Advances in Computational Science and Engineering
Vol. 2, No. 1, March 2024, pp. 37-72
doi:10.3934/acse.2024005

A COMPUTATIONAL FRAMEWORK FOR THE DESIGN OF TENSILE STRUCTURES

Marcio S. V. de Souza* and Ruy M. O. Pauletti

Structural and Geotechnical Department of the Polytechnic School of the University of São Paulo, Brazil

Abstract. This paper introduced a parametric workflow for the design of tensile structures, encompassing form-finding, patterning, flattening, and geometrically nonlinear structural analysis of cables and membranes, and how these design steps can be sequenced and iterated. The concepts of parametric design and visual programming languages were outlined, with a particular focus on structural design. The implementation of these concepts was straightforward, and the paper explained their application into a new Grasshopper plug-in, called BATS - basic analysis of tensile structures, capable of performing the full design cycle of this type of structures.

1. **Introduction.** Tensile structures, such as membrane and cable structures, have become prominent components within the realm of civil engineering and architecture with a broad range of applications, from iconic architectural landmarks and large enclosures to small, prosaic canopies. They usually consist of polyvinyl chloride (PVC) or polytetrafluoroethylene (PTFE) coated fabrics made of polyester of fiberglass strands and reinforced with steel cables, which present very low resistance to bending moments, and, therefore, also very low resistance to compressive forces, since the material buckles under any compression state. These conditions render tensile structures flexible structural systems that withstand only with internal tensile forces. This often leads to large displacements and wrinkles during a load response, which requires a nonlinear static analysis framework to properly evaluate the stress and displacements.

Several authors [84, 4, 16, 22, 17, 26] commonly divide the design of tensile structures into four main steps: (1) form-finding of the membrane shape in a prestressed state, (2) geometrically nonlinear structural analysis, (3) patterning, and (4) flattening of the fabric strips. Figure 1 shows one classic approach of a design workflow. The form-finding process is performed to find a viable shape in equilibrium under a predetermined initial stress field input and the boundary conditions (supports, external loads) that fulfill architectural requirements. Once a viable shape is found, structural analysis is performed to determine the response of the system to external load conditions, where safety and functional requirements must be met.

2020 *Mathematics Subject Classification.* Primary: 58F15, 58F17; Secondary: 53C35.

Key words and phrases. Tensile structures, Parametric design,Visual programming language, Tensile architecture, Computational engineering.

*Corresponding author: Marcio S. V. de Souza.

Tensile structure fabrics are commonly manufactured as planar sheets, which requires the 3D shape to be flattened in a 2D geometry, presenting some difficulties. Tensile structure surfaces commonly present a high degree of double curvature, and thus are non-developable, and it is impossible to find a flattened shape without deviations. To minimize deviation from this process, the patterning (or cutting pattern generation) step is performed, where the 3D surface is normally subdivided into a set of fabric strips, which are flattened individually. Patterning reduces the total level of double curvature and helps to reduce the deviation from the idealized 3D geometry and also fulfills the manufacturing restriction on the width of a 2D fabric sheet, which is generally manufactured in 2-3m widths. Finally, the flattening process takes each segmented strip and finds the unstressed 2D strip in which the deviation from the 3D counterpart is minimized. Since the idealized assembled fabric is under prestress, the flattening process also needs to take into account the effect of compensation, in which the unstressed 2D fabric will achieve the desired prestress when assembled into the idealized shape, and results in a slightly shorter fabric than one from a flattening process with no considerations of the stress state of the assembled shape. More details on this topic are given in Section 5.

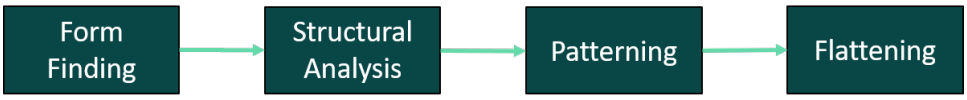

FIGURE 1. Linear workflow in the design of tensile structures

In practice, this process is limited to flexible isotropic or slightly orthotropic materials, since in structural analysis the stress field of the assembled surface is assumed to be the same as the idealized stress field from the form-finding process. Due to the inherent deviation error in the flattening process, the assembled surface will also contain deviations in the resulting stress field, which increases with the stiffness of the fabric. In nonlinear orthotropic materials, this process can also lead to distortions because the layout of the fibers over the membrane surface usually is not known *a priori*, usually requiring some design iterations to find a satisfactory layout of the orthotropy direction for each fabric pattern.

Figure 2 shows alternatives to the design process, obtained by reordering the design steps presented in Figure 1. The main idea is to move the patterning and flattening processes prior to structural analysis. In this way, it is possible to determine the real initial stress field that differs from the one prescribed in form-finding due to the patterning and flattening geometry deviations, which is then used for further load response analysis. The real initial stress field also contains information about possible regions in which the fabric can wrinkle; although it is an acceptable effect in load response, it is not desired to happen in the assembled state under normal conditions. References [40, 54, 16] propose linear workflows to retrieve the stress deviation from the flattening process and accurately define the fiber orientation of each tensile seam for the final analysis. Other workflows were proposed by defining energy minimization optimization problems between design steps [7, 42, 33, 48, 47].

The final design is usually the result of an iterative process between the outlined design steps, a process that normally differs from the more 'one-way' process used in the design of more conventional structural systems, which usually have an

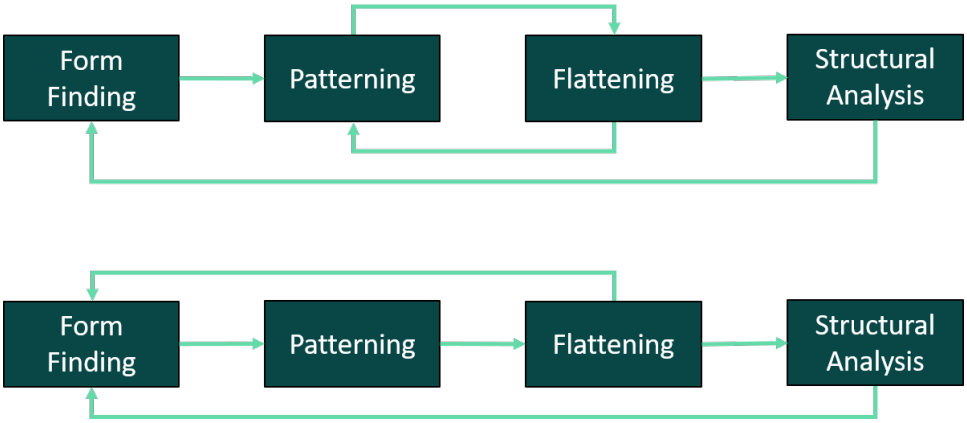

FIGURE 2. Iterative workflows in the design of tensile structures

architectural shape given beforehand, to which the structure must comply, while in tensile structures geometry is strongly coupled to the structural performance.

The complexities arising from the aforementioned and design phases render the use of analytical tools impractical for the proper design and analysis of tensile structures, except for few particular cases. Only with advances in CAD (computer-aided design) and CAE (computer-aided engineering) did the design of tensile structures reach maturity. In fact, the history of modern tensile architecture is closely related to the development of numerical methods for their design [1].

However, the use of general CAD and CAE software may be challenging when it comes to tensile structural design, as it might require considerable effort to accurately define all design stages and their information flow. To address this issue, many specialized software have been developed for structural design [43, 31, 20] with built-in features to handle these design steps. Additionally, some general CAE software also offers special modules for form-finding, patterning, and flattening of membranes in addition to the analysis itself. Although these solutions significantly improve the user experience in the design process, they are usually provided as sandbox software solutions that do not communicate directly with other CAD and CAE systems, and provide only limited options to automate the integration between the design phases.

Going beyond this paradigm, many CAD products offer visual programming language (VPL) interfaces to enable the user to model geometry with the use of parameters and design rules. Rhinoceros3D (Rhino) [50] is a well-known CAD software in architecture and engineering industries and provides its built-in VPL Grasshopper [71], which has components for manipulation of many Rhino geometries, allowing the creation and modification of geometries ruled by parameters instead of manual modeling. The user can define a logical sequence of events describing the project as a function of predefined parameters and instantly retrieve geometry and analysis feedback within the changes to them.

Besides, Rhino and Grasshopper both have their own application programming interfaces (APIs), which enable the development of custom components and parameters for a variety of purposes. Karamba3d [66] was a pioneer in creating a finite element analysis plug-in for Grasshopper, which revolutionized the way architectural and structural modeling systems are integrated. This plug-in eliminates the need

for external CAE systems, allowing users to define their architectural intentions and automatically convert them into finite element models. This means that structural analysis can be done directly within a CAD environment, with (almost) instantaneous analysis feedback, resulting in the use of a unified environment for CAD and CAE modeling. This workflow can also be combined with other tools inside a Grasshopper environment, such as heuristic optimization tools to minimize defined objective functions in terms of the parameters prescribed in the Grasshopper script. Integration of these optimization tools with structural analysis enables the setting of structural optimization routines to minimize or maximize a given output of the structure (weight, cost, stiffness, span, etc.) [75, 64, 65, 12, 13].

Grasshopper has become a popular choice for structural design, both for its ability to handle complex shapes and for automating design tasks and geometry integration with CAE systems. Some analysis softwares with tensile structure features have developed Grasshopper interfaces to enable a parametric definition of their CAE models, such as ix-Cube[31] and RFEM [19]. However, most of these packages only provide a way to transfer Rhino/Grasshopper data to the original simulation system, meaning that the user needs to manually open the CAE software and initiate the analysis, which still requires the operation of separate CAD and CAE frameworks. However, some plug-ins were proposed to enable an approach to the design of tensile structures fully inside the Grasshopper environment. Kangaroo [63] is a general purpose physics simulation engine, which allows mesh-based form-finding of tensile membranes and could theoretically support the full design cycle if custom components for membranes are implemented. Kiwi3d is a Grasshopper plug-in that allows parametric definition of the tensile structure design cycle, which includes form-finding, analysis, patterning, and assembly of tensile structures with native Grasshopper / Rhino surface CAD geometries combined with isogeometric analysis [6, 22, 21]. EQlib [55] is a tool for the form-finding and analysis of tensile structures considering a wide range of computational geometries such as meshes, nonuniform rational b-splines (NURBS) surfaces, and subdivision surfaces. Other plug-ins are available for the specific task of form-finding, such as RhinoMembrane[30] and RhinoVault [23].

This paper presents BATS (basic analysis of tensile structures), a new mesh-based Grasshopper plug-in for the complete design process of tensile structures. The first of the BATS developments was on a high performance form-finding tool [15], and the present version includes a series of components that allow the user to define structural membranes and cables to carry out the entire design workflow, including form-finding, patterning, flattening, and structural analysis considering geometric and material nonlinearity. A relevant feature is the ability to handle wrinkling and slackening effects naturally on the membrane and cable elements. The main objective of the tool is to be a simple, fast, and flexible tool to handle the entire tensile structures design workflow. For this purpose, simplex finite elements are implemented in a dedicated C++ numerical solver, and tools to manipulate the meshes for each specific design step are available, taking advantage of native Grasshopper mesh editing capabilities. The use of meshes also has the advantage of being compatible with a wide range of other analysis and editing tools for Grasshopper.

This paper introduces the fundamentals of geometrically nonlinear structural analysis for cables and membranes, as well as the strategies for form-finding, patterning, and flattening, and how these design steps can be sequenced and iterated. The paper discusses the concepts of parametric design and VPL programming and how

they allow users with little knowledge of traditional programming to automate their CAD/CAE workflows, with a special emphasis on structural design. Finally, the implementation of the BATS parametric workflow for the design of tensile structures is discussed, and some selected applications are briefly described.

2. **Structural analysis.** Structural analysis of possible load cases in a tensile structure must be considered to check design ultimate (ULS) and serviceability (SLS) limit states. This is the fundamental step to validate the structural safety and determine the cross-sections needed for all elements. An extra type of analysis for tensile structures is the assembly analysis, which is performed considering the deformation from the flattened segments to the final structure in order to correctly define the prestress conditions, which unavoidably deviates from the idealized stress field determined by the form-finding algorithms, due to inherent error that comes from the flattening of double curvature strips.

From the prestressed state (either idealized or real), the structure shall be analyzed for all combinations of live loads. Traditionally, the analysis of tensile membranes and cables net disregards gravity effects, since weight loads are small when compared to the live loads, but they can also be considered without extra computational cost. The main external loads that affect common tensile structures are wind and snow loads, as tensile roofs usually consist of large-span systems with low stiffness. The definition of wind loads can be challenging due to the surface shapes of membranes, which increases the complexity in determining the pressure coefficients. Reference [77] is a recent report on the design of tensile structures that gives general guidelines for determining the pressure coefficients of the tensile shapes, which can be done by small-scale physical models tested in wind tunnels or computational fluid dynamics (CFD) models, although it is not recommended to rely only on CFD analysis. The guidelines also refer to other works in which common shape pressure coefficients such as hyperbolic paraboloids, conoids, and stadium roofs are given [17]. Other specific geometries are studies in references [67, 68, 69, 28].

The evaluation of displacements and stresses can be done with the help of numerical methods, especially the finite element method (FEM). In this paper, we focus on very simple finite elements to model the main structural components of tensile structures: straight trusses and cable elements, and flat triangular membrane elements. In recent years, isogeometric analysis (IGA) for tensile structures has been proposed [6, 22, 55] as an alternative to design and analysis. IGA directly uses CAD generated surfaces as shape functions of the finite element problem, such as NURBS, which is a valuable feature in terms of CAD/CAE integration and a promising research field.

Nevertheless, in the authors' experience, the choice of such simple elements with more refined meshes provides good numerical performance and reliable results, and simplifies both pre and post-processing of the structural model. The selection of simple elements allowed us to focus on the bigger scope of this work, which is the implementation of a computational framework capable of performing the whole design cycle for those structural systems. However, the BATS framework is sufficiently general to allow the future inclusion of more sophisticated elements.

2.1. **Trusses and cables.** Trusses are the simplest type of structural system widely used in civil engineering. Each truss member is considered to be pin-jointed at its ends, and external loads are considered to be applied only at the nodes, and, consequently, the members are subjected only to an axial force. This leads also to a

simplified mechanical formulation, in which only the structural axis of the truss is relevant. The high axial stiffness presented in common civil structures makes the displacement behavior of these systems mostly small enough to consider infinitesimal displacements in the mechanical model.

Cables are a special type of truss system presented in most of tensile structures. They can be represented, in a simplified way, by a chain of trusses that can only withstand tensile forces, because of its lack of bending stiffness. This causes cables to automatically buckle under compression, and a simplified approach of the behavior is to neglect the stiffness of the cable during compression. This normally results in highly flexible systems, and cable displacements cannot be considered infinitesimal anymore, which requires a geometrically exact formulation [86, 58, 9, 5, 11]. Cables can then be simplified as a chain of small truss elements.

