Computer-Aided Design 155 (2023) 103435

Contents lists available at ScienceDirect

Computer-Aided Design

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

Constrained Form-Finding of Tension–Compression Structures using Automatic Differentiation

Rafael Pastrana [a,*], Patrick Ole Ohlbrock [b], Thomas Oberbichler [c], Pierluigi D'Acunto [d], Stefana Parascho [e]

[a] School of Architecture, Princeton University, United States of America
[b] Chair of Structural Design, ETH Zürich, Switzerland
[c] Chair of Structural Analysis, Technische Universität München, Germany
[d] Professorship of Structural Design, Technische Universität München, Germany
[e] Lab for Creative Computation, École Polytechnique Fédérale de Lausanne, Switzerland

ARTICLE INFO

Article history:
Received 30 September 2021
Received in revised form 25 August 2022
Accepted 9 October 2022

Keywords:
Form-finding
Shape optimization
Automatic differentiation
Structural design
Design tool
Combinatorial equilibrium modeling

ABSTRACT

This paper proposes a computational approach to form-find pin-jointed bar structures subjected to combinations of tension and compression forces. The generated equilibrium states can meet structural and geometrical constraints via gradient-based optimization. We achieve this by extending the combinatorial equilibrium modeling (CEM) framework in three important ways. First, we introduce a new topological object, the auxiliary trail, to expand the range of structures that can be form-found with the framework. Then, we leverage automatic differentiation (AD) to obtain an exact value of the gradient of the sequential and iterative calculations of the CEM form-finding algorithm, instead of a numerical approximation. Finally, we encapsulate our research developments in an open-source design tool written in Python that is usable across different CAD platforms and operating systems. After studying four different structures – a self-stressed tensegrity, a tree canopy, a curved bridge, and a spiral staircase – we demonstrate that our approach enables the solution of constrained form-finding problems on a diverse range of structures more efficiently than in previous work.

1. Introduction

A *form-finding* method generates the shape and the internal force state of a structure so that, given a design load and a set of support conditions, the structure is in a state of *static equilibrium* [1,2]. Different numerical form-finding methods exist that fall into one of three categories: stiffness-matrix [3,4], dynamic equilibrium [5–7] and geometric [8–15] methods. An in-depth review of this taxonomy is found in [1,16]. Advancements in all categories have been propelled over the last decade by the development of multiple computational design tools [17–21].

In a form-finding method, a structure is often modeled as a discrete network of straight bars that are connected by pinned joints at the nodes of the network. The design load is transferred from one node to another exclusively through axial forces in the bars and a state of static equilibrium is reached when the sum of forces incident to every node is zero. Conceptually, the axial-dominant load-carrying mechanism of a structure in static equilibrium implies that it will require less material volume to withstand the applied design load [22].

1.1. The CEM framework

The Combinatorial Equilibrium Modeling (CEM) framework is a geometric form-finding method for 3D structures modeled as pin-jointed bar networks and subjected to combinations of tension and compression forces [23–25]. Examples of such mixed structures are space frames, bridges, stadium roofs, multistory buildings and tensegrities.

The CEM framework consists of two operative parts: the CEM form-finding algorithm and an optimization-based constrained form-finding solver. This framework represents a structure with three diagrams: a topology diagram T describes the internal connectivity and the internal tension–compression state of a structure. Meanwhile, a form diagram F and a force diagram F^* display the geometric and force attributes of the calculated state of static equilibrium. Fig. 1 presents a graphical overview of how these components interact and Section 2 provides a thorough review of their theoretical underpinnings.

A distinctive feature of the CEM form-finding algorithm is that static equilibrium is computed sequentially and iteratively, unlike other geometric form-finding methods [8–10,26]. Nevertheless, this computation approach is precisely what allows the algorithm to ensure the generation of a static equilibrium state for a mixed

* Corresponding author.
E-mail address: arpastrana@princeton.edu (R. Pastrana).

https://doi.org/10.1016/j.cad.2022.103435
0010-4485/

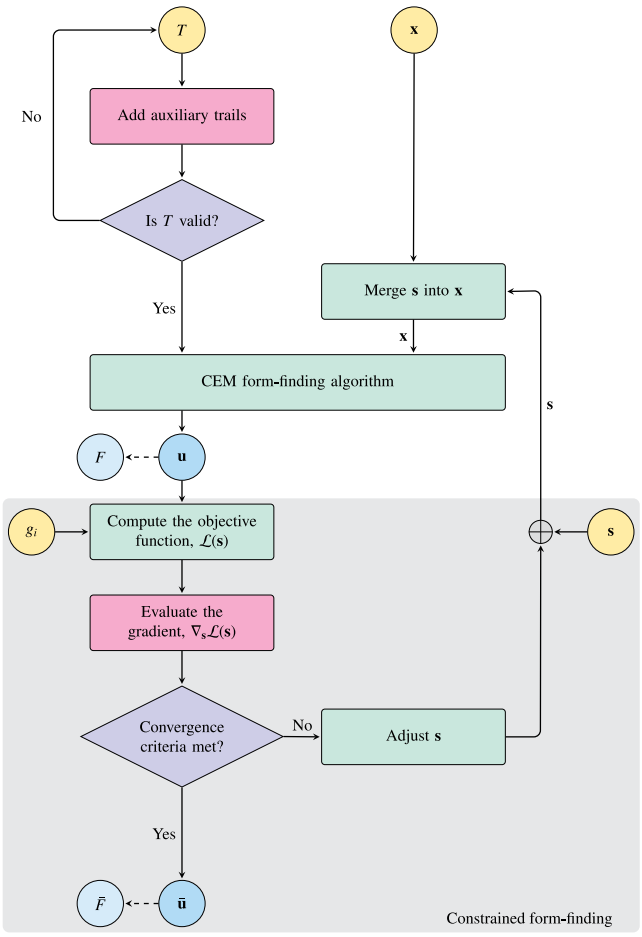


Fig. 1. Overview of the Combinatorial Equilibrium Modeling (CEM) framework and our extensions. The inputs to the framework are a topology diagram T and the design parameters $\mathbf{x}$. The CEM form-finding algorithm calculates a state of static equilibrium $\mathbf{u}$ from which a form diagram F can be optionally constructed. To find a constrained equilibrium state $\bar{\mathbf{u}}$ that best satisfies force and geometric constraints g_i, the CEM framework minimizes an objective function $\mathcal{L}(\mathbf{s})$ by iteratively adjusting the optimization parameters $\mathbf{s}$ using the gradient $\nabla_{\mathbf{s}}\mathcal{L}(\mathbf{s})$ until the convergence criteria $\mathcal{L}(\mathbf{s}) \leq \epsilon$ or $\|\nabla\mathcal{L}(\mathbf{s})\| \leq \kappa$ is reached. The vector of optimization parameters $\mathbf{s}$ contains a user-defined portion of the design parameters $\mathbf{x}$. Two of the extensions we make in this paper are highlighted in pink. Auxiliary trails simplify the construction of a larger variety of valid topology diagrams T. Reverse-mode automatic differentiation computes an exact value of $\nabla_{\mathbf{s}}\mathcal{L}(\mathbf{s})$ thus allowing for more efficient and stable solutions to constrained form-finding problems. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

tension–compression structure as long as the input topology diagram T fulfills the requirements listed in Section 1.2.1.

The CEM form-finding algorithm allows designers to explore different equilibrium states for a fixed diagram T by manipulating a portion of the nodal positions, and the lengths and the forces in a subset of the bars of a structure (see Section 2.2). However, realistic structural design scenarios often pose constraints a priori where it is important to find a specific equilibrium state that best satisfies them. Examples of such constraints include fitting a target shape [27,28], restraining bar forces and lengths [29–31], and controlling the reaction forces at the supports of a structure [20,32,33]. The challenge is that while constraints can be readily enumerated in a design problem, it is often not straightforward to discern what combination of input design parameters is conducive to the envisioned result.

One way to solve such constrained form-finding problems is to manually tweak the input parameters until the required constraints are met, one by one. This can quickly become a cumbersome process. Instead, the CEM framework form-finds 3D structures subjected to design constraints following an automatic approach: the constraints are aggregated into a single objective function and a computer assists the designer in calculating the values of the design parameters that minimize the function via gradient-based optimization [24,34].

1.2. Limitations of the CEM framework

The CEM framework as presented in [23,24,34] faces two limitations. One of them is related to its topological modeling flexibility and the other to its computational performance when solving a constrained form-finding problem.

1.2.1. Strict topological modeling rules

Every topology diagram T must fulfill two requirements in order to be considered a valid input to the CEM form-finding algorithm:

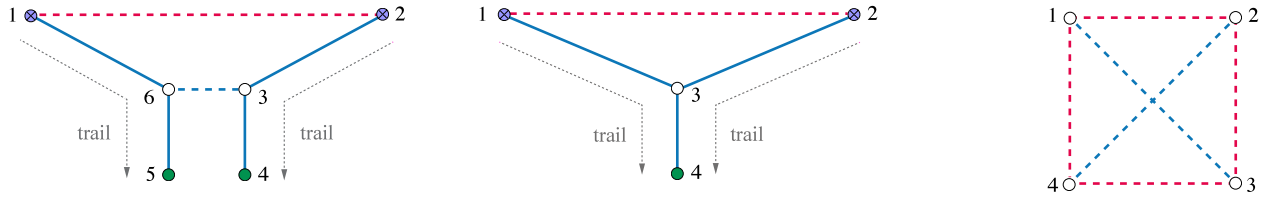
1. Every node $\mathbf{v}$ needs to be part of exclusively one trail ω (Section 2.1.1).
2. Every trail ω must have one and only one support assigned to its last node.

Modeling a topology diagram that abides by these rules can become a daunting task without a sound knowledge of the CEM theoretical background, especially for structures that do not have a clear load-transfer hierarchy. As a result, the type of structures that can be readily form-found with the CEM form-finding algorithm is limited. Fig. 2(a) shows an example of a topology diagram wherein the two topological modeling rules are satisfied. In contrast, the diagram in Fig. 2(b) violates the first rule because the two proposed trails $\omega_1 = \{1, 3, 4\}$ and $\omega_2 = \{2, 3, 4\}$ share nodes 3 and 4. Fig. 2(c) depicts a self-stressed tensegrity structure which has no supports and consequently infringes rule number two.

1.2.2. Approximate gradient computation

To solve a constrained form-finding problem, the CEM framework approximates the gradient of an objective function to minimize via finite differences (FD) [24,34]. This is in stark contrast to other geometric form-finding precedents wherein analytical equations are used to calculate an exact gradient of the objective function [9,27,28,33,35,36]. FD circumvents the derivation problems we discuss in Section 3.2 as it does not require the calculation of analytical derivatives of the CEM form-finding algorithm to obtain an estimate of the gradient.

Using FD poses a number of challenges nonetheless. FD requires choosing an adequate step size h to compute an approximation of the gradient [37]. If the step size h is too large, then the gradient approximation can be inaccurate, whereas if it is too small it can lead to significant round-off errors due to floating-point underflow or to a much extended optimization runtime. Furthermore, calculating gradients with FD is a computationally taxing process since the objective function has to be evaluated at least once for every input optimization parameter [37,38]. An expensive calculation of the gradient can be detrimental to an interactive exploration of constrained static equilibrium states for a structure, particularly if the number of optimization parameters is large.



(a) Two-sided cantilever structure (b) Branching structure (c) Self-stressed tensegrity

Fig. 2. Topology diagrams T that correspond to three different structural systems. Diagrams 2(b) and 2(c) do not meet the CEM topology requirements listed in Section 1.2. The former shows nodes 3 and 4 as members of two different trails, whereas the latter omits the assignment of trails and support nodes.

1.3. Automatic differentiation

Automatic differentiation (AD), also known as algorithmic differentiation, comprises a set of techniques that evaluate the derivatives of a differentiable function that is expressed algorithmically by means of the automated and repeated application of the chain rule [39]. In contrast to FD, derivatives obtained with AD are exact up to floating-point precision and do not require the specification of a step size h to be computed [37]. Unlike symbolic differentiation, AD enables the evaluation of derivatives through control flow statements, such as if-else clauses, loops and recursion [39].

One of the most prominent applications of AD today is in training machine learning models via backpropagation, in particular neural networks that learn via gradient descent [40]. In structural analysis, AD has been used to derive complex isogeometric elements from an energy functional, showcasing its ease of use and its numerical efficiency for high-dimensional analysis problems [41]. Other precedents of AD applied to various engineering problems are the sizing of the frame of an injection molding machine [38], the shape optimization of a supersonic aircraft [42], and the weight minimization of steel frames under seismic loads [43].

In the context of form-finding, Cuvilliers recently proposed the use AD to form-find pin-jointed bar structures subjected to geometric constraints [44]. Unlike their work, we use the CEM form-finding algorithm and not the *Force Density Method* [8,9] as the equilibrium state calculator. Furthermore, they focus on compression-only shells while we study various types of mixed tension–compression structures.

1.4. Outline and contribution

Table 1 lists the symbols we use to display topology, form and constrained form diagrams, T, F and $\bar{F}$, respectively.

This paper is organized in six sections.

In Section 2 we present the theoretical concepts that underpin the current state of the CEM framework. We review the steps the framework follows to sequentially and iteratively compute a state of static equilibrium from an algorithmic perspective, and then discuss the mathematical formulation of the objective function that it minimizes to solve a constrained form-finding problem.

Section 3 develops the extensions we make to the CEM framework to overcome the limitations we outlined in Section 1.2, which constitute the core of our contribution. We first introduce a new topological helper object, the auxiliary trail. We show next how we leverage AD to evaluate an exact gradient of the CEM form-finding algorithm and guide the reader through this process with a simple constrained form-finding example. We also present a new standalone design tool called `compas_cem` that encapsulates the two above-mentioned extensions.

Table 1
Symbols that describe the elements of topology (T), form (F) and constrained form ($\bar{F}$) diagrams in the extended CEM framework. Refer to Section 2.1 for element definitions. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Diagram	Symbol	Description
T	○	Node
	●	Origin node
	●	Support node
	×	Load
	———	Trail edge in tension
	———	Trail edge in compression
	- - -	Deviation edge in tension
	- - -	Deviation edge in compression
	———	Auxiliary trail edge
$F, \bar{F}$	○	Node
	——→	Load or reaction force vector
	———	Edge in tension
	———	Edge in compression

In Section 4, we benchmark and validate the extended CEM framework by studying multiple constrained form-finding problems on three different types of structures: a self-stressed tensegrity, a tree canopy and a curved bridge subjected to eccentric point loads. We showcase the applicability of our work in practical structural design problems with the case study of a spiral staircase in Section 5. For reproducibility, we share the data and the code we use to solve all the constrained form-finding problems we discuss in Sections 4 and 5 in an open-access repository [45].

The paper concludes in Section 6 with a discussion of our experimental findings, the limitations of our approach and future research directions.

2. Theoretical background

We review how the Combinatorial Equilibrium Modeling (CEM) framework works. This is relevant to guide the discussion in subsequent sections. The CEM framework was introduced in [23] and further developed in [24,25]. It consists of two operative parts: (i) a form-finding algorithm that builds equilibrium sequentially and iteratively (Section 2.3), and (ii) a constrained form-finding routine that utilizes an optimization solver (Section 2.4). The goal of the former is to generate a numerical *state of static equilibrium*, $\mathbf{u}$. That of the latter is to produce a *constrained state of static equilibrium* $\bar{\mathbf{u}}$ that has been restricted by a set of geometric and force constraints. Regardless, there are two necessary inputs to calculate these states: a valid topology diagram T and a vector of design parameters $\mathbf{x}$.

