbf{n}}_i$	Unit normal vector of internal face f_i
$\mathbf{e}_i^\dagger$	3D vector representation of the edge $e_i^\dagger$
p_i	Target 3D point of the constrained vertex $v_i^\dagger$
p_i'	p_i in a local coordinate system
$\mathbf{v}$	3D vector between the constrained vertex pair
$\mathbf{v}'$	$\mathbf{v}$ in a local coordinate system
n_x	x-coords of $\hat{\mathbf{n}}_i$
n_y	y-coords of $\hat{\mathbf{n}}_i$
n_z	z-coords of $\hat{\mathbf{n}}_i$
n_x'	x-coords of $\hat{\mathbf{n}}_i$ in a local coordinate system
n_y'	y-coords of $\hat{\mathbf{n}}_i$ in a local coordinate system
n_z'	z-coords of $\hat{\mathbf{n}}_i$ in a local coordinate system
b_i	Column vector indicating a constrained edge $e_i^\dagger$
l	Target lengths of the constrained edges
d	Column vector of u_i, v_i, and w_i
t	Column vector by stacking 0, l, and d
q	Solution to the system of equations
Parameters	
ξ	Parameter for the Moore–Penrose inverse method
Other	
r	Rank of $\mathbf{M}$
u	x-coord differences of a constrained vertex pair
v	y-coord differences of a constrained vertex pair
w	z-coord differences of a constrained vertex pair
u′	u in a local coordinate system
v′	v in a local coordinate system
w′	w in a local coordinate system

2. Method

In the context of 3DGS using polyhedral reciprocal diagrams, both form and force diagrams consist of polyhedral cells, planar faces, straight edges, and vertices. Each edge in the reciprocal diagrams shares an identical vertex with its adjacent edges. As described by Akbarzadeh [11] and Hablicsek et al. [29], the edges and faces in the reciprocal diagrams have specific directions. The face normal must be consistent with its edge directions following the right-hand rule, the face orientations must be consistent within each cell, and the cell orientations must be consistent within each diagram. Therefore, each pair of adjacent faces shares two versions of an identical edge that have opposite directions. Similarly, each pair of adjacent cells shares two versions of an identical face that have opposite normal directions.

A simple example can be used to illustrate the topological relationship between reciprocal polyhedral diagrams (Fig. 2). The starting diagram is called *primal*, Γ, which can either be a form diagram or a force diagram, and the reciprocal diagram is called *dual*, $\Gamma^\dagger$. The numbers of vertices, edges, faces, and cells of the primal are denoted by v, e, f, and c respectively, and the ones of the dual are super-scripted with a dagger symbol (†).

These two diagrams are topologically reciprocal: i.e. the vertices v, edges e, faces f, and cells c of the primal topologically map to the cells $c^\dagger$, faces $f^\dagger$, edges $e^\dagger$, and vertices $v^\dagger$ of the dual diagram respectively [36]. One characteristic of this reciprocity is that the numbers of dual elements in both diagrams are the same. For instance, the number of edges e in the primal is equal to the number of faces $f^\dagger$ in the dual, and the number of vertices v of the primal is equal to the number of cells $c^\dagger$ in the dual, etc. Adjacency is another key characteristic of reciprocity, meaning that if two elements in the primal

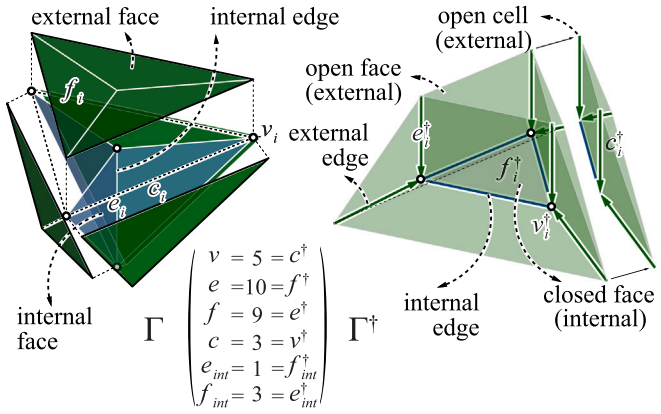

Fig. 2. An example force diagram (Γ) and its corresponding form diagram ($\Gamma^{\dagger}$). The numbers of vertices, edges, faces, cells, internal edges, and internal faces of the force diagram (denoted by v, e, f, c, e_{int}, and f_{int}) are equal to the number of cells, faces, edges, vertices, internal faces, and internal edges of the form diagram (denoted by $c^{\dagger}$, $f^{\dagger}$, $e^{\dagger}$, $v^{\dagger}$, $f^{\dagger}_{int}$, and $e^{\dagger}_{int}$) respectively.

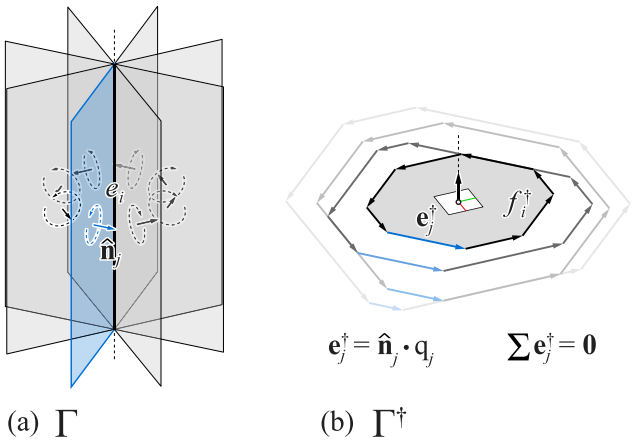

Fig. 3. Closing equations can be written based on the closed edge vectors associated with each face. (a) The faces f_j are connected to an internal edge e_i in a force diagram. (b) The corresponding edges $e^{\dagger}_j$ in the form diagram surrounding an internal face $f^{\dagger}_j$ that is planar since the edges $e^{\dagger}_j$ share a common normal vector.

are adjacent, then their corresponding elements in the dual diagram are also adjacent. Moreover, each edge e_i of the primal is perpendicular to its corresponding face $f^{\dagger}_i$ in the dual.

Notably, the external faces of a force diagram correspond to the external edges that represent applied load and reaction forces in the form diagram. Those external faces and edges are usually colored green as shown in Fig. 2. Unlike a force diagram where all elements are closed, a form diagram contains open elements. For example, vertices only exist at one end of the external edges in a form diagram, so such edges are open-ended. Also, all faces and cells containing open edges are also considered open. The closed faces in a form diagram are defined as internal, and they correspond to the internal edges of the force diagram. The edges of the internal faces are also defined as internal, and they correspond to the internal faces of the force diagram. All internal faces of the force diagram are shaded in blue, with the darkness indicating their respective areas. The darker the color, the larger the corresponding area. A similar color code is applied to the form diagram, where edges are colored blue for compression and red for tension. The darkness and thickness of each edge correspond to the area of its respective face in the force diagram.

The algebraic reciprocal construction with constraints starts from an input primal diagram. It aims to represent the dual diagram using a system of linear equations. Although both user-created form and force diagrams can be used as input as shown by Hablicsek et al. [29], this paper focuses on the form-finding aspect of it and always starts from force diagrams, so Γ and $\Gamma^{\dagger}$ always denote force and form diagrams respectively. Moreover, starting from a form diagram, meaning that an analysis of its internal force distribution is wanted, is relatively difficult in the context of 3DGS as the forms with polyhedral features are not easy to construct directly.

2.1. Linear closing equations for the form diagram

Before having any constraints, the linear closing equations need to be developed based on the closed coplanar edge vectors associated with each internal face in the form diagram. This follows the convention established by Hablicsek et al. [29]. Let e_{int}, f_{int} denote the number of internal edges and faces respectively in the force diagram; let $e^{\dagger}_{int}$, $f^{\dagger}_{int}$ denote the number of internal edges and internal faces respectively in the form diagram. For each internal face $f^{\dagger}_i$ in the form diagram to be constructed, we can write an equation that shows its closed edges based on the edge lengths (Fig. 3). The term edge length in this paper means a scalar that represents the signed length of an edge vector $\mathbf{e}^{\dagger}_i$ and is denoted by q_i. All edge lengths of the form diagram can be assembled in the vector $\mathbf{q}$.

Since each edge of the force diagram is perpendicular to its corresponding face in the form diagram, the unit direction vector of edge $e^{\dagger}_i$ in the form diagram is equal to the unit normal vector of face f_i in the force diagram. Let $\mathbf{N}_x$, $\mathbf{N}_y$, and $\mathbf{N}_z$ be the $[f_{int} \times f_{int}]$ diagonal matrices whose diagonal entries are the x-, y-, and z-coordinates (respectively) of the unit normal vectors of the internal faces in the force diagram. According to Hablicsek et al. [29], the $[e_{int} \times f_{int}]$ connectivity matrix $\mathbf{C}_{e_{int}\times f_{int}}$ shows the connectivity between the internal faces and internal edges in the form diagram. Then the closed edge vectors can be represented as

$$\mathbf{C}_{e_{int}\times f_{int}}\mathbf{N}_x\mathbf{q} = \mathbf{0} \quad \mathbf{C}_{e_{int}\times f_{int}}\mathbf{N}_y\mathbf{q} = \mathbf{0} \quad \mathbf{C}_{e_{int}\times f_{int}}\mathbf{N}_z\mathbf{q} = \mathbf{0}. \tag{1}$$

Those equations can be combined as

$$\mathbf{A}\mathbf{q} = \mathbf{0} \tag{2}$$

where $\mathbf{A}$ is a $[3e_{int} \times f_{int}]$ closing matrix written as

$$\mathbf{A} = \begin{pmatrix} \mathbf{C}_{e_{int}\times f_{int}}\mathbf{N}_x \\ \hline \mathbf{C}_{e_{int}\times f_{int}}\mathbf{N}_y \\ \hline \mathbf{C}_{e_{int}\times f_{int}}\mathbf{N}_z \end{pmatrix} \tag{3}$$

Each solution for $\mathbf{q}$ shows a possible combination of edge lengths in the form diagram that satisfies the closed edge vectors associated with every internal face in the form diagram.

2.1.1. Closing matrix with reduced size

The closing matrix $\mathbf{A}$ can be constructed in another way such that its size can be reduced to $[2e_{int} \times f_{int}]$. This optional step could significantly reduce the execution time when solving the equations systems [37].

In the previous formulation shown by Eq. (1), each closed face in the form diagram is enforced by three equations that correspond to the x-, y-, and z-dimensions. This is redundant due to the planarity of all faces. Under a local coordinate system aligned with the face, this can be enforced by only two equations that correspond to the local x-dimension and local y-dimension (Fig. 4).

Let $\mathbf{n}_x$, $\mathbf{n}_y$, $\mathbf{n}_z$ be the $[e^{\dagger}_{int} \times 1]$ (equivalent to $[f_{int} \times 1]$) vector whose entries are the x-, y-, and z-coordinates (respectively) of the unit directional vectors of the $e^{\dagger}_{int}$ internal edges of the form diagram (equivalent to the unit normal vectors of the f_{int} internal faces of the force diagram). For each face $f^{\dagger}_i$ in the form diagram, let $\mathbf{P}_i$ be a

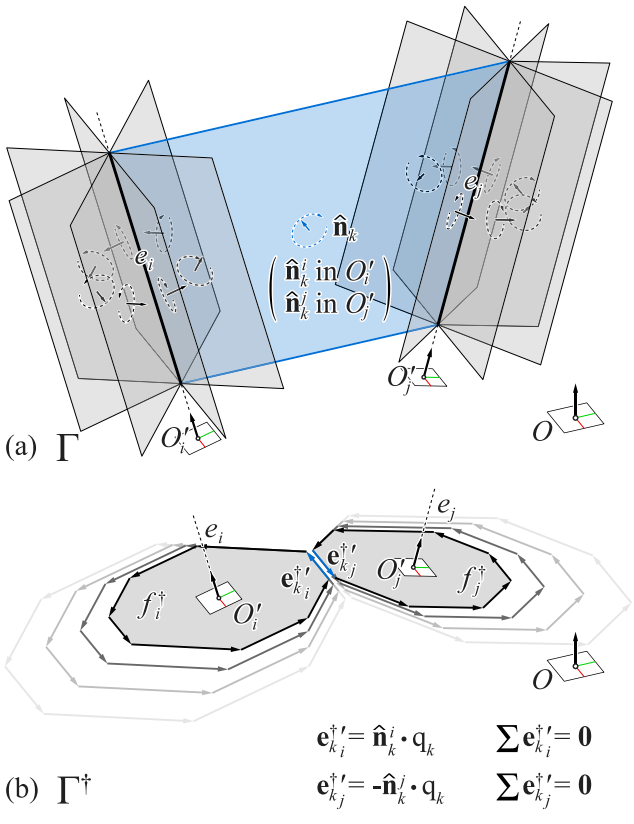

Fig. 4. Closing equations are written based on the local coordinate system aligned with each face $f_i^\dagger$. (a) Two internal edges e_i, e_j, and all connected faces f_k in a force diagram. (b) The corresponding internal face $f_i^\dagger$, $f_j^\dagger$, and the surrounding edges $e_k^\dagger$ in the form diagram. The closed face $f_i^\dagger$ or $f_j^\dagger$ means that the corresponding edge vectors add up to a zero vector.