The coordinate vector $\boldsymbol{x}$ for a 2-node truss element is given by

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{bmatrix} \tag{1}$$

where $\boldsymbol{x}_i$ is the 3D coordinate of the node i of the truss. We can define the displacement field in terms of the difference in length Δl between different configurations.

$$\Delta l = l - l^r \tag{2}$$

$$l\boldsymbol{v} = \boldsymbol{x}_2 - \boldsymbol{x}_1 \tag{3}$$

$$l^2 = (\boldsymbol{x}_2 - \boldsymbol{x}_1) \cdot (\boldsymbol{x}_2 - \boldsymbol{x}_1) = (l\boldsymbol{v}) \cdot (l\boldsymbol{v}) \tag{4}$$

where l and l^r are, respectively, the current length and the reference length of the truss, and $\boldsymbol{v}$ is the unit vector of the axis of the truss. It is possible to write equation 3 in terms of a linear transformation given by $\boldsymbol{\Psi} = \begin{bmatrix} -\boldsymbol{I} & \boldsymbol{I} \end{bmatrix}$

$$l\boldsymbol{v} = \boldsymbol{\Psi}\boldsymbol{x} \tag{5}$$

The uniform Green strain tensor g and its variation δg within the element is given by the quadratic difference of the lengths in reference and in the current configuration.

$$g = \frac{1}{2}\frac{l_i^2 - {l_i^r}^2}{{l_i^r}^2} = \frac{1}{2}\frac{(\boldsymbol{\Psi}\boldsymbol{x}) \cdot (\boldsymbol{\Psi}\boldsymbol{x}) - {l_i^r}^2}{{l_i^r}^2} \tag{6}$$

$$\delta g = \frac{1}{2{l_i^r}^2}\delta(l_i^2) = \frac{1}{{l_i^r}^2}\boldsymbol{\Psi}\boldsymbol{x} \cdot (\boldsymbol{\Psi}\delta\boldsymbol{x}) = \frac{l}{{l_i^r}^2}\boldsymbol{\Psi}^T\boldsymbol{v} \cdot \delta\boldsymbol{x} \tag{7}$$

The internal part of the weak form of the truss equilibrium δW_{int} is

$$\delta W_{int} = \int_0^l (N\delta g)dl = \int_0^{l^r} \left(N\frac{l}{{l^r}^2}\boldsymbol{\Psi}^T\boldsymbol{v} \cdot \delta\boldsymbol{x}\right) dl^r \tag{8}$$

where N is the normal force acting on the truss, and it is given by the constitutive equation

$$N = Eg + N_{pre} \tag{9}$$

where N_{pre} is the force related to the prestress applied to the truss and E is the linear Young modulus of the material. We can define equation 8 in terms of the discretized internal force vector $\boldsymbol{f}^{int}$:

$$\delta W_{int} = \boldsymbol{f}^{int} \cdot \delta\boldsymbol{x} = \left(\frac{l}{l^r}N\boldsymbol{\Psi}^T\boldsymbol{v}\right) \cdot \delta\boldsymbol{x} \tag{10}$$

For cables, there is a constraint to the normal force to not withstand compression, which gives the following conditions:

$$\boldsymbol{f}^{int} = \begin{cases} 0, & N < 0 \\ \dfrac{l}{l^r} N \boldsymbol{\Psi}^T \boldsymbol{v}, & N \geq 0 \end{cases} \tag{11}$$

The variation of the internal force vector results in the discretized stiffness matrix $\boldsymbol{K}$ multiplied by the variation of the position vector.

$$\delta \boldsymbol{f}^{int} = \frac{1}{l^r} \left(l \delta(N) \boldsymbol{\Psi}^T \boldsymbol{v} + N \boldsymbol{\Psi}^T \delta(l\boldsymbol{v}) \right) = \left(\boldsymbol{K}^e + \boldsymbol{K}^g \right) \delta \boldsymbol{x} \tag{12}$$

where $\boldsymbol{K}^e$ is the elastic part and $\boldsymbol{K}^g$ the geometric part of the stiffness matrix $\boldsymbol{K} = \boldsymbol{K}^e + \boldsymbol{K}^g$, defined by recalling equations 12 and 5:

$$\boldsymbol{K}^e = \frac{l^2}{l^{r3}} EA \boldsymbol{\Psi}^T (\boldsymbol{v} \otimes \boldsymbol{v}) \boldsymbol{\Psi} \tag{13}$$

$$\boldsymbol{K}^g = \frac{N}{l^r} \boldsymbol{\Psi}^T \boldsymbol{\Psi} \tag{14}$$

For cables when $N < 0$, then also $\boldsymbol{K} = \boldsymbol{0}$.

2.2. **Membranes.** Membrane elements are surface elements that can only resist in-plane forces, having negligible bending stiffness and, therefore, are highly flexible systems. Sophisticated formulations of membrane elements using continuum mechanics is given by [24, 10, 80, 18, 81, 6].

2.2.1. *Natural membrane discretization.* We herein focus on the natural membrane finite element formulation given by [3] and further developed by [58]. Reference [3] introduces the concept of natural stresses and strains in simplex finite elements. The most common simplex membrane finite element is the 3-node CST (constant strain triangle), in which the displacements are linearly interpolated, resulting in constant strain and stress along its surface area [62, 61]. Figure 3 defines the membrane element in three configurations. The shape is defined by a set of three coordinates $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3\}$, in which a local basis is set by the basis vectors $\{\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3\}$. The coordinates are defined in the reference $(\cdot^r)$, initial $(\cdot^0)$, and current configuration $(\cdot)$. The initial configuration refers to the element configuration before deformation on any arbitrary basis, the reference configuration is the initial configuration transformed to the working basis, and the current configuration describes the coordinates after displacement.

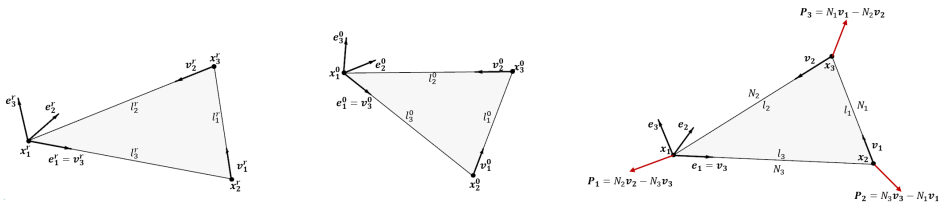

FIGURE 3. Natural membrane element: a) reference configuration, b) initial configuration, and c) current configuration

In the natural approach, the displacement field is given by vector $\boldsymbol{\Delta l}$, representing the change in length between the edges of the element.

$$\boldsymbol{\Delta l} = \boldsymbol{l} - \boldsymbol{l}^r \tag{15}$$

$$l_i \boldsymbol{v}_i = \boldsymbol{x}_k - \boldsymbol{x}_j \tag{16}$$

$$l_i^2 = (\boldsymbol{x}_k - \boldsymbol{x}_j) \cdot (\boldsymbol{x}_k - \boldsymbol{x}_j) = (l_i \boldsymbol{v}_i) \cdot (l_i \boldsymbol{v}_i) \tag{17}$$

where $\boldsymbol{l}$ and $\boldsymbol{l}^r$ are, respectively, the current and reference length vectors, $\boldsymbol{v}_i$ is the unit vector of the membrane edge i, and $i, j, k \in \{1, 2, 3\}$ permute cyclically. It is possible to write equation 16 in terms of a linear transformation of the global position vector $\boldsymbol{x}$ with the operator $\boldsymbol{\Psi}$:

$$l_i \boldsymbol{v}_i = \boldsymbol{\Psi}_i \boldsymbol{x} \tag{18}$$

where

$$\boldsymbol{\Psi}_1 = \begin{bmatrix} \boldsymbol{0} & -\boldsymbol{I} & \boldsymbol{I} \end{bmatrix}$$

$$\boldsymbol{\Psi}_2 = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & -\boldsymbol{I} \end{bmatrix} \tag{19}$$

$$\boldsymbol{\Psi}_3 = \begin{bmatrix} -\boldsymbol{I} & \boldsymbol{I} & \boldsymbol{0} \end{bmatrix}$$

The natural Green strain $\boldsymbol{g}^n$ is then defined by

$$g_i^n = \frac{1}{2} \frac{l_i^2 - {l_i^r}^2}{{l_i^r}^2} \tag{20}$$

Introducing equations 17 and 18 in 20, we have

$$g_i^n = \frac{1}{2} \frac{(l_i \boldsymbol{v}_i) \cdot (l_i \boldsymbol{v}_i) - {l_i^r}^2}{{l_i^r}^2} = \frac{1}{2} \frac{(\boldsymbol{\Psi}_i^T \boldsymbol{\Psi}_i \boldsymbol{x}) \cdot \boldsymbol{x} - {l_i^r}^2}{{l_i^r}^2} \tag{21}$$

The variation of the strain vector $\delta \boldsymbol{g}^n$ in function of $\boldsymbol{x}$ can be obtained as

$$\delta g_i^n = \frac{l_i}{{l_i^r}^2} \boldsymbol{\Psi}_i^T \boldsymbol{v}_i \cdot \delta \boldsymbol{x} \tag{22}$$

The conventional representation of the Green strain tensor $\boldsymbol{E}$ is given by

$$\boldsymbol{E} = \boldsymbol{g_\alpha} \otimes \boldsymbol{e}_{\boldsymbol{\alpha}}^r + \boldsymbol{g_3} \otimes \boldsymbol{e}_{\boldsymbol{3}}^r \tag{23}$$

where $\boldsymbol{g_3} = g_{33} \boldsymbol{e}_{\boldsymbol{3}}^r$ represents the membrane out-of-plane strain in the reference configuration. An interesting property of the Green strain tensor is that the strain vectors $\boldsymbol{g_\alpha}$ are defined in the reference plane ${\boldsymbol{e_1}}^r, {\boldsymbol{e_2}}^r$, which gives

$$g_{13} = g_{23} = 0 \quad for \quad i = 1, 2 \tag{24}$$

Also, $\boldsymbol{E}$ is a symmetric tensor, which gives $g_{12} = g_{21}$. The tensor can be represented in matrix form as

$$\boldsymbol{E} = \begin{bmatrix} \overline{\boldsymbol{E}} & \boldsymbol{0} \\ \boldsymbol{0} & g_{33} \end{bmatrix} = \begin{bmatrix} \overline{\boldsymbol{g}}_1 & \overline{\boldsymbol{g}}_2 & \boldsymbol{0} \\ 0 & 0 & g_{33} \end{bmatrix} \tag{25}$$

where $\overline{(\cdot)}$ is the 2d representation of $(\cdot)$.

The natural Green strain in 20 can also be represented in terms of the conventional Green strain tensor as

$$g_i^n = \overline{\boldsymbol{v}}_i^r \cdot \overline{\boldsymbol{E}} \overline{\boldsymbol{v}}_i^r = \overline{\boldsymbol{v}}_i^r \cdot (\overline{\boldsymbol{g}}_\alpha \otimes \overline{\boldsymbol{e}}_\alpha^r) \overline{\boldsymbol{v}}_i^r = (\overline{\boldsymbol{v}}_i^r \cdot \overline{\boldsymbol{e}}_\alpha^r)(\overline{\boldsymbol{g}}_\alpha \cdot \overline{\boldsymbol{v}}_{\boldsymbol{i}}^{\boldsymbol{r}}) \tag{26}$$

Since the transformation between the reference configuration and the initial configuration is only an imposed rigid body motion, we can define equation 26 in terms of the initial configuration 3d vectors since $\overline{\boldsymbol{v}}_i^r \cdot \overline{\boldsymbol{e}}_\alpha^r = \boldsymbol{v}_i^0 \cdot \boldsymbol{e}_\alpha^0$. Also, noting in

3 that $\boldsymbol{v}_3^0 \cdot \boldsymbol{e}_1^0 = 1$ and $\boldsymbol{v}_3^0 \cdot \boldsymbol{e}_2^0 = \boldsymbol{v}_3^0 \cdot \boldsymbol{e}_3^0 = 0$, we can rewrite the relation as a linear transformation $\boldsymbol{T}$:

$$\boldsymbol{g}^n = \boldsymbol{T}\boldsymbol{g}^v \tag{27}$$

$$\boldsymbol{T} = \begin{bmatrix} (\boldsymbol{v}_1^0 \cdot \boldsymbol{e}_1^0)^2 & (\boldsymbol{v}_1^0 \cdot \boldsymbol{e}_2^0)^2 & (\boldsymbol{v}_1^0 \cdot \boldsymbol{e}_1^0)(\boldsymbol{v}_1^0 \cdot \boldsymbol{e}_2^0) \\ (\boldsymbol{v}_2^0 \cdot \boldsymbol{e}_1^0)^2 & (\boldsymbol{v}_2^0 \cdot \boldsymbol{e}_2^0)^2 & (\boldsymbol{v}_2^0 \cdot \boldsymbol{e}_1^0)(\boldsymbol{v}_2^0 \cdot \boldsymbol{e}_2^0) \\ 1 & 0 & 0 \end{bmatrix} \tag{28}$$

where $\boldsymbol{g}^v$ is the green strain vector in the Voigt notation:

$$\boldsymbol{g}^v = \begin{bmatrix} g_{11} & g_{22} & 2g_{12} \end{bmatrix}^T \tag{29}$$

We can also define the second Piola-Kirchoff stress tensor $\boldsymbol{S} = \boldsymbol{s}_i \otimes \boldsymbol{e}_i$ in the Voigt notation as $\boldsymbol{s}^v$:

$$\boldsymbol{s}^v = \begin{bmatrix} s_{11} & s_{22} & s_{12} \end{bmatrix}^T \tag{30}$$

It is possible to define a natural stress vector $\boldsymbol{s}^n$, which is energetically conjugate to the natural Green strain tensor with the use the theorem of virtual displacements

$$\boldsymbol{s}^v \cdot \delta\boldsymbol{g}^v = \boldsymbol{s}^n \cdot \delta\boldsymbol{g}^n = \boldsymbol{s}^n \cdot \delta(\boldsymbol{T}\boldsymbol{g}^v) = \boldsymbol{T}^T\boldsymbol{s}^n \cdot \delta\boldsymbol{g}^v \tag{31}$$

which gives

$$\boldsymbol{s}^n = \boldsymbol{T}^{-T}\boldsymbol{s}^v \tag{32}$$

The relationship between stress and strain is given by the constitutive equation, also in terms of the Voigt notation $\boldsymbol{C}^v$

$$\boldsymbol{s}^v = \boldsymbol{C}^v\boldsymbol{g}^v + \boldsymbol{s}^v_{pre} \tag{33}$$

It is also possible to write a natural constitutive tensor $\boldsymbol{C}^n$ by inserting equations 27 and 32 into equation 33;

$$\boldsymbol{s}^n = \boldsymbol{T}^{-T}\boldsymbol{C}^v\boldsymbol{T}^{-1}\boldsymbol{g}^n + \boldsymbol{T}^{-T}\boldsymbol{s}^v_{pre} = \boldsymbol{C}^n\boldsymbol{g}^n + \boldsymbol{s}^n_{pre} \tag{34}$$

$$\boldsymbol{C}^n = \boldsymbol{T}^{-T}\boldsymbol{C}^v\boldsymbol{T}^{-1} \tag{35}$$

The discretized weak form of the internal work δW^{int} can be written in terms of natural variables.