2.1. Topology diagram

A *topology diagram* T is an undirected graph of N nodes $\mathcal{V}$ connected by M edges $\mathcal{E}$. It captures the internal connectivity of a structure modeled as a pin-jointed network of straight bars.

Every edge $\mathbf{e}_{i,j}$ connecting two nodes $\mathbf{v}_i, \mathbf{v}_j$ must be labeled as either a *trail edge* $\mathbf{e}^{\mathrm{t}}_{i,j}$ or a *deviation edge*, $\mathbf{e}^{\mathrm{d}}_{i,j}$. Therefore, the total number of trail edges R and the total number of deviation edges D in T must add up to M, i.e. $R + D = M$. Trail edges outline the trails of a structure (Section 2.1.1). These edges determine the primary paths that the loads applied to a structure follow towards the supports. Deviation edges connect nodes on different trails to redirect the load trajectories determined by the trails.

The entries in the adjacency matrix $\mathbf{C} \in \{-1, 0, 1\}$ of the topology diagram define the expected internal force state $c_{i,j}$ of the bars in the structure [24]. If $\mathbf{C}[i,j] = c_{i,j} = -1$, the corresponding edge $\mathbf{e}_{i,j}$ is in compression. Conversely, if $c_{i,j} = 1$, then $\mathbf{e}_{i,j}$ is in tension. A topology diagram furthermore prescribes the subset $\mathcal{S}$ of size L with the nodes where a support is assigned to the structure, $\mathcal{S} \subset \mathcal{V}$.

2.1.1. Trails

Trails are critical to assess the validity of a topology diagram T. A *trail* ω is an ordered set of nodes linked exclusively by trail edges, $\omega = \{\mathbf{v}^{\mathrm{o}}, \dots, \mathbf{v}^{\mathrm{s}}\}$. The first node in a trail $\mathbf{v}^{\mathrm{o}}$ is referred to as an *origin node*. The last node $\mathbf{v}^{\mathrm{s}}$ must have a support assigned, $\mathbf{v}^{\mathrm{s}} \in \mathcal{S}$, and it is thus referred to as a *support node*. A trail must contain at least two nodes. The set of all trails in a topology diagram is denoted Ω. There must be as many trails as there are support nodes, $|\mathcal{S}| = |\Omega|$. Trails need not have the same number of nodes each.

2.1.2. Sequences

Once a trail ω is constructed, the nodes within are sorted based on how distant they are to the origin node $\mathbf{v}^{\mathrm{o}}$ in the trail. For every node $\mathbf{v}_i$, this topological distance is defined as the number of intermediate trail edges $\mathbf{e}^{\mathrm{t}}$ plus one between $\mathbf{v}_i$ and $\mathbf{v}^{\mathrm{o}}$. Nodes that are equally distant to their corresponding $\mathbf{v}^{\mathrm{o}}$ across all defined trails Ω belong to the same *sequence*, k. While the first sequence $k = 1$ groups all the origin nodes $\mathbf{v}^{\mathrm{o}}$ in T, the last sequence k^{last} contains the support nodes of the trails with the most nodes. The list of sequences $\boldsymbol{k}$ is an ordered set of consecutive integers between $k = 1$ and k^{last}.

2.2. Design parameters

The vector of design parameters $\mathbf{x}$ prescribes an immutable portion of the state of static equilibrium $\mathbf{u}$ that is calculated by the CEM form-finding algorithm. It concatenates:

- A vector $\boldsymbol{\mu} \in \mathbb{R}^M_+$ with the absolute magnitude of the internal force $\mu^{\mathrm{d}}_{i,j}$ of every deviation edge $\mathbf{e}^{\mathrm{d}}_{i,j}$.
- A vector $\boldsymbol{\lambda} \in \mathbb{R}^M_+$ with the length $\lambda^{\mathrm{t}}_{i,j}$ of each trail edge $\mathbf{e}^{\mathrm{t}}_{i,j}$. Alternatively, a matrix $\boldsymbol{\Phi} \in \mathbb{R}^{R\times 6}$ with an intersection plane $\boldsymbol{\phi}_{i,j} \in \mathbb{R}^6$ per trail edge $\mathbf{e}^{\mathrm{t}}_{i,j}$ to implicitly compute $\lambda^{\mathrm{t}}_{i,j}$.
- A matrix $\mathbf{P} \in \mathbb{R}^{N\times 3}$ with the position $\mathbf{p}^{\mathrm{o}} \in \mathbb{R}^3$ of every origin node $\mathbf{v}^{\mathrm{o}}$
- A matrix $\mathbf{Q} \in \mathbb{R}^{N\times 3}$ with the load vectors $\mathbf{q} \in \mathbb{R}^3$ applied to the nodes $\mathbf{v}$. Only one load vector per node is permitted. If the modeled structure is self-stressed, all the entries in $\mathbf{Q}$ are null vectors.

Algorithm 1: The CEM form-finding algorithm

Input : Topology diagram, T
Sequences, $\boldsymbol{k}$
Trails, Ω
Design parameters, $\mathbf{x}$
Maximum # of equilibrium iterations, $\tau^{\max}$
Minimum distance threshold, $\eta^{\min}$
Output: State of static equilibrium, $\mathbf{u}$

$\mathcal{V}, \mathcal{E}, \mathcal{S}, \mathbf{C}, \leftarrow T$
$\boldsymbol{\mu}, \boldsymbol{\lambda}, \boldsymbol{\Phi}, \mathbf{P}, \mathbf{Q} \leftarrow \mathbf{x}$
iteration, $\tau \leftarrow 1$
distance, $\eta \leftarrow \infty$
while $\tau \leq \tau^{max}$ **or** $\eta \geq \eta^{min}$ **do**
  for sequence **in** sequences, $k \in \boldsymbol{k}$ **do**
    for trail **in** trails, $\omega \in \Omega$ **do**
      $\mathbf{v}_i \leftarrow$ NodeInTrailAtSequence(ω, k)
      if node exists **in** trail, $\mathbf{v}_i \in \omega$ **then**
        if first sequence, $k = 1$ **then**
          $\mathbf{t}_h \leftarrow \mathbf{0}$
          $\mathbf{p}_i \leftarrow$ PositionOriginNodeInTrail(ω)
        $\mathbf{d}_i \leftarrow$ DeviationEdgesVector($\mathbf{v}_i, \mathbf{p}_i$) ▷ Eq. (4)
        $\mathbf{t}_i \leftarrow$ ResidualForceVector($\mathbf{t}_h, \mathbf{d}_i, \mathbf{q}_i$) ▷ Eq. (3)
        if node is **not** a support node, $\mathbf{v}_i \notin \mathcal{S}$ **then**
          if plane $\boldsymbol{\phi}_{i,j}$ exists **then**
            $\lambda^{\mathrm{t}}_{i,j} \leftarrow$ PlaneIntersection($\mathbf{p}_i, \mathbf{t}_i, \boldsymbol{\phi}_{i,j}$) ▷ Eq. (2)
            $\mathbf{p}_j \leftarrow$ NodePosition($\mathbf{p}_i, \mathbf{t}_i, c_{i,j}, \lambda^{\mathrm{t}}_{i,j}$) ▷ Eq. (1)
            $\mu^{\mathrm{t}}_{i,j} \leftarrow$ TrailEdgeForce($\mathbf{t}_i$) ▷ Eq. (5)
            $\mathbf{t}_h \leftarrow \mathbf{t}_i$
            $\mathbf{p}_i \leftarrow \mathbf{p}_j$
          else
            $\mathbf{r}_i \leftarrow \mathbf{t}_i$
  for deviation edge **in** edges, $\mathbf{e}^{d}_{i,j} \in \mathcal{E}$ **do**
    $\lambda^{\mathrm{d}}_{i,j} \leftarrow$ DeviationEdgeLength($\mathbf{p}_i, \mathbf{p}_j$) ▷ Eq. (6)
  if not first iteration, $\tau > 1$ **then**
    $\eta \leftarrow$ NodeDistances($\mathbf{P}^{(\tau)}, \mathbf{P}^{(\tau-1)}$) ▷ Eq. (7)
  $\tau \leftarrow \tau + 1$
$\mathbf{u} \leftarrow \boldsymbol{\mu}, \boldsymbol{\lambda}, \mathbf{P}, \mathbf{Q}, \mathbf{R}$

2.3. Form-finding algorithm

The CEM form-finding algorithm completes the attributes in $\mathbf{u}$ following Algorithm 1. The numerical outputs of the algorithm consist of:

- The absolute magnitude of the internal force $\mu^{\mathrm{t}}_{i,j} \in \boldsymbol{\mu}$ of every trail edge $\mathbf{e}^{\mathrm{t}}_{i,j}$.
- The length $\lambda^{\mathrm{d}}_{i,j} \in \boldsymbol{\lambda}$ of every deviation edge $\mathbf{e}^{\mathrm{d}}_{i,j}$.
- The position $\mathbf{p}_i \in \mathbf{P}$ of every non-origin node, $\mathbf{v}_i \neq \mathbf{v}^{\mathrm{o}}_i$.
- A matrix $\mathbf{R} \in \mathbb{R}^{L\times 3}$ with the reaction force vector $\mathbf{r}_i \in \mathbb{R}^3$ incident to every support node, $\mathbf{v}^{\mathrm{s}}_i \in \mathcal{S}$.

The outputs are concatenated into a single vector. Once $\mathbf{u}$ is complete, the form diagram F and the force diagram F^* of the structure can be built using vector-based graphic statics [14] to visualize the resulting equilibrium state of the structure.

2.3.1. Sequential equilibrium

Static equilibrium is calculated at the nodes of the diagram T one sequence at a time [23,24]. The form-finding process starts off by computing equilibrium at the nodes at the first sequence

$k = 1$ and continues to the next sequence $k+1$ until the last one, k^{last}, is reached. The calculation ends for every trail ω when the node at the current sequence k is a support node.

Except for the nodes at the first sequence $k = 1$, a state of static equilibrium at node $\mathbf{v}_j$ at sequence $k \neq 1$ is subordinated to the equilibrium state of the nodes preceding it. Let the triplet of nodes $\mathbf{v}_h$, $\mathbf{v}_i$ and $\mathbf{v}_j$ be three consecutive nodes along a sequence-ordered trail ω. Let $\mathbf{v}_i$ be the node on the trail at sequence k, $\mathbf{v}_h$ the node at the previous sequence, $k-1$, and $\mathbf{v}_j$ the node at the next sequence, $k+1$. The calculation of static equilibrium at node $\mathbf{v}_i$ outputs the position $\mathbf{p}_j$ of the next node $\mathbf{v}_j$ and the force magnitude $\mu^{\text{t}}_{i,j}$ of the trail edge $\mathbf{e}^{\text{t}}_{i,j}$ connecting them. The position $\mathbf{p}_j$ is calculated as:

$$\mathbf{p}_j = \mathbf{p}_i + c_{i,j}\,\lambda^{\text{t}}_{i,j}\,\frac{\mathbf{t}_i}{\|\mathbf{t}_i\|} \tag{1}$$

where $\mathbf{p}_i$ is the position of node $\mathbf{v}_i$, $c_{i,j}$ and $\lambda^{\text{t}}_{i,j}$ are the internal force state (-1 for compression, $+1$ for tension) and the length of $\mathbf{e}^{\text{t}}_{i,j}$, respectively; and $\mathbf{t}_i$ is a residual force vector incident to node $\mathbf{v}_i$ as per Eq. (3).

If a plane $\boldsymbol{\phi}_{i,j}$ is supplied instead of a specific trail edge length $\lambda^{\text{t}}_{i,j}$, then the absolute magnitude of $\lambda^{\text{t}}_{i,j}$ is computed by intersecting the line of action of the vector $\mathbf{t}_i$ onto $\boldsymbol{\phi}_{i,j}$ [25]:

$$\lambda^{\text{t}}_{i,j} = \left| \frac{\mathbf{n}^{\boldsymbol{\phi}_{i,j}} \cdot (\mathbf{p}^{\boldsymbol{\phi}_{i,j}} - \mathbf{p}_i)}{\mathbf{n}^{\boldsymbol{\phi}_{i,j}} \cdot (\mathbf{t}_i / \|\mathbf{t}_i\|)} \right| \tag{2}$$

where $\mathbf{p}^{\boldsymbol{\phi}_{i,j}} \in \mathbb{R}^3$ is the base point and $\mathbf{n}^{\boldsymbol{\phi}_{i,j}} \in \mathbb{R}^3$ the vector normal that describe the plane $\boldsymbol{\phi}_{i,j}$.

To estimate vector $\mathbf{t}_i$, all the forces acting on $\mathbf{v}_i$ are summed:

$$\mathbf{t}_i = \mathbf{t}_h - \mathbf{d}_i - \mathbf{q}_i \quad \text{where} \quad \mathbf{t}_h = \begin{cases} \mathbf{0} & \text{if } k = 1 \\ \mathbf{t}_i^{(k-1)} & \text{otherwise} \end{cases} \tag{3}$$

where $\mathbf{t}_h$ is the residual force vector at node $\mathbf{v}_h$. The vector $\mathbf{q}_i$ denotes the load applied to node $\mathbf{v}_i$, if any, and $\mathbf{d}_i$ corresponds to the resultant force vector generated by all the deviation edges $\mathbf{e}^{\text{d}}_{i,m}$ connected to $\mathbf{v}_i$:

$$\mathbf{d}_i = \sum_m c_{i,m}\,\mu^{\text{d}}_{i,m}\,\frac{\mathbf{p}_i - \mathbf{p}_m}{\|\mathbf{p}_i - \mathbf{p}_m\|} \tag{4}$$

The terms $c_{i,m}$, $\mu^{\text{d}}_{i,m}$ and $\mathbf{p}_m$ encode the force state, the force magnitude and the position of the node $\mathbf{v}_m$ that is connected to $\mathbf{v}_i$ by the deviation edge $\mathbf{e}^{\text{d}}_{i,m}$, respectively. If no deviation edges are connected to $\mathbf{v}_i$, then $\mathbf{d}_i = \mathbf{0}$.