$[f_{int} \times f_{int}]$ (or $[e_{int}^\dagger \times e_{int}^\dagger]$) diagonal matrix that describes the edge-face connectivity for a face $f_i^\dagger$, and its diagonal entries are defined as

$$\mathbf{P}_{j,j} = \begin{cases} +1 & \text{if edge } e_j^\dagger \text{ is an edge of face } f_i^\dagger \\ -1 & \text{if the opposite of edge } e_j^\dagger \text{ is an edge of face } f_i^\dagger. \\ 0 & \text{otherwise.} \end{cases}$$

Then, a local coordinate system of $f_i^\dagger$ can be constructed by taking its center as the local origin and its normal vector as the local z-direction (Fig. 4). This is followed by transforming the unit directional vectors of the $e_{int}^\dagger$ edges in the form diagram to the local coordinate system. Let $\mathbf{n}'_{x_i}$, $\mathbf{n}'_{y_i}$ be the $[e_{int}^\dagger \times 1]$ vectors whose entries are the x- and y-coordinates (respectively) of the transformed unit directional vectors in the local coordinate system of face $f_i^\dagger$. The two closing equations can be written as

$$(\mathbf{P}_i \mathbf{n}'_{x_i})^\intercal = 0 \quad (\mathbf{P}_i \mathbf{n}'_{y_i})^\intercal = 0. \tag{4}$$

Therefore, for all $f^\dagger$ faces, the simplified $[2f_{int}^\dagger \times e_{int}^\dagger]$ (or $[2e_{int} \times f_{int}]$) closing matrix $\mathbf{A}$ can be rewritten as

$$\mathbf{A} = \begin{pmatrix} \vdots \\ (\mathbf{P}_i \mathbf{n}'_{x_i})^\intercal \\ (\mathbf{P}_i \mathbf{n}'_{y_i})^\intercal \\ (\mathbf{P}_{i+1} \mathbf{n}'_{x_{(i+1)}})^\intercal \\ (\mathbf{P}_{i+1} \mathbf{n}'_{y_{(i+1)}})^\intercal \\ \vdots \end{pmatrix}. \tag{5}$$

2.2. Additional linear equations for edge constraints

Edge constraints set target lengths for the internal edges in the form diagram, and each constrained edge requires one additional constraint equation. For example, if the ith internal edge $e_i^\dagger$ is constrained to a target length l, its constraint equation may be written as

$$\mathbf{b}_i^\intercal \mathbf{q} = l \tag{6}$$

where $\mathbf{b}_i$ is the $[f_{int} \times 1]$ column vector with all entries zero (0) except at the index of $e_i^\dagger$ where it is one (1). For a total number of b constrained edges, all b edge constraint equations may be assembled in the matrix form as

$$\mathbf{B}\mathbf{q} = \mathbf{l} \tag{7}$$

where the rows of the $[b \times f_{int}]$ edge constraint matrix $\mathbf{B}$ are the row vectors $\mathbf{b}_i^\intercal$, and $\mathbf{l}$ is a $[b \times 1]$ column vector with the entries being the target lengths.

2.3. Additional linear equations for vertex constraints

Vertex constraints set target locations for the vertices, and they can be points, lines, or planes. A target point fixes all three coordinates of the constrained vertex, while target lines and target planes allow the vertex to move on them. Non-linear geometries like curves and curved surfaces are not covered in this study as these geometric constraints are difficult to represent through a linear equation system.

If there is only one constrained vertex for the form diagram, it can be achieved simply by a translation originating from the original vertex location to the target location, and it does not affect the lengths of the edges. Vertex constraints have influences on the edge lengths only if the number of constrained vertices d is greater than one, and, in such cases, additional vertex constraint equations are needed.

2.3.1. Points as vertex constraints

From all constraint types, point constraint is the simplest and hereby used to start the description. The vertex constraint equations are developed based on paired constrained vertices. For a pair of vertices $v_i^\dagger$ and $v_j^\dagger$ that are constrained to points $\mathbf{p}_i$ and $\mathbf{p}_j$, a path of edges that connects them can be found using Breadth First Search algorithm. Then the equations can be written since the edges on the path must have compatible lengths in order to fit in between the two vertices (Fig. 5).

Let $\mathbf{v}$ be the vector going from $\mathbf{p}_i$ to $\mathbf{p}_j$, and let u, v, w be the x-, y-, and z-component of $\mathbf{v}$, respectively. Let $\mathbf{Q}$ be a $[f_{int} \times f_{int}]$ (or $[e_{int}^\dagger \times e_{int}^\dagger]$) diagonal matrix, and its diagonal entries are defined as

$$\mathbf{Q}_{k,k} = \begin{cases} 0 & \text{if edge } e_k^\dagger \text{ not on the path} \\ +1 & e_k^\dagger \text{ on the path, its direction goes from } v_i^\dagger \text{ to } v_j^\dagger \\ -1 & e_k^\dagger \text{ on the path, its direction goes from } v_j^\dagger \text{ to } v_i^\dagger. \end{cases}$$

Then the three equations for this pair of constrained vertices can be described as

$$(\mathbf{Q}\mathbf{n}_x)^\intercal \mathbf{q} = \mathrm{u} \quad (\mathbf{Q}\mathbf{n}_y)^\intercal \mathbf{q} = \mathrm{v} \quad (\mathbf{Q}\mathbf{n}_z)^\intercal \mathbf{q} = \mathrm{w}. \tag{8}$$

Note that although there could be multiple paths connecting $v_i^\dagger$ and $v_j^\dagger$, the result does not depend on the choice of path.

When the number of constrained vertices d is greater than 2, all relative positions can be extensively described by $d-1$ pairs of constrained vertices. For the ith vertex pair, the three constraint equations can be written as

$$(\mathbf{Q}_i \mathbf{n}_x)^\intercal \mathbf{q} = \mathrm{u}_i \quad (\mathbf{Q}_i \mathbf{n}_y)^\intercal \mathbf{q} = \mathrm{v}_i \quad (\mathbf{Q}_i \mathbf{n}_z)^\intercal \mathbf{q} = \mathrm{w}_i, \tag{9}$$

and the $3 \times (d-1)$ constraint equations can be assembled in matrix form as

$$\mathbf{D}\mathbf{q} = \mathbf{d} \tag{10}$$

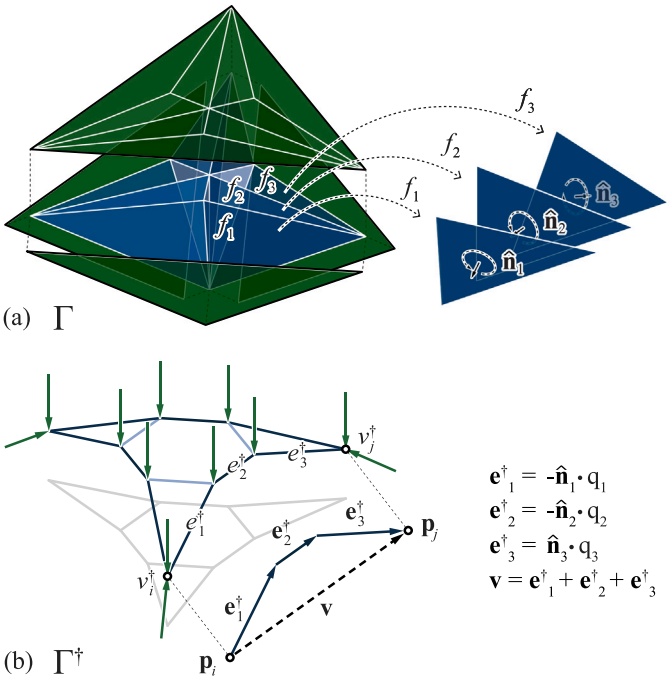

Fig. 5. The path of edges connecting a pair of constrained vertices $v_i^\dagger$ and $v_j^\dagger$ need to have compatible lengths to fit in between. (a) The normal directions of faces f_1, f_2, and f_3 are denoted as $\hat{\mathbf{n}}_1$, $\hat{\mathbf{n}}_2$, and $\hat{\mathbf{n}}_3$. (b) The path of vectors $\mathbf{e}_1^\dagger$, $\mathbf{e}_2^\dagger$, and $\mathbf{e}_3^\dagger$ that goes from vertex $v_i^\dagger$ to $v_j^\dagger$ should add up to $\mathbf{v}$, which is the vector starting from point constraint $\mathbf{p}_i$ to point constraint $\mathbf{p}_j$. Vectors $\mathbf{e}_1^\dagger$, $\mathbf{e}_2^\dagger$, and $\mathbf{e}_3^\dagger$ can be represented using $\hat{\mathbf{n}}_1$, $\hat{\mathbf{n}}_2$, and $\hat{\mathbf{n}}_3$ respectively. A negative sign is needed if a normal direction is opposite to the path direction.

where $\mathbf{D}$ is the $[3(d-1)\times f_{int}]$ vertex constraint matrix, written as

$$\mathbf{D} = \begin{pmatrix} \vdots \\ (\mathbf{Q}_i\mathbf{n}_x)^\intercal \\ (\mathbf{Q}_i\mathbf{n}_y)^\intercal \\ (\mathbf{Q}_i\mathbf{n}_z)^\intercal \\ \vdots \end{pmatrix}, \tag{11}$$

and $\mathbf{d}$ is the $[3(d-1)\times 1]$ column vector written as

$$\mathbf{d} = \begin{pmatrix} \vdots \\ u_i \\ v_i \\ w_i \\ \vdots \end{pmatrix}. \tag{12}$$

2.3.2. Mixed constraint types of points, lines, and planes

Unlike point constraints where vertex locations are completely fixed, a line constraint or plane constraint releases the restriction in one or two dimensions and therefore allows it to move. The vertex constraints equations for mixed constraint types are also based on constrained vertex pairs $v_i^\dagger$ and $v_j^\dagger$.

There is an assumption in this paper that one vertex of each pair, say $v_i^\dagger$, needs to be constrained to a point, while the constraint of $v_j^\dagger$ can be any type. That also means for d constrained vertices, at least one vertex of them needs to be constrained to a point. This is because when none of the vertices has a fixed position, the difference between their positions will be difficult to represent. Although this can be resolved by setting up a reference point for one of the vertices and introducing its coordinates as three extra variables, they still need to be manually chosen after solving the equation system. Therefore, it is convenient to have at least one vertex fixed from the beginning to simplify the problem.

Fig. 6 shows form diagram examples with line constant and plane constraint, all derived from the same force diagram illustrated in Fig. 5a. Since target lines and planes may have 3D orientations, a local coordinate system O' that is aligned with the line or plane can help reduce the number of equations needed and thus simplify the problem. If $v_j^\dagger$ is constrained to a line, then the local coordinate system can be constructed with the unit directional vector of the line as the local z-axis, and any point on the line as the origin. In such case the x- and y-differences of $v_i^\dagger$ and $v_j^\dagger$ in the local coordinate system are constant (Fig. 6a). If it is constrained to a plane, then the unit normal direction of the plane will be used as the local z-axis, and the origin can be any point on the plane. In this case, the z-difference of $v_i^\dagger$ and $v_j^\dagger$ in the local coordinate system is constant (Fig. 6b). All points and vectors changed to this local coordinate system are super-scripted with a prime symbol ($'$). Let $\hat{\mathbf{x}}'$, $\hat{\mathbf{y}}'$, $\hat{\mathbf{z}}'$ denote the unit directional vectors of the axes of the local coordinate system.

In the coordinate system, the path of edges connecting $v_i^\dagger$ and $v_j^\dagger$ must still have compatible lengths. In order to have the equations in the local system, the target location $\mathbf{p}_i$ and the unit direction vectors of all edges $e_i^\dagger$ need to perform a basis change. In the local coordinate system, let $\mathbf{v}'$ be the vector going from $\mathbf{p}_i$ to any point $\mathbf{p}_j$ on the constraint geometry. Here the origin of the local coordinate system is used as $\mathbf{p}_j$ for the sake of clarity; let u', v', w' be the x-, y-, and z-component of $\mathbf{v}'$ respectively; let $\mathbf{n}'_x$, $\mathbf{n}'_y$, $\mathbf{n}'_z$ be the $[e_{int}^\dagger \times 1]$ (equivalent to $[f_{int}\times 1]$) vectors whose entries are the x-, y-, and z-coordinates (respectively) of the transformed unit directional vectors of the $e_{int}^\dagger$ internal edges of the form diagram (equivalent to the normal vectors of the f_{int} internal faces of the force diagram). If $v_j^\dagger$ is constrained to a line, then only two equations are enough to describe the constraints and they are written as

$$\left(\mathbf{Q}\mathbf{n}'_x\right)^\intercal \mathbf{q} = u' \quad \left(\mathbf{Q}\mathbf{n}'_y\right)^\intercal \mathbf{q} = v'. \tag{13}$$

This is because only the local x- and y-coordinates of $v_j^\dagger$ are constrained, and the local z-coordinate is free to change. If $v_j^\dagger$ is constrained to a plane, then only one equation is needed, and it can be described as

$$\left(\mathbf{Q}\mathbf{n}'_z\right)^\intercal \mathbf{q} = w'. \tag{14}$$

Similarly, this is because only the local z-coordinate of $v_j^\dagger$ is constrained, and the local x- and y-coordinates are free to change.