$$\delta W^{int} = t\int_A \boldsymbol{s}_i^n \cdot \delta\boldsymbol{g}_i^n dA = \frac{tAl_i}{l_i^{r^2}} s_i^n \boldsymbol{\Psi}_i^T \boldsymbol{v}_i \cdot \delta\boldsymbol{x} = \boldsymbol{f}^{int} \cdot \delta\boldsymbol{x} \tag{36}$$

where $\boldsymbol{f}^{int}$ is the discretized internal force vector. The external work contribution δW^{ext} is given as

$$\delta W^{ext} = \boldsymbol{f}^{ext} \cdot \delta\boldsymbol{x} \tag{37}$$

where $\boldsymbol{f}^{ext}$ is the discretized external force vector given at each node as

$$\boldsymbol{f}_a^{ext} = \boldsymbol{f}^q + \boldsymbol{f}^p = \frac{A}{3}(q\boldsymbol{e}_q + p\boldsymbol{n}) \tag{38}$$

where $\boldsymbol{f}^q$ is the constant force vector and $\boldsymbol{f}^p$ the pressure force vector, q is the constant load magnitude, $\boldsymbol{e}_q$ the direction of the load, p is the pressure load magnitude, and $\boldsymbol{n} = \boldsymbol{e}_1 \times \boldsymbol{e}_2$ the normal vector of the CST element. The internal stiffness matrix of the natural membrane $\boldsymbol{K}^{int}$ is given as

$$\boldsymbol{K}^{int} = \boldsymbol{K}^e + \boldsymbol{K}^g \tag{39}$$

The elastic part of the stiffness contribution $\boldsymbol{K}^e$ is

$$\boldsymbol{K}^e = tA\frac{l_i l_j}{l_i^{r^2} l_j^{r^2}} C_{ij}^n \boldsymbol{\Psi}_i^T (\boldsymbol{v}_i \otimes \boldsymbol{v}_j)\boldsymbol{\Psi}_j \tag{40}$$

The geometrical part $\boldsymbol{K}^g$ is defined as

$$\boldsymbol{K}^g = \frac{tA}{l_i^{r^2}} s_i^n \boldsymbol{\Psi}^T \boldsymbol{\Psi} \tag{41}$$

The contribution to external stiffness $\boldsymbol{K}^{ext}$ is given only by the pressure forces. Defining $\boldsymbol{\Psi}^p = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{I} & \boldsymbol{I} \end{bmatrix}^T$,

$$\boldsymbol{K}^{ext} = \frac{p}{6} \boldsymbol{\Psi}^p \left(\boldsymbol{V}_1 \boldsymbol{\Psi}_2 - \boldsymbol{V}_2 \boldsymbol{\Psi}_1 \right) \tag{42}$$

where $\boldsymbol{V}_i = Skew(\boldsymbol{v}_i)$ is the skew-symmetric matrix of vector $\boldsymbol{v}_i$.

2.2.2. *Wrinkling.* The inability of tensile membrane materials to withstand compression makes them susceptible to easily buckle when there is a compressive load in their plane, an effect commonly called *wrinkle*. To accurately retrieve the wrinkling behavior of membrane elements, a dense mesh of shell elements with low flexural stiffness can be used [53, 56], but this leads to high computational cost due to the addition of rotations degrees of freedom, mesh quality, and post-buckling effects.

However, in most applications of structural analysis of tensile structures, the exact shape of the wrinkles does not matter to the engineer, but rather to the correct in-plane stress and strain distribution of the membrane. In this sense, the tension field theory [82, 76, 53] removes any compressive stress on the membrane element, ignores out-of-plane displacements handling only in-plane strains, and states that if a membrane region wrinkles, it is subjected to an uniaxial tension state. This leads to a new nonlinearity as the new membrane stiffness become also a function of the wrinkled uniaxial direction.

A way to add this behaviour to the membrane model is by a modified constitutive tensor $\hat{\boldsymbol{C}}^v$, as several authors proposed [2, 32, 36, 85]. It is defined that a membrane can be in 3 types of stress states : *taut*, *winkled*, or *slack*. The definition of the membrane state can be defined by a mixed criterion of the principal stresses s_1, s_2 and strains g_1, g_2, which are obtained as the eigenvalues of the stress and strain tensors. The definition of the membrane state can be defined by a mixed criterion of the principal stresses s_1, s_2 and strains g_1, g_2

- $s_2 > 0 \longrightarrow$ taut
- $s_2 \leqslant 0$ and $g_1 > 0 \longrightarrow$ wrinkled
- $g_1 \leqslant 0 \longrightarrow$ slack

When the membrane is taut, there is no need for modifications of the constitutive equations, and the constitutive tensor remains the same.

$$\hat{\boldsymbol{C}}^v = \boldsymbol{C}^v \tag{43}$$

If the membrane is slack, the membrane loses all compressive stiffness and all components of the constitutive tensor are penalized.

$$\hat{\boldsymbol{C}}^v = P \boldsymbol{C}^v \tag{44}$$

where P is the penalty factor that can be conveniently defined as

$$P = \begin{Bmatrix} 1 & ; & s_{max} \leqslant s_2 \\ s_{max}/s_2 & ; & s_{max} > s_2 \end{Bmatrix} \tag{45}$$

The penalty factor is used to avoid ill-conditioning of the stiffness matrix that would appear if the whole constitutive tensor is set to zero.

When the membrane is wrinkled, the first step is to find the wrinkling direction $\boldsymbol{w}(\theta) = \boldsymbol{Q}(\theta)\boldsymbol{e}_{\boldsymbol{2}}^{r}$ and uniaxial direction $\boldsymbol{t}(\theta) = \boldsymbol{Q}(\theta)\boldsymbol{e}_{\boldsymbol{1}}^{r}$, where $\boldsymbol{Q}(\theta)$ is the tensor representing the rotation of a vector in the plane of the element basis.

$$\boldsymbol{Q} = \begin{bmatrix} cos^2\theta & sin^2\theta & 2cos\theta sin\theta \\ sin^2\theta & cos^2\theta & -2cos\theta sin\theta \\ -cos\theta sin\theta & cos\theta sin\theta & cos^2\theta - sin^2\theta \end{bmatrix} = \begin{bmatrix} \boldsymbol{\phi}_1^T \\ \boldsymbol{\phi}_2^T \\ \boldsymbol{\phi}_3^T \end{bmatrix} \tag{46}$$

A method for evaluating the direction of wrinkles is given by [32] by enforcing the constraints of the wrinkle condition.

$$s_{22}^{w} = \boldsymbol{\phi_2}(\theta) \cdot \boldsymbol{s}^{v} = 0, \qquad s_{12}^{w} = \boldsymbol{\phi_3}(\theta) \cdot \boldsymbol{s}^{v} = 0 \tag{47}$$

The following condition leads, after some algebra [32],to the following nonlinear equation, which solution θ gives the wrinkling angle:

$$[\boldsymbol{\phi_2}(\theta) \cdot \boldsymbol{C}^{v}\boldsymbol{\phi_2}(\theta)][\boldsymbol{\phi_3}(\theta) \cdot \boldsymbol{s}^{v}] - [\boldsymbol{\phi_2}(\theta) \cdot \boldsymbol{C}^{v}\boldsymbol{\phi_3}(\theta)][\boldsymbol{\phi_2}(\theta) \cdot \boldsymbol{s}^{v}] = 0 \tag{48}$$

Equation 48 can be solved numerically with the Newton method. However, another constraint to solution is the existence of uniaxial stresses in order that $s_{11}^{w} > 0$. If a non-favorable solution is found during the Newton method procedure ($s_{11}^{w} \leqslant 0$), a phase shift of $\pm\frac{\pi}{2}$ can be performed to force convergence into a feasible domain.

Finally, the constitutive tensor in the wrinkled direction is defined by

$$\hat{\boldsymbol{C}}^{v} = \boldsymbol{C}^{v} - \frac{(\boldsymbol{C}^{v}\boldsymbol{\phi_2}) \otimes (\boldsymbol{\phi_2}\boldsymbol{C}^{v})}{\boldsymbol{\phi_2} \cdot \boldsymbol{C}^{v}\boldsymbol{\phi_2}} \tag{49}$$

For isotropic materials, the principal stress and strain directions are the same as the wrinkling and uniaxial direction ($\theta = \psi_s, \psi_g$), which gives

$$\boldsymbol{w}(\theta) = \boldsymbol{w}(\psi_s); \qquad \boldsymbol{t}(\theta) = \boldsymbol{t}(\psi_s) \tag{50}$$

In orthotropic materials, the principal strain and stress angles differ, and so does the wrinkling angle ($\psi_s \neq \psi_s \neq \theta$). Some authors [87, 81] approximated the wrinkling direction with the principal stress direction and showed a small deviation from the exact solution.

2.3. **Solution.** As stated in the previous section, we want to find the displacement field of the system in order that for all nodes i, the residual force is $\boldsymbol{r} = \boldsymbol{0}$. There are multiple methods to find the equilibrium configuration, and this choice depends mostly on the particular problem to solve. For tensile structures, there are two main solution procedures: the Newthon-Raphson method (NRM) and the dynamic relaxation method (DRM) [14]. As will be discussed further, the NRM is a classical solver for nonlinear problems used in almost all finite element applications. However, some particularities from tensile structures can lead to slow convergence or even divergence of the solution using the NRM. To cope with this problem, the DRM is a strong candidate, as it does not depend on the derivatives of the residual vector and uses a pseudo-dynamic approach, which leads to a more stable solution. In this work, both DRM and NRM are implemented, and they are options available for all the steps that require structural analysis, since the best method needs to be defined according to the specific problem to be solved. A mixed approach in which DRM is used with a higher error tolerance and then moves to NRM for a refined solution is also presented in a similar approach to [51].

2.3.1. *Newthon-Raphson method.* The NRM is a classical method to numerically solve nonlinear equations with multiple degrees of freedom. Generally, the problem starts by defining the global initial current position as $\boldsymbol{x}^{t=0} = \boldsymbol{\xi}$, and for each iteration we find an increment $\Delta\boldsymbol{x}_{\boldsymbol{i}}^{t}$ by solving the following linear equation system:

$$\boldsymbol{K}(\boldsymbol{x}^t)\Delta\boldsymbol{x}^t = \boldsymbol{r}(\boldsymbol{x}^t) \tag{51}$$

where $\boldsymbol{K}$ is the global stiffness matrix and $\boldsymbol{r}$ is the global residual vector. Here, the position in the next iteration is defined as

$$\boldsymbol{x}^{t+1} = \boldsymbol{x}^t + (\Delta\boldsymbol{x})^t \tag{52}$$

The iterations are then repeated until the norm of the residual vector $\boldsymbol{r}$ is below a given threshold $\epsilon \approx 0$.

$$||\boldsymbol{r}|| < \epsilon \tag{53}$$

The NRM is very versatile and solves a wide range of structural problems. However, in systems where the stiffness matrix is not well conditioned, it can easily lose convergence. In tensile structures, the stiffness matrix can be highly ill-conditioned because of mechanisms in the structure due to low prestress and wrinkling conditions, making this method limited to cope with these structural types and requiring stabilization methods in order to guarantee a solution, such as load steps and penalty factors to artificially control the stiffness of the system to avoid an ill-conditioned state [70, 32, 2, 60].

2.3.2. *Dynamic relaxation method.* The DRM was first proposed by [14] in the field of linear elastic analysis of pressure vaults. In order to avoid high consumption of memory required by the stiffness matrix, the DRM proposes the equilibrium condition as the stationary phase of an explicit dynamic system, with time integration usually made using the central difference method. As explicit time integration does not require the use of the stiffness matrix, it is possible to operate only with the vector global residual ($\boldsymbol{r}$) and displacement ($\boldsymbol{u}$) measures. The guarantee of the stability of the explicit motion is directly related to the stiffness matrix, but there are many methodologies to find a stable time step without the use of the global stiffness matrix.

The avoidance of the use of the stiffness matrix in DRM made it widely used to solve membrane displacement problems, especially when low or no prestress is determined, which can easily lead to singular stiffness matrices. First uses of the DRM to solve membrane problems are referred to in [4] and [38]. References [59, 61, 26] give a systematic understanding of the history and development of the dynamic relaxation method, with an eye on minimizing the computational cost of the method.

For each time step Δt, the new nodal velocity $\dot{\boldsymbol{u}}_{\boldsymbol{a}}^{t+\frac{\Delta t}{2}}$ and displacement $\boldsymbol{u}_{\boldsymbol{a}}^{t+\Delta t}$ can be determined using the central difference method:

$$\dot{\boldsymbol{u}}_{\boldsymbol{a}}^{t+\frac{\Delta t}{2}} = \left(\frac{m_a}{\Delta t} + \frac{c_a}{2}\right)^{-1}\left[\dot{\boldsymbol{u}}_{\boldsymbol{a}}^{t-\frac{\Delta t}{2}}\left(\frac{m_a}{\Delta t} + \frac{c_a}{2}\right) - \boldsymbol{r}_{\boldsymbol{a}}\right] \tag{54}$$

$$\boldsymbol{u}_{\boldsymbol{a}}^{t+\Delta t} = \boldsymbol{u}_{\boldsymbol{a}}^{t} + \dot{\boldsymbol{u}}_{\boldsymbol{a}}^{t+\frac{\Delta t}{2}}\Delta t \tag{55}$$

3. **Form-finding.** Generally, it is not possible to define the shape of a taut structure *a priori*, but an appropriate shape is sought, compatible with an initial stress field imposed to the system. This shape searching is commonly referred to as *form-finding*. There are multiple methods for the form-finding of tensile structure shapes, and all

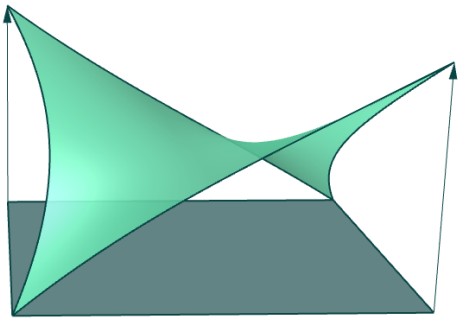

FIGURE 4. Form-finding process from an initial flat mesh

of them are based on the equilibrium of the structural system for a given imposed stress field and boundary conditions.

The force density method (FDM) [41] provides one of the most convenient alternatives for shape finding, defining a linearized solution for a network of linear elements. By its turn, the natural force density method (NFDM) [57] is an extension of the FDM that preserves the linearity of the original method and overcomes some of its limitations to cope with irregular meshes, which may arise from unstructured tessellations of surface geometries. Other common form-finding methods are the update reference stategy (URS) [8] and the DRM [1].

3.1. **Force density method.** One of the first alternatives of form-finding methods was the FDM, proposed by [41] and [73], in the context of cable nets. Given a line truss element, it is possible to define the force density n of the element as

$$n = N/l^r \tag{56}$$

where N is the normal force and l^r is the reference length of the truss element. Recovering the weak form of the 2 node truss element, and assuming that n is a constant value, we have

$$\boldsymbol{f}^{int} = n\boldsymbol{\Psi}^T\boldsymbol{\Psi}\boldsymbol{x} \tag{57}$$

Equilibrium conditions are held, such as

$$\boldsymbol{f}^{int} = \boldsymbol{f}^{ext} \tag{58}$$

Since the term $n\boldsymbol{\Psi}^T\boldsymbol{\Psi}$ is constant, it is actually possible to define the coordinates of the nodes as the solution of a linear system of equations.

$$\boldsymbol{K}^{fd}\boldsymbol{x} = \boldsymbol{f}^{ext} \tag{59}$$

$$\boldsymbol{K}^{fd} = n\boldsymbol{\Psi}^T\boldsymbol{\Psi} \tag{60}$$

where $\boldsymbol{K}^{fd}$ is the force density sitffness matrix. The authors [41] and [73] derived the equations 63 and 64 by finding the equilibrium of each node in a cable net. As discussed in [8] and [62], the force density method is equivalent to a linearization of the internal forces of the truss assuming that a given second-Piola Kirchhoff stress is constant, in which the tangent stiffness results exactly in the geometric part of the truss stiffness matrix.