To maintain the equilibrium of forces at $\mathbf{v}_i$, the residual vector $\mathbf{t}_i$ is taken by the trail edge $\mathbf{e}^{\text{t}}_{i,j}$ such that the vector formed between positions $\mathbf{p}_i$ and $\mathbf{p}_j$ point in the same direction as $\mathbf{t}_i$, and the absolute magnitude of the force $\mu^{\text{t}}_{i,j}$ passing through the trail edge is equal to the norm of $\mathbf{t}_i$:

$$\mu^{\text{t}}_{i,j} = \|\mathbf{t}_i\| \tag{5}$$

If $\mathbf{v}_j$ is a support node, then the residual vector $\mathbf{t}_i$ is parsed as the reaction force vector incident to the support node $\mathbf{v}^{\text{s}}_i$, that is, $\mathbf{r}_i = \mathbf{t}_i$. The length of any deviation edge $\mathbf{e}^{\text{d}}_{i,j}$ is lastly calculated as the distance between the positions of the two nodes it links:

$$\lambda^{\text{d}}_{i,j} = \|\mathbf{p}_i - \mathbf{p}_j\| \tag{6}$$

2.3.2. Iterative equilibrium

The process described in Section 2.3.1 must be run iteratively whenever (i) form-dependent load cases like wind or self-weight are applied to the structure; or (ii) deviation edges $\mathbf{e}^{\text{d}}_{i,j}$ that connect any two nodes $\mathbf{v}_i$, $\mathbf{v}_j$ that do not belong to the same sequence k exist (these edges are also called *indirect deviation edges*) [24]. The termination conditions for iterative equilibrium are to exhaust a maximum number of iterations $\tau^{\max}$ or to reach a minimum distance threshold $\eta^{\min}$ close to zero, such that $\eta \leq \eta^{\min}$. The distance η measures the cumulative displacement of the position $\mathbf{p}_i$ of every node $\mathbf{v}_i$ at iteration τ in relation to the previous one:

$$\eta = \sum_i^{\mathcal{V}} \|\mathbf{p}_i^{(\tau)} - \mathbf{p}_i^{(\tau-1)}\| \tag{7}$$

The value of η can be normalized by dividing it by N to make it independent of the total number of nodes in the structure. If indirect deviation edges exist, their contribution to $\mathbf{d}_i$ in Eq. (4) is set to $\mathbf{0}$ during the first iteration, $\tau = 1$ [24].

Table 2
Selection of constraint functions supported by the CEM framework. From a geometrical vantage point, g_1 sets a target position $\bar{\mathbf{p}}_i$ for node $\mathbf{v}_i$; g_2 a target orientation vector $\bar{\mathbf{a}}_{i,j}$ for edge $\mathbf{e}_{i,j}$; while g_3 a target length $\bar{\lambda}^{\text{d}}_{i,j}$ for deviation edge $\mathbf{e}^{\text{d}}_{i,j}$. Similarly, but from a force perspective, g_4 prescribes a desired force magnitude $\bar{\mu}^{\text{t}}_{i,j}$ for trail edge $\mathbf{e}^{\text{t}}_{i,j}$; g_5 a target individual load path $\bar{\varphi}_{i,j}$ for edge $\mathbf{e}_{i,j}$; and g_6 a target reaction force vector $\bar{\mathbf{r}}_i$ at support node $\mathbf{v}^{\text{s}}_i$. The edge load path $\varphi_{i,j}$ in g_5 corresponds to Maxwell's load path [46]. The minimization of this non-negative quantity over all the edges of a structure is conducive to a minimum-volume, pin-jointed bar structure [47,48].

Type	Target	Constraint function
Geometry	Node position, $\bar{\mathbf{p}}_i$	$g_1(\mathbf{u}(\mathbf{s})) = \|\mathbf{p}_i - \bar{\mathbf{p}}_i\|$
	Edge direction, $\bar{\mathbf{a}}_{i,j}$	$g_2(\mathbf{u}(\mathbf{s})) = \left\| \frac{\mathbf{p}_j - \mathbf{p}_i}{\|\mathbf{p}_j - \mathbf{p}_i\|} \cdot \bar{\mathbf{a}}_{i,j} \right\| - 1$
	Edge length, $\bar{\lambda}^{\text{d}}_{i,j}$	$g_3(\mathbf{u}(\mathbf{s})) = \lambda^{\text{d}}_{i,j} - \bar{\lambda}^{\text{d}}_{i,j}$
Force	Edge force, $\bar{\mu}^{\text{t}}_{i,j}$	$g_4(\mathbf{u}(\mathbf{s})) = \mu^{\text{t}}_{i,j} - \bar{\mu}^{\text{t}}_{i,j}$
	Edge load path, $\bar{\varphi}_{i,j}$	$g_5(\mathbf{u}(\mathbf{s})) = \lvert\mu_{i,j}\lambda_{i,j}\rvert - \bar{\varphi}_{i,j}$
	Reaction force, $\bar{\mathbf{r}}_i$	$g_6(\mathbf{u}(\mathbf{s})) = \|\mathbf{r}_i - \bar{\mathbf{r}}_i\|$

2.4. Constrained form-finding

The CEM framework can determine the parameters that lead to a constrained state of static equilibrium $\bar{\mathbf{u}}$ that best satisfies a priori geometric and structural design requirements. This is accomplished by minimizing an objective function using gradient-based optimization.

2.4.1. Optimization parameters

The vector of optimization parameters $\mathbf{s}$ defines the potential solution space of a constrained form-finding problem. It collects a subset of the design parameters $\mathbf{x}$ (see Section 2.2). Design parameters that are not included in $\mathbf{s}$ stay constant throughout the optimization process.

2.4.2. System solution

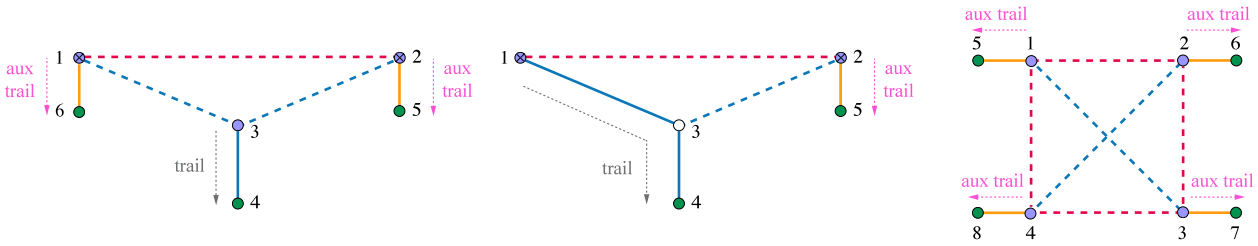
The CEM form-finding algorithm provides an explicit solution $\mathbf{u}(\mathbf{s})$ for a given a choice of optimization parameters $\mathbf{s}$. As per Section 2.3, this solution contains the missing node positions $\mathbf{p}$, the internal forces in the trail edges μ^{t}, the lengths of the deviation edges λ^{d}, and the reaction forces $\mathbf{r}$. The output solution described by $\mathbf{s}$ and $\mathbf{u}(\mathbf{s})$ is in static equilibrium.

2.4.3. Constraints

Vector $\mathbf{s}$ is modified to satisfy nonlinear equality constraints. Each constraint g_i is formulated as a function of the optimization parameters and the system solution $\mathbf{u}(\mathbf{s})$:

$$g_i(\mathbf{u}(\mathbf{s})) = 0 \tag{8}$$

The constraint functions g_i can be formulated arbitrarily. However, the complexity of the formulation may affect the solution and the convergence rate of the optimization problem. We list



(a) Branching structure A (b) Branching structure B (c) Self-stressed tensegrity

Fig. 3. Auxiliary trails in a topology diagram T. To ensure topological validity, in Fig. 3(a) and Fig. 3(b) we append auxiliary trails to the diagram of the branching structure first displayed in Fig. 2(b). Similarly, we insert four auxiliary trails to that of the self-stressed structure in Fig. 3(c).

in Table 2 the most frequently used geometric- and force-related constraint functions which measure the distance between the current value and a target value for one of the attributes in $\mathbf{u}(\mathbf{s})$. These functions can be freely combined in Eq. (9).

2.4.4. Objective function

Every nonlinear equality constraint g_i is weighted by a penalty factor w_i and aggregated into a single objective function $\mathcal{L}$ that is minimized to solve a constrained form-finding problem.

$$\mathcal{L}(\mathbf{s}) = \frac{1}{2}\sum_i w_i\, g_i(\mathbf{u}(\mathbf{s}))^2 \tag{9}$$

2.4.5. Gradient-based optimization

Eq. (9) can be efficiently minimized using first-order gradient descent or any other gradient-based optimization algorithm, such as the Limited-Memory Broyden–Fletcher–Goldfarb–Shanno algorithm (L-BFGS) [49], Sequential Least Squares Quadratic Programming (SLSQP) [50] or Truncated Newton (TNEWTON) [51].

2.4.6. Optimization convergence

The selected optimization algorithm minimizes Eq. (9) over a prescribed number of optimization iterations $\upsilon^{\max}$. The algorithm converges to an optimal instance of the optimization parameters $\mathbf{s}$ when one of the two conditions given by Eq. (10) is fulfilled:

$$\begin{aligned} \mathcal{L}(\mathbf{s}) &\leq \epsilon \\ \|\nabla_{\mathbf{s}}\mathcal{L}(\mathbf{s})\| &\leq \kappa \end{aligned} \tag{10}$$

The objective convergence threshold ϵ and the gradient convergence threshold κ are two scalars close to zero (for example, $\epsilon = 1 \times 10^{-6}$). The first condition in Eq. (10) indicates that the output value of the objective function $\mathcal{L}(\mathbf{s})$ approaches zero, which implies that the defined constraints g_i satisfy Eq. (8). An instance of $\mathbf{s}$ that fulfills Eq. (8) does not exist when the supplied constraints contradict each other. In such case, the optimizer converges to a local minimum of the objective function where the norm of the gradient vanishes, $\|\nabla_{\mathbf{s}}\mathcal{L}(\mathbf{s})\| \leq \kappa$. The computation of the gradient is discussed in Section 3.2.

3. Extensions to the CEM framework

We extend the CEM framework to overcome the limitations outlined in Section 1.2: auxiliary trails facilitate the creation of a valid topology diagram T and automatic differentiation enables the computation of more reliable and efficient solutions to constrained form-finding problems. The extended CEM framework is implemented in a standalone design tool.

3.1. Auxiliary trails

An auxiliary trail $\omega^{\mathrm{a}} = \{\mathbf{v}_i^{\mathrm{o}}, \mathbf{v}_j^{\mathrm{s}}\}$ is a short helper trail with an origin node $\mathbf{v}_i^{\mathrm{o}}$ and a support node $\mathbf{v}_j^{\mathrm{s}}$ linked by a single trail edge $\mathbf{e}_{i,j}^{\mathrm{t}}$ of unit length $\lambda_{i,j}^{\mathrm{t}} = 1$. We automatically attach an auxiliary trail to any node $\mathbf{v}_i$ in a topology diagram T that has not been assigned to another trail before the application of the CEM form-finding algorithm (see Section 2.3). Such trail-free nodes are characteristic at the intersection between one or more deviation edges and no trail edges. The attachment operation transforms node $\mathbf{v}_i$ into the origin node $\mathbf{v}_i^{\mathrm{o}}$ of ω^{a}. The extensive use of auxiliary trails enables the explicit construction of the topology diagram T of a structure using only deviation edges. Given an input T wherein every bar of a structure is modeled as a deviation edge $\mathbf{e}_{i,j}^{\mathrm{d}}$, appending an auxiliary trail to every node $\mathbf{v}_i$ in T converts this initially invalid diagram into a diagram T that complies with the topological modeling rules of the CEM form-finding algorithm (Section 1.2.1). Such a deviation-only modeling strategy circumvents the manual edge labeling process described in Section 2.1 as no distinction has to be made upfront by a designer on whether an edge $\mathbf{e}_{i,j}$ is a trail edge $\mathbf{e}_{i,j}^{\mathrm{t}}$ or a deviation edge $\mathbf{e}_{i,j}^{\mathrm{d}}$. We show two examples of structures that are modeled using the deviation-only modeling strategy in Sections 4.1 and 4.2.

However, the topological modeling flexibility enabled by auxiliary trails comes at a computation price. The attachment of an auxiliary trail creates a local artificial subsystem wherein a specific state of static equilibrium must be computed: the auxiliary trails must not carry any loads in order to capture the originally intended load-carrying behavior of the structure. An analogy for the role auxiliary trails play is that they provide additional temporary support to a structure while it is being form-found. Since we are interested in the "self-standing" version of the structure, we must find an equilibrium state in which the magnitude of the loads these temporary supports carry is zero.

Form-finding a structure whose topology diagram T contains at least one auxiliary trail hence becomes a constrained form-finding task. The initially-desired static equilibrium state for a structure is obtained only after solving the optimization problem discussed in Section 2.4, where the objective function $\mathcal{L}$ is extended with extra penalty terms, one per auxiliary trail, as we show in Eq. (11):

$$\frac{1}{2}\sum_a w_a\,(\mu_{i,j}^{\mathrm{t}} - \bar{\mu}_{i,j}^{\mathrm{t}})_a^2 \tag{11}$$

The purpose of the additional penalty terms is to minimize the difference between the target force $\bar{\mu}_{i,j}^{\mathrm{t}} = 0$ and the current force $\mu_{i,j}^{\mathrm{t}}$ in the trail edge $\mathbf{e}_{i,j}^{\mathrm{t}}$ of each auxiliary trail ω^{a} in T. These terms are equivalent to constraint function g_4 in Table 2. By extension, when the force in the trail edge of an auxiliary trail is zero, the reaction forces incident to its corresponding support node also vanish.

Fig. 3 shows three topology diagrams T that use auxiliary trails to remedy the modeling challenges posed by the diagrams in Fig. 2. Fig. 3(a) depicts a diagram T where auxiliary trails $\omega_2^a = \{1, 6\}$ and $\omega_3^a = \{2, 5\}$ are appended to nodes 1 and 2. Meanwhile, in Fig. 3(b), trail $\omega_2 = \{2, 3, 4\}$ is deleted, thus leaving node $\mathbf{v}_2$ trail unassigned. An auxiliary trail $\omega_2^a = \{2, 5\}$ is attached to $\mathbf{v}_2$ to make T valid. As portrayed by Fig. 3(c), an auxiliary trail is annexed to each of the four nodes of the tensegrity structure in Fig. 2(c) to correct its initial topological invalidity.

3.2. Automatic and exact computation of the gradient

The gradient required to determine a minimum of Eq. (9) results from the first derivative of $\mathcal{L}$ with respect to the optimization parameters $\mathbf{s}$:

$$\nabla_{\mathbf{s}} \mathcal{L} = \sum_i w_i \, g_i \cdot \nabla_{\mathbf{s}} \, g_i \tag{12}$$

The computation of the gradient $\nabla_{\mathbf{s}} \mathcal{L}$ requires the calculation of the derivatives of the individual constraint functions $\nabla_{\mathbf{s}} \, g_i$ and the derivative of the system solution $\mathbf{u}(\mathbf{s})$:

$$\nabla_{\mathbf{s}} \, g_i = \frac{\partial g_i}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial \mathbf{s}} \tag{13}$$

While Eqs. (1)–(7) are compact algebraic manipulations for which derivatives can be found analytically, manually applying the chain rule through the control flow structure of the CEM form-finding algorithm (see Algorithm 1) to calculate the partial derivative of the system solution with respect to the optimization parameters has been a complex task [24]. Instead of circumventing its sequential and iterative characteristics, we exploit the computational implementation of the CEM form-finding algorithm we develop in Section 3.3 by using AD in reverse mode to obtain a version of $\nabla_{\mathbf{s}} \mathcal{L}(\mathbf{s})$ that is exact up to floating-point precision. AD ingests the function $\mathcal{L}$, and generates another function $\nabla_{\mathbf{s}} \mathcal{L}$ that calculates the associated gradient. The key insight is that $\nabla_{\mathbf{s}} \mathcal{L}$ is automatically generated by a computer program.