There is an alternative approach for developing the constraint equations for a mixed type of constraint geometries without going through the change of coordinate system. Let $v_i^\dagger$ be a constrained vertex, and assume that $v_j^\dagger$ is bounded to a plane P given by the equation

$$\mathbf{n}_P\mathbf{p}_j = d_P \tag{15}$$

where $\mathbf{n}_P$ denotes a normal vector of the plane and $\mathbf{p}_j$ denotes the coordinate vector of the vertex $v_j^\dagger$.

Subtracting $\mathbf{n}_P\mathbf{p}_i$ from both sides of Eq. (15), the following equation can be obtained

$$\mathbf{n}_P(\mathbf{p}_j - \mathbf{p}_i) = d_P - \mathbf{n}_P\mathbf{p}_i. \tag{16}$$

The left-hand side of the equation above can be rewritten using Eq. (8)

$$\mathbf{n}_P\,(\mathbf{Q}\mathbf{N})^\intercal\,\mathbf{q} = d_P - \mathbf{n}_P\mathbf{p}_i \tag{17}$$

where the matrix $\mathbf{Q}$ describes a path from vertex $v_i^\dagger$ to $v_j^\dagger$ and $\mathbf{N}$ is the $[e_{int}^\dagger \times 3]$ matrix obtained by stacking the vectors $\mathbf{n}_x$, $\mathbf{n}_y$ and $\mathbf{n}_z$ column-wise .

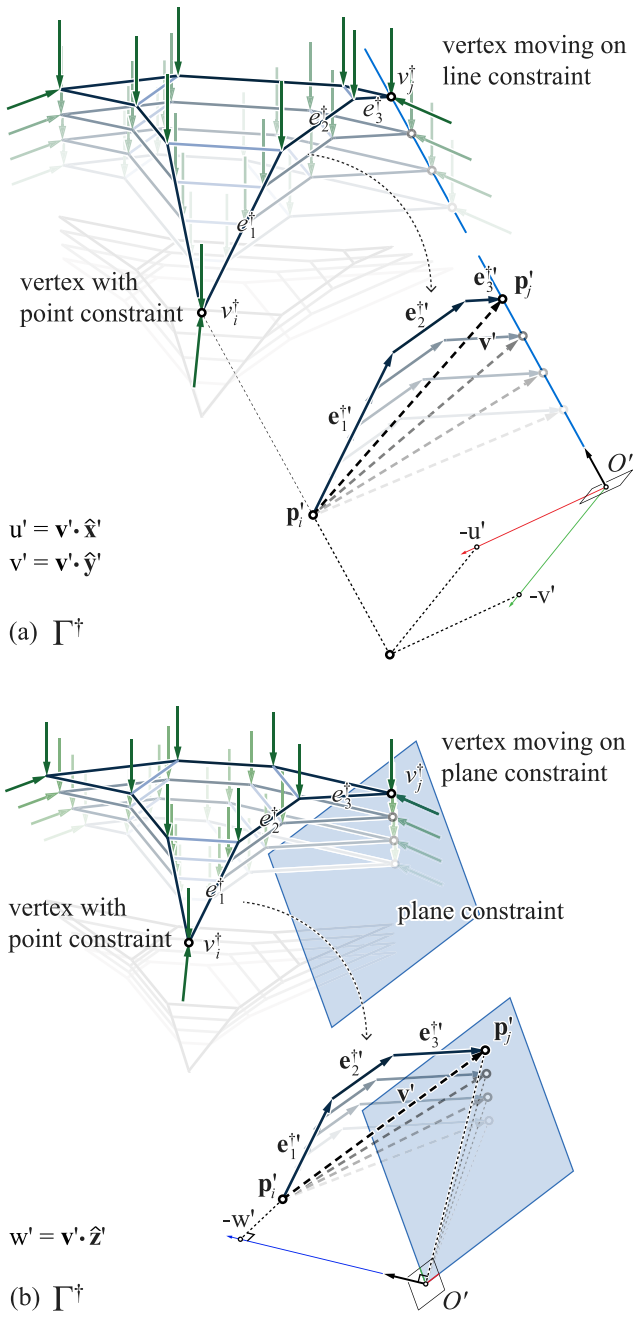

Fig. 6. (a) When $v_j^\dagger$ is constrained to a line, the x- and y-differences of $v_i^\dagger$ and $v_j^\dagger$ in the local coordinate system are constant. (b) When $v_j^\dagger$ is constrained to a plane, the z-difference of $v_i^\dagger$ and $v_j^\dagger$ in the local coordinate system is constant.

Therefore, a constraint that bounds a vertex to a plane provides one additional linear equation in **q**. Similarly, a constraint that bounds a vertex to a line provides two additional linear equations as the line can be represented as the intersection of two planes; a constraint that bounds a vertex to a point provides three additional linear equations as the point can be represented as the intersection of three planes.

To summarize the above, for d ($d > 2$) vertices constrained to p points, l lines, and n planes where d equals the sum of p, l, and n, a total number of $3(p-1)+2l+n$ equations are needed to describe the constraints. Accordingly, the shape of the vertex constraint matrix **D** becomes $[(3(p-1)+2l+n)\times f_{int}]$, and that of the column vector **d** becomes $[(3(p-1)+2l+n)\times 1]$.

2.4. Constrained linear closing equations and solutions

All closing equations, edge constraint equations, and vertex constraint equations are assembled as a system of non-homogeneous constrained linear closing equations written as follows:

$$\mathbf{Mq} = \mathbf{t}. \tag{18}$$

Here the $[(2e_{int}+b+3(p-1)+2l+n)\times f_{int}]$ matrix **M** is named the constrained closing matrix, obtained by vertically stacking the closing matrix **A**, the edge constraint matrix **B**, and the vertex constraint matrix **D**:

$$\mathbf{M} = \begin{pmatrix} \mathbf{A} \\ \mathbf{B} \\ \mathbf{D} \end{pmatrix}. \tag{19}$$

The $[(2e_{int}+b+3(p-1)+2l+n)\times 1]$ column vector **t** is obtained by vertically stacking the $[2e_{int}\times 1]$ zero vector, the vector **l**, and the vector **d**. Each solution of **q** represents a set of edge lengths of the form diagram that satisfy all closing requirements and constraints.

The form diagram may have over-constraining problems when constraints are incompatible with the planarity or competing with each other. In such cases, the constrained closing equations are overdetermined, and Eq. (18) has no solution. However, the least-square solution defines the closest solution to satisfying all closing and constraint requirements such that the Euclidean norm is minimized. Some point constraints can be replaced with line constraints or plane constraints to help mitigate over-constraining problems because they allow for more flexibility in vertex locations.

2.5. The geometric degrees of freedom of the form diagram

When the constrained closing equations are underdetermined, the solution may not be unique. In fact, in this case, the set of solutions forms an affine subspace inside $\mathbb{R}^{f_{int}}$ meaning that all solutions lie on a linear subspace translated by some vector.

The dimension of the linear subspace indicates the aforementioned GDoF of the form diagram. In other words, the GDoF is defined by the number of edges m that need to have their lengths assigned before an exact solution can be determined. This GDoF can be calculated according to the relationship between the numbers of independent equations and unknowns, where the rank r of the constrained closing matrix **M** is used:

$$m = f_{int} - r. \tag{20}$$

If a solution to Eq. (18) exists, the GDoF is always greater than or equal to zero. Zero GDoF indicates that there is a unique solution for **q** and the form diagram is determinate, whereas a positive GDoF means there are infinitely many solutions.

2.6. Algebraic representation of the force diagram

A system of linear equations can also be constructed to represent the input force diagram itself based on the closed coplanar edge vectors of its own f faces rather than the internal faces of its dual diagram such that the force diagram can be designed and modified. In this case, $\hat{\mathbf{n}}_i$ will be the unit direction vector of the edge e_i in the force diagram, the dimension of the closing matrix **A** becomes $[2f\times e]$, and the dimension of the constrained closing matrix **M** becomes $[(2f+b+3(p-1)+2l+n)\times e]$. The details are similar to the above described and hence omitted. The solutions of this system of equations show all force diagrams that have the same topology as well as edge and face orientations, and they represent different force distributions for the corresponding form diagram. This can be helpful because the force diagram can be adjusted without breaking the reciprocity between form and force diagrams. Related examples are presented in Section 4.2.

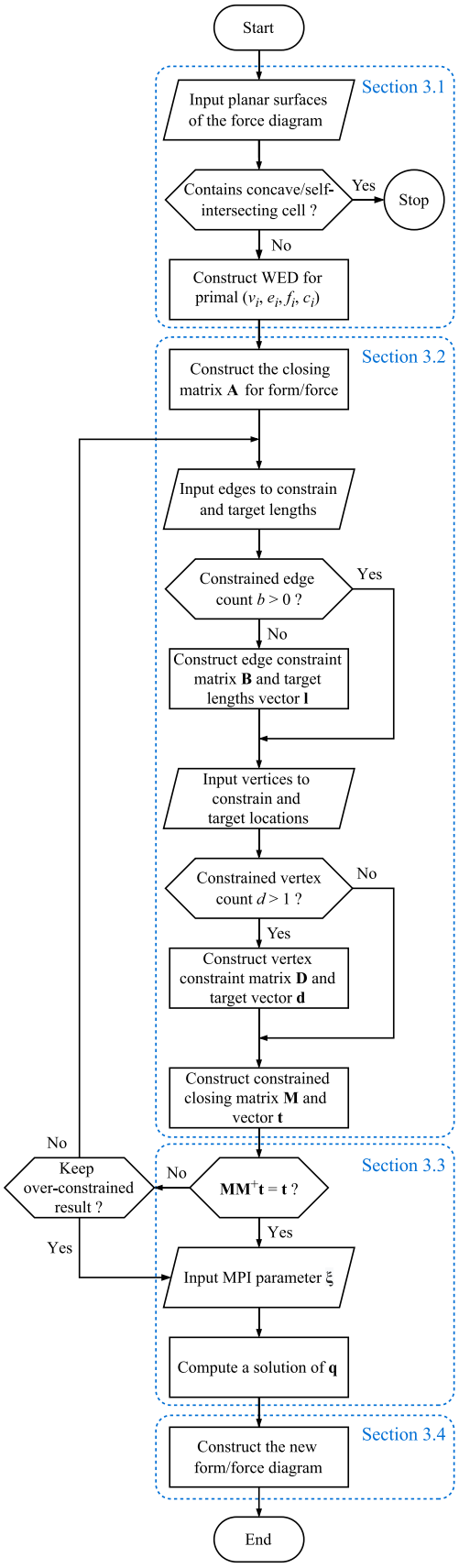

Fig. 7. The computational pipeline.

3. Computational implementation

The procedures described above are implemented in the 3D modeling environment of Rhino3D® and Grasshopper3D® as a plug-in named PolyFrame 2 [38] that is readily available for researchers and designers to explore. The handy data management framework of the reciprocal diagrams is developed by Nejur and Akbarzadeh [28] based on a winged-edge data structure (WED) proposed by Kremer et al. [39]. Matrix computations are accomplished using Numpy.NET, a .NET binding for the scientific computing library Numpy [40]. The computational pipeline is shown in a flowchart (Fig. 7).

In general, it takes a force diagram as input and creates the algebraic representation of either the force diagram itself or its corresponding form diagram, up to the user's choice. The algebraic representation encompasses all possible solutions of edge lengths **q** within the solution space. Through a second input of ξ, a specific solution can be selected from the solution space. After getting a solution **q** for the edge lengths, the corresponding form diagram or a new force diagram can be constructed. The examples in this section utilize only the algebraic representation of form diagrams for the sake of clarity and simplicity.

3.1. Construction of the data structure

The construction of the data structure starts from a group of planar surfaces with connecting boundary edges which represent the force diagram. Then the vertices, edges, faces, and cells can be extracted and the data structure can be found through the process described by Nejur and Akbarzadeh [28]. As mentioned in this previous publication, although the data structure allows self-intersecting faces and cells to exist through later manipulation, the initial construction of the data structure cannot accept self-intersecting surfaces, i.e. all c cells in the force diagram need to be convex, otherwise, the topology cannot be defined properly. This handy data structure allows for fast queries of topological and geometrical information from the force diagram, which is a foundation for the following steps.

3.2. Construction of the constrained closing equations

The construction of the closing matrix **A** has been explained thoroughly by Hablicsek et al. [29] as well as in Section 2.1. The construction edge constraint matrix **B** is straightforward and has also been adequately covered in Section 2.2. The construction of vertex constraint matrix **D** needs to find a connecting path for each constrained vertex pair $v_i^\dagger$ and $v_j^\dagger$ in the form diagram. The Breadth-First search (BFS) algorithm can be used for this purpose. It starts by using one constrained vertex as the origin to perform a BFS to traverse the network of the form diagram. During the BFS, a search tree can be constructed with a constrained vertex being the root node, other vertices being its descendant, and links being the connecting edges. The constrained vertex pairs can be obtained from this search tree. For all other constrained vertices, their connecting paths with the origin vertex can be obtained by traversing the tree up until the root is reached (Fig. 8). Once the paths are found, the vertex constraint matrix **D** can be written as described in Section 2.3. The constrained closing matrix **M** can then be obtained by Eq. (19).