3.2. **Natural force density method.** The NFDM preserves the linearity of the original FDM for the surface continua. The natural force density derives from the natural forces defined at the natural membrane finite element, and was first proposed by [57] with an extended discussion in [59].

The NFDM defines a similar approach to the FDM by defining a vector of natural force densities $\boldsymbol{n}$ defined by

$$n_i = \frac{tA}{l_i^{r^2}} s_i^n \tag{61}$$

We can rewrite the vector of internal forces for the natural membrane as

$$\boldsymbol{f}^{int} = n_i \boldsymbol{\Psi}_i^T \boldsymbol{\Psi}_i \boldsymbol{x} \tag{62}$$

Assuming $\boldsymbol{n}$ is constant and using the same approach as in equations 63 and 64, we get the linear system of equations for the node coordinates of the natural membrane:

$$\boldsymbol{K}^{nfd} \boldsymbol{x} = \boldsymbol{f}^{ext} \tag{63}$$

$$\boldsymbol{K}^{nfd} = n_i \boldsymbol{\Psi}_i^T \boldsymbol{\Psi}_i \tag{64}$$

where $\boldsymbol{K}^{nfd}$ is the natural force density stiffness matrix. As with the FDM, the natural force density stiffness matrix is also equivalent to the geometric portion of the natural membrane stiffness matrix with a constant prescribed Second-Piola Kirchhoff stress. The NFDM was also extended to 4node elements and anisotropic stress states by [60].

3.3. **Iterative force density method.** The linear solution for a shape in an equilibrium state will result in a nonuniform distribution of stresses along the shape. As proposed by [62], it is possible to impose a uniform stress field by an iteration method in which each iteration uses the last iteration as a reference geometry.

$$\boldsymbol{K}^i \boldsymbol{x}^{i+1} = \boldsymbol{f}^{ext} \tag{65}$$

where $\boldsymbol{K}^i$ is the force density stiffness matrix given a reference configuration $\boldsymbol{x}^{r^i} = \boldsymbol{x}^i$. As noted by [62], this iterative method leads to a stress state condition in which the Second-Piola Kirchhoff stress is equivalent to the Cauchy stress, resulting in a minimal surface.

Unlike solving the uniform stress problem by means of a nonlinear system with the help of the NRM, the iterative force density method does not show a quadratic convergence rate. However, the main characteristic of this method is that every iteration results in an equilibrium shape, even if the distribution of stresses will not be uniform like in the case of the converged minimal surface, which is a very handy feature to have in a form-finding framework that is not satisfied with the nonlinear approach.

4. **Patterning.** The final shape of the tensile membrane is rarely developable, and the only exception to the exact extraction of developable flattened tensile sheets is with single-curved shapes. For double-curved situations, it is not possible to flatten the entire surface in a single fabric sheet, and the shape has to be subdivided into strips, which are flattened later. However, these segmented stripes still present some degree of double curvature and some error in the flattening process is expected.

The fundamental importance of the patterning process is to determine an optimum point for maximum flattening performance and minimum surface subdivision. An excessive subdivision, although significantly reducing the flattening error, can make the assembly of the surface impracticable due to the excess of seam joints. Large

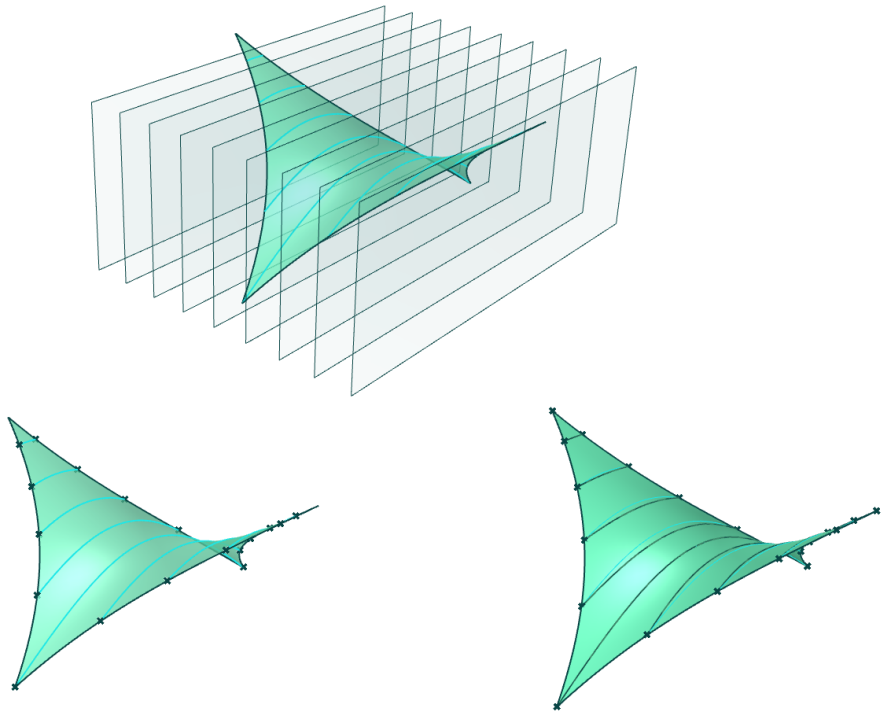

FIGURE 5. (a,b) Plane cut segmentation to find initial straight lines and (c) geodesic curve definition

subdivisions, on the other hand, dramatically increase the flattening error and are limited in width due to fabric sheet fabrication.

An industry standard method of patterning is to divide surfaces using geodesic lines [4, 84, 46, 26]. The geodesic line tends to deliver less distortions in the flattening process, as it results in straighter lines when flattened, which helps to avoid material waste during the fabric cutting.

When dealing with mesh geometries, a challenging aspect of this process is how to define the subdivisions of the mesh between the geodesic lines and subsequently divide the mesh into a set of mesh strips. Reference [16] proposes a method that defines the geodesic line in the edges of the mesh using a geodesic line finite element. The problem is set to find first the shortest path between two nodes of the mesh and sequentially distort the mesh so that the short-path edges are close to a geodesic line. However, this method has the disadvantage of distorting the mesh, which can compromise their quality, especially for coarse meshes.

[26] gives a geodesic line finding method by finding the equilibrium shape of a frictionless sliding cable in the membrane mesh projected to the mesh surface, which allows the geodesic line nodes to walk through the edges and faces of the mesh. [63] also gives a mesh-independent alternative to find geodesic lines by applying a minimum energy optimization problem for a minimum length line constrained to the mesh surface. Although the mesh is not distorted during those methods, the

patterning still requires the mesh to be divided over the geodesic lines, which also can lead to low quality distorted meshes (Figure 4).

An alternative to define the patterning is to find the geodesic lines directly in a CAD surface geometry such as NURBS and BRep (boundary representation), which can be obtained from IGA form-finding [6, 55] and can be obtained with the help of geometric algorithms normally presented in CAD software. Using this approach, the segmentation of the strips can be set as a collection of trimmed patches of the CAD surface [22].

We propose a mesh-based solution in which geodesic lines are found with the help of the Kangaroo Grasshopper plug-in [63], and instead of directly splitting the mesh with the resulted segmentation lines, a complete re-mesh of each strip is performed with the help of Grasshopper mesh management tools in order to guarantee good mesh quality for each strip, imposing that nodes of the edges of the strips match their neighbors, as seen in Figure 4.

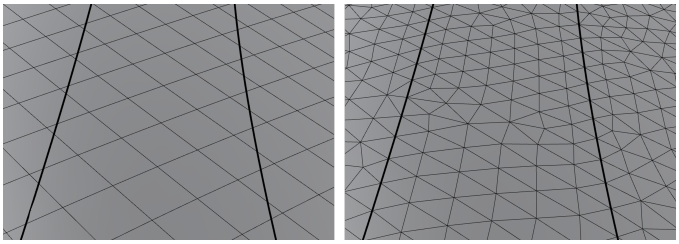

FIGURE 6. (a) Division of a mesh from arbitrary curves generates meshes with nonuniform elements. (b) Removing the strips with node constraints in the edges to enforce they are coupled sort out the issue.

However, the re-meshing occurs in a previous mesh in which a form-finding process was executed with a defined stress field. To be able to use the new mesh in the subsequent steps, a stress field mapping between the old and new meshes also needs to be performed. We propose an algorithm in which we reduce the stress tensor of an element in a set of the principal stress, second principal stress, and the direction of the first principal stress $\{s1, s2, \boldsymbol{t}\}$.

- For each reference mesh node, get stress contribution from each element weighted by its area.
- With the stress defined by means of the reference mesh nodes, find the stress in the target mesh nodes by means of interpolation based on the distance of the three closest points.
- Find the target element stress by an average of the nodal stresses of the element.

5. **Flattening.** With the correct definition of the surface patterning, each strip is then flattened in a two-dimensional plane for production. The major drawback of this process is that distortions between the flattened fabric surface and the double-curved strip will always appear in a certain rate due to the non-developability of the patterned surfaces. In addition, the flattening process of structural membranes is not only a geometric problem, but also an elasticity one. Since the idealized shape is assumed to be in a given prestress field, the flattened fabric needs to take into

account the compensation of the amount of stretch required to achieve the desired stress field and fabric geometry. These combined effects result in that the real stress field obtained by the assembly of the fabric will also contain deviations from the idealized stress field from the form-finding. Author [83] summarizes the main effects of this deviation as: (a) orientation of the fabric, (b) shear deformation of the fabric, (c) orthotropic behavior of the fabric, and (d) distortion of the flattening process from a spatial double-curved shape. Some materials, such as PVC fabric coats, tend to have low shear stiffness, and many manufacturers are pushing the material behavior to be as isotropic as possible, leading to small errors due to the flattening process, as the distortions may be accommodated due to the shear flexibility and isotropic behavior in the prestressing. However, shear-stretch and orthotropic materials, such as PTFE coated fabric and ethylene tetrafluoroethylene (ETFE) foils, are highly sensitive to these errors, and a careful flattening process is mandatory to deliver good performance of the fabric assembly.

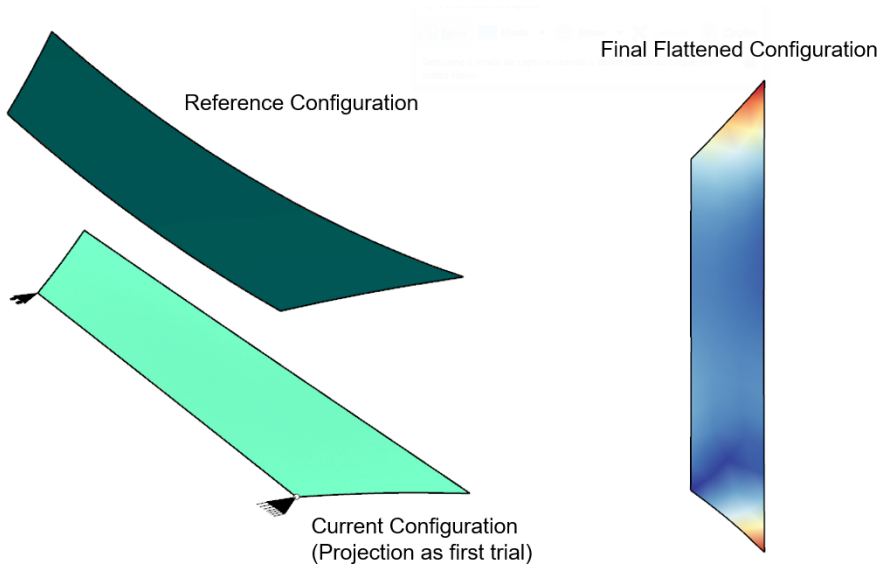

FIGURE 7. Flattening process of tensile surface strips. (a) Definition of the structural problem with the 3D geometry in the reference configuration and the projected geometry as a first trial for the 2D current configuration. (b) Resulted flattened strip with the corresponding displacement field from the projected initial geometry.

In this work, the flattening process used is performed by a structural analysis of each patterned strip constrained to a plane as in [46, 26]. The three-dimensional shape is defined as the reference configuration with its initial stress field, the two-dimensional flattened shape is defined as the current configuration, and a pinned and a roller support are added on opposite sides to provide sufficient boundary conditions. A first trial for the flattened shape can be obtained by projecting the mesh into the plane. This process can be interpreted as pressing the prestressed reference configuration on a flat surface without friction [16]. The strip will deform to accommodate the initial stress given, resulting in a flattened strip that takes into account the compensation of the stretch of the material for the defined initial stress.

Other approaches are given by [40, 16], which proposes a minimal energy optimization problem to minimize the strain energy of the assembly process by using a modified weak form of equilibrium and defining the spatial configuration with Cauchy stress and Euler-Almansi strain tensors. The 3D shape is defined as the current configuration, and the flattened shape as an unknown reference configuration. Although the derivatives of the Euler-Almansi strain are more complex to compute, this approach also enables optimized flattening for large strains. For small strains, the process is quite similar to the approaches of [46, 26], as all nonlinear strain tensors can be approximated to the infinitesimal strain tensor of the theory of linear elasticity.

6. **Parametric design workflow.** The design and analysis steps of the tensile structures present some computational challenges, especially in communication between the CAD and CAE systems. CAD generally has resources for the geometrical definition of complex shapes and details, where CAE is used for computational physical analysis. [27] addresses the main issues of CAD/CAE integration on loss of data, compatibility during the process, and lack of automation. These issues arise due to the different characteristics between processes. A CAD model is mainly a computational representation of the geometry, which does not necessarily have attributes and properties. These features are crucial inside a CAE environment, as it needs data such as material properties, physical interfaces, and element types.

Many CAE softwares offer CAD modeling features inside its interfaces, but are still very limited compared to CAD specialized software. Even when advanced geometry manipulation is enabled, many times it is bounded to the features of the CAE program.