We stress that AD provides a numerical value of $\nabla_{\mathbf{s}} \mathcal{L}(\mathbf{s})$ evaluated at a specific instance of $\mathbf{s}$ instead of generating an analytical expression for it. Nevertheless, the AD output is adequate for our purposes since we use the value of $\nabla_{\mathbf{s}} \mathcal{L}(\mathbf{s})$ to minimize Eq. (9), regardless of what the underlying analytical expression might be. We refer the reader to [39,40] for a detailed theoretical treatment on how reverse-mode AD evaluates derivatives of algorithmically expressed functions. To illustrate how AD operates specifically through the calculations of the CEM framework, we supplement the discussion with a toy constrained form-finding example.

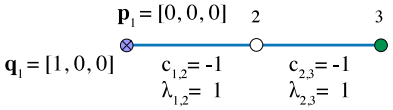
3.2.1. Example

Consider a two-segment strut subjected to a horizontal compressive force $\mathbf{q}_1$. Fig. 4(a) shows the topology diagram T, the internal force states $\mathbf{C}$ and the design parameters $\mathbf{x}$. The compression-only diagram F corresponding to the state of static equilibrium $\mathbf{u}$ output by the CEM form-finding algorithm is given in Fig. 4(b). In this example, we impose a geometric restriction on the position of node $\mathbf{v}_3$: it should land at $\bar{\mathbf{p}}_3 = [3, 0, 0]$. However, the resulting position $\mathbf{p}_3 = [2, 0, 0]$ is away from the target.

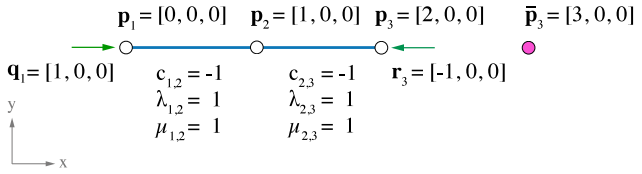
We are deliberately unsure of what combination of trail edge lengths $\lambda_{i,j}^t$ would result in a constrained state of static equilibrium $\bar{\mathbf{u}}$ that matches $\bar{\mathbf{p}}_3$. To solve this constrained form-finding problem, we set the two trail edge lengths as optimization variables such that $\mathbf{s} = [\lambda_{1,2}, \lambda_{2,3}] = [1, 1]$. The superscript t in $\lambda_{i,j}^t$ is dropped in this example for legibility. The only constraint function used is g_1 from Table 2 and the penalty factor w_1 is set to $w_1 = 1$.

The value of the gradient that reverse-mode AD computes is $\nabla_{\mathbf{s}} \mathcal{L}(\mathbf{s}) = [-1, -1]$. As expected, the negative magnitude of the partial derivatives in the gradient indicates that adjusting $\mathbf{s}$ in the opposite direction of the gradient would elongate both trail edges for $\mathbf{p}_3$ to move closer to the target position $\bar{\mathbf{p}}_3$ after the next optimization step. Reverse-mode AD arrives at $\nabla_{\mathbf{s}} \mathcal{L}(\mathbf{s}) = [-1, -1]$ after processing one forward and one reverse computation trace, one after the other, as we depict in Fig. 5 and in Table 3.

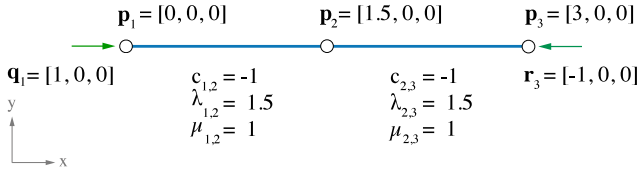
First, AD builds a forward evaluation trace (also called a Wengert list [52]) to calculate the output value of the objective function $\mathcal{L}(\mathbf{s})$. During the assembly of the forward trace, AD keeps track of the sequence of operations that interact with the entries in $\mathbf{s}$ in their journey towards $\mathcal{L}(\mathbf{s})$. The forward evaluation trace is



(a) Topology diagram T, edge internal force states $c_{1,2}, c_{2,3}$; and input design parameters $\mathbf{x}$ (the position $\mathbf{p}_1$ of origin node $\mathbf{v}_1^o$; the edge lengths $\lambda_{1,2}, \lambda_{2,3}$; and the applied load $\mathbf{q}_1$).



(b) Form diagram F and the equilibrium state $\mathbf{u}$ generated with the CEM form-finding algorithm, which consists of the node positions $\mathbf{p}_2, \mathbf{p}_3$; the edge absolute internal forces $\mu_{1,2}, \mu_{2,3}$; and the support reaction force $\mathbf{r}_3$. To solve the constrained form-finding problem, position $\mathbf{p}_3$ should reach the target position $\bar{\mathbf{p}}_3$.



(c) Constrained form diagram $\bar{F}$ and the equilibrium state $\bar{\mathbf{u}}$ post-optimization. The node position $\mathbf{p}_3$ moves to $\bar{\mathbf{p}}_3$ once the gradient-based optimizer modifies the edge lengths such that $\lambda_{1,2} = \lambda_{2,3} = 1.5$.

Fig. 4. Constrained form-finding of a two-segment strut.

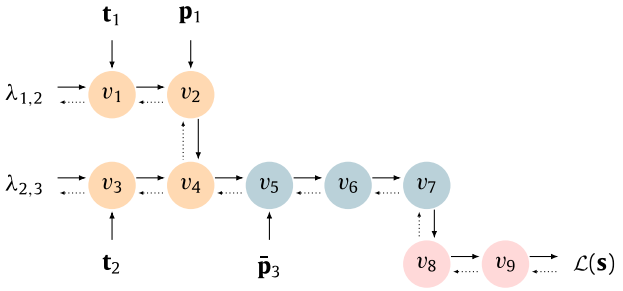


Fig. 5. Computation graph of the two-segment strut described in Section 3.2.1. The graph traces the operations involved in the evaluation the objective function $\mathcal{L}(\mathbf{s})$ at optimization parameters $\mathbf{s} = [\lambda_{1,2}, \lambda_{2,3}]$ during the forward pass (solid arrows). The nodes v_i store the output of each of the intermediary operations that modify $\mathbf{s}$ on their way to $\mathcal{L}(\mathbf{s})$. Nodes v_1 to v_4 correspond to operations that occur within the CEM form-finding algorithm (see Algorithm 1), while nodes v_5 to v_7 correspond to those executed in the evaluation of the constraint function g_1. Nodes v_8 and v_9 evaluate Eq. (9). To evaluate the gradient $\nabla_{\mathbf{s}} \mathcal{L}(\mathbf{s})$, reverse-mode AD propagates the partial derivatives of each node as per Eq. (14) in the opposite direction of the forward pass (backpropagation, dotted arrows). We unpack the operations of both the evaluation trace and the derivatives trace in Table 3.

Table 3
Automatic differentiation (AD) applied to the CEM framework to solve the constrained form-finding problem depicted in Fig. 4(b). To evaluate the gradient $\nabla_{\mathbf{s}}\mathcal{L}(\mathbf{s})$ of Eq. (9), reverse-mode AD operates on two computation traces: firstly, one forward evaluation trace (left-hand side) and secondly, one derivatives trace (right-hand side, also called backpropagation). To construct the former, AD evaluates Eq. (9) and keeps of all the elementary operations that modify the optimization parameters $\mathbf{s} = [\lambda_{1,2}, \lambda_{2,3}]$ (in this case, the length of the edges $\mathbf{e}_{1,2}$ and $\mathbf{e}_{2,3}$), and the sequence in which these operations alter them. The output of each elementary operation is stored in intermediary variables v_i which finally become interconnected nodes in the computation graph we show in Fig. 5. AD calculates the partial derivative of each of the nodes with respect to $\mathcal{L}(\mathbf{s})$ (i.e. the *adjoints*, $\bar{v}_i$) walking in reverse over the edges of the graph and applying the chain rule. The graph walk starts from the last operation tracked in the evaluation trace and ends when $\mathbf{s}$ is reached. The adjoints of the optimization parameters with respect to Eq. (9) are finally the partial derivatives in the gradient, $\nabla_{\mathbf{s}}\mathcal{L}(\mathbf{s}) = [-1, -1]$.

Evaluation trace (Forward pass)				Derivatives trace (Backpropagation)			
$\lambda_{1,2}$			$= 1$	$\bar{\lambda}_{1,2}$	$= \bar{v}_1 \frac{\partial v_1}{\partial \lambda_{1,2}}$	$= \bar{v}_1^\top c_{1,2} \frac{\mathbf{t}_1}{\lVert\mathbf{t}_1\rVert}$	$= -1$
$\lambda_{2,3}$			$= 1$	$\bar{\lambda}_{2,3}$	$= \bar{v}_3 \frac{\partial v_3}{\partial \lambda_{2,3}}$	$= \bar{v}_3^\top c_{2,3} \frac{\mathbf{t}_2}{\lVert\mathbf{t}_2\rVert}$	$= -1$
v_1	$= \lambda_{1,2} \times c_{1,2} \frac{\mathbf{t}_1}{\lVert\mathbf{t}_1\rVert}$	$= 1 \times \frac{[1,0,0]}{1}$	$= [\ 1, 0, 0]$	$\bar{v}_1$	$= \bar{v}_2 \frac{\partial v_2}{\partial v_1}$	$= \bar{v}_2 \times 1$	$= [-1, 0, 0]$
v_2	$= v_1 + \mathbf{p}_1$	$= [1, 0, 0] + [0, 0, 0]$	$= [\ 1, 0, 0]$	$\bar{v}_2$	$= \bar{v}_4 \frac{\partial v_4}{\partial v_2}$	$= \bar{v}_4 \times 1$	$= [-1, 0, 0]$
v_3	$= \lambda_{2,3} \times c_{2,3} \frac{\mathbf{t}_2}{\lVert\mathbf{t}_2\rVert}$	$= 1 \times \frac{[1,0,0]}{1}$	$= [\ 1, 0, 0]$	$\bar{v}_3$	$= \bar{v}_4 \frac{\partial v_4}{\partial v_3}$	$= \bar{v}_4 \times 1$	$= [-1, 0, 0]$
v_4	$= v_3 + v_2$	$= [1, 0, 0] + [1, 0, 0]$	$= [\ 1, 0, 0]$	$\bar{v}_4$	$= \bar{v}_5 \frac{\partial v_5}{\partial v_4}$	$= \bar{v}_5 \times 1$	$= [-1, 0, 0]$
v_5	$= v_4 - \bar{\mathbf{p}}_3$	$= [2, 0, 0] - [3, 0, 0]$	$= [-1, 0, 0]$	$\bar{v}_5$	$= \bar{v}_6 \frac{\partial v_6}{\partial v_5}$	$= \bar{v}_6 \times 2 \times v_5$	$= [-1, 0, 0]$
v_6	$= v_5^\top \cdot v_5$	$= [-1, 0, 0]^\top \cdot [-1, 0, 0]$	$= 1$	$\bar{v}_6$	$= \bar{v}_7 \frac{\partial v_7}{\partial v_6}$	$= \bar{v}_7 \times 0.5 \times \sqrt{v_6}$	$= 0.5$
v_7	$= \sqrt{v_6}$	$= \sqrt{1}$	$= 1$	$\bar{v}_7$	$= \bar{v}_8 \frac{\partial v_8}{\partial v_7}$	$= \bar{v}_8 \times 2 \times v_7$	$= 1$
v_8	$= v_7 \times v_7$	$= 1 \times 1$	$= 1$	$\bar{v}_8$	$= \bar{v}_9 \frac{\partial v_9}{\partial v_8}$	$= \bar{v}_9 \times 0.5$	$= 0.5$
v_9	$= v_8 \times 0.5$	$= 1 \times 0.5$	$= 0.5$	$\bar{v}_9$	$= \bar{\mathcal{L}}(\mathbf{s}) \frac{\partial \mathcal{L}(\mathbf{s})}{\partial v_9}$	$= \bar{\mathcal{L}}(\mathbf{s}) \times 1$	$= 1$
$\mathcal{L}(\mathbf{s})$			$= 0.5$	$\bar{\mathcal{L}}(\mathbf{s})$			$= 1$

broken next into a sequence of elementary mathematical operations, such as sums, divisions and multiplications, whose outputs are stored in intermediary variables v_i. Dependency relations between the variables v_i are finally represented as nodes and edges in a computation graph [53], which we show for this example in Fig. 5. In this example, from all the steps that the CEM form-finding algorithm comprises (see Algorithm 1), only one modifies $\mathbf{s}$, which corresponds to Eq. (1). We highlight the capability of AD to register this automatically despite the multiple loops and conditional statements in the CEM form-finding algorithm.

Once the forward evaluation trace is complete, reverse-mode AD backpropagates the derivatives on the nodes of the computation graph displayed in Fig. 5. We present the derivatives trace of this example on the right-hand side of Table 3. This reverse derivatives trace starts off at the value of the objective function $\mathcal{L}(\mathbf{s})$ and finishes once the nodes in the graph that correspond to the optimization parameters $\mathbf{s}$ are reached. Unlike forward-mode AD, only one pass over the entire computation graph suffices to compute $\nabla_{\mathbf{s}}\mathcal{L}(\mathbf{s})$ [54].

As the AD process walks in reverse over the graph, it calculates the partial derivative of $\mathcal{L}(\mathbf{s})$ with respect to every intermediate node using the chain rule. This partial derivative $\bar{v}_i$, called an *adjoint*, quantifies the sensitivity of the output $\mathcal{L}(\mathbf{s})$ to changes in the value of an intermediary variable v_i. The adjoint is expressed as:

$$\bar{v}_i = \frac{\partial \mathcal{L}(\mathbf{s})}{\partial v_i} = \sum_j \bar{v}_j \frac{\partial v_j}{\partial v_i} \tag{14}$$

The calculation of $\bar{v}_i$ is carried out by looking at each of the j children nodes of the variable v_i in the graph [53,55]. In our example, the value of the gradient finally results from the adjoints of the optimization parameters $\nabla_{\mathbf{s}}\mathcal{L}(\mathbf{s}) = [\bar{\lambda}_{1,2}, \bar{\lambda}_{2,3}]$ where their partial derivatives are scaled by the adjoints $\bar{v}_1$ and $\bar{v}_3$, such that $\bar{\lambda}_{1,2} = \bar{v}_1\, \partial v_1/\partial \lambda_{1,2}$ and $\bar{\lambda}_{2,3} = \bar{v}_3\, \partial v_3/\partial \lambda_{2,3}$.

3.3. Design tool

With the goal of making our work usable and reproducible, the CEM framework and the extensions presented hitherto are consolidated in a standalone, open-source design tool called `compas_cem` [56]. The tool is written in Python [57] and is integrated into the COMPAS framework, a computational ecosystem for collaboration and research in architecture, engineering, fabrication, and construction [58]. As a COMPAS extension, `compas_cem` can interface seamlessly with other packages in the COMPAS framework to perform additional downstream tasks on the structures generated with this tool [59–61].

A first CEM toolkit was presented in [62] as a plugin bound to the Windows version of Grasshopper [63]. In contrast, `compas_cem` runs independently from 3D modeling software, and it makes it possible to solve constrained form-finding problems on tension–compression structures readily from the command line interface of any of three major computer operating systems: Windows, MacOS and Linux. Furthermore, COMPAS provides our tool with the necessary interfaces to be invoked directly inside Blender [64], Rhino for Windows, Rhino for MacOS [65], and Grasshopper [63]. We illustrate this possibility in the structural design application we discuss in Section 5.