3.3. Solution of the constrained linear closing equations

All solutions of the non-homogeneous constrained linear closing equation system (Eq. (18)) may be calculated using the Moore–Penrose inverse (MPI) method [41,42] that provides a fast and interactive way to construct a dual diagram with well-distributed edge lengths [29]. The Moore–Penrose inverse of the constrained matrix **M** is denoted $\mathbf{M}^+$. A solution to Eq. (18) exists only when the following identity holds

$$\mathbf{MM^+t = t}. \tag{21}$$

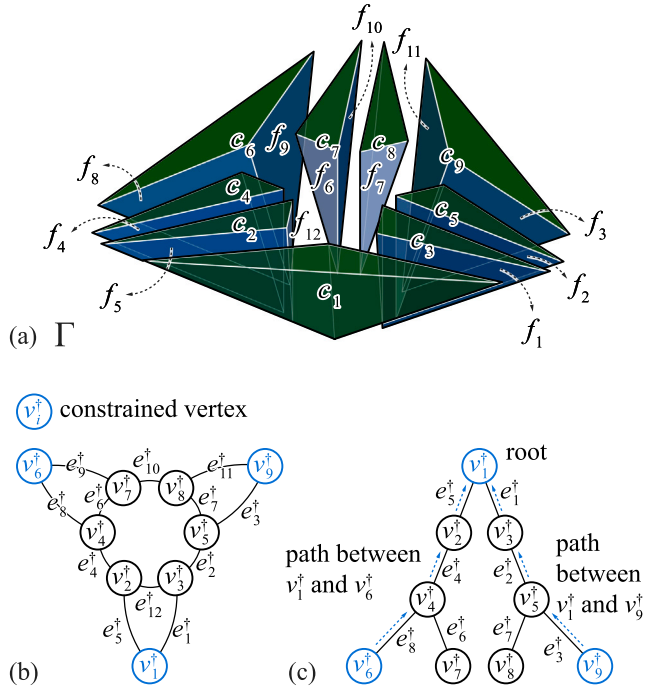

Fig. 8. (a) Exploded force diagram showing individual cells. (b) The adjacency graph of the cells. (c) The BFS tree.

Moreover, all solutions of $\mathbf{q}$ are given by

$$\mathbf{q} = \mathbf{M}^{+}\mathbf{t} + [\mathbf{I} - \mathbf{M}^{+}\mathbf{M}]\xi \tag{22}$$

where $\mathbf{I}$ is the $[f_{int} \times f_{int}]$ identity matrix and ξ is a user-specified $[f_{int} \times 1]$ column vector input. When $\mathbf{M}$ has full column rank, which means $\mathbf{I} - \mathbf{M}^{+}\mathbf{M}$ is a zero matrix, the solution $\mathbf{q} = \mathbf{M}^{+}\mathbf{t}$ is unique. Otherwise, the solution $\mathbf{q}$ depends on the parameter ξ that users can freely choose. In general, the ith entry of ξ loosely controls the signed length of edge $e_i^\dagger$ when it is not constrained. In fact, if ξ is a solution of Eq. (18), meaning that $\mathbf{M}\mathbf{M}^{+}\xi = \xi$, then Eq. (22) yields

$$\mathbf{q} = \mathbf{M}^{+}\mathbf{t} + [\mathbf{I} - \mathbf{M}^{+}\mathbf{M}]\xi = \mathbf{M}^{+}\mathbf{t} + \xi - \mathbf{M}^{+}\mathbf{t} = \xi, \tag{23}$$

showing that if ξ is the input parameter, then the output of the MPI method agrees with ξ. In other words, the MPI method returns the solution ξ. On the contrary, if edge $e_i^\dagger$ has its length constrained, the ith entry of ξ will have no effect on the solution.

The most computationally intensive task is computing $\mathbf{M}^{+}$. However, once computed and stored, assigning ξ and getting $\mathbf{q}$ is extremely fast, allowing for the efficient generation of a wide range of solutions. It is noted in the previous research paper [29] that the accumulation of numerical error may lead to an incorrect result. This can be resolved by cutting off small singular values when calculating $\mathbf{M}^{+}$.

3.4. Construction of the form diagram

After getting a solution for $\mathbf{q}$ which contains signed lengths for all $e_{int}^\dagger$ edges, the vertex positions can be calculated through methods such as the algebraic approach and graph-search approach [29] for the construction of the form diagram. Once again, the BFS algorithm is used. It traverses from a vertex $v_i^\dagger$ that has known coordinates, then gradually visits the adjacent vertices and calculates their positions based on the lengths of the connecting edges until all vertices are visited (Fig. 9). If the form diagram has constrained vertices, the starting vertex $v_i^\dagger$ can be chosen from them. Otherwise, if no vertex is constrained, meaning that they can be constructed anywhere in the 3D space, a starting vertex needs to be specified and its position needs to be assigned. Finally, the tension or compression state of the $e^\dagger$ edges can be determined based on the procedure described by Hablicsek et al. [29].

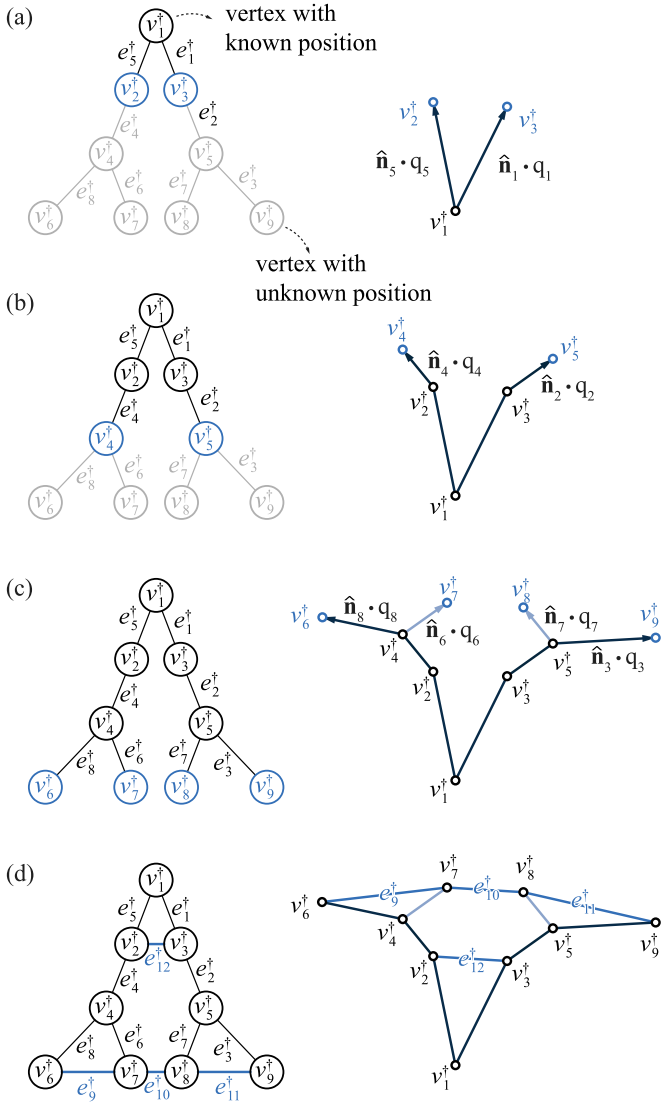

Fig. 9. The construction of the form diagram once a solution $\mathbf{q}$ is obtained.

3.5. Detection of over-constraining problems

As mentioned in Section 2.4, the form diagram may be over-constrained if the vertex and edge constraints are incompatible with the planarity or competing with each other. The system of equations allows the over-constraining problems to be explicitly detected.

When a form diagram is over-constrained, Eq. (21) does not hold, meaning that the equation system is overdetermined and there will be no solution. This serves as a criterion to assess whether a form diagram is over-constrained. However, when ξ is a zero vector, Eq. (22) gives a minimum norm solution (namely $\mathbf{M}^{+}\mathbf{t}$) that is the closest to satisfying all constraints, meaning that

$$\|\mathbf{M}\mathbf{q} - \mathbf{t}\| \geq \|\mathbf{M}\mathbf{M}^{+}\mathbf{t} - \mathbf{t}\| \tag{24}$$

holds for any possible vector $\mathbf{q}$ where $\|.\|$ denotes the Euclidean norm. Although this minimum norm solution of edge lengths can be used to construct a form diagram using the process described in Section 3.4, the resulting form diagram is usually incorrect due to nonplanarity and edge angle deviations. Fig. 10a shows the minimum norm solution of an over-constrained form diagram generated from the same force diagram as Fig. 8a. Vertices $v_1^\dagger$, $v_6^\dagger$, and $v_9^\dagger$ are constrained to points $\mathbf{p}_1$, $\mathbf{p}_2$, and $\mathbf{p}_3$ respectively, where the three vertex constraints cannot be

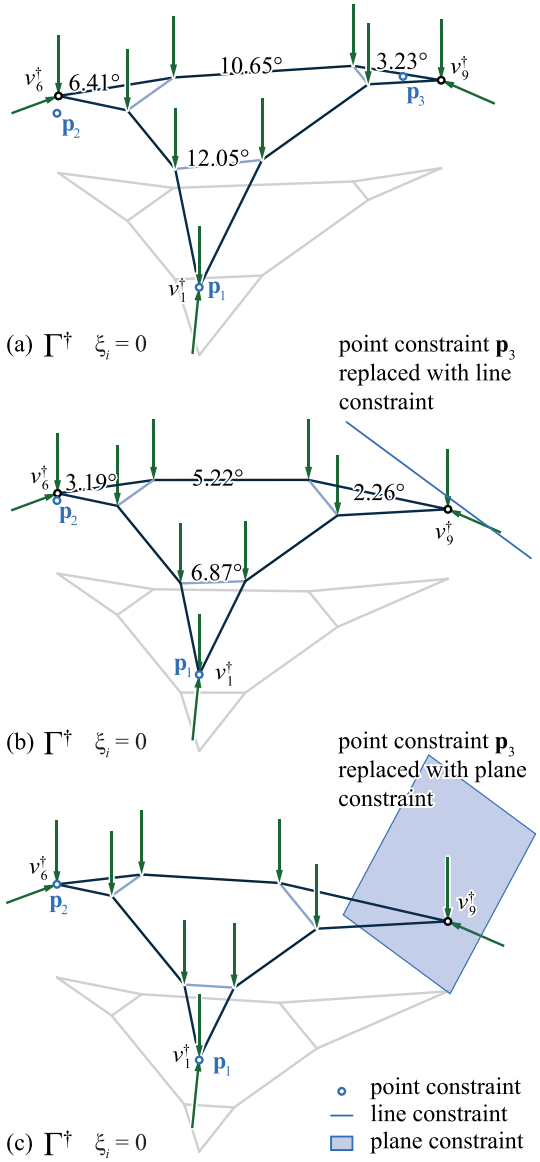

Fig. 10. The problem of over-constraining can be mitigated or resolved by using more forgiving constraint types such as lines and planes. Over-constrained form diagrams usually have angle deviations on some edges as shown in (a) and (b). (a) The minimum norm solution of an over-constrained form diagram with vertices $v_1^\dagger$, $v_6^\dagger$, and $v_9^\dagger$ constrained to points $\mathbf{p}_1$, $\mathbf{p}_2$, and $\mathbf{p}_3$ respectively. The positions of $v_6^\dagger$ and $v_9^\dagger$ deviate from the target points. (b) The angle deviations are reduced after replacing point constraint $\mathbf{p}_3$ with a line constraint. (c) The over-constraining issue is resolved after replacing point constraint $\mathbf{p}_3$ with a plane constraint.

simultaneously satisfied. The non-zero angle deviations are displayed on each respective edge.

The problem of over-constraining may be mitigated or resolved by using more forgiving constraint types such as lines and planes. Fig. 10b shows the minimum norm solution of the form diagram with point constraint $\mathbf{p}_3$ replaced with a line constraint. This form is less over-constrained as can be seen from the reduced angle deviation. Fig. 10c shows that the over-constraining issue is resolved by replacing the point constraint $\mathbf{p}_3$ with a plane constraint.

3.6. Comparison with the iterative method

Several benchmark models are tested to compare the algebraic approach with the iterative approach described by Nejur and Akbarzadeh [28] (Fig. 11). The algebraic approach outperforms the iterative approach in terms of both precision and speed. The iterative approach needs a maximum allowed deviation angle for perpendicularity as the stopping criteria, and the smaller the angle is the longer it will take. In most cases, the result cannot reach zero angle deviation. For the algebraic approach, the results without over-constraining are precise and have almost zero angle deviation due to the nature of close-form solutions. For over-constraining problems, the algebraic method can explicitly tell whether a form or force diagram is over-constrained as described in Section 3.5, whereas the iterative methods cannot. In terms of speed, different constraint conditions are less likely to affect the runtime of the algebraic approach, whereas the runtime of the iterative approach heavily depends on how different the goal is from the initial state. In all benchmark models, the algebraic approach is approximately tens to hundreds of times faster than the iterative approach that uses 1 degree as the maximum allowed deviation. Besides, the algebraic method works diagrams with concave polyhedrons, which is difficult to achieve using the iterative method.