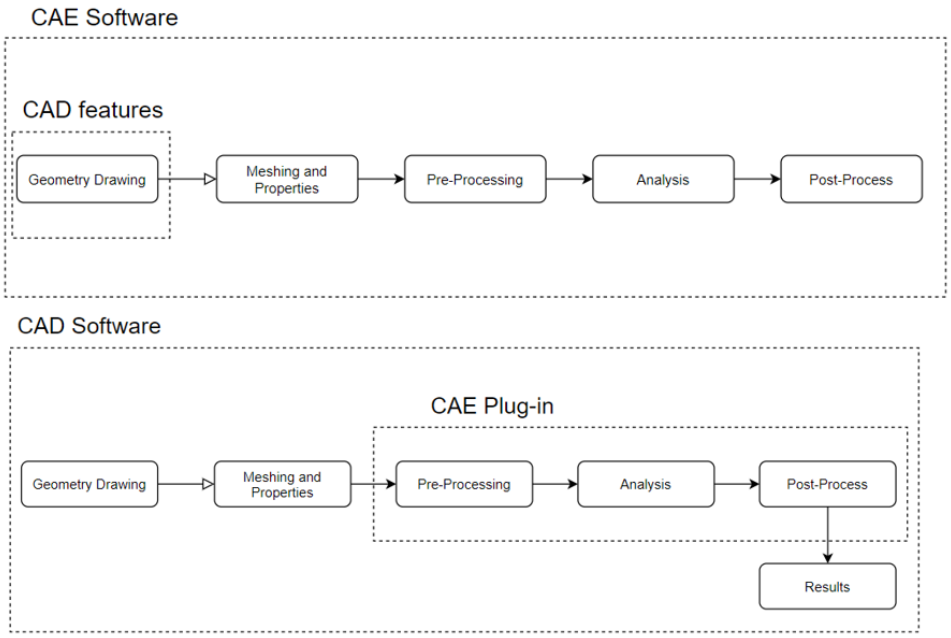

FIGURE 8. Possibilities in the integration of CAD and CAE interfaces

Some specialized CAD software such as Rhino3D [50] have been delivering scripting features with API giving the developer access to the geometrical objects of the CAD system inside an object-oriented programming language. With these features, it is possible to develop CAE structural applications with full use of CAD capabilities, or to communicate CAD and CAE systems by API integration. With this functionality, a constant mapping between the CAD geometry and the finite element model can be established, which avoids data loss during the design process, enables automation of tedious tasks such as meshing, patterning, and flattening, and drastically reduces manual data transfer between different softwares.

In the CAD scripting context, mainly two types of programming are featured: VPL and textual programming languages (TPL). A VPL consists of visual block elements, which contain algorithms and can be manipulated in a logical sequence of inputs and outputs. On the other hand, TPL systems are based on a sequence of linear characters that describes the commands the program should execute [37].

VPLs are advantageous, as they do not require a broad knowledge of programming syntax and are easier to use for general CAD users. The possibility of association of inputs and outputs in VPLs is interesting for CAE, as association between geometrical and analysis data can be made. However, VPLs are strictly dependent on TPLs, as each block component on the VPL is produced from a TPL code.

A big trend in computational design is the use of parametric (or algorithm-oriented) design in architecture and engineering with the use of VPLs. Parametric design environments usually offer the designer a user-friendly interface in which he visually assembles a sequence of components blocks that receive, process, and output data. The addition of CAE components to these systems is very promising in terms of task automation, design flexibility, and architecture/engineering integration. Rhino3D offers its VPL Grasshopper [71], which has become an industry standard for architectural design and has built a wide ecosystem of third-party plug-ins for a wide range of applications, from simulations to rendering and detailing. [49]. There are also VPL-focused platforms like Synera [79] that offer the definition of automated engineering processes in a “low code” environment for industrial engineers.

With the implementation of a structural analysis tool kit inside a VPL language such as Grasshopper, it is possible to integrate the architectural model into the structural analysis by assigning rules for the geometry, meshing, and definition of boundary conditions. The environment allows the user to implement routines for specific and interconnected tasks based on the geometry and properties of the structure, which can even be integrated with the architectural model if it is also defined parametrically. The conversion of data is automated by assembling a series of VPL algorithms, where any change in geometry returns a suitable new structural model, avoiding rework in mesh generation and other possible tedious tasks.

[66] describes parametric structural analysis as a powerful tool for geometry development because it feeds the architectural design space with structural feedback. In this perspective, these tools are known to be very effective in developing the geometry and materiality of complex structures. In addition, these are also interesting tools for the structural design itself, even when based on predefined geometries, considering the possibility of constructing automation’s for detailing and member sizing [64], or feed production plant with specific machining files. The benefits of adopting a parametric approach for the structural design can be summarized as follows.

- Integration with the architectural model, further processes like detailing and fabrication by common parameters, and rules shared between architects, engineers, manufacturers, and builders.
- Immediate feedback of the structural analysis within changes in the architectural and structural model.
- Flexibility in the automation of the workflow as the user has freedom to define the design rules using a user-friendly programming interface.
- Direct interface with other tools such as thermal, acoustics, wind, and circulation analysis, as well physic simulation, fabrication algorithms, and heuristic optimization by sharing common design parameters of the architectural and analysis models.

7. **Proposed framework and implementation.** Our proposed framework for tensile structures is based on a set of VPL components in a custom Grasshopper plug-in named BATS [74], which enables the automation of tensile structural design processes with the definition of a logic sequence of components that describes a set of defined parameters and design rules (Figure 9).

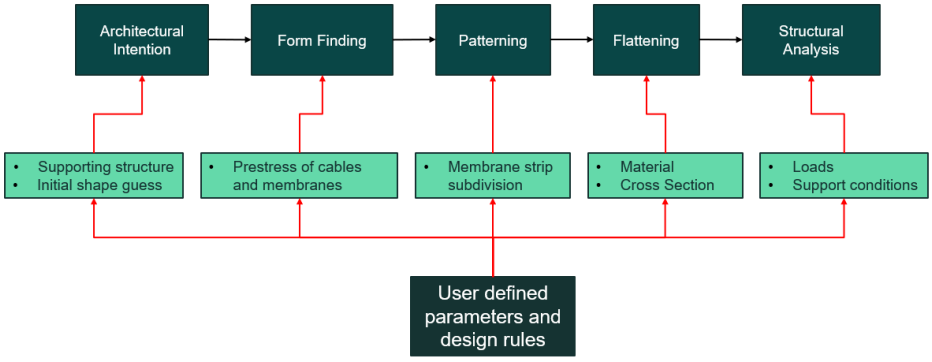

FIGURE 9. Workflow driven from parameter and design rules with integrated feedback between the design steps.

The main requirements for BATS:

- The data model is the same for all the design steps, and the transfer between them should be as automatic as possible.
- All relevant properties and objects should be accessible by the user as an output during any design step.
- Fast response in each design process to enable design feedback in almost real-time.
- Advanced users should be able to use BATS API to develop their own custom functionality, even outside Grasshopper.

To achieve those requirements, we propose the software architecture described in Figure 10. *BATScore* is the main data model that provides a set of constructors and modeling algorithms for the structural model. *BATS++* is the numerical library for finite element analysis calculations used by *BATScore*, written in C++, and designed for performance with the help of low-level memory management resources. *BATSgh* is the model with the definition of Grasshopper components that exposes the *BATScore* API to the VPL user.

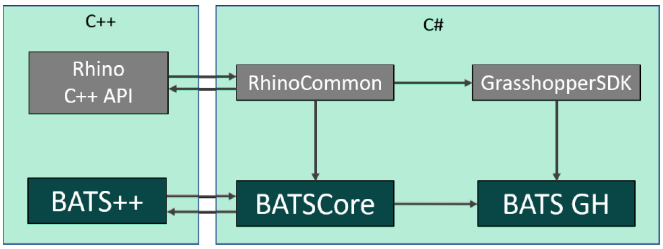

FIGURE 10. Software architecture of BATS

The choice of using different languages for model management and the FEA solution is to balance healthy development of the application and communication with external software such as Rhino and Grasshopper with a higher level programming language such as C#, but also taking advantage of C++ resources and libraries to improve memory management and performance of intense computations required for FEA.

7.1. **Data model.** Figure 11 shows the data model defined in *BATScore*. It provides a set of structural objects to be created and modified, as well as a structural model in which objects can be assigned and built in a set of FEA nodes and elements. The model can be constructed automatically by assigning a set of structural elements, supports, and loads, where the nodes are assigned by collecting the element nodes and using an RTree spatial data structure [44] to increase the performance of finding duplicate nodes.

Since the finite elements proposed in this work are defined by low-order polynomials, the representation of real structural members (beams, cables, and fabric) needs to be a set of individual finite elements that constitute the discretization of a given member. Having a model capable of mapping the individual elements to their original members is of great interest to enable enhanced preprocessing and post-processing of the model.

We propose in our data model element patches for that purpose, in which a real structural member can be defined in a patch that contains the discretized set of finite elements. For the specific case of fabric membranes, the patch also contains a set of strips that can be flattened and reassembled to the main patch.

7.1.1. *Element patches.* Element patches are data structures for cable, membrane, and beams that describe them in a set of discrete finite elements and can be observed in Figure 12. The results of the analysis can be accessed per patch for further design checks, and the re-mesh of patches is possible with the stress and strain field mapped from the old to the new mesh.

7.1.2. *Surface strips.* For surface patches, an additional dataset is the definition of the membrane strip geometries. Each strip contains its mesh geometry in both a flattened state and an assembled state. Starting with the desired assembly geometry, this data structure provides algorithms to perform the flattening step on each strip, thus providing the flattened state and then being pulled back to the assembly state for the assembly static analysis, as given in Figure 13.

Assembly automation is possible because each strip is also informed about the mapping from its own nodes and elements to the global patch mesh, and instead of

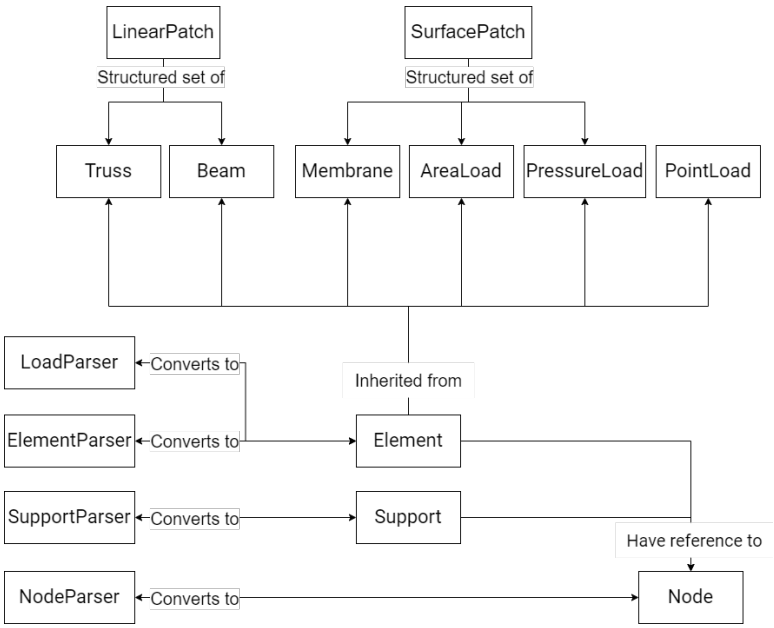

FIGURE 11. BATS data model

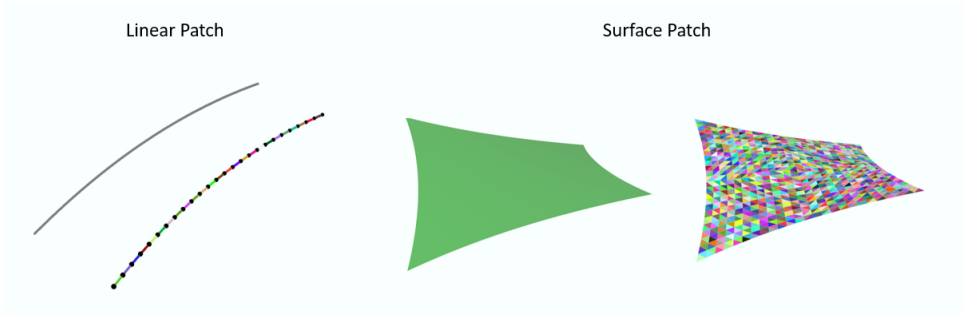

FIGURE 12. Linear and surface patch for cable and membrane members.

defining the ideal stress field from the form-finding for the analysis, the real prestress condition can be found by a static analysis where the flattened strips are defined as each element reference geometry and the form-found shape is set as the first solution guess.

7.2. **Numerical engine.** The numerical engine *BATS++* is a library written in C++ in which the elements and solvers are described in Sections 2, 3, and 5. In addition, a geometrically exact beam is implemented from the formulation described in [52], where a special routine for finite rotations is implemented for the dynamic relaxation method, details of which will be given in a forthcoming paper. The data model is consistent with the one defined in *BATScore* and contains the functionality to calculate the residual force vectors and stiffness matrices of the local elements and to further integrate the global residual force vector and the global stiffness matrix.

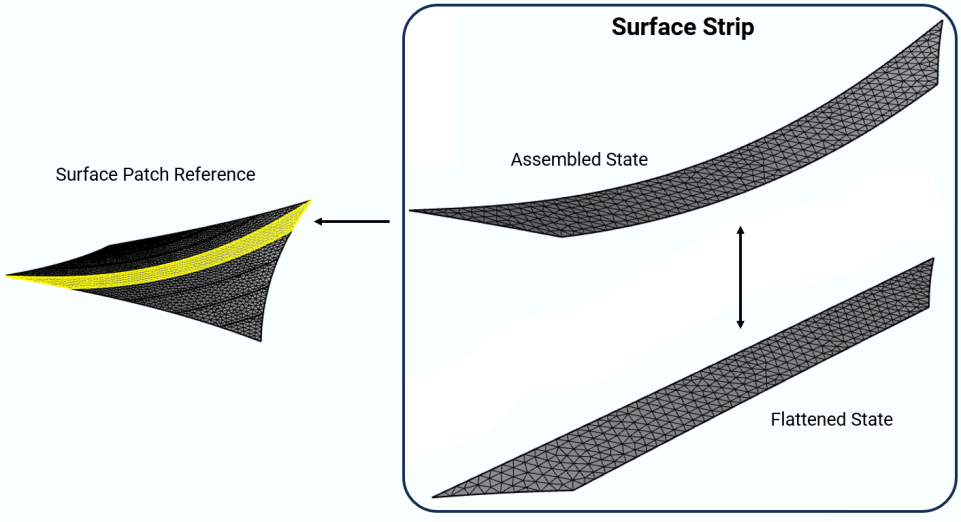

FIGURE 13. Definition of surface strips. A surface strip contains the assembled and flattened geometry and its topological reference to the main surface patch.

7.2.1. *Parsers.* The communication between *BATScore* and *BATS++* is made by external functions written in C++, which can be called from the C# code using platform invocation (P/Invoke). P/Invoke enables the declaration of a C# method that calls code from external dynamic linked libraries (dll). The declaration must match the corresponding C++ function, with special attention to the variable types used in the variables. The C# and C++ classes are different concepts, and it is not possible to directly parse an object from the C# class to a C++ function, which requires considerable effort to produce equivalent wrappers between classes. An alternative approach is to transfer structure objects since both can be defined as immutable types, with just some attribute declarations regarding array sizes in the C# counterpart.

BATScore have external function declarations for form-finding, nonlinear structural analysis, and flattening of membrane strips, with the conversions between the structural model and the structure parser objects. On the other hand, *BATS++* receives the structures as input and output data to construct and solve the model.

7.2.2. *Optimized code generation with symbolic programming.* One of the most time-consuming steps in the FEA process is to calculate the residual forces and the stiffness matrices of the elements. For the NRM, this process is not as relevant since there is a solution of the linear system of equations, which tends to be the bottleneck for this solver. On the other hand, the matrix-free nature of the DRM makes this step the critical path from a performance point of view, since the number of total iterations for solutions is higher.

In nonlinear analysis schemes, the calculation of the force and stiffness of the element is performed at each iteration step and is composed of a series of matrix and vector operations. A lot of computation overhead can occur due to those operations if the implementation is done naively, like numerically defining the inverse of a matrix where the inverse could be computed directly using the analytical expression.