Currently, `compas_cem` uses the implementation of the optimization algorithms in the `NLopt` library [66] to minimize Eq. (9), and delegates the evaluation of the gradient $\nabla_{\mathbf{s}}\mathcal{L}(\mathbf{s})$ shown in Eq. (12) with AD to `autograd` [67]. This choice of dependencies is not restrictive. In the future, we envision integrating other Python optimization and automatic differentiation libraries such as `scipy` [68] and `hyperjet` [69].

The codebase of the design tool we propose follows a modular and object-oriented structure. For a granular overview of the objects and functions that it comprises, we refer the interested reader to the latest version of the `compas_cem` manual available online [56]. We offer instead a walk-trough over a minimal working example of `compas_cem` written in 59 lines of Python code and discuss how it relates to the theoretical concepts developed in Sections 2 and 3. This example, shown in Fig. 6, generates the structure we presented in Section 3.2.

The required `compas_cem` imports occur in lines 1–11. A `TopologyDiagram()` object, instantiated in line 16 is a child class of the `Network()` relational datastructure from COMPAS. A `Network()` is a graph that facilitates the storage of attributes on

```python
from compas_cem.diagrams import TopologyDiagram
from compas_cem.elements import Node
from compas_cem.elements import DeviationEdge
from compas_cem.elements import TrailEdge
from compas_cem.equilibrium import static_equilibrium
from compas_cem.loads import NodeLoad
from compas_cem.optimization import Optimizer
from compas_cem.optimization import PointConstraint
from compas_cem.optimization import TrailEdgeParameter
from compas_cem.plotters import Plotter
from compas_cem.supports import NodeSupport


# 1. Define inputs
# Instantiate an empty topology diagram
topology = TopologyDiagram()

# Add nodes
topology.add_node(Node(1, xyz=[0., 0., 0.]))
topology.add_node(Node(2))
topology.add_node(Node(3))

# Add edges
# The sign of the length indicates the force state
topology.add_edge(TrailEdge(1, 2, length=-1.))
topology.add_edge(TrailEdge(2, 3, length=-1.))

# Indicate support node
topology.add_support(NodeSupport(3))

# Add load
topology.add_load(NodeLoad(1, vector=[1., 0., 0.]))

# Build trails automatically
topology.build_trails(auxiliary_trails=False)

# 2. Form-Finding
# Generate form diagram in static equilibrium
form = static_equilibrium(topology, tmax=1, eta=1e-5)

# 3. Constrained Form-Finding
# Instantiate an optimizer
optimizer = Optimizer()

# Set target position for node 3
constraint = PointConstraint(3, point=[3., 0., 0.])
optimizer.add_constraint(constraint)

# Define optimization parameters
optimizer.add_parameter(TrailEdgeParameter((1, 2)))
optimizer.add_parameter(TrailEdgeParameter((2, 3)))

# Solve constrained form-finding problem
cform = optimizer.solve(topology, "SLSQP", eps=1e-6)

# 4. Visualize constrained form diagram
plotter = Plotter()
plotter.add(cform)
plotter.show()
```

Fig. 6. Python code that models the structure shown in Figs. 4(a) and 4(b) with the version of `compas_cem` at the time of writing [56].

its vertices and edges as Python dictionaries. Therefore, the inputs to the CEM form-finding algorithm, the topology diagram T and the design parameters $\mathbf{x}$, are stored in the same `TopologyDiagram()` object. In lines 25–26, we set the internal force state of the trail edges $c_{i,j}$ as the sign of the length that parametrizes them (see Section 2.1). The negative length indicates that the edges are under compression, $c_{1,2} = c_{2,3} = -1$. A positive length would conversely assign them a tension state. After defining nodes, edges, supports and loads in lines 19–32, the composition of T can be formally expressed as $\mathcal{V} = \{1, 2, 3\}$, $\mathcal{E} = \{(1, 2), (2, 3)\}$, $\mathcal{S} = \{3\}$.

A graph traversal algorithm that automatically searches for trails Ω is invoked on line 35. This algorithm takes the support node $\mathbf{v}_3^{\mathrm{s}}$ as the search starting point and moves recursively over the next connected trail edge $\mathbf{e}_{i,j}$ until no more trail edges are found. If the boolean argument `auxiliary_trails` were set to `True`, the trail-search algorithm would attach an auxiliary trail ω^{a} to any node that was not automatically assigned to a trail by the graph traversal, as exposed in Section 3.1. Only one trail is expected in this example, $\omega_1 = \{1, 2, 3\}$. In line 39, the CEM form-finding algorithm is invoked to output a form diagram F, conditioned on $\tau^{\max} = 1$ and $\eta^{\min} = 1 \times 10^{-5}$. As with `TopologyDiagram()`, F is a subclass of `Network()` that self-contains the attributes that describe the computed state of static equilibrium $\mathbf{u}$.

The generation of a constrained form diagram $\bar{F}$ is spread over lines 43–54. In line 46, we specify that the desired position for node $\mathbf{v}_3$ is $\bar{\mathbf{p}}_3 = [3, 0, 0]$, and in lines 50–51, we define the optimization parameters $\mathbf{s}$ as the length of the trail edges created in lines 25–26 such that $\mathbf{s} = [\lambda_{1,2}^{\mathrm{t}}, \lambda_{2,3}^{\mathrm{t}}]$. The formulated constrained form-finding problem is solved in line 54. Eq. (9) is minimized with the SLSQP optimization algorithm [50] from `NLopt` [66] using an optimization convergence threshold of $\epsilon = 1 \times 10^{-6}$. Internally, `autograd` [67] evaluates the gradient, as required by the optimization process. We plot the resulting instance of $\bar{F}$ in line 59 of the code and show it in Fig. 4(c).

4. Numerical validation

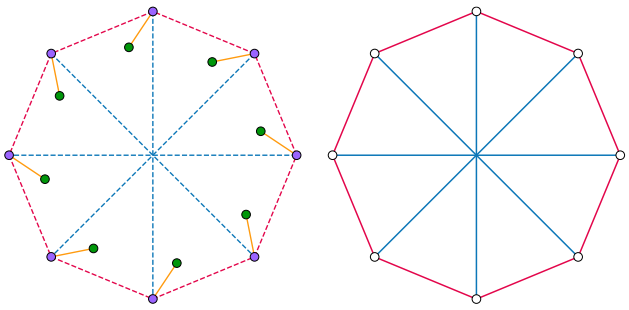
The intent of this section is to quantitatively benchmark the extensions we make to the CEM framework. We study three structures that leverage auxiliary trails to be topologically compatible with the CEM form-finding algorithm: a self-stressed tensegrity wheel, a tree canopy and a bridge curved on plan.

We assume that all the structures are pin-jointed and only bear axial forces. We model the first two structures using only deviation edges to illustrate how, in an extreme case, inserting an auxiliary trail to every node in T can relieve designers from the trail-deviation edge labeling process (see Section 2.1). The bridge structure follows a more conventional topological modeling approach and only appends auxiliary trails at the tip of the cantilevering hangers.

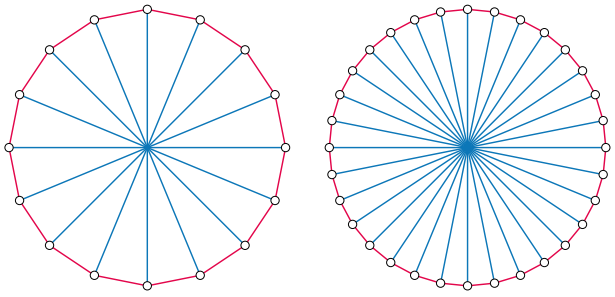
The primary goal of all the constrained form-finding experiments in this section is to minimize the forces in the auxiliary trails by setting the target edge force $\hat{\mu}_i$ to zero. We impose additional geometric constraints to the bridge to test the auxiliary trails extension we propose in combination with more constraint types. The penalty factors for all the constraints are equal to one, $w = 1$, and the distance threshold for iterative equilibrium in the CEM form-finding algorithm is $\eta^{\min} = 1 \times 10^{-6}$ (Section 2.3.2).

We solve each constrained form-finding experiment using automatic differentiation (AD) and finite differences (FD) with different step sizes h, following the implementation of the baseline version of the CEM framework [24,25,62]. We fix the optimization convergence thresholds to $\epsilon = 1 \times 10^{-6}$ and $\kappa = 1 \times 10^{-8}$ in all experiments, and compare the impact of the two differentiation schemes by looking at the total convergence runtime (i.e. the elapsed time in seconds it takes an optimization algorithm to converge as per Eq. (10)), the number of optimization parameters $\mathbf{s}$, and the output value of the objective function after convergence to a constrained equilibrium state $\bar{\mathbf{u}}$. We run every experiment ten times and report the resulting mean values per experiment.

For reference, we execute the work we present in this section in a collection of jupyter notebooks [70] using `compas_cem` on MacOS, on a quad-core Intel CPU clocked at 2.9 GHz. We make these notebooks available as supplementary data in [45].



(a) Topology T (left) and constrained form diagram $\bar{F}$ (right) of a tensegrity wheel.
12 parameters ($n = 3$)
AD: 0.04 sec / FD: 0.09 sec

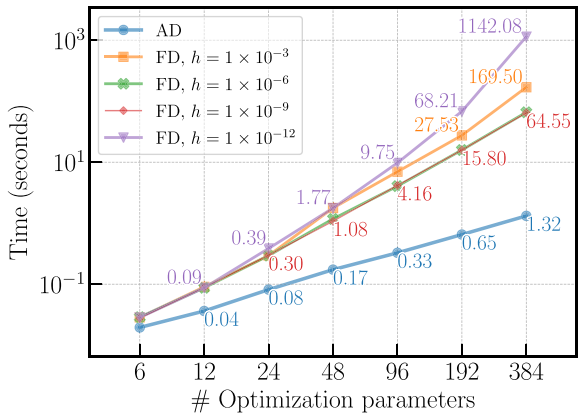


24 parameters ($n = 4$)
AD: 0.08 sec / FD: 0.30 sec

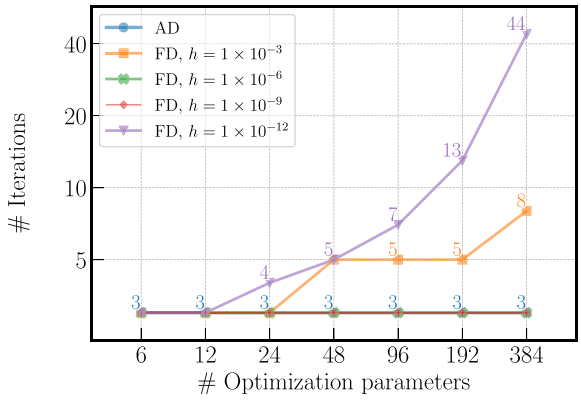
48 parameters ($n = 5$)
AD: 0.17 sec / FD: 1.08 sec

(b) Constrained form diagrams $\bar{F}$

Fig. 7. Planar tensegrity wheels form-found using the extended CEM framework. Auxiliary trails are not drawn in the diagrams $\bar{F}$ as they bear no force post optimization. We contrast the total convergence runtime for AD and FD with $h = 1 \times 10^{-9}$ per diagram $\bar{F}$.



(a) Convergence time



(b) Number of iterations for convergence

Fig. 8. Performance comparison between AD and FD to optimize a planar tensegrity wheel. Fig. 8(a): using FD is more expensive than AD as the number of optimization parameters increases. Fig. 8(b): inadequate values of the step size h raise the number of iterations required for convergence and thus extend the optimization runtime. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4.1. Auxiliary trails in 2D

Self-stressed structures are not subjected to external loads and are support-free. Here, we model a 2D self-stressed tensegrity wheel such that its perimeter is entirely in tension and the internal spokes in compression (Fig. 7). We carry out a parametric study where we progressively increase the number of optimization parameters as we increment the number of edges on the perimeter of the wheel in steps of size 2^n, where $n \in \{2, \ldots, 8\}$. For each configuration, the topological diagram of the wheel comprises of 2^{n+1} nodes and $2.5(2^n)$ edges, of which $1.5(2^n)$ are deviation edges. The remainder of the edges are the edges of the auxiliary trails. We consider the force in every deviation edge an optimization parameter in $\mathbf{s}$ and test FD with four different step sizes h, three orders of magnitude apart, $h \in \{1 \times 10^{-3}, 1 \times 10^{-6}, 1 \times 10^{-9}, 1 \times 10^{-12}\}$. Eq. (9) is minimized with the L-BFGS algorithm [49]. All the wheel experiments converge by satisfying the condition $\mathcal{L}(\mathbf{s}) \leq \epsilon$ in Eq. (10).

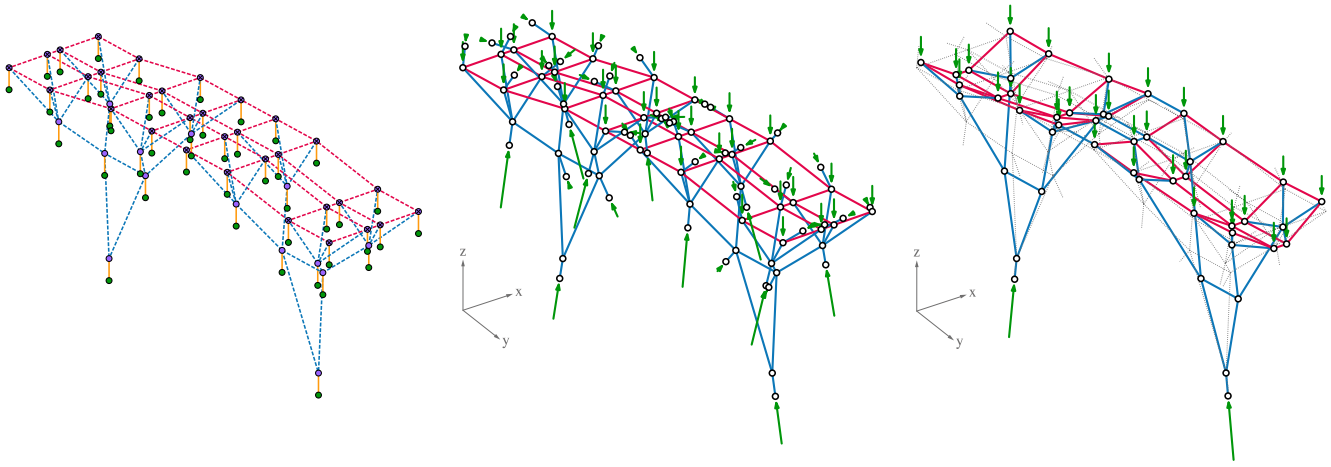
While AD and FD show comparable performance when $n < 4$, the total time for convergence with FD surges as the number of parameters increases, irrespective of the step size h (see Fig. 8(a)). When the number of parameters is the highest, form-finding the tensegrity wheel with AD gradients takes only 2.1% of the time it takes to do so with the best FD performance (1.32 vs. 64.55 s, with $h = 1 \times 10^{-9}$).