3.7. Computation complexity

Another performance test is also presented to show the computation complexity of the algebraic implementation (Fig. 12). The tested models are 3DGS approximations of Schwarz P minimal surface in different resolutions [25]. Despite the exponential growth in runtime, the algorithm can efficiently process complex geometries within a relatively short timeframe. All tests mentioned above were performed single-threaded on a laptop with a Xeon E3-1505M CPU and 32 GB of RAM.

4. Application

The previous section has shown that the enhanced algebraic 3DGS equipped with edge and vertex constraints offers comprehensive and flexible control while maintaining a high computation efficiency in exploring the solution space. Those capabilities are further explored through various case studies in this section. As mentioned earlier, the algebraic representation can be established for both form and force diagrams. Thus, the case studies are classified into those two categories: the generation of various form diagrams and the manipulation of force diagrams.

4.1. Generation of various form diagrams

4.1.1. Convenient form control based on site conditions

The algebraic approach allows the form to be generated and easily controlled with desired constraint geometries, which is convenient for accommodating real-world site conditions. Fig. 13 shows a simple design scenario where a bridge is needed above a 10 m-wide river. In order to design a funicular bridge that fits between the banks, lines and points can be used as constraint geometries to constrain the four corners of the bridge. For different site conditions, the bridge geometry can be easily adjusted by moving the constraint points of the constrained vertices.

4.1.2. Intuitive exploration of the solution space

Due to the geometric degrees of freedom (GDoF), it is possible to generate an infinitude of drastically different form diagrams from the same force diagram. This solution space of the form diagram can be explored by specifying a different input parameter ξ. Although this is theoretically doable, modifying ξ can be tedious and counter-intuitive. This is because ξ is a $[f_{int} \times 1]$ vector, and its dimension can be large for complex form diagrams. Moreover, it is difficult to directly perceive the relationship between ξ and the resulting form diagram. Edge and vertex constraints can provide a different approach for intuitively exploring the solution space. For example, Fig. 14 illustrates different bridge

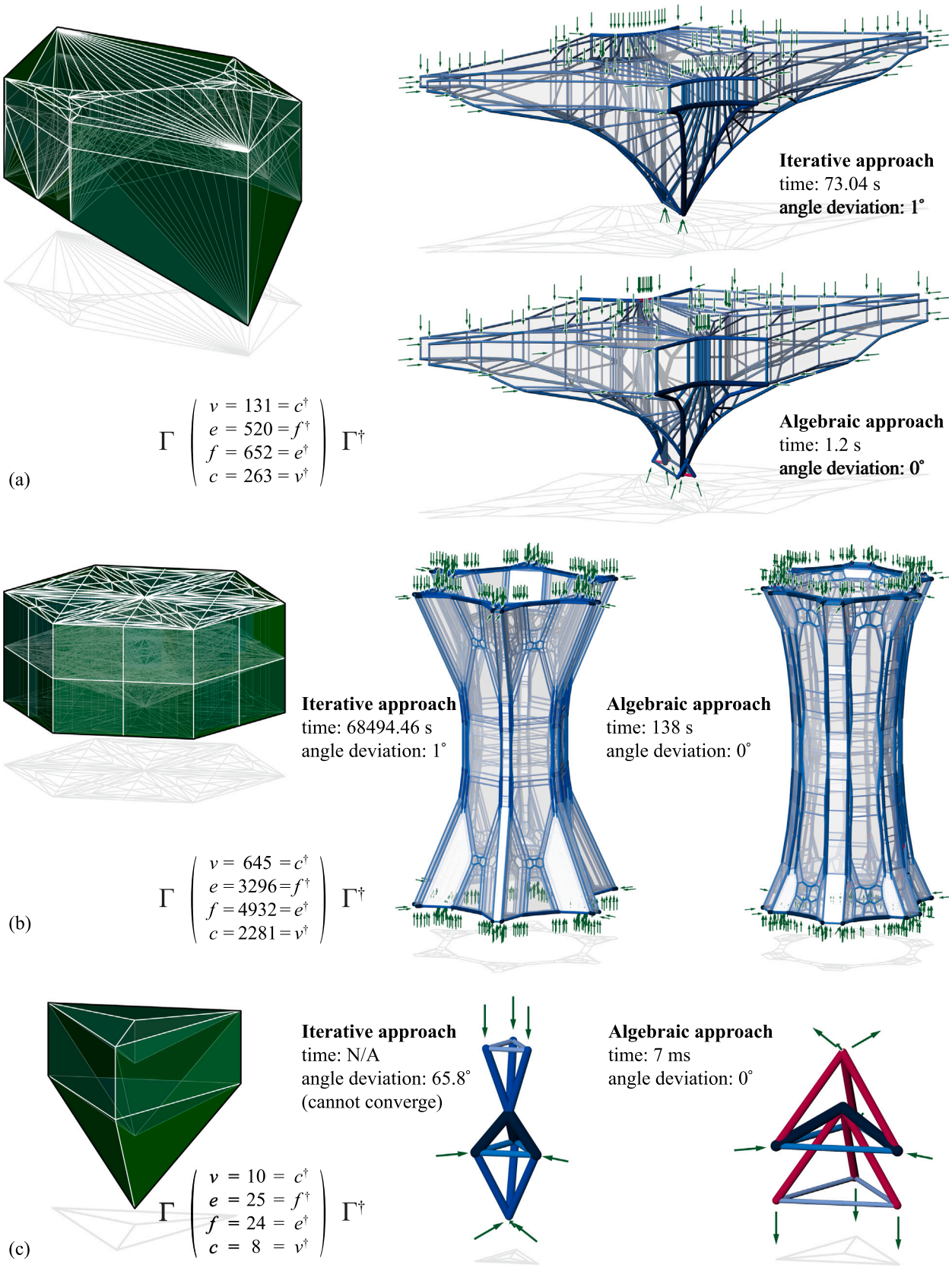

Fig. 11. A comparative performance study between the algebraic approach and the iterative approach.

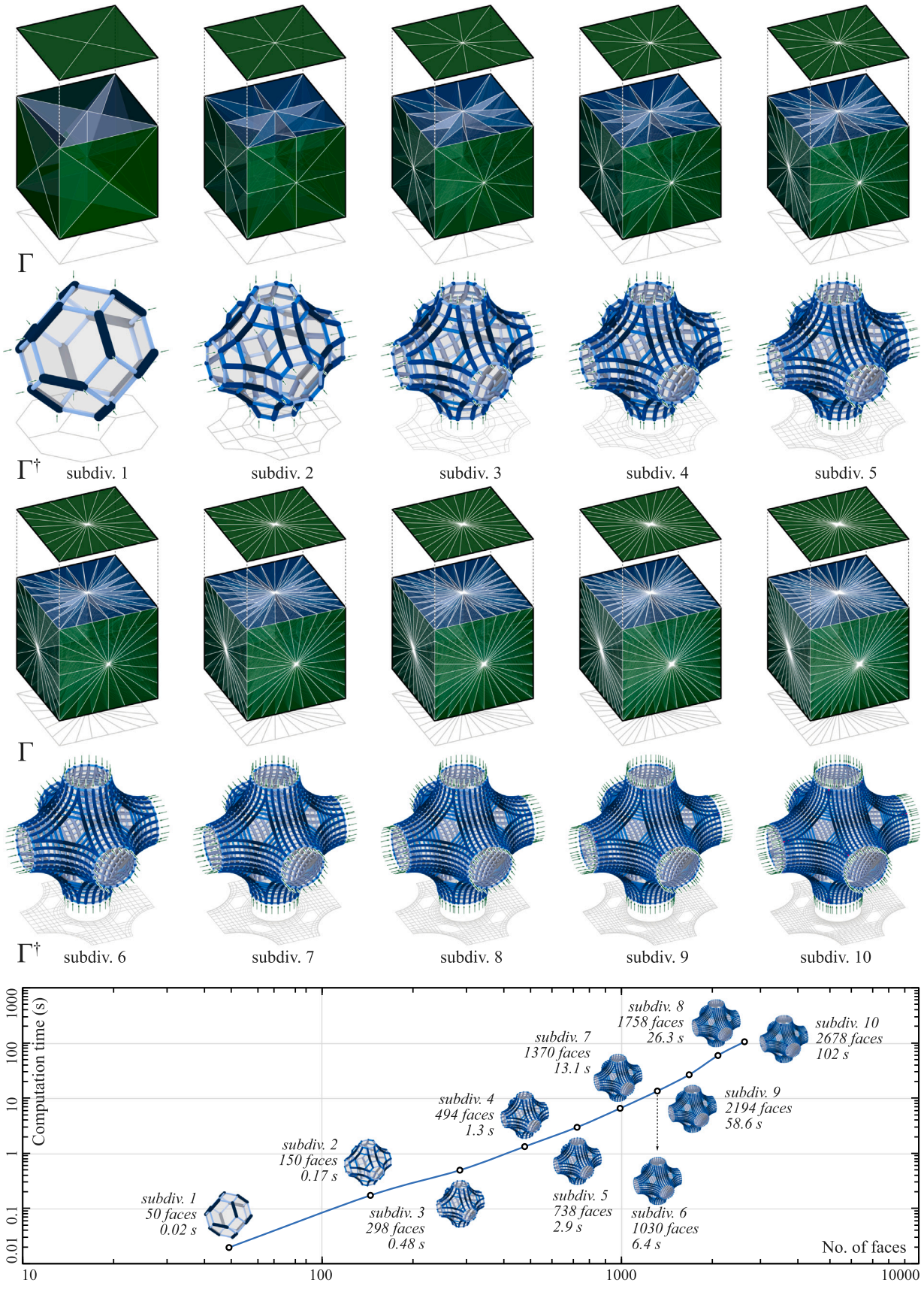

Fig. 12. Runtime of 3DGS approximations of Schwarz P minimal surface in different resolutions.

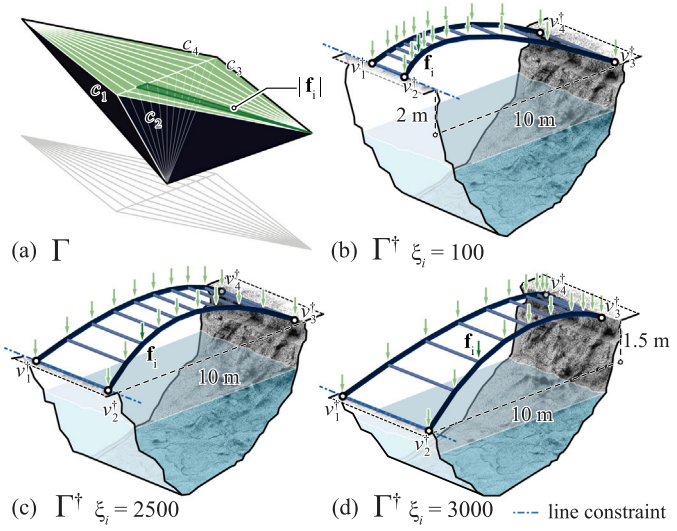

Fig. 13. Simple bridge designs with constraints set up in accordance with different site conditions.

geometries designed for the same site. Those different geometries have different forms and states of tension and compression. This is achieved by precisely modifying the height and end positions of the bridge. Fig. 15 shows a force diagram that is shared by a family of funicular shells depicted in Fig. 16, whose four corner vertices $v_1^\dagger$, $v_2^\dagger$, $v_3^\dagger$, and $v_4^\dagger$ are constrained to different points. Changing their target point locations results in an array of forms, ranging from compression or tension only to tension–compression combined, from synclastic to anticlastic.

4.1.3. Form-finding for various loading scenarios

A force diagram is only able to represent the equilibrium of a single loading scenario, and the corresponding form diagram will only be in equilibrium under that specific situation. In practice, the initial force diagram usually represents the combination of structural self-weight and the primary static loads. However, dynamic loads must also be taken into consideration if their magnitudes are comparable to the dead loads. Fig. 17 shows a 3D bridge design example that accounts for vertical live loads by using multiple force diagrams to indicate different live load situations. All force diagrams have the same topology but different areas on the top faces. In Fig. 17a, each top face of Γ_1 represents the tributary dead load of the structure with an area of $|\mathbf{f}|$, while Γ_2 to Γ_8 have several enlarged top faces. Each enlarged face has an additional area of $|2\mathbf{f}|$, indicating the live load at the corresponding vertex. For each force diagram, a corresponding form diagram is generated and constrained to the same span, load position, and height (Fig. 17b). By overlaying those form diagrams, a new bridge geometry can be obtained, which contains load paths for all live load cases (Fig. 17c). While this new bridge geometry may alter the static load represented by Γ_1, this preliminary exploration provides insights for further design developments.

4.1.4. Auxetic structures

Auxetic metamaterials show unusual reaction behaviors under uniaxial compression or tension forces. They contract perpendicular to the load direction under an applied compressive force and expand under tension. This behavior, namely negative Poisson's ratio, heavily depends on the internal concave cellular geometries [44,45]. A previous research project has demonstrated that the algebraic formulation of graphic statics is a suitable tool for systematically designing two-dimensional auxetic metamaterials with concave geometric arrangements [46]. This may also be extended to three dimensions.