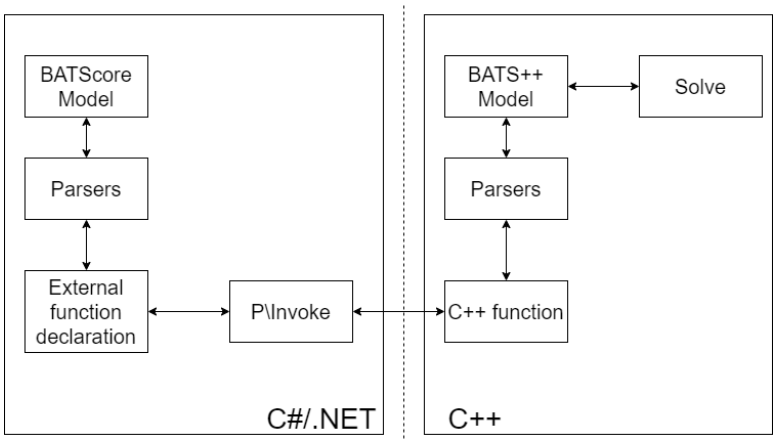

FIGURE 14. Flowchart of the communication between BATScore and BATS++

Multiplying multiple matrices often introduces overheads since common terms during multiplication can be repeated excessively. However, finding the exact analytical expression of residual forces and the stiffness matrix based on the elements variables is often impossible to do by hand.

The use of symbolic programming languages can be quite beneficial in these terms. A symbolic programming language enables the evaluation of mathematical expressions in their exact form, without introducing numbers. They can be used to find the exact form of the individual components of the force vectors and stiffness matrix in terms of the input parameters, thus reducing repetitive operations due to numerical matrix and vector operations. Languages such as Mathematica offer an extensive library for symbolic math with automatic differentiation and expression simplification functionality, which can be used in third-party libraries like AceGen [35] to automatically write optimized C++ functions. AceGen defines a hierarchy of symbolic variables in a graph structure in which common products are found for subsequent steps and drastically reduces the number of required numerical operations compared to traditional numerical evaluation processes.

All element computations in our work were first written in AceGen and the generated code was inserted into the *BATS++* code. A simple benchmark was performed for the evaluation of a relevant and complex operator for the analysis of the geometrically exact beam [52] by comparing the pure numerical calculation of the function with native C++ arrays and third-party libraries Eigen [25] and Armadillo [72], as well as the code generated by AceGen, both in single thread and parallel threads, and considerable performance was obtained by running the AceGen code. Another use of automatic differentiation for structural design frameworks is given by [55].

7.2.3. *Solver.* Both NRM and DRM solvers were implemented in BATS++ and can be called from the same data model. The code uses multi-thread parallelism to accelerate the computations when possible. While the calculation of each element local residual force and stiffness matrix can be parallelized, the assembly of the global variables in parallel can find race-condition problems where different threads might try to access the same memory location at the same time.

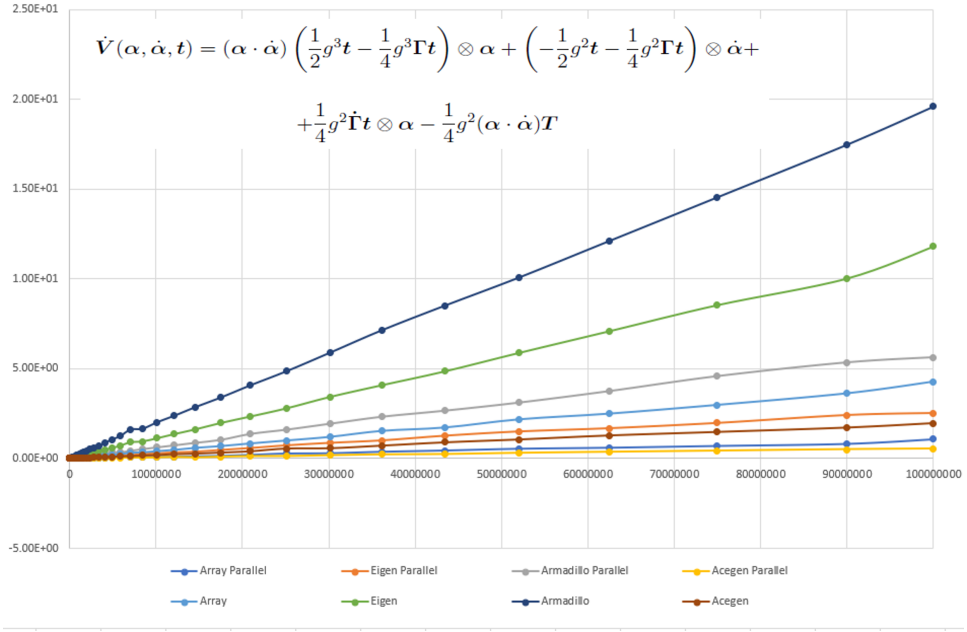

FIGURE 15. Benchmark of function implemented using third-party linear algebra libraries, native C++ arrays, and AceGen generated code.

For the NRM, we compute each individual element local variables in parallel, but run both the global vector/matrix global assembly and the linear system solver sequentially. In order to solve the linear equations, the C++ Eigen library was used, which provides fast solvers with some degree of parallelism under the hood.

The DRM method is more interesting in terms of parallel computing. Since we do not need to handle with global matrices, the only processes in which race condition can occur are during the global residual vector assembly and kinetic energy evaluation, which are the only processes done sequentially, with element evaluation and time-step updates done in parallel.

7.3. **Grasshopper components.** To allow the use of BATS in Grasshopper, a series of components were developed for the creation, editing, process, and analysis of tensile structural models. An overview of the automated design workflow it generates can be observed in Figure 17.

- Assemble Model: From geometric inputs from Rhino and Grasshopper and relevant structural data, it is possible to define BATS objects such as element patches, supports, and loads, and assign them to an FEA model.
- Form-Finding: A component performs the form-finding process using the NFDM (see Section 3) from the boundary conditions of the assembled model and using the input mesh as a reference mesh for the shape-finding algorithm.
- Patterning/Remap: The patterning algorithm in BATS is the only one that is not directly implemented as a component. The main idea is that the patterning step can be set as a mechanism in which the user can recover the meshes from the form-found model and perform its own patterning using Grasshopper

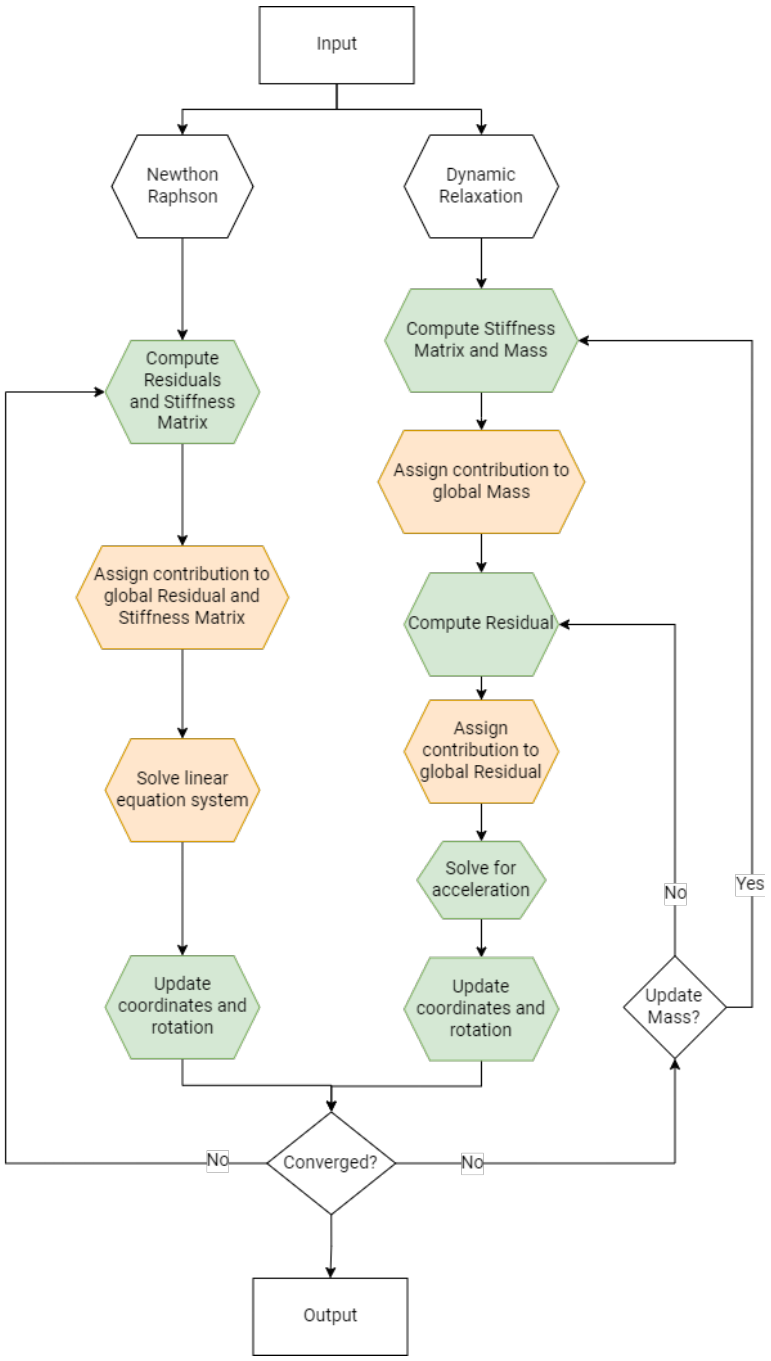

FIGURE 16. BATS Solvers flowchart. Green represents actions using parallelism and yellow actions computed in a single thread.

algorithms, to then feedback the re-meshed strips inside the surface patches, where the remap of the stresses and strain fields are computed using the method in Section 4. A standard patterning process defined by a segmented division

via geodesic lines is implemented as a Grasshopper cluster, in which we use the native Grasshopper physic engine Kangaroo [63] to perform the geodesic line computation, and a special re-meshing algorithm is set to generate mesh strips that are compatible with good mesh quality. Users can then use the cluster as a base to implement more advanced patterning techniques.

- Flattening: With the model updated with the segmented surface patches in strips, a component is available for performing the flattening process on all strips. The user can see the results of the flattening to obtain the flattened shape for production and can also analyze the residual stresses left after the flattening process.
- Assembly Analysis: Once a model contains the surface patches with their corresponding strips, it is possible to use the assembly analysis component to get the actual resultant stress field of the real assembly, which is different from the idealized form-found shape due to the inherent errors generated from the flattening process. This step is important to ensure that no wrinkles will occur during the construction stage and to have a more accurate initial stress field for subsequent analysis.
- Multiphase Load Response Analysis: Once the assembled geometry and stress field are determined, it is possible to add new loads to the model to perform a load response analysis due to wind, snow, and pressure. In BATS Grasshopper, each model that is given as an output of a process can be used as an initial condition for a subsequent analysis, enabling the user to set a chain of load actions and track the response of the load combinations.

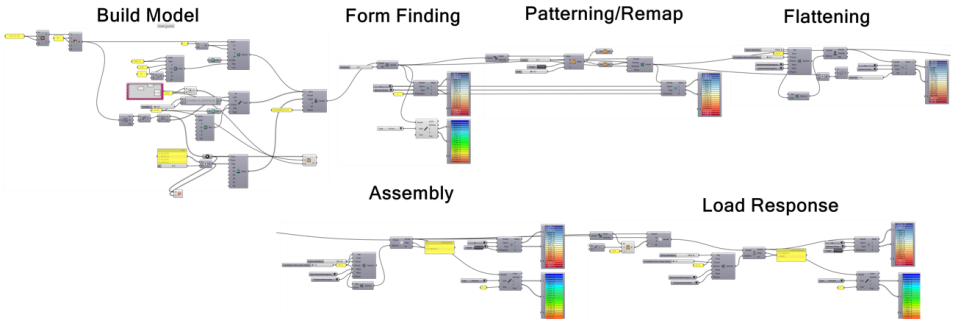

FIGURE 17. Parametric workflow for design and analysis of tensile structure implemented using BATS Grasshopper plug-in.

In addition to building this classic workflow automation, the BATS components are flexible and each step can be called totally independent of the others. If the user already has a shape with an initial stress field, it can process the patterning and flattening algorithm directly. In addition, the multiphase load response solver can be called at any time for any geometrically exact elasticity problem that combines truss, membrane, and beam elements. In the following section, we present some study cases that gives a clearer understanding of the benefits and limitations of the proposed workflow.

8. **Applications and discussion.** We present some study cases for the design of tensile structures using the proposed tool to show the capabilities of the workflow in

terms of design exploration and automation. We then highlight some discussions concerning the performance, flexibility, and limitations of the framework.

8.1. **Applications.** The design of a hyperbolic canopy is proposed as follows. From an initial 20x20 meters flat surface with cables on its edges, a form-finding procedure is established by imposing a vertical displacement on diagonal points and the intended stress field for the fabric and the cables. Figure 18 shows the initial conditions and alternative shapes of different displacement magnitudes and the cable-to-fabric stress ratio.