Figs. 8(a) and 8(b) expose the effect that changing the value of h has on the optimization convergence with FD. The computation time with FD for a single iteration is equivalent for all the values of h we tested. However, if h is too large ($h = 1 \times 10^{-3}$) or too small ($h = 1 \times 10^{-12}$), then convergence with FD for this tensegrity structure is much extended because the optimizer needs more iterations to reach an optimal solution for $\mathbf{s}$ due to an inaccurate approximation of the gradient $\nabla_{\mathbf{s}}\mathcal{L}(\mathbf{s})$. For example, the optimizer requires 5 and 41 more iterations than AD to solve a tensegrity wheel with $n = 8$, when $h = 1 \times 10^3$ and $h = 1 \times 10^{-12}$, respectively.

4.2. Auxiliary trails in 3D

We test the addition of auxiliary trails to a three-dimensional tree canopy structure (see Fig. 9). We observe how the performance of AD and FD differs as the choice of optimization algorithm changes. We model the initial tree structure with 46 nodes and 72 deviation edges. The total number of nodes and edges in the topological diagram doubles after inserting the auxiliary trails. We apply a point load $\mathbf{q} = [0, 0, -0.5]$ to the topmost nodes of the structure.

To minimize the forces in the auxiliary trails, we define 186 optimization parameters. These parameters consist of the positions of the origin nodes, which are allowed to translate only in the Y and Z Cartesian directions, and the force magnitude in every deviation edge. We test three different gradient-based optimization algorithms: L-BFGS [49], SLSQP [50], and AUGLAG [71] and run them for a maximum of $\upsilon^{\max} = 200$ optimization iterations.



(a) Topology diagram, T (b) Form diagram, F (c) Constrained form diagram, $\bar{F}$

Fig. 9. Tree canopy. The auxiliary trails on the branches take a large portion of the reaction forces in the form diagram F. After optimization, the reaction forces in the constrained form diagram $\bar{F}$ are taken only by the two supports at the base of the structure while the internal forces in the auxiliary trails vanish.

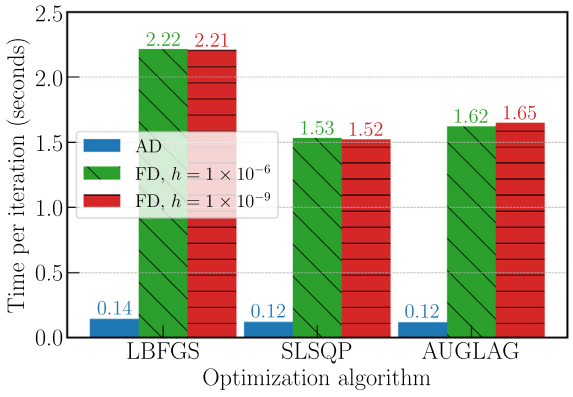


Fig. 10. Tree canopy. Convergence with AD is at least 10 times faster per iteration than FD. This disparity in computational performance is consistent for this constrained form-finding problem regardless of the selected optimization algorithm. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

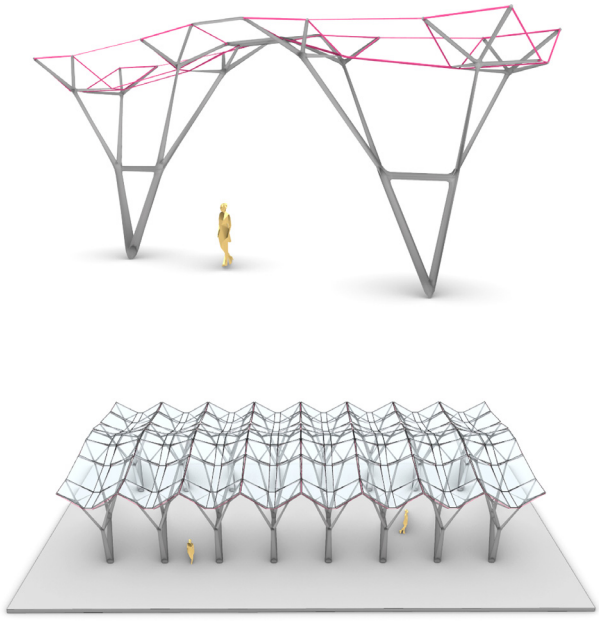


Fig. 11. Architectural vision for the constrained form diagram $\bar{F}$ shown in Fig. 9. The form-found tree canopy is repeated sequentially to create a colonnade of load-bearing trees.

The FD step sizes we discuss for this structure are $h \in \{1 \times 10^{-6}, 1 \times 10^{-9}\}$.

In Fig. 10, we report the elapsed time per iteration that the optimization algorithms take to converge by meeting the condition $\mathcal{L}(\mathbf{s}) \leq \epsilon$ in Eq. (10) with $\epsilon = 1 \times 10^{-6}$. Fig. 11 displays an architectural representation of the optimized tree structure. We calculate the time per iteration by dividing the convergence runtime over the total number of iterations incurred. This ratio is nearly equal for both FD step sizes across the three optimization algorithms we assess. Minimizing the forces in the auxiliary trails of the tree structure is at least 10 times faster per iteration using AD than with FD. The optimization time with AD is at most 0.14 s per iteration with L-BFGS, whereas this value is at least 1.52 s with FD in combination with SLSQP.

Among the three optimization algorithms we test, L-BFGS takes the fewest number of iterations to solve this constrained form-finding problem for both differentiation schemes, requiring 78 and 120 iterations to converge for AD and FD, respectively. Moreover, the minimization of Eq. (9) with AD is at least one order of magnitude faster for the 2D tensegrity structure with $n = 7$ described in Section 4.1 than it is with the tree structure presented here, despite the size of the two constrained form-finding problems is similar and the optimization algorithm is the same: 192 parameters and a convergence runtime of 0.65 s for the tensegrity versus 182 parameters and 11.15 s for the tree.

A plausible reason for this discrepancy is that, for the tree structure, both the node positions and the deviation edge forces are set as optimization parameters, whereas for the spoke wheel the optimization parameters only contemplate the forces in the deviation edges. The minimization of the auxiliary trail forces utilizing both the node positions and the internal forces of the structure is a non-linear problem that can be computationally more expensive to solve.

4.3. Auxiliary trails in 3D and additional constraints

We combine force and geometric constraints to steer the form-finding of a bridge with no intermediary supports (Fig. 12).

The bridge has two chords that curve on plan. A series of triangular hangers perpendicular to the longitudinal axis of the

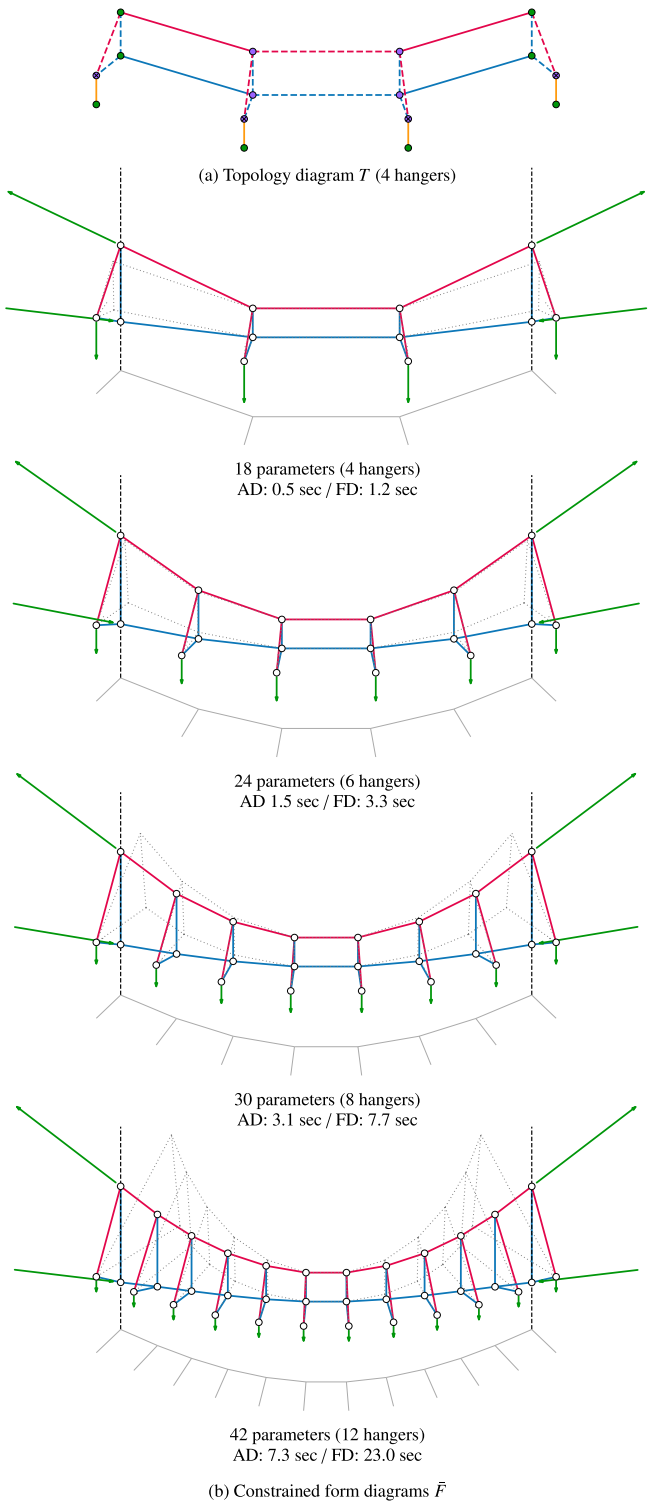


Fig. 12. Sensitivity analysis of a bridge curved on plan. The form diagrams F obtained after the first run of the CEM form-finding algorithm are drawn in the background with dotted lines. The forces in the auxiliary trails at the tip of the hangers have been minimized and the bridge endpoints pulled to the dashed vertical lines. The solution to this constrained form-finding problem is consistently faster with AD.

bridge take a uniformly distributed line load produced by an eccentric deck. We convert this line load into point loads via tributary lengths and apply them to the tip of the hangers. Unlike the spoke wheel and the tree structure discussed in Sections 4.1 and 4.2, we model the topological diagram T of the bridge using a hybrid strategy: we append auxiliary trails only to the nodes at the tip of the hangers because we assign all the other nodes in T along the bridge chords to a standard trail.

The geometric constraint for this structure is to pull the support nodes towards two predetermined vertical line rays located at either extreme of the bridge. The rationale behind this constraint is to arrive at a form in static equilibrium subject to a limited range of locations where to anchor the bridge abutments. The forces in all the deviation edges in T are considered optimization parameters, in addition to the length of the four trail edges connected to a support node.

We do a sensitivity analysis and compare AD and FD by monotonically increasing the number of hangers, from 4 to 22 in even steps, and then solving the resulting constrained form-finding problem with SLSQP [50]. The number of optimization parameters ranges from 18, when the number of triangular hangers is the smallest, to 72 when it is the greatest. Our goal is to estimate the time this constrained form-finding problem would take to converge to $\mathcal{L}(\mathbf{s}) \leq \epsilon$, where $\epsilon = 1 \times 10^{-6}$, when we add auxiliary trails only to a portion of the nodes in T of the bridge. We set the step size for FD to $h \in \{1 \times 10^{-3}, 1 \times 10^{-6}, 1 \times 10^{-9}, 1 \times 10^{-12}\}$ and we restrict the number of optimization iterations to $\upsilon^{\max} = 100$ for both differentiation methods, AD and FD.

Fig. 13(a) shows that the time for convergence with FD is equivalent for three different step sizes h across all experiments. The optimizer did not converge with $h = 1 \times 10^{-3}$ for this structure. This is different from the observations we make after studying a planar tensegrity in Section 4.1, where FD converges with a step size of $h = 1 \times 10^{-3}$ at the expense of significantly extending the convergence runtime. This finding illustrates that the impact of h on the quality of the gradient approximation with FD is problem dependent.

Fig. 13(b) provides further insight into the inadequacy of $h = 1 \times 10^{-3}$ to tackle this constrained form-finding problem. The final value of the objective function $\mathcal{L}(\mathbf{s})$ is, on average, two orders of magnitude higher than the desired optimization convergence threshold ϵ. In contrast, AD and FD with the three other step sizes h meet the target value of ϵ within the iteration budget $\upsilon^{\max} = 100$ since they reach the goal of $\mathcal{L}(\mathbf{s})/\epsilon \leq 1$. Nevertheless, optimizing the bridge is consistently more expedite with AD: convergence with a AD is 2.4 and 5 times faster when the number of parameters is the smallest (22) and the largest (72), respectively.

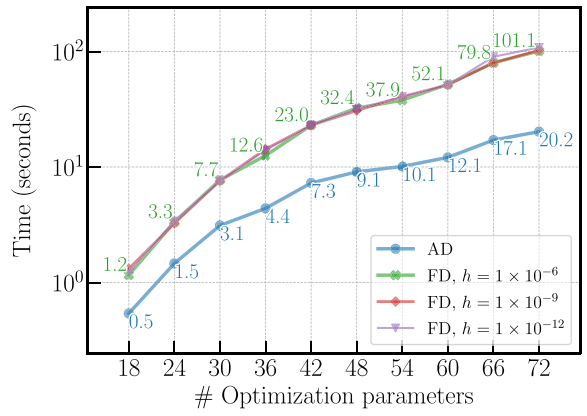
5. Case study

We illustrate the potential of the extended CEM framework to support designers in practical structural design problems, especially during the conceptual design stage.

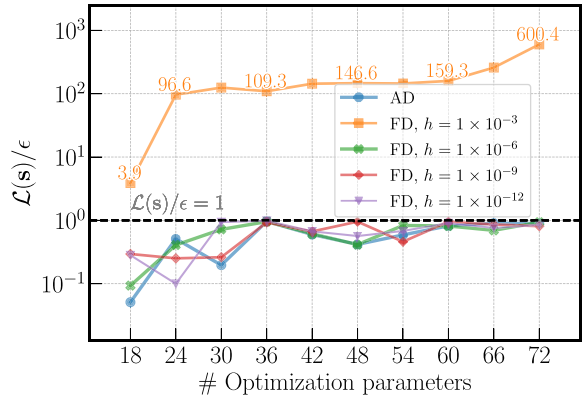
5.1. Design task

We design the load-bearing structure of a spiral staircase subjected to the design constraints listed in Section 5.3. The external perimeter of the staircase follows a semicircle on-plan with a diameter of 4 m (Fig. 14(b)). The staircase is planned to connect the ground floor to a mezzanine slab with a single run of 18 equidistant steps. Every step is 1 meter wide and perpendicular to the semicircle. The top of the mezzanine slab is 3.4 m above the ground and the design load is of 1 kN per step.

Inspired by the Fourth Bridge over the Grand Canal in Venice [72], we use the extended CEM framework to form-find a spiraling truss-like structure for the staircase. We imagine the structure to carry the applied loads via cross-shaped ribs suspended on two curving chords. The chords are initially proposed to be 1 meter apart from each other, running parallel to the run of stairs, and



(a) Convergence time



(b) Convergence rate

Fig. 13. Curved bridge. Fig. 13(a): Optimizing the bridge with FD is consistently more expensive than with AD regardless of the step size h. Fig. 13(b): the optimizer misses the optimization convergence threshold $\epsilon = 1 \times 10^{-6}$ by at least two orders of magnitude with FD and $h = 1 \times 10^{-3}$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

to be anchored at their intersections with the ground floor and the side of the mezzanine slab.