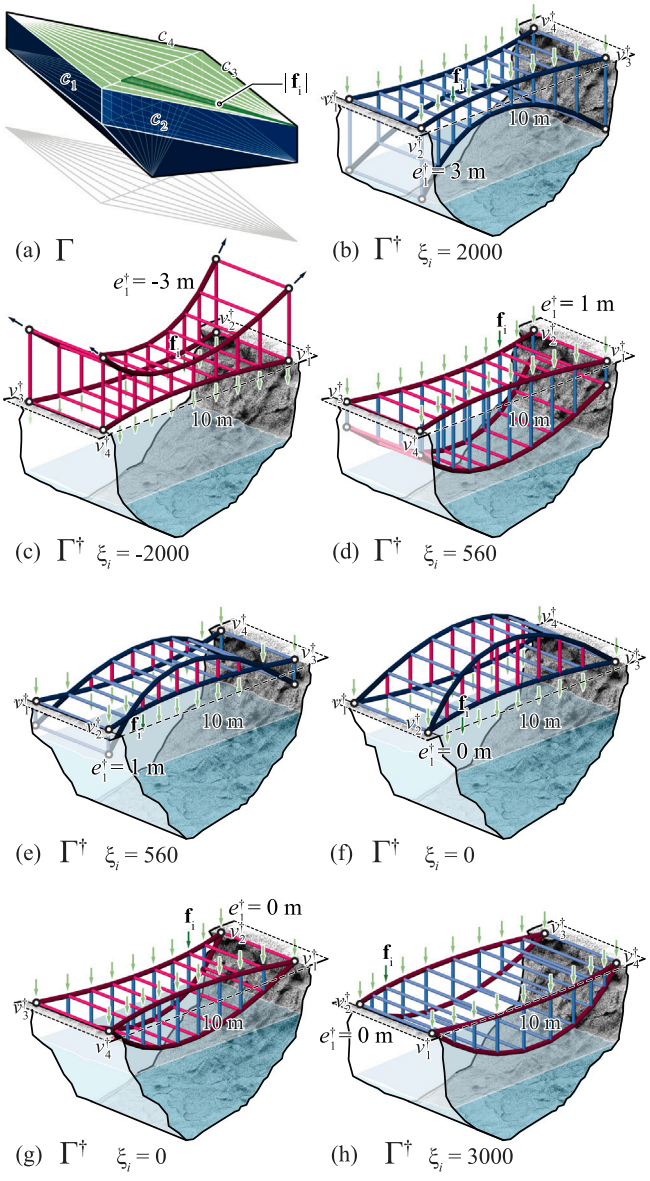

Fig. 14. A family of bridge designs with different geometries and states of tension and compression. (a) The shared force diagram. (b) A compression-only geometry. (c) A tension-only geometry. (d)–(h) Tension–compression combined geometries.

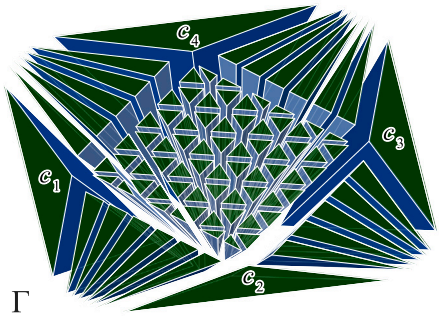

Fig. 15. The exploded force diagram shared by the family of funicular shells.

Algebraic 3DGS is capable of creating those 3D geometries with concave cells in two different ways. First, it can start from a convex

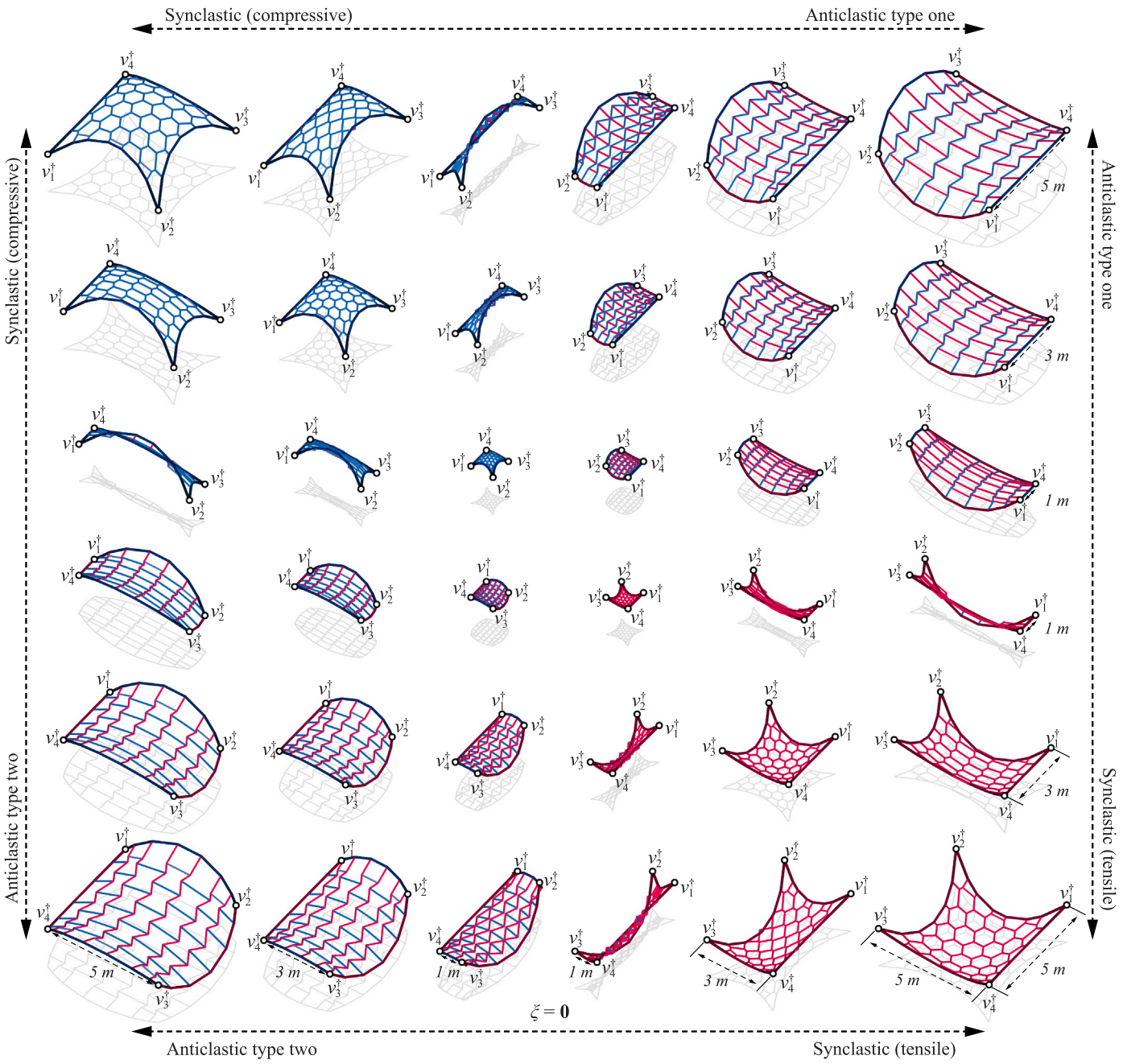

Fig. 16. A family of funicular shells generated based on the same force diagram, and their four corner vertices $v_1^\dagger$, $v_2^\dagger$, $v_3^\dagger$, and $v_4^\dagger$ are constrained to different points.

configuration and modify the strut lengths based on the GDoF to turn certain cells into a concave configuration (Fig. 18a–e). Alternatively, the concave configuration can be generated directly from a force diagram with concave cells (Fig. 18f–j). The auxetic behaviors of the two structures are then verified using the finite element method (FEM). The specimens have a footprint of 50 mm by 50 mm, with all edges materialized as 1.2 mm diameter struts using a linear elastic material. The bottom vertices are fixed in z-direction, and a total vertical force of 10 kN is distributed to all top vertices. The simulation results are shown in Fig. 18e and 18j, which indicate a negative Poisson's ratio. These approaches have the potential to significantly accelerate the development and exploration of various auxetic metamaterials. It provides a powerful tool for the science and engineering communities to further investigate and develop new auxetic metamaterials, and explore their potential applications in areas such as biomedicine, aerospace, and advanced manufacturing.

4.2. Manipulation of force diagrams

4.2.1. Removal of unwanted external forces

One additional advantage of having control over edge lengths is the ability to manipulate face areas, this is especially useful for force diagram manipulations as the face areas directly relate to force magnitudes in the form diagram. A structural form found using 3DGS sometimes has undesired external forces as they are difficult to provide in reality. By constraining their corresponding faces to zero area in the force diagram, such forces can be removed and a new internal force distribution can be obtained. Compared to the existing approach of constraining face areas through quadratic formulations [35], changing face areas through edge lengths is more intuitive and computationally efficient. Fig. 19 illustrates examples from two built design projects [19,22] where the lateral external forces were difficult to provide in the actual construction. In Fig. 19b where the external side faces of the force diagram

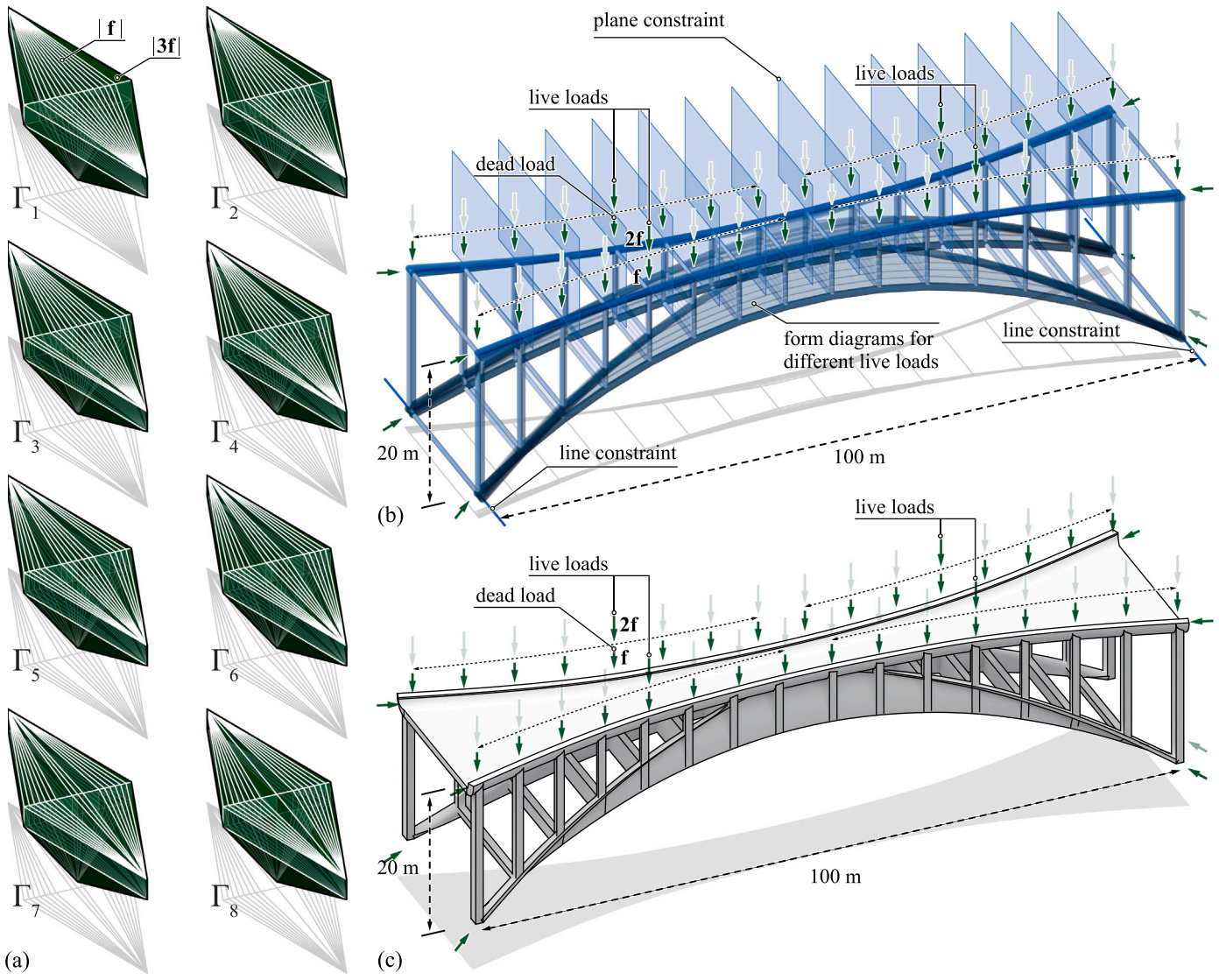

Fig. 17. A 3D bridge design example that accounts for vertical live loads by using multiple force diagrams to indicate different situations of live loads. (a) Eight force diagrams that show different live load situations. Γ_1 represents the dead loads of the structure, each top face has an area of $|\mathbf{f}|$ indicating the tributary dead load on the top vertex. Γ_2 to Γ_8 have several enlarged top faces, representing vertices with applied live loads. (b) The four bottom-corner vertices of each form diagram are constrained to two line constraints with 100 m spacing, and the top vertices with external loads are constrained to equally-spaced plane constraints. The longest vertical edge of each form diagram is constrained to 20 m. (c) A final bridge design is obtained by overlaying those form diagrams.

are rectangles, their areas can be reduced to zero by constraining the length of edge e_1 to zero. In the more complex example shown in Fig. 19e where the external side faces are trapezoids, their areas are turned to zero through self-intersection. The area calculation for self-intersecting faces has been exhaustively explained by Akbarzadeh and Hablicsek [35] and is captured in Fig. 20. In the new equilibrium states of both examples, some edges are turned into tension, which can be materialized and treated differently from the compression members. The main limitation of area control through edge lengths is that a face with n edges requires $n-2$ edge length constraints to determine its area, and this can be tedious when n is large.