The form-finding component provides real-time feedback for the resultant shape due to the high performance of the implemented form-finding procedure [15]. This enables the use of parameters to quickly navigate through design solutions and find the best shape for a given context, and heuristic optimization algorithms can be applied over the parameters by use of Galapagos, a genetic algorithm presented in the ecosystem of Grasshopper. After the form-finding, we can retrieve the mesh

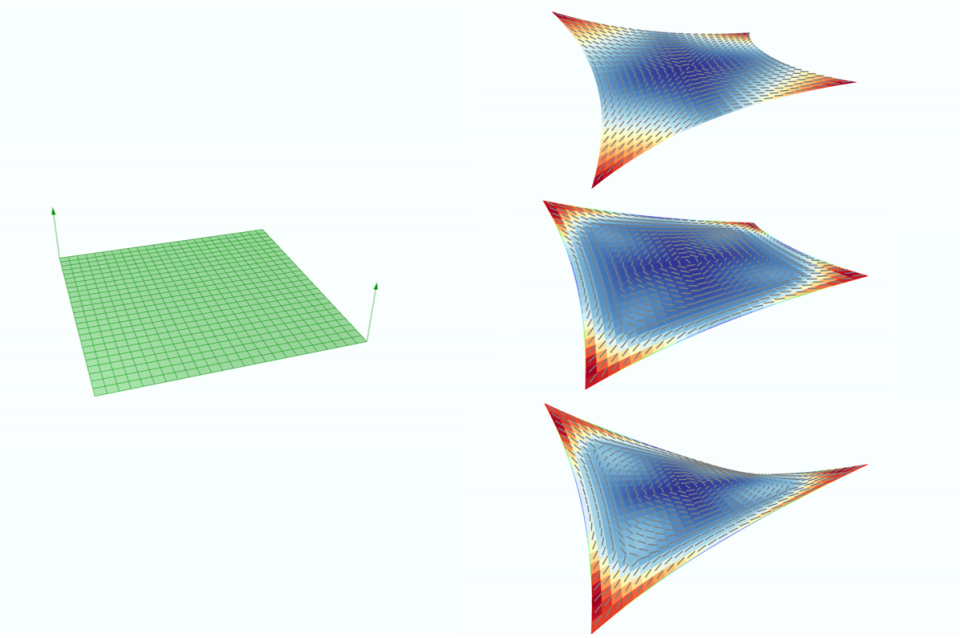

FIGURE 18. Model definition and solutions from the form-finding component for different vertical displacement of the supports and different ratio between the membrane and cable stresses.

geometry of the fabric shapes and process the patterning step. We implemented a Grasshopper script that can take a 4 side mesh surface, find an equally spaced array of geodesic lines around the surface, and create the strip meshes by reconstructing the meshes split by the geodesic lines. This algorithm can be modified to create other patterning logic as well as custom algorithms for arbitrary patterning. This is possible since the requirement for the patterning is simply to take the resultant geometry, create a logic to process in strips, and feedback the strips to the model. Once fed back to the model, the strips are set as surface strips of the surface patch, and the stress-field remap algorithm finds the equivalent stresses on the elements

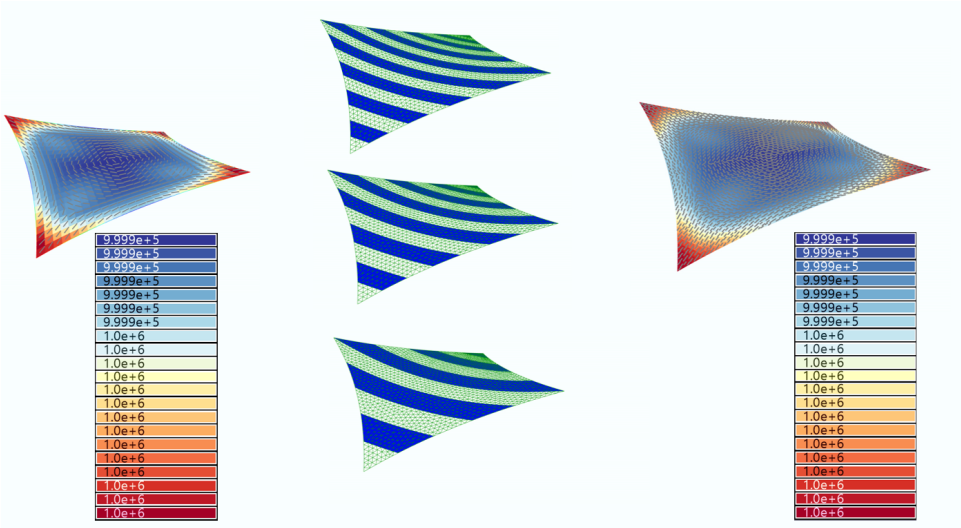

FIGURE 19. Patterning and re-meshing of the defined shape into different number of strips and remapping of the stress field to the new mesh. First principal stress field displayed in Pa.

of the new segmented meshes. Figure 19 shows the patterning process for different numbers of strips generated from the algorithm and the stress field of the new mesh, which is ready to be processed by the flattening step.

Figure 20 shows the flattening and assembly processes. The flattening component takes the model updated with strips on its surface patches and applies the flattening algorithm to each strip, and the resulted flattened strips are stored in the surface strip. With this new model, it is possible to export the flattened strips meshes for production and send them to the assembly component, which solves the displacement of the flattened strips to their assembled position, resulting in the real prestress applied to the surface. This method enables the validation of the flattening algorithm and can also predict whether any wrinkles can occur to the fabric when assembled. The assembled mesh model can be further edited to add and combine new loads to the meshes to perform geometrically nonlinear static analysis. Figure 21 shows the first principal stress, second principal stress, and displacement fields of a membrane under a load pressure of 800Pa inward in half of the surface section and upward in the other section. It is possible to see the wrinkled region in the second principal stress plot, where the stresses are zero (blue). In this scenario, a membrane model without wrinkling could result in a large error on the support reactions, which is why before the adoption of a wrinkling model, the condition for safety was to never let the membrane wrinkle. With the wrinkling model, it is possible to reduce the prestress on the membrane by letting the membrane wrinkle under extreme loading conditions, since they are temporary and aesthetics is not a priority as is in the assembled state.

Figure 22 shows the same workflow for the design of the surface of a conoid. The conoid was designed to explode possibilities of having an inclined axis for the top ring as well as to define a torsional patterning of the strips. This example is one of

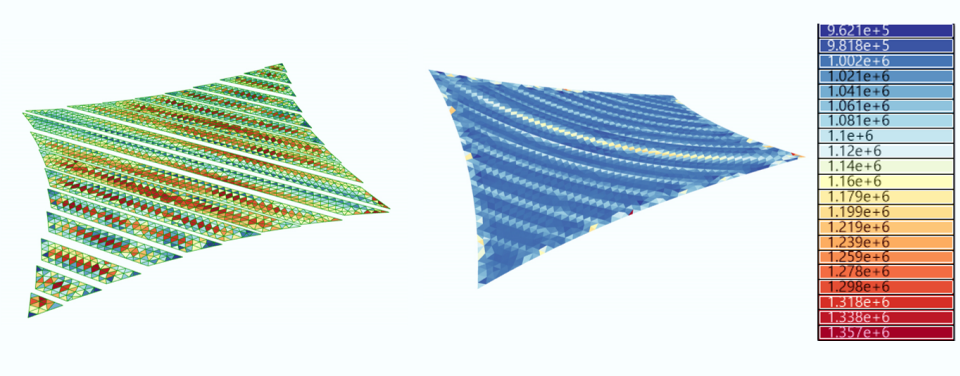

FIGURE 20. Flattening process generates the production strips of the fabric, which can be used for the further assembly analysis to find the real stress field resultant from the assembly.First principal stress field of the assembled fabric is displayed in Pa.

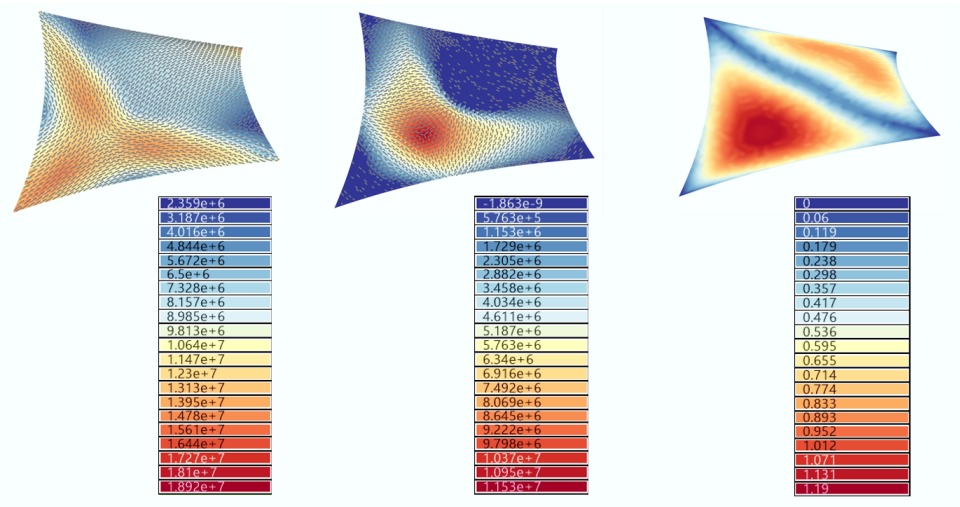

FIGURE 21. Load response analysis of a wind pressure of 800Pa. From left to right: first principal stress (Pa), second principal stress (Pa), and displacement fields (m).

particular difficulties for patterning and flattening, since the best-fit plane having an initial guess of the flattened geometry can be hard to achieve.

Figure 23 shows an example of a tensile structure formed by multiple surface patches. Those typologies cannot have their strips directly defined from a single mesh, requiring the definition of patches that must be segmented individually and match their nodes on their edges for proper FEA analysis of the assembly and load response analysis.

Both examples show the convenience of letting the designer choose his own patterning strategies directly from Grasshopper. Those strategies can be automated for the design in question, and the rest of BATS workflow will just ask for the final

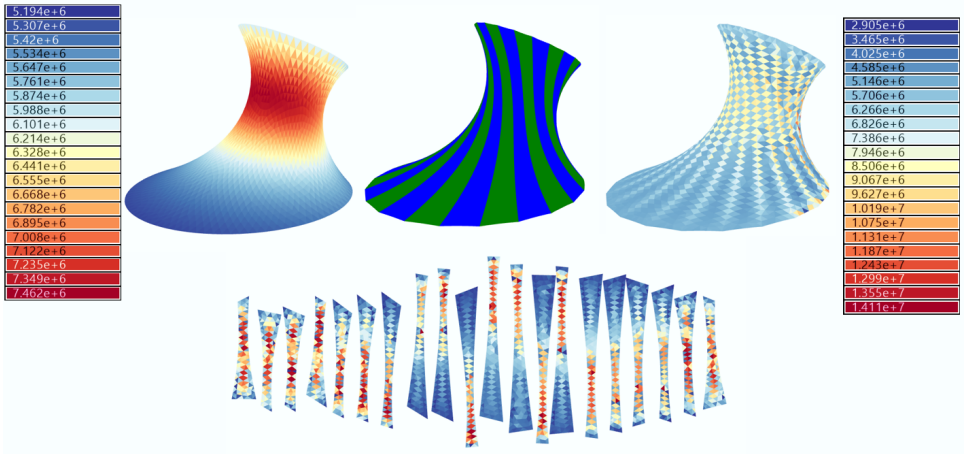

FIGURE 22. Example of conoid with torsioned strips. At the top from left to right: a) Geometry and first principal stress obtained from form-finding (Pa), b) Patterned geometry, c) First principal stress from real assembly (Pa). At the bottom: Flattened strips with residual stress with legend omitted for simplicity.

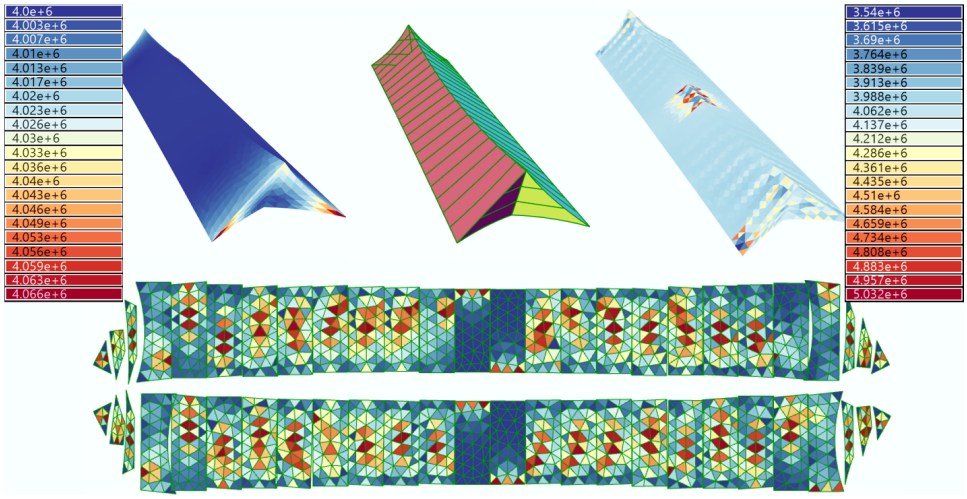

FIGURE 23. Example of multi-patch membrane. At the top from left to right: a) Geometry and first principal stress obtained from form-finding (Pa), b) Patterned geometry, c) First principal stress from real assembly (Pa). At the bottom: Flattened strips with residual stress with legend omitted for simplicity.

patterning solution to continue the design workflow. In Figure 22, it was possible to define the points for the generation of torsioned geodesic lines, as well as customize the planes in which the first flattening guesses will be produced from the projection of the surface strip. In Figure 23, it was possible to customize the patterning strategy to match both the individual patterning of each surface patch and to make each

mesh patch share the same nodes with others at their edges, allowing the use of FEA algorithms for assembly and load response.

8.2. **Discussion.** The implementation of this design automation gives feedback throughout the design cycle of tensile structures, and any change in a design step will provide immediate feedback for the next steps. This approach makes the need to locally iterate between steps in a manual fashion unnecessary, since the output and inputs on each step are strongly defined, allowing the user to just worry about the input data for each step, and the transitions between them are handled by the software and can save considerable time during design modeling for both design exploration and optimization. The algorithmic nature of tensile structural design makes parametric design an excellent tool for handling those systems, since the algorithms can be set in components that communicate between others, enabling also the use of external components of diverse fields to complete the design, such as CFD and daylight simulation tools.

It is important that these systems give flexibility in the workflow to avoid the biased design imposed by the tool. This requirement shows the importance of having separate modules for the plug-in *BATSgh* and the data model *BATScore*, since the plug-in can receive and send structured data as the design model, and the Grasshopper components can modify this data independently, allowing users to implement their own routines for any of the design steps if the *BATScore* API is used correctly. This modular behavior can enable the implementation of a diverse range of new form-finding, patterning, flattening, and structural analysis algorithms inside the environment, and even components that establish local optimization routines between the steps. In terms of patterning, enabling mesh division to be set directly in Grasshopper opens up a wide range of custom patterning algorithms without the need for deep programming experience and cannot be constrained to only geodesic lines depending on the context. If the designer wants to have colored strips that produce some kind of drawing in the mesh, creating a proper script to segment the shape in the desired strips for this drawing is sufficient, since the rest of functionality is automatically handled by BATS.

However, there are still some limitations to the proposed tool, and they can be topics for future research. Although the form-finding process is fast and provides real-time feedback, the same does not apply for the subsequent processes. The patterning process requires a series of algorithms to find geodesic lines, split, and reconstruct the original mesh, which can be consuming. Also, the flattening and structural analysis steps both require the use of geometrically nonlinear analysis, which is by nature much slower than linear algorithms such as those used in the form-finding. Even with the large set of optimizations proposed for the solver, there is still a gap to fulfill real-time feedback for all the steps, and the current implementation might take some seconds to complete the full computation of the entire workflow. Having the ability to process all those steps in seconds is already a good achievement, but real-time feedback would be ideal to unlock the full potential of parametric systems. Graphics processing unit (GPU) based solvers are promising, especially for DRM since they have an algorithm well suited for parallelization [45, 29, 34]. Deep learning models that predict results are also promising candidates for this task [39].

Another issue concerns the limitation of parametric workflows with respect to design edge cases. Since all designs are defined on the basis of VPLs, adding edge cases requires defining algorithms to detect them and treat them separately. As the

number of edges cases increases, the VPL code starts to grow and becomes hard to manage. To overcome this issue, an alternative is to combine the data model to be accessible both by the CAD and its VPL, in this case enabling BATS objects to be modeled directly in Rhino and have bidirectional communication with the Grasshopper components. This requires a modeling framework that can automate the workflow using both traditional and parametric modeling, which requires a totally new approach to implement such technology. An example of such systems is Branch [78], an internal software product of Structurecraft, a timber engineering company that proposes automated design and detailing of timber structural design using this approach.

9. **Conclusion.** In this work, we show the implementation of BATS, a new computational framework for the parametric design of tensile structures in Grasshopper that enables the automation of the design workflow of tensile structures that enhance the design exploration and optimization of those systems. We showed that parametric design is very suitable for the design of tensile structures to handle complex shapes, automate the design workflow, and set optimization routines. With our tool, the user can define its own design automation for the full tensile structure cycle that includes form-finding, patterning, flattening, and structural analysis steps. The presented tool is available for public testing [74].