The static equilibrium concept is to first use the tread in every step of the staircase as a tension element that ties the upper half of the compression rib underneath (see callout rectangle in Fig. 14(c)). Next, the goal is to resist the torsional effects produced by the forces coming from the ribs by coupling together the two chords of the staircase as a pair of tension–compression rings.

5.2. Topology diagram

Fig. 14(a) displays the topology diagram T we build to form-find the staircase. We represent each of the chords with two trails connected by a single deviation edge at the middle as the chords are the main paths for load transfer towards the supports. As secondary load-bearing elements, we model the 18 ribs and ties with deviation edges and auxiliary trails.

5.3. Design constraints

The form-found shape of the staircase structure has to conform to a number of design requirements in addition to the minimization of the forces in the 54 auxiliary trails in T. We show these graphically in Fig. 14(b).

The side of the mezzanine slab where the chords have to be anchored has a maximum pull-out force capacity of 35 kN at position $\bar{\mathbf{p}}_1 = [0, 4, 3.4]$. As a result, the position of support node $\mathbf{v}_1^s$ in the tension chord must coincide with $\bar{\mathbf{p}}_1$, and the target absolute force magnitude passing through edge $\mathbf{e}_{1,2}$ has to be constrained to $\bar{\mu}_{1,2} = 35$ kN. The position of the support node on the compression chord $\mathbf{v}_{21}^s$ is restricted to slide on a horizontal line 0.20 m below $\bar{\mathbf{p}}_1$, parallel to the soffit of the mezzanine slab. We set these design requirements as optimization constraints.

Additionally, we define a sequence of planes to constrain, with the CEM form-finding algorithm, the position of the chord nodes to the plane formed by the upper portion of the rib they connect to (see Section 2.3.1). For example, nodes $\mathbf{v}_5$ and $\mathbf{v}_{25}$ in T should lie on the plane formed by $\mathbf{v}_{50}^o, \mathbf{v}_{51}^o, \mathbf{v}_{52}^o$, which is plane $\boldsymbol{\phi}^d$ in Fig. 14(b). Numerically, plane $\boldsymbol{\phi}^d$ corresponds to the intersection planes $\boldsymbol{\phi}_{6,5}$ and $\boldsymbol{\phi}_{26,25}$ of trail edges $\mathbf{e}_{6,5}^t$ and $\mathbf{e}_{26,25}^t$ in T, respectively. The reasoning behind this planarity constraint is to enable the fabrication of the ribs from flat sheets of material. Similarly, we pull the positions of the bottom support node per chord to the ground floor plane $\boldsymbol{\phi}^s$ and that of the support nodes connecting to the slab to $\boldsymbol{\phi}^t$ to explicitly restrict the feasible range of positions of these nodes can take during the optimization process. Plane $\boldsymbol{\phi}^s$ is described by base point $\mathbf{p}^{\phi^s} = [0, 0, 0]$ and normal $\mathbf{n}^{\phi^s} = [0, 0, 1]$, whereas plane $\boldsymbol{\phi}^t$ is defined by $\mathbf{p}^{\phi^t} = \bar{\mathbf{p}}_1$ and $\mathbf{n}^{\phi^t} = [1, 0, 0]$.

5.4. Constrained form diagram

We parametrize this constrained form-finding problem by setting the absolute force magnitude in all the deviation edges μ^d as entries in the vector of optimization parameters $\mathbf{s}$. We also allow the position of the origin nodes on the chords $\mathbf{v}_{10}^o, \mathbf{v}_{11}^o, \mathbf{v}_{30}^o, \mathbf{v}_{31}^o$ to translate vertically. The resulting optimization problem is minimized with L-BFGS [49].

We show the resulting constrained form diagram $\bar{F}$ in Fig. 14(c). The output values of the gradient $\nabla_{\mathbf{s}}\mathcal{L}(\mathbf{s})$ and of the objective function $\mathcal{L}(\mathbf{s})$ reaches the optimization convergence threshold $\epsilon = 1 \times 10^{-6}$, implying that the generated $\bar{F}$ for the staircase is satisfactory, meeting all the imposed design constraints. We can observe that some of the trade-offs made to solve this constrained form-finding task are that the initial distance between the chord supports on the mezzanine slab increased from 1 meter to 1.28 m and that the absolute magnitude of the reaction forces at the compression chord supports are about 25% and 60% higher than that at node $\mathbf{v}_1^s$ post-optimization.

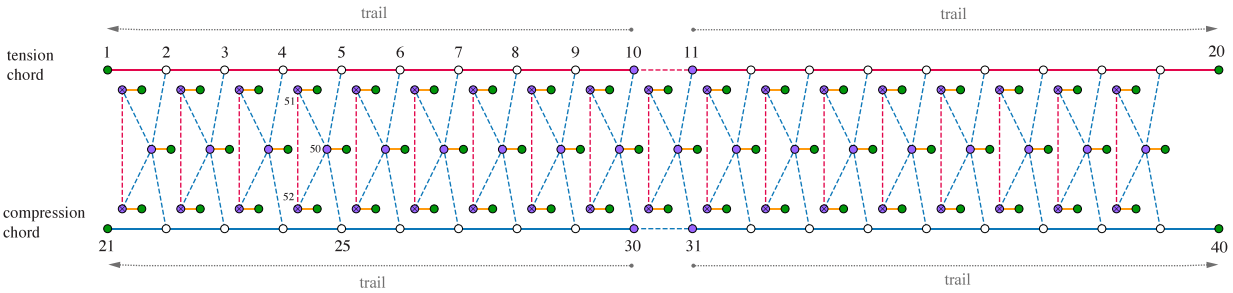
Fig. 15 finally shows one architectural interpretation of $\bar{F}$.

6. Conclusion

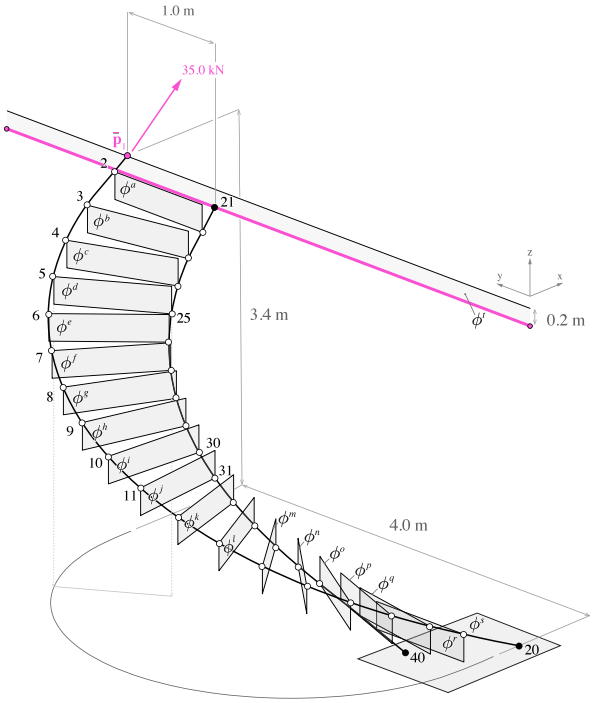
In this paper, we presented, developed and validated three extensions to the CEM framework: auxiliary trails, automatic differentiation (AD) and `compas_cem`. Compared to its baseline version [23–25], our work positions the extended CEM framework as a more efficient, general and accessible approach to generate structurally efficient shapes that meet force and geometric design requirements.

Auxiliary trails simplified the construction of valid topology diagrams for more types of structures that were difficult to be topologically modeled otherwise, such as branching structures, triangular cantilevers, and self-stressed systems. These helper trails also made it possible to explicitly model a structure only with deviation edges while still fulfilling the topological modeling rules of the CEM form-finding algorithm.

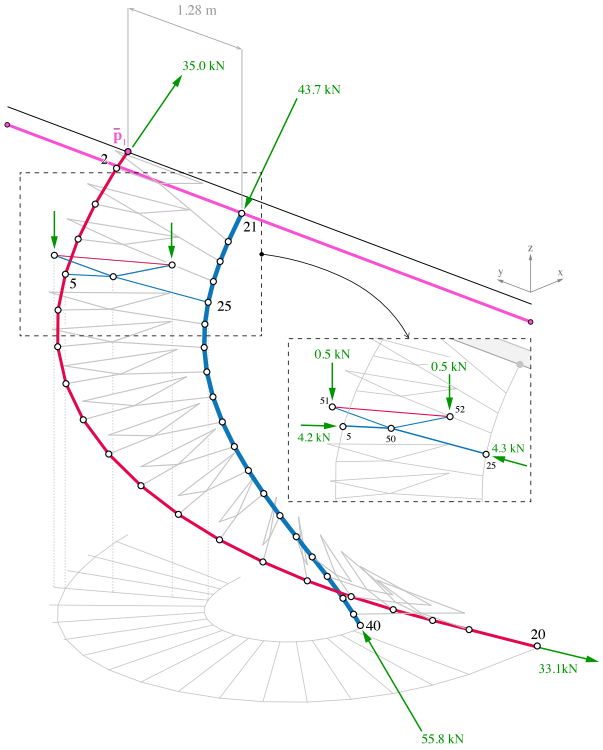
The application of AD enabled the automatic calculation of an exact gradient value of the CEM form-finding algorithm, no longer an approximation that depends on the calibration of a step size as in previous work. While the FD baseline and AD saw comparable performance for small constrained form-finding problems, our



(a) Topology diagram with auxiliary trails, T. The tension and compression chords of the staircase are represented by two trails and one deviation edge each. Edges $\mathbf{e}^{\mathrm{d}}_{5,50}$, $\mathbf{e}^{\mathrm{d}}_{25,50}$, $\mathbf{e}^{\mathrm{d}}_{51,50}$, $\mathbf{e}^{\mathrm{d}}_{52,50}$ model one of the 18 compression ribs and $\mathbf{e}^{\mathrm{d}}_{51,52}$ represents its matching tension tie. The design load was of 1 kN per step was applied as $\mathbf{q} = [0, 0, -0.5]$ kN to the end-nodes on the tension ties (e.g. to nodes $\mathbf{v}^{\mathrm{o}}_{51}$, $\mathbf{v}^{\mathrm{o}}_{52}$).



(b) Design constraints



(c) Constrained form diagram, $\bar{F}$

Fig. 14. Application of the extended CEM framework to design the load-bearing structure of a spiral staircase. After constructing a topology diagram T (Fig. 14(a)), a constrained form diagram $\bar{F}$ (Fig. 14(c)) that complies with a priori structural and fabrication constraints is computed using AD. The design constraints input to Eq. (9) in addition to the minimization of the forces in the auxiliary trails are shown in pink in Fig. 14(b) and comprise: (i) restraining the position of node $\mathbf{v}^{\mathrm{s}}_1$ to target position $\bar{\mathbf{p}}_1 = [0, 4, 3.4]$, (ii) fixing the magnitude of support force $\mathbf{r}_1$ to 35 kN; and (iii) constraining $\mathbf{p}_{21}$ to lie on a continuous horizontal line 0.2 m below $\bar{\mathbf{p}}_1$. We also restrict via the CEM form-finding algorithm the position of every pair of unsupported nodes on the chords to the plane defined by the nodes of the funicular rib they connect to (e.g. $\mathbf{v}_5$ and $\mathbf{v}_{25}$ to $\boldsymbol{\phi}^d$), the position that of support nodes $\mathbf{v}^{\mathrm{s}}_{20}$, $\mathbf{v}^{\mathrm{s}}_{40}$ to the ground floor plane $\boldsymbol{\phi}^s$ and that of nodes $\mathbf{v}^{\mathrm{s}}_1$, $\mathbf{v}^{\mathrm{s}}_{21}$ to the slab plane $\boldsymbol{\phi}^t$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

experiments demonstrated that calculating a constrained state of static equilibrium can be substantially faster with AD as the number of optimization parameters increases. Using AD opens up the possibility to accelerate design exploration cycles with the extended CEM framework, especially for large constrained form-finding problems.

With `compas_cem`, we consolidated our work into an open-source design tool. The tool enables the formulation and the solution of constrained form-finding problems in plain and simple Python code. Furthermore, `compas_cem` enables designers to use the extended CEM framework on three different operating systems and on three distinct pieces of 3D modeling software.

The work presented herein has limitations. The computation of an equilibrium state where auxiliary trails are not load-bearing depends on the solution of a constrained form-finding problem and not on a single run of the CEM form-finding algorithm. This optimization dependency is starker when a structure is modeled entirely with deviation edges. Consequently, the risk of using auxiliary trails is to end up with an under-parametrized or an over-constrained problem where neither equilibrium nor any other design constraint is met. We also hypothesize that the calculation of equilibrium states for deviation-only models with the extended CEM framework may be comparable to the numerical formulation of the *Update Reference Strategy* [10] and the *Force Density Method* [8,9], and as such, form-finding deviation-only models with our approach may share their drawbacks. A deeper investigation of this relationship is left to subsequent publications.

Fig. 15. Architectural interpretation of the constrained form diagram $\bar{\Gamma}$ of a spiral staircase generated with the extended CEM framework.

Future work should look into hybrid modeling strategies that guide designers to best combine auxiliary trails with standard trail and deviation edges during the construction of a topology diagram. Other future research directions are to add regularization terms to our current penalty approach to handle outlier constraints more robustly and to experiment with more types of objective functions as presented in [44]. We are also interested in leveraging more complex gradient-based optimization techniques such as Newton-based optimization methods that utilize the second-order derivatives (i.e. the Hessian) of the CEM form-finding algorithm to solve constrained form-finding problems more efficiently [37]. By delegating the calculation of derivative values to a computer with AD, we can now compose arbitrary design constraints and calculate higher-order derivatives with minimal friction: the only requisites are that the constraint and the objective functions are differentiable and written in (Python) code.

The adoption of computational techniques like AD can make gradient-based optimization more accessible to researchers in the field and it can propel the development of integrative and efficient frameworks that generate forms imbued with structural and other non-structural design requirements. We ultimately hope our work helps positioning constrained form-finding as a viable approach to tackle practical structural design problems on a wider spectrum of structural typologies, beyond the conventional catalog of shells and cable nets.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We thank Isabel Moreira de Oliveira from the Form-Finding Lab at Princeton University for her valuable suggestions during the development of this paper. This work was supported in part by the U.S. National Science Foundation under grant OAC-2118201 and the Deutsche Forschungsgemeinschaft, Germany under grant 434336509.