4.2.2. Self-stressed structure

A force diagram may have all its external face areas reduced to zero while having non-zero internal face areas (Fig. 21). In this case, the corresponding form diagram represents a self-stressed structure, because the edges have forces without any external load. This can be used to detect the static indeterminacy in a form diagram since statically determinate structures cannot have any state of self-stress. However, when the internal faces cannot keep a non-zero area, it does not mean that the form diagram is statically determinate. This is due to the polyhedral nature of the force diagram, which only shows a subset of all possible equilibrium states.

5. Discussion and conclusion

This paper introduces an improved algebraic formulation for 3DGS equipped with edge and vertex constraints, which enables the precise manipulation of edge lengths and vertex locations of the reciprocal diagrams while maintaining the computation speed and flexibility of exploring the solution space. A software tool that implements this enhanced formulation within the environment of Rhino3D®and

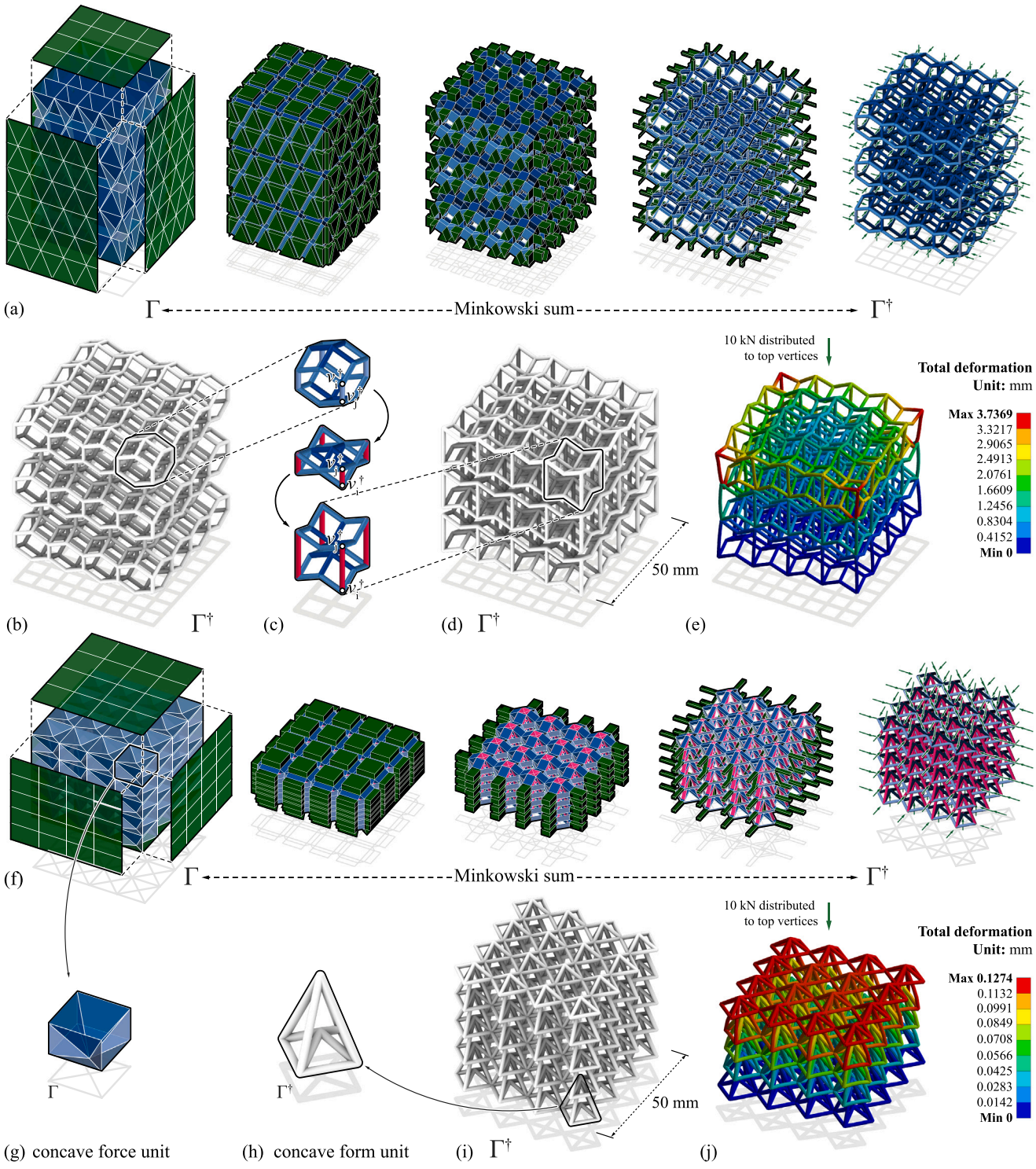

Fig. 18. Two ways of generating structures with auxetic behavior using algebraic 3DGS. (a)–(e) A form diagram is turned from a convex configuration to a concave configuration through edge length manipulation. (f)–(j) a form diagram with concave configuration directly generated from a force diagram with concave cells. (a) and (f) use Minkowski sum [43] to show the transition between force and form diagrams and to illustrate the reciprocity. (e) and (j) show analysis results using FEM that indicate negative Poisson's ratios.

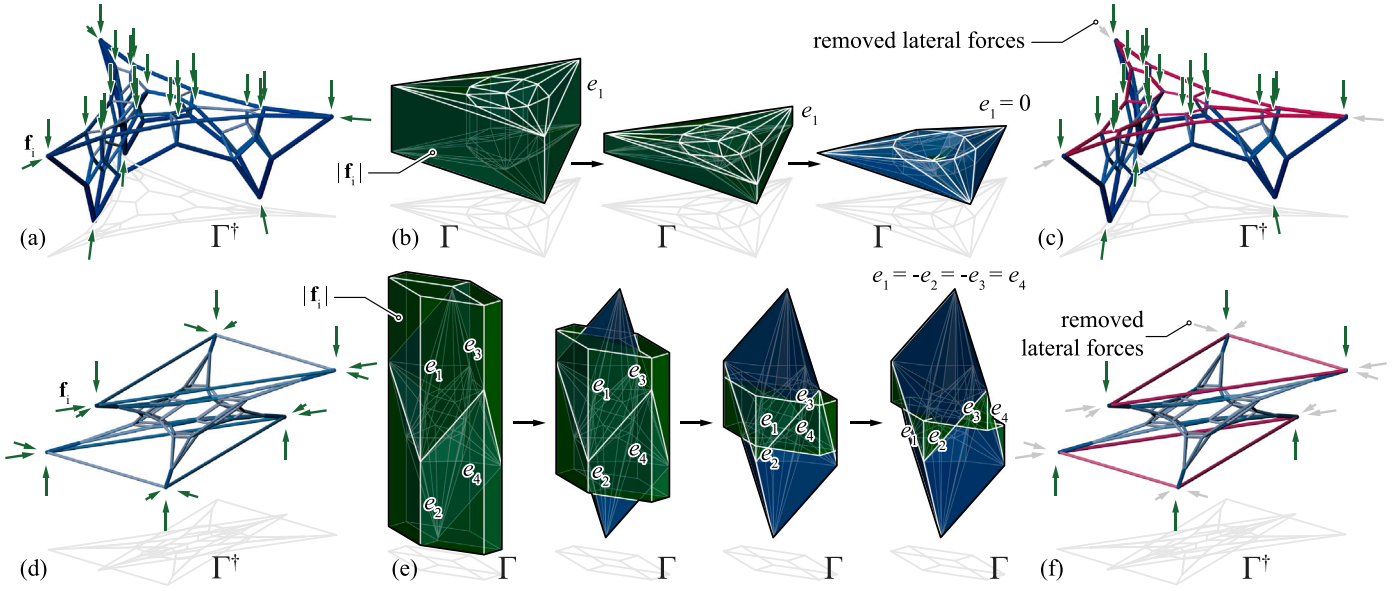

Fig. 19. Two built design projects are used as examples to show the removal of lateral external forces. Those forces are removed by constraining the corresponding faces to zero area in the force diagram.

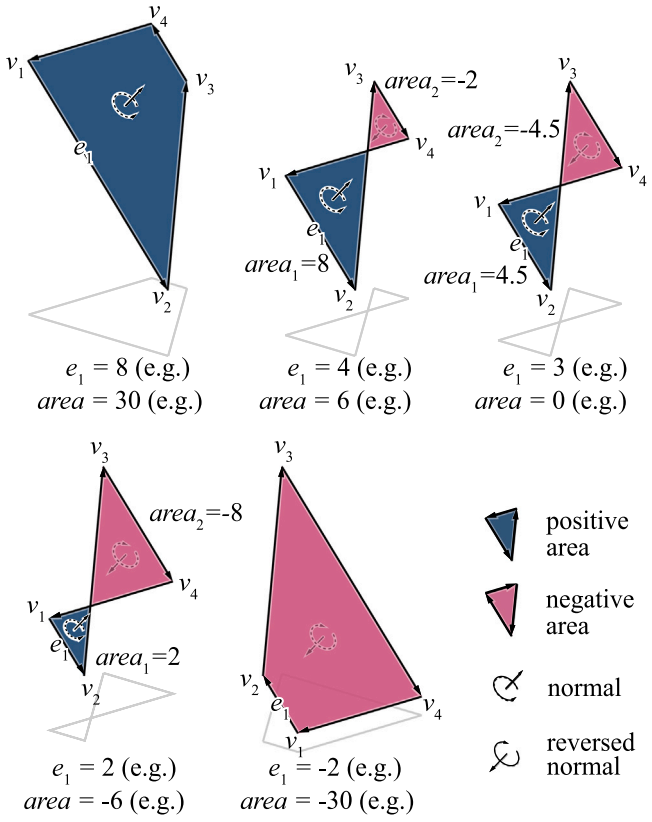

Fig. 20. A series of faces showing how the face areas are calculated. If any part of a face has a reversed normal based on the right-hand rule, its area will be considered negative.

Grasshopper3D®has been released and is readily available for researchers, designers, and educators to exploit. Furthermore, the potential applications of this tool were demonstrated through various case studies that showcase the flexibility and versatility of this approach.

There are numerous potential applications of this enhanced formulation and software tool. In addition to the demonstrated case studies, this approach can also be applied to teaching because its efficient computation enables interactivity and allows users to interact with the inputs and see outputs instantaneously. Furthermore, this algebraic formulation can be transferred to web platforms to provide a widely accessible educational tool for 3DGS.

This proposed formulation and its implementation can be further improved to add more convenient features and increase the ease of use. One potential improvement is to incorporate linear programming approaches to enable more flexible constraint conditions. At present, the edge constraints are set as fixed numbers, which can result in over-constraining problems when too many edges have target lengths. By extending the target length from a single number to a range, linear programming can greatly reduce the chance of over-constraining. Linear programming can also help find compression or tension-only forms, which are difficult to achieve for complex geometries under the algebraic framework. Moreover, in advanced use cases, linear programming can be employed for load-path optimizations [47].

Another area where further improvement can be made is in the computation speed of the software tool. To achieve this, high-performance numerical packages can be leveraged to enhance the efficiency of the computation process. The developer notes that the use of Numpy.NET is approximately four times slower than using Numpy directly in Python, suggesting significant potential for improvement. By utilizing better numerical packages, the computation speed can be boosted, making it more feasible for larger and more complex projects.

Overall, the proposed formulation and its implementation offer a promising algebraic framework for efficient form-finding using 3DGS, with significant potential for further development and refinement. The incorporation of linear programming approaches and high-performance numerical packages can lead to a more user-friendly and efficient tool, which can facilitate the exploration of a wider range of structural solutions for a variety of design challenges. With its versatility and accessibility, this proposed formulation can be useful not only for professional engineers and architects but also for students and educators seeking to explore the possibilities of 3DGS.