We highlighted the benefits of our parametric tensile structural design workflow and also some of the limitations, especially with respect to the performance and manual modeling of edge cases, which are good candidates for further research. Although not extensively mentioned in this work, the tool also includes a geometrically exact beam element, which is not yet in maturity to be inserted into the present workflow and is going to be the topic of a forthcoming paper.

Acknowledgments. We would like to thank you for **following the instructions above** very closely. It will save us lot of time and expedite the process of your article's publication.

REFERENCES

[1] S. Adriaenssens, P. Block, D. Veenendaal and C. Williams, *Shell Structures for Architecture: Form Finding and Optimization*, Routledge/ Taylor & Francis Group, 2014.

[2] T. Akita, K. Nakashino, M. C. Natori and K. C. Park, A simple computer implementation of membrane wrinkle behaviour via a projection technique, *International Journal for Numerical Methods in Engineering*, **71** (2007), 1231-1259.

[3] J. H. Argyris, P. C. Dunne, T. Angelopoulos and B. Bichat, Large natural strains and some special difficulties due to non-linearity and incompressibility in finite elements, *Computer Methods in Applied Mechanics and Engineering*, **4** (1974), 219-278.

[4] M. R. Barnes, Form-finding and analysis of prestressed nets and membranes, *Computers and Structures*, **30** (1988), 685-695.

[5] K. Bathe, *Finite Element Procedures*, Prentice Hall, Watertown, 2006.

[6] A. M. Bauer, *CAD-integrated Isogeometric Analysis and Design of Lightweight Structures*, PhD thesis, Technische Universität München, 2020.

[7] K.-U. Bletzinger, J. Linhard and R. Wüchner, Extended and integrated numerical form finding and patterning of membrane structures, **50** (2009), 35-49.

[8] K.-U. Bletzinger and E. Ramm, A general finite element approach to the form finding of tensile structures by the updated reference strategy, *International Journal of Space Structures*, **14** (1999), 131-145.

[9] J. Bonet and R. D. Wood, *Nonlinear Contiuum Mechanics for Finite Element Analysis*, Cambridge University Press, Cambridge, 2008.

[10] J. Bonet, R. D. Wood, J. Mahaney and P. Heywood, Finite element analysis of air supported membrane structures, *Computer Methods in Applied Mechanics and Engineering*, **190** (2000), 579-595.
[11] M. A. Crisfield, *Non-linear Finite Element Analysis of Solids and Structures*, vol. 1, John Wiley & Sons, Chichester, 1991.
[12] R. G. Cruz, C. M. Arcipreste, R. L. Pinheiro and R. A. Ribas Generative design in the design development of metallic constructions, *SIGRADI*, Sao Paulo, (2018), 211-218.
[13] R. G. Cruz, C. M. Arcipreste, R. L. Pinheiro and R. A. Ribas Generative design: Information flow between genetic algorithm and parametric design in a steel structure construction, *Ambient. Constr.*, **21** (2021), 271-289.
[14] A. Day, An introduction to dynamic relaxation, *The Engineer*, **219** (1965), 218-221.
[15] M. S. V. de Souza and R. M. O. Pauletti, An overview of the natural force density method and its implementation on an efficient parametric computational framework, *Curved and Layered Structures*, **8** (2021), 47-60.
[16] F. H. Dieringer, *Numerical Methods for the Design and Analysis of Tensile Structures*, PhD thesis, Technische Universität München, 2014.
[17] B. Forster and M. Mollaert, *The European Design Guide for Tensile Surface Structures*, Tensinet, 2004.
[18] A. J. Gil and J. Bonet, Finite element analysis of prestressed structural membranes, *Finite Elements in Analysis and Design*, **42** (2006), 683-697.
[19] D. S. GmbH, Rfem 6, https://www.dlubal.com/en/products/rfem-fea-software/what-is-rfem.
[20] T. Gmbh, Easy, https://www.technet-gmbh.com/en/products/easy/.
[21] A. Goldbach and K.-U. Bletzinger, Cad-integrated parametric design cycle for structural membranes, *Journal of the International Association for Shell and Spatial Structures*, **60** (2019), 266-272.
[22] A.-K. Goldbach, *The CAD-Integrated Design Cycle for Structural Membranes*, PhD thesis, Technische Universität München, 2021.
[23] B. R. Group, Rhinomembrane, https://www.food4rhino.com/en/app/rv2.
[24] F. Gruttmann and R. L. Taylor, Theory and finite element formulation of rubberlike membrane shells using principal stretches, *International Journal for Numerical Methods in Engineering*, **35** (1992), 1111-1126.
[25] G. Guennebaud, B. Jacob et al., Eigen v3, http://eigen.tuxfamily.org, 2010.
[26] D. Guirardi, *O Método da Relaxação Dinâmica Aplicado À Análise de Estruturas de Cabos e Membranas*, Ph.d, University of São Paulo, 2011.
[27] G. P. Gujarathi and Y.-S. Ma, Parametric CAD/CAE integration using a common data model, *Journal of Manufacturing Systems*, **30** (2011), 118-132.
[28] K. Hincz and M. Gamboa-Marrufo, Deformed shape wind analysis of tensile membrane structures, *Journal of Structural Engineering*, **142** (2016), 04015153.
[29] P. Iványi, CUDA accelerated implementation of parallel dynamic relaxation, *Advances in Engineering Software*, **125** (2018), 200-208.
[30] ix Ray, Rhinomembrane, https://www.food4rhino.com/en/app/rhinomembrane-v-30-rhino-60.
[31] ixRay Ltd., ixcube, http://www.ixcube.com/ixcube/.
[32] A. Jarasjarungkiat, R. Wüchner and K.-U. Bletzinger, A wrinkling model based on material modification for isotropic and orthotropic membranes, *Computer Methods in Applied Mechanics and Engineering*, **197** (2008), 773-788.
[33] J.-Y. Kim and J.-B. Lee, A new technique for optimum cutting pattern generation of membrane structures, *Engineering Structures*, **24** (2002), 745-756.
[34] U. Kiran, S. S. Gautam and D. Sharma, Gpu-based matrix-free finite element solver exploiting symmetry of elemental matrices, *Computing*, **102** (2020), 1941-1965.
[35] J. Korelc, Multi-language and multi-environment generation of nonlinear finite element codes, *Engineering with Computers*, **18** (2002), 312-327.
[36] E.-S. Lee and S.-K. Youn, Finite element analysis of wrinkling membrane structures with large deformations, *Finite Elements in Analysis and Design*, **42** (2006), 780-791.
[37] A. Leitão, L. Santos and J. Lopes, Programming languages for generative design: A comparative study, *International Journal of Architectural Computing*, **10** (2012), 139-162.
[38] W. Lewis, M. Jones and K. Rushton, Dynamic relaxation analysis of the non-linear static response of pretensioned cable roofs, *Computers & Structures*, **18** (1984), 989-997.

[39] L. Liang, M. Liu, C. Martin and W. Sun, A deep learning approach to estimate stress distribution: A fast and accurate surrogate of finite-element analysis, *Journal of The Royal Society Interface*, **15** (2018), 20170844.
[40] J. Linhard, R. Wüchner and K.-U. Bletzinger, *Introducing Cutting Patterns in Form Finding and Structural Analysis*, Springer Netherlands, Dordrecht, 2008, 69-84.
[41] K. Linkwitz and H.-J. Schek, Einige Bemerkungen zur Berechnung von vorgespannten Seilnetzkonstruktionen, *Ingenieur-Archiv*, **40** (1971), 145-158.
[42] T. Linthout, A. c. Rezaei and W. p. v. Van Paepegem, *Form-Finding and Patterning of Fabric Structures Using Shape Optimization Techniques*, PhD thesis, 2016.
[43] R. T. Ltd., Wintess, `https://www.wintess.com/`.
[44] Y. Manolopoulos, A. Nanopoulos, A. Papadopoulos and Y. Theodoridis, *R-Trees: Theory and Applications*, Advanced Information and Knowledge Processing, Springer, 2006.
[45] J. Martínez-Frutos, P. J. Martínez-Castejón and D. Herrero-Pérez, Fine-grained gpu implementation of assembly-free iterative solver for finite element problems, *Computers & Structures*, **157** (2015), 9-18.
[46] B. Maurin and R. Motro, Fabric membranes cutting pattern, *Springer Netherlands*, Dordrecht, 2005, 195-212.
[47] J. McCartney, B. K. Hinds and K. W. Chong, Pattern flattening for orthotropic materials, *Computer-Aided Design*, **37** (2005), 631-644, CAD Methods in Garment Design.
[48] J. McCartney, B. K. Hinds and B. L. Seow, The flattening of triangulated surfaces incorporating darts and gussets, *Computer-Aided Design*, **31** (1999), 249-260.
[49] McNeel, Food for rhino, `https://www.food4rhino.com/`.
[50] McNell, Rhino3D, `https://www.rhino3d.com/`.
[51] K. Nakashino, A. Nordmark and A. Eriksson, Geometrically nonlinear isogeometric analysis of a partly wrinkled membrane structure, *Computers & Structures*, **239** (2020), 106302.
[52] A. G. Neto, C. A. Martins and P. M. Pimenta, Static analysis of offshore risers with a geometrically-exact 3D beam model subjected to unilateral contact, *Computational Mechanics*, **53** (2014), 125-145.
[53] A. Niewiarowski, S. Adriaenssens and R. Pauletti, Illustrating membrane-dominated regimes in pressurized thin shells, *Journal of the International Association for Shell and Spatial Structures*, **62** (2021), 125-137.
[54] T. Nouri-Baranger, Computational methods for tension-loaded structures, *Archives of Computational Methods in Engineering*, **11** (2004), 143-186.
[55] T. J. Oberbichler, *A Modular and Efficient Implementation of Isogeometric Analysis for the Interactive Cad-Integrated Design of Lightweight Structures*, PhD thesis, Technische Universität München, 2023.
[56] I. Oliveira, P. Pimenta and R. Pauletti, Consideration of wrinkling in the design of orthotropic membranes, in *II Simposio Latinoamericano de Tensoestructuras, 2005. At: Caracas, Venezuela*, 2005.
[57] R. M. O. Pauletti, An extension of the force density procedure to membrane structures, IASS Symposium, Beijing, 2006, 490-491.
[58] R. Pauletti, *História, Análise e Projeto Das Estruturas Retesadas*, Habilitation, University of São Paulo, 2003.
[59] R. M. O. Pauletti, Static analysis of taut structures, in *Textile Composites and Inflatable Structures II*, Springer, (2008), 117-139.
[60] R. M. O. Pauletti and F. L. Fernandes, An outline of the natural force density method and its extension to quadrilateral elements, *International Journal of Solids and Structures*, **185-186** (2020), 423-438.
[61] R. Pauletti, A. Pappalardo and D. Guirardi, The method of dynamic relaxation for the static nonlinear analysis of cable and membrane structures, 2008.
[62] R. M. O. Pauletti and P. M. Pimenta, The natural force density method for the shape finding of taut structures, *Computer Methods in Applied Mechanics and Engineering*, **197** (2008), 4419-4428.
[63] D. Piker, Kangaroo: Form finding with computational physics, *Architectural Design*, **83** (2013), 136-137.
[64] J. Pini and M. S. V. Souza, Beaver: A computational parametric approach for conception, analysis and design of timber structures, in *World Conference on Timber Engineering*, WCTE, 2020.

[65] R. Prandini, J. T. Pini and M. S. V. de Souza, Beaver: A parametric design framework for timber engineering, in *World Conference on Timber Engineering (WCTE 2023)*, WCTE, World Conference on Timber Engineering (WCTE 2023), 2023.
[66] C. Preisinger, Linking structure and parametric geometry, *Architectural Design*, **83** (2013), 110-113.
[67] F. Rizzo, P. D'Asdia, M. Lazzari and L. Procino, Wind action evaluation on tension roofs of hyperbolic paraboloid shape, *Engineering Structures*, **33** (2011), 445-461.
[68] F. Rizzo, P. D'Asdia, F. Ricciardelli and G. Bartoli, Characterisation of pressure coefficients on hyperbolic paraboloid roofs, *Journal of Wind Engineering and Industrial Aerodynamics*, **102** (2012), 61-71.
[69] F. Rizzo and F. Ricciardelli, Design pressure coefficients for circular and elliptical plan structures with hyperbolic paraboloid roof, *Engineering Structures*, **139** (2017), 153-169.
[70] R. Rossi, M. Lazzari, R. Vitaliani and E. Oñate, Simulation of light-weight membrane structures by wrinkling model, *International Journal for Numerical Methods in Engineering*, **62** (2005), 2127-2153.
[71] D. Rutten, Grasshopper3D, https://www.grasshopper.com/.
[72] C. Sanderson and R. Curtin, Armadillo: a template-based C++ library for linear algebra, *Journal of Open Source Software*, **1** (2016), 26.
[73] H.-J. Schek, The force density method for form finding and computation of general networks, *Computer Methods in Applied Mechanics and Engineering*, **3** (1974), 115-134.
[74] M. Souza, Bats, https://www.food4rhino.com/app/bats.
[75] M. S. V. Souza and R. Pauletti, Parametric design and optimization of shell structures using the natural force density method, IASS Symposium, Tokyo, 2016, 10.
[76] D. J. Steigmann, Tension-field theory, *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, **429** (1990), 141-173.
[77] N. Stranghöner, J. Uhlemann, F. Bilginoglu, K. Bletzinger, H. Bögner-Balz, E. Corne, N. Gibson, P. Gosling, R. Houtman, J. Llorens et al., *Prospect for European Guidance for the Structural Design of Tensile Membrane Structures: Support to the Implementation, Harmonisation and Further Development of the Eurocodes*, EUR (Luxembourg. Online), Publications Office of the European Union, 2023.
[78] Structurecraft, A computational approach to mass timber, https://www.youtube.com/watch?v=8pRsMjhAXCI.
[79] Synera, Synera gmbh, https://www.rhino3d.com/.
[80] R. L. Taylor, E. Oñate and P.-A. Ubach, Finite element analysis of membrane structures, *Springer Netherlands*, Dordrecht, 2005, 47-68.
[81] J. G. Valdés, J. Miquel and E. Oñate, Nonlinear finite element analysis of orthotropic and prestressed membrane structures, *Finite Elements in Analysis and Design*, **45** (2009), 395-405.
[82] H. Wagner, *Flat Sheet Metal Girders with Very Thin Metal Web*, Flat Sheet Metal Girders with Very Thin Metal Web, National Advisory Committee for Aeronautics, 1931.
[83] R. Wagner, On the design process of tensile structures, *Springer Netherlands*, Dordrecht, 2005, 1-16.
[84] D. S. Wakefield, Engineering analysis of tension structures: Theory and practice, *Engineering Structures*, **21** (1999), 680-690.
[85] X. Wang, L. Yin and Q. Yang, Numerical analysis of the wrinkling behavior of thin membranes, *Archive of Applied Mechanics*, **89** (2019), 2361-2380.
[86] P. Wriggers, *Nonlinear Finite Element Methods*, Springer-Verlag, Berlin, 2018.
[87] Q. Yang, F. Tan and X. Wang, Loading and wrinkling analysis of membrane structures, *Science China Technological Sciences*, **54** (2011), 2597-2604.

Received February 2024; revised March 2024; early access March 2024.