References

[1] Lewis WJ. Tension structures: form and behaviour. London: Thomas Telford; Distributors, ASCE Press; 2003.
[2] Bletzinger K. Fifty years of progress for shell and spatial structures : in celebration of the 50th anniversary jubilee of the IASS (1959-2009). Madrid: International Association for Shell and Spatial Structures; 2011.
[3] Argyris JH, Angelopoulos T, Bichat B. A general method for the shape finding of lightweight tension structures. Comput Methods Appl Mech Engrg 1974;3(1):135–49. http://dx.doi.org/10.1016/0045-7825(74)90046-2.
[4] Tabarrok B, Qin Z. Nonlinear analysis of tension structures. Comput Struct 1992;45(5–6):973–84. http://dx.doi.org/10.1016/0045-7949(92)90056-6.
[5] Barnes MR. Form Finding and Analysis of Tension Structures by Dynamic Relaxation. Int J Space Struct 1999;14(2):89–104. http://dx.doi.org/10.1260/0266351991494722.
[6] Kilian A, Ochsendorf J. Particle-Spring Systems for Structural Form Finding. J Int Assoc Shell Spat Struct 2005;46(2):77–84.
[7] Adriaenssens S, Ney L, Bodarwe E, Williams C. Finding the form of an irregular meshed steel and glass shell based on construction constraints. J Archit Eng 2012;18(3):206–13. http://dx.doi.org/10.1061/(asce)ae.1943-5568.0000074.
[8] Linkwitz K, Schek HJ. Einige Bemerkungen zur Berechnung von vorgespannten Seilnetzkonstruktionen. Ing-Arch 1971;40(3):145–58. http://dx.doi.org/10.1007/BF00532146.
[9] Schek H-J. The force density method for form finding and computation of general networks. Comput Methods Appl Mech Engrg 1974;3(1):115–34. http://dx.doi.org/10.1016/0045-7825(74)90045-0.
[10] Bletzinger K-U, Ramm E. A General Finite Element Approach to the form Finding of Tensile Structures by the Updated Reference Strategy. Int J Space Struct 1999;14(2):131–45. http://dx.doi.org/10.1260/0266351991494759.
[11] Nouri Baranger T. Form Finding Method of Tensile Fabric Structures: Revised Geometric Stiffness Method. J Int Assoc Shell Spat Struct 2002;43(1):13–21.
[12] Block P, Ochsendorf J. Thrust network analysis: A new methodology for three-dimensional equilibrium. J Int Assoc Shell Spat Struct 2007;48(3):167–73.
[13] Pauletti RMO, Pimenta PM. The natural force density method for the shape finding of taut structures. Comput Methods Appl Mech Engrg 2008;197(49):4419–28. http://dx.doi.org/10.1016/j.cma.2008.05.017.
[14] 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. http://dx.doi.org/10.1016/J.IJSOLSTR.2019.02.008.
[15] Hablicsek M, Akbarzadeh M, Guo Y. Algebraic 3D graphic statics: Reciprocal constructions. Comput Aided Des 2019;108:30–41. http://dx.doi.org/10.1016/j.cad.2018.08.003.
[16] Veenendaal D, Block P. An overview and comparison of structural form finding methods for general networks. Int J Solids Struct 2012;49(26):3741–53. http://dx.doi.org/10.1016/j.ijsolstr.2012.08.008.
[17] Fivet C, Zastavni D. A fully geometric approach for interactive constraint-based structural equilibrium design. Comput Aided Des 2015;61:42–57. http://dx.doi.org/10.1016/j.cad.2014.04.001.
[18] Rippmann M. Funicular shell design: geometric approaches to form finding and fabrication of discrete funicular structures (Ph.D. thesis), ETH Zurich; 2016, p. 374. http://dx.doi.org/10.3929/ETHZ-A-010656780.
[19] Senatore G, Piker D. Interactive real-time physics: An intuitive approach to form-finding and structural analysis for design and education. Comput Aided Des 2015;61:32–41. http://dx.doi.org/10.1016/j.cad.2014.02.007.
[20] Lee J, Mele TV, Block P. Disjointed force polyhedra. Comput Aided Des 2018;99:11–28. http://dx.doi.org/10.1016/j.cad.2018.02.004.
[21] Nejur A, Akbarzadeh M. PolyFrame, Efficient Computation for 3D Graphic Statics. Comput Aided Des 2021;134:103003. http://dx.doi.org/10.1016/j.cad.2021.103003.
[22] Michell A. The limits of economy of material in frame-structures. Lond Edinb Dublin Philos Mag J Sci 1904;8(47):589–97. http://dx.doi.org/10.1080/14786440409463229.
[23] Ohlbrock PO, Schwartz J. Combinatorial Equilibrium Modeling. Int J Space Struct 2016;31(2–4):177–89. http://dx.doi.org/10.1177/0266351116660799.

[24] Ohlbrock PO, D'Acunto P. A Computer-Aided Approach to Equilibrium Design Based on Graphic Statics and Combinatorial Variations. Comput Aided Des 2020;121:102802. http://dx.doi.org/10.1016/J.CAD.2019.102802.
[25] Ohlbrock PO. Combinatorial equilibrium modelling: a computational framework for equilibrium-based structural design (Ph.D. thesis), ETH Zurich; 2020, p. 238. http://dx.doi.org/10.3929/ETHZ-B-000478732.
[26] Bahr M. Form finding and analysis of shells and slabs based on equilibrium solutions (Ph.D. thesis), ETH Zurich; 2017, p. 141. http://dx.doi.org/10.3929/ETHZ-B-000182853.
[27] Panozzo D, Block P, Sorkine-Hornung O. Designing Unreinforced Masonry Models. ACM Trans Graph SIGGRAPH 2013 2013;32(4):91:1–91:12. http://dx.doi.org/10.1145/2461912.2461958.
[28] Tamai H. Advanced application of the force density method in multidisciplinary design practice by incorporating with optimization using analytical derivatives. In: Obrebski J, Tarczewski R, editors. Proceedings of the international association for shell and spatial structures (IASS) symposium 2013. Wroclaw, Poland; 2013, p. 9.
[29] Zhang JY, Ohsaki M. Adaptive force density method for form-finding problem of tensegrity structures. Int J Solids Struct 2006;43(18):5658–73. http://dx.doi.org/10.1016/j.ijsolstr.2005.10.011.
[30] Allen E, Zalewski W. Form and forces: designing efficient, expressive structures. John Wiley & Sons; 2009.
[31] Miki M, Kawaguchi K. Extended Force Density Method for Form-Finding of Tension Structures. J Int Assoc Shell Spat Struct 2010;51(4):13.
[32] Malerba P, Patelli M, Quagliaroli M. An Extended Force Density Method for the form finding of cable systems with new forms. Struct Eng Mech 2012;42:191–210. http://dx.doi.org/10.12989/SEM.2012.42.2.191.
[33] Quagliaroli M, Malerba PG. Flexible bridge decks suspended by cable nets. A constrained form finding approach. Int J Solids Struct 2013;50(14):2340–52. http://dx.doi.org/10.1016/j.ijsolstr.2013.03.009.
[34] Ohlbrock PO, D'acunto P, Jasienski J-P, Fivet C. Constraint-Driven Design with Combinatorial Equilibrium Modelling. In: Proceedings of IASS annual symposia, Vol. 2017. Hamburg, Germany: International Association for Shell and Spatial Structures (IASS); 2017, p. 1–10, URL https://www.ingentaconnect.com/content/iass/piass/2017/00002017/00000015/art00013.
[35] Takahashi K, Ney L. Advanced form finding by constraint projections for structural equilibrium with design objectives. In: Proceedings of the IASS symposium 2018. Boston, USA; 2018, p. 8.
[36] Cuvilliers P, Danhaive R, Mueller C. Gradient-based optimization of closest-fit funicular structures. In: Kawaguchi K, Ohsaki M, Takeuchi T, editors. Proceedings of the IASS annual symposium 2016. Tokyo, Japan; 2016 p. 10.
[37] Nocedal J, Wright SJ. Numerical optimization. Springer series in operations research, 2nd ed.. New York: Springer; 2006.
[38] Haase G, Langer U, Lindner E, Mühlhuber W. Optimal Sizing Using Automatic Differentiation. In: Hoffmann K-H, Hoppe RHW, Schulz V, editors. Fast solution of discretized optimization problems. ISNM international series of numerical mathematics, Basel: Birkhäuser; 2001, p. 120–38. http://dx.doi.org/10.1007/978-3-0348-8233-0_10.
[39] Corliss G, Faure C, Griewank A, Hascoet L, Naumann U. Automatic differentiation of algorithms: from simulation to optimization. New York, NY: Springer; 2013, URL https://link.springer.com/book/10.1007/978-1-4613-0075-5.
[40] Baydin AG, Pearlmutter BA, Radul AA, Siskind JM. Automatic differentiation in machine learning: A survey. 2018, arXiv:1502.05767 [cs, stat]. URL http://arxiv.org/abs/1502.05767.
[41] Oberbichler T, Wüchner R, Bletzinger K-U. Efficient computation of nonlinear isogeometric elements using the adjoint method and algorithmic differentiation. Comput Methods Appl Mech Engrg 2021;381:113817. http://dx.doi.org/10.1016/j.cma.2021.113817.
[42] Unger E, Hall L. The use of automatic differentiation in an aircraft design problem. In: 5th symposium on multidisciplinary analysis and optimization. Panama City Beach,FL,U.S.A.: American Institute of Aeronautics and Astronautics; 1994, p. 64–72. http://dx.doi.org/10.2514/6.1994-4260.
[43] Cho H-N, Min D-H, Lee K-M, Kim H-K. Multi-Level and Multi-Objective Optimization of Framed Structures Using Automatic Differentiation. In: Proceedings of the computational structural engineering institute conference. Computational Structural Engineering Institute of Korea; 2000 URL https://koreascience.kr/article/JAKO200010102436328.view.
[44] Cuvilliers P. The constrained geometry of structures: optimization methods for inverse form-finding design (Ph.D. thesis), Massachusetts Institute of Technology; 2020, URL http://dspace.mit.edu/handle/1721.1/7582.
[45] Pastrana R, Ohlbrock PO, D'Acunto P, Parascho S. Supplementary data for the paper constrained form-finding of tension-compression structures using automatic differentiation. 2021, URL https://github.com/arpastrana/cem_ad_cad.
[46] Maxwell JC. On Reciprocal Figures, Frames, and Diagrams of Forces. Trans R Soc Edinb 1870;26(1):1–40. http://dx.doi.org/10.1017/S0080456800026351.
[47] Beghini LL, Carrion J, Beghini A, Mazurek A, Baker WF. Structural optimization using graphic statics. Struct Multidiscip Optim 2014;49(3):351–66. http://dx.doi.org/10.1007/s00158-013-1002-x.
[48] Liew A, Avelino R, Moosavi V, Van Mele T, Block P. Optimising the load path of compression-only thrust networks through independent sets. Struct Multidiscip Optim 2019;60(1):231–44. http://dx.doi.org/10.1007/s00158-019-02214-w.
[49] Nocedal J. Updating quasi-Newton matrices with limited storage. Math Comp 1980;35(151):773. http://dx.doi.org/10.1090/S0025-5718-1980-0572855-7.
[50] Kraft D. Algorithm 733: TOMP–Fortran modules for optimal control calculations. ACM Trans Math Software 1994;20(3):262–81. http://dx.doi.org/10.1145/192115.192124.
[51] Dembo RS, Steihaug T. Truncated-Newton algorithms for large-scale unconstrained optimization. Math Program 1983;26(2):190–212. http://dx.doi.org/10.1007/BF02592055.
[52] Wengert RE. A simple automatic derivative evaluation program. Commun ACM 1964;7(8):463–4. http://dx.doi.org/10.1145/355586.364791.
[53] Bauer FL. Computational graphs and rounding error. SIAM J Numer Anal 1974;11(1):87–96, URL http://www.jstor.org/stable/2156433.
[54] Griewank A, Walther A. Evaluating derivatives: principles and techniques of algorithmic differentiation. SIAM; 2008, http://dx.doi.org/10.1137/1.9780898717761, URL https://epubs.siam.org/doi/abs/10.1137/1.9780898717761.
[55] Oktay D, McGreivy N, Aduol J, Beatson A, Adams RP. Randomized Automatic Differentiation. In: International conference on learning representations. 2020, p. 1–19, URL https://openreview.net/forum?id=xpx9zj7CUlY.
[56] Pastrana R, Ohlbrock PO, D'Acunto P, Parascho S. COMPAS CEM: The Combinatorial Equilibrium Modeling framework for COMPAS. 2021, http://dx.doi.org/10.5281/zenodo.5705740, URL https://arpastrana.github.io/compas_cem.
[57] The Python Software Foundation. Python. 2021, URL https://www.python.org/.
[58] Mele TV, et al. COMPAS: A framework for computational research in architecture and structures. 2017-2021, http://dx.doi.org/10.5281/zenodo.2594510, http://compas.dev.
[59] Liew A, Mendez-Echenagucia T, Ranaudo F, Van Mele T. COMPAS FEA: Finite element analysis using Abaqus, Ansys, or OpenSEES. 2021, URL https://https://compas.dev/compas_fea/.
[60] Bernhard M, Van Mele T, Casas G, Clemente R, Feihl N. COMPAS VOL: Volumetric modelling with signed distance functions. 2021, URL https://github.com/dbt-ethz/compas_vol.
[61] Rust R, Casas G, Parascho S, Jenny D, Dörfler K, Helmreich M, Gandia A, Ma Z, Ariza I, Pacher M, Lytle B, Huang Y. COMPAS FAB: Robotic fabrication package for the COMPAS framework. 2018, http://dx.doi.org/10.5281/zenodo.3469478, Gramazio Kohler Research, ETH Zürich. https://github.com/compas-dev/compas_fab/.
[62] Ohlbrock PO, D'Acunto P. CEM: Combinatorial Equilibrium Modeling. 2021, Release 2.00. URL http://github.com/OleOhlbrock/CEM.
[63] Rutten D. Grasshopper. 2007, URL https://www.grasshopper3d.com/.
[64] Blender Online Community. Blender - free and open 3D creation software. 2021, URL http://www.blender.org.
[65] Robert McNeel & Associates. Rhinoceros3d. 2007, URL https://www.rhino3d.com/.
[66] Johnson SG. The NLopt nonlinear-optimization package. 2021, URL https://github.com/stevengj/nlopt.
[67] Maclaurin D, Duvenaud D, Adams RP. Autograd: Effortless gradients in numpy. In: ICML 2015 AutoML workshop, Vol. 238. 2015, p. 5, URL https://github.com/HIPS/autograd.
[68] Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, Burovski E, Peterson P, Weckesser W, Bright J, van der Walt SJ, Brett M, Wilson J, Millman KJ, Mayorov N, Nelson ARJ, Jones E, Kern R, Larson E, Carey CJ, Polat İ, Feng Y, Moore EW, VanderPlas J, Laxalde D, Perktold J, Cimrman R, Henriksen I, Quintero EA, Harris CR, Archibald AM, Ribeiro AH, Pedregosa F, van Mulbregt P, SciPy 10 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods 2020;17:261–72. http://dx.doi.org/10.1038/s41592-019-0686-2.
[69] Oberbichler T. HyperJet. 2021, http://dx.doi.org/10.5281/zenodo.5093152.
[70] Kluyver T, Ragan-Kelley B, Pérez F, Granger B, Bussonnier M, Frederic J, Kelley K, Hamrick J, Grout J, Corlay S, Ivanov P, Avila D, Abdalla S, Willing C, development team J. Jupyter Notebooks - a publishing format for reproducible computational workflows. In: Loizides F, Scmidt B, editors. Positioning and power in academic publishing: players, agents and agendas. IOS Press; 2016, p. 87–90, URL https://eprints.soton.ac.uk/403913/.

[71] Conn AR, Gould NIM, Toint P. A Globally Convergent Augmented Lagrangian Algorithm for Optimization with General Constraints and Simple Bounds. SIAM J Numer Anal 1991;28(2):545–72. http://dx.doi.org/10.1137/0728030.

[72] Zordan T, Briseghella B, Siviero E. The Fourth Bridge over the Grand Canal in Venice: From Idea to Analysis and Construction. Struct Eng Int 2010;20(1):6–12. http://dx.doi.org/10.2749/101686610791555667.