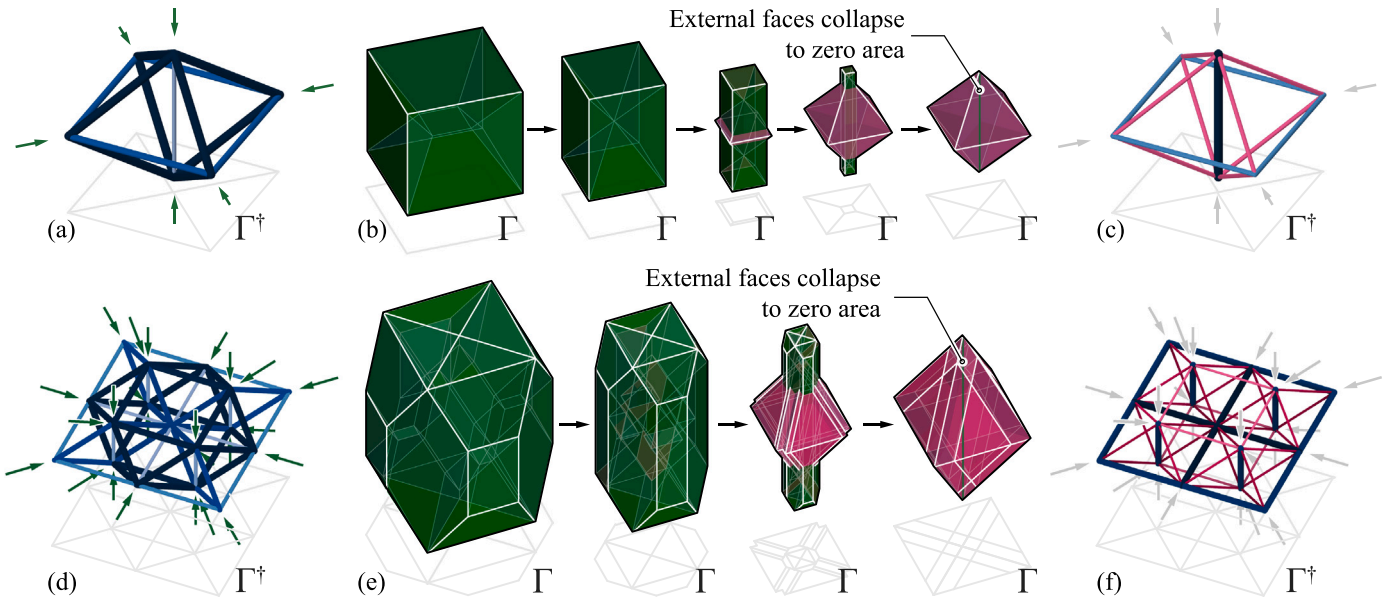

Fig. 21. Some force diagrams may allow all external faces constrained to zero area while keeping internal face areas, and the corresponding form diagrams will represent a self-stressed structure. Some faces of the force diagram are colored in red because their area is flipped to negative.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Masoud Akbarzadeh reports financial support was provided by National Science Foundation.

Data availability

Data will be made available on request.

Acknowledgments

This research is funded by the National Science Foundation CAREER Award (NSF CAREER-CMMI 1944691) and the National Science Foundation Future Eco Manufacturing Research Grant (NSF FMRG-CMMI 2037097) to Masoud Akbarzadeh.

References

[1] Rankine WJM. A manual of applied mechanics. London: Griffin; 1858.
[2] Rankine WJM. Principle of the equilibrium of polyhedral frames. Philos Mag Ser 4 1864;27(180):92.
[3] Maxwell JC. On reciprocal figures and diagrams of forces. Philos Mag Ser 4 1864;27(182):250–61.
[4] Culmann K. Die graphische statik. Zürich: Verlag Meyer und Zeller; 1864.
[5] Bow RH. Economics of construction in relation to framed structures. London: Spon; 1873.
[6] Cremona L. Graphical statics: Two treatises on the graphical calculus and reciprocal figures in graphical statics. Translated By Thomas Hudson Beare. Oxford: Clarendon Press; 1890.
[7] Wolfe WS. Graphical analysis: A text book on graphic statics. New York: McGraw-Hill Book Co. Inc.; 1921.
[8] Billington DP. Robert Maillart's bridges: The art of engineering. Princeton University Press: N9ZAYMXBmnMC; 1979, Google-Books-ID.
[9] Maxwell JC. On reciprocal figures, frames and diagrams of forces. Trans R Soc Edinburgh 1870;26(1):1–40.
[10] D'Acunto P, Jasienski J-P, Ohlbrock PO, Fivet C, Schwartz J, Zastavni D. Vector-based 3D graphic statics: A framework for the design of spatial structures based on the relation between form and forces. Int J Solids Struct 2019;167:58–70.
[11] Akbarzadeh M. 3D graphic statics using reciprocal polyhedral diagrams [Ph.D. thesis], Zurich, Switzerland: ETH Zurich; 2016.
[12] Lee J. Computational design framework for 3D graphic statics (Ph.D. thesis), ETH Zurich; 2018, Accepted: 2019-03-14T06:19:28Z.
[13] Konstantatou M, D'Acunto P, McRobie A. Polarities in structural analysis and design: n-dimensional graphic statics and structural transformations. Int J Solids Struct 2018;152–153:272–93.
[14] Lu Y, Cregan M, Chhadeh P, Seyedahmadian A, Bolhassani M, Schneider J, et al. All glass, compression-dominant polyhedral bridge prototype: form-finding and fabrication. In: Inspiring the next generation: Proceedings of the 7th international conference on spatial structures and the annual symposium of the IASS. Surrey, UK; 2021, p. 326–36.
[15] Akbari M, Lu Y, Akbarzadeh M. From design to the fabrication of shellular funicular structures. In: 2021 association for computer aided design in architecture annual conference. Association for computer aided design in architecture annual conference, ACADIA 2021, Virtual, Online: ACADIA; 2021.
[16] Lu Y, Seyedahmadian A, Chhadeh PA, Cregan M, Bolhassani M, Schneider J, et al. Funicular glass bridge prototype: Design optimization, fabrication, and assembly challenges. Glass Struct Eng 2022;7(2):319–30.
[17] Chai H, Bolhassani M, Akbarzadeh M. Structural form-finding of multi-span undulating funicular beam structure. In: Proceedings of IASS 2022 symposium affiliated with APCS 2022 conference. Beijing, China; 2022.
[18] Lu Y, Alsalem T, Akbarzadeh M. A method for designing multi-layer sheet-based lightweight funicular structures. J Int Assoc Shell Spat Struct 2022;63(4):252–62.
[19] Bolhassani M, Akbarzadeh M, Mahnia M, Taherian R. On structural behavior of a funicular concrete polyhedral frame designed by 3D graphic statics. Structures 2018;14:56–68.
[20] Heisel F, Lee J, Schlesier K, Rippmann M, Saeidi N, Javadian A, et al. Design, cultivation and application of load-bearing mycelium components. Int J Sustain Energy Dev 2018;6(1):296–303.
[21] Bhooshan V, Louth H, Bieling L, Bhooshan S. Spatial developable meshes. In: Gengnagel C, Baverel O, Burry J, Ramsgaard Thomsen M, Weinzierl S, editors. Impact: Design with all senses. Cham: Springer International Publishing; 2020, p. 45–58.
[22] Akbarzadeh MG. Saltatur. In: ACADIA 2020: Distributed proximities / Volume II: Projects [Proceedings of the 40th annual conference of the association of computer aided design in architecture (ACADIA) ISBN 978-0-578-95253-6]. Online and Global. 24-30 October 2020. Edited By M. Yablonina, a. Marcus, S. Doyle, M. Del Campo, V. Ago, B. Slocum. 108-113.. CUMINCAD; 2020, URL http://papers.cumincad.org/cgi-bin/works/paper/acadia20_108p.
[23] Liu Y, Lu Y, Akbarzadeh M. Kerf bending and zipper in spatial timber tectonics: A polyhedral timber space frame system manufacturable by 3-axis CNC milling machine. In: 2021 Association for computer aided design in architecture annual conference, ACADIA 2021, November 3, 2021 - November 6, 2021. Association for computer aided design in architecture annual conference, ACADIA 2021, Virtual, Online: ACADIA; 2021.
[24] Naboni R, Zomparelli A. Complex modelling automation for 3D polyhedral structures built with additive formwork manufacturing. Archit, Struct Constr 2023.
[25] Akbari M, Mirabolghasemi A, Bolhassani M, Akbarzadeh A, Akbarzadeh M. Strut-based cellular to shellular funicular materials. Adv Funct Mater 2022;32(14):2109725.
[26] Lee J, Mele TV, Block P. Disjointed force polyhedra. Comput Aided Des 2018;99:11–28.

[27] PSL. PolyFrame. 2018, URL https://www.food4rhino.com/app/polyframe.
[28] Nejur A, Akbarzadeh M. PolyFrame, efficient computation for 3D graphic statics. Comput Aided Des 2021;134:103003.
[29] Hablicsek M, Akbarzadeh M, Guo Y. Algebraic 3D graphic statics: Reciprocal constructions. Comput Aided Des 2019;108:30–41.
[30] Akbarzadeh M, Hablicsek M. Geometric degrees of freedom and non-conventional spatial structural forms. In: Impact: Design with all senses, design modelling symposium. Berlin, Germany; 2020.
[31] Micheletti A. On generalized reciprocal diagrams for self-stressed frameworks. Int J Space Struct 2008;23(3):153–66.
[32] Van Mele T, Block P. Algebraic graph statics. Comput Aided Des 2014;53:104–16.
[33] Alic V, Åkesson D. Bi-directional algebraic graphic statics. Comput Aided Des 2017;93:26–37.
[34] Akbarzadeh M, Hablicsek M. Geometric degrees of freedom and non-conventional spatial structural forms. In: Gengnagel C, Baverel O, Burry J, Ramsgaard Thomsen M, Weinzierl S, editors. Impact: Design with all senses. Cham: Springer International Publishing; 2020, p. 3–17.
[35] Akbarzadeh M, Hablicsek M. Algebraic 3D graphic statics: Constrained areas. Comput Aided Des 2021;141:103068.
[36] Akbarzadeh M, Van Mele T, Block P. On the equilibrium of funicular polyhedral frames and convex polyhedral force diagrams. Comput Aided Des 2015;63:118–28.
[37] Akbarzadeh M, Hablicsek M, Guo Y. Developing algebraic constraints for reciprocal polyhedral diagrams of 3D graphic statics. In: Mueller C, Adriaenssens S, editors. Proceedings of the international association for shell and spatial structures (IASS) symposium: Creativity in structural design. MIT, Boston, US; 2018.
[38] PS. PolyFrame 2. 2023, URL https://www.food4rhino.com/app/polyframe-2.
[39] Kremer M, Bommes D, Kobbelt L. OpenVolumeMesh – A versatile index-based data structure for 3D polytopal complexes. In: Jiao X, Weill J-C, editors. Proceedings of the 21st international meshing roundtable. Berlin, Heidelberg: Springer; 2013, p. 531–48.
[40] Harris CR, Millman KJ, van der Walt SJ, Gommers R, Virtanen P, Cournapeau D, et al. Array programming with NumPy. Nature 2020;585(7825):357–62.
[41] Moore EH. On the reciprocal of the general algebraic matrix. Bull Amer Math Soc 1920;26(9):385–96.
[42] Penrose R. A generalized inverse for matrices. Math Proc Camb Phil Soc 1955;51(3):406–13.
[43] McRobie A. Maxwell and rankine reciprocal diagrams via Minkowski sums for two-dimensional and three-dimensional trusses under load. Int J Space Struct 2016;31(2–4):203–16.
[44] Reid DR, Pashine N, Wozniak JM, Jaeger HM, Liu AJ, Nagel SR, et al. Auxetic metamaterials from disordered networks. Proc Natl Acad Sci 2018;115(7):E1384–90.
[45] R Reid D, Pashine N, S Bowen A, R Nagel S, Pablo JJd. Ideal isotropic auxetic networks from random networks. Soft Matter 2019;15(40):8084–91.
[46] Hablicsek M, Akbarzadeh M. Structural form-finding of auxetic materials using graphic statics. In: Proceedings of IASS symposium and spatial structures conference 2020/21, inspiring the next generation. Guildford, UK; 2021.
[47] Beghini LL, Carrion J, Beghini A, Mazurek A, Baker WF. Structural optimization using graphic statics. Struct Multidiscip Optim 2013;49(3):351–66.

Yao Lu is currently a Ph.D. Candidate at the Polyhedral Structures Laboratory, Weitzman School of Design, University of Pennsylvania. He has a Master of Science in Matter Design Computation from Cornell University, a Master of Architecture and a Bachelor in Engineering from Tongji University.

Márton Hablicsek is currently a tenured assistant professor of mathematics at Leiden University in the Netherlands, and a research associate at PSL. He was a postdoctoral fellow at the University of Copenhagen and the Centre of Symmetry and Deformation and the University of Pennsylvania. He holds a Ph.D. in Mathematics from the University of Wisconsin-Madison and his research lies in algebraic geometry and its applications.

Masoud Akbarzadeh is an Assistant Professor of Architecture at Weitzman School of Design, University of Pennsylvania. He is also the director of Polyhedral Structures Laboratory (PSL), which focuses on Structures and Advanced Technologies. He holds a D.Sc. from the Institute of Technology in Architecture, ETH Zurich, and a Master of Science in Architecture Studies and a Master of Architecture from MIT.