A Computational Framework for Interactive Form Finding of Textile Hybrid Structures through Dynamic Topology Control

Textile Hybrid Structures refer to the coupling of tensile form- and bending-active components into a stiffer construct. Several computational frameworks built upon the Dynamic Relaxation method have been developed for interactive explorations of material and geometric properties during form finding. However, efforts are still required when addressing dynamic alterations of topology without completely resetting the simulation. The main problem to face is the data structure design when information is dynamized. In other words, dynamic resizing of topological data without losing consistency of connectivity. As a result, data structure design needs to be considered as a transcendental aspect when looking for increased flexibility and interactivity in form finding.

In this paper, we present the development of a computational framework for form-finding textile hybrid structures enabling dynamic explorations of complex topological configurations during solver’s execution. A so-called evolving network formulation used to model mutable assemblies of interconnected particles is presented as well as the numerical scheme adopted to find the equilibrium state of such structures. The implementation of the framework is further described through the development of ElasticSpace, an interactive form finding tool for textile hybrid structures built with Java.

Keywords: Form-Finding, Evolving Networks, Textile Hybrid Structures, Dynamic Relaxation, Dynamic Topology, Data Structure

Introduction

Textile hybrid structures have rapidly become an important research area in architectural design. The main advantage of such structures is the production of highly intricate geometries by means of simplest components. Major contributions regarding its form finding have been conducted, in particular, at the Institute for Computational Design (ICD) and Institute for Building Structures and Structural Design (ITKE) based on multiple computational methods including particle-spring systems (PSS), finite element method (FEM) and physical form finding. For instance, Lienhard et al.
(Lienhard et al. 2014) have introduced the approach of contraction cables in FEM for simulating the bending behaviour of elastic beams while Ahlquist et al. (Ahlquist & Menges 2013) have described the use of particle-spring systems for the interactive design of force-active structures.

Numerical simulations based on FEM can generate a complete mechanical description of physical structures including those based on large deformations, but lacks a key element necessary for architectural design, real-time interactivity. On the other hand, PSS being a by-product of the popular Dynamic Relaxation (DR) method allows to reduce computational cost leading towards enhanced interactivities between designers and models. Because of this, designers can dynamically modify forces, parameters, stiffness coefficients and constraints of the system until finding a satisfactory equilibrium shape. However, if design intentions are not reached, numerical solvers need to be stopped and relaunched after setting topological modification. Although both methods enables different design workflows, they share the unique requirement of pre-defining an initial topology from where calculations can start. In other words, they require the discretization of a static solution region into a finite set of interconnected nodes/particles.

In this paper, it is suggested that such design strategy should be categorized as a “top-down” approach for form finding since the resulting geometry is only dependent on the initial and static topology. This approach has proven to be highly efficient when topologies have lowest degrees of complexity and can be intuitively solved as in the case of the Umbrella Marrakech (ITKE) or the Gggallery Installation (ICD). Nevertheless, when highly intricate material relationships are involved and designer’s intuition cannot provide a valid solution, as in the case of the M1 project in France (ICD/ITKE), the problem of finding a valid topology requires the use of physical
models (Ahlquist et al. 2013a). A condition that paradoxically implies additional amounts of manual calculations when dealing with highest degrees of complexity.

At the core of such ambiguity, lays the necessity for new ways of controlling the flow of information in numerical form finding when exploring the potential of textile hybrid systems. In the following, a different design strategy that is suggested to be categorized as a “bottom-up” form finding approach is proposed by using the concept of dynamic topologies. In a “bottom-up” approach, form finding processes are govern by mutable topological and geometrical spaces. Thereby, form-found geometry is not anymore the immediate result of a static topology since this one is progressively rebuilt during the simulation. A condition that is more closely related to the implicit nature of physical form finding where material’s behaviour and organization are inseparably associated.

![Diagram of form-finding approaches](image)

Figure 1. Design approaches for form-finding a) Top-Down b) Bottom-Up

This paper introduces the development of an advanced computational framework for the exploration of complex form/force equilibrium shapes of textile hybrid structures following the “bottom-up” approach for form finding. On this basis, an evolving network formulation is presented for guiding the dynamization of topological data. In addition, it has been assumed that the most convenient method to explore such approach is DR given its highest flexibility in comparisons to FEM.
Background

Textile Hybrid Structures

A textile hybrid structure defines the combination of tensile form- and bending-active structures into a unique construct given shared characteristic regarding flexibility, pre-stress and lightness. In the context of this research the coupling of one-dimensional bending-active components with one- and two-dimensional tensile form-active components is investigated. Related works have addressed this specific coupling as a complex forming process driven by reciprocal interactions between membrane surfaces and systems of elastic rods (Lienhard & Ahlquist et al. 2013). That is pre-stress in membranes is introduced through the elastic deformation of bending-active components whose shape is in turn controlled by the deformation of membranes (Lienhard & Alpermann et al. 2013). In general, this type of structural system has been addressed with interchangeable terms as hybrid bending-active structures (Obrębski & Tarczewski 2013), bending-active tensile membrane hybrid structures (Thomsen et al. 2015) and textile hybrid structures (Knippers et al. 2011; Ahlquist et al. 2013b; Lienhard 2014).

Numerical Form-Finding

Originated from the analogy of tidal flow computations by Day (Day 1965), DR has been further developed as an explicit method for static analysis of structures by M.R. Barnes (Barnes 1999). Compared to other numerical methods, DR is more suitable for interactive form-finding applications as no global stiffness matrix is needed to find a solution. This means flexible and interactive design spaces in which the discretization of an arbitrary body can be organized as a network-type assembly of interconnected particles. An extensive comparative review of dynamic methods for structural form-finding is presented by Veenendaal and Block (Veenendaal & Block 2012).
In the context of textile hybrid structures, special attention demands the research conducted by Ahlquist (Ahlquist & Menges 2013; Ahlquist et al. 2015) regarding the development of an interactive form finding tool called springFORM built with the Processing programming language and Simon Greenwood’s particle library (Greenwold 2004). Greenwood’s library is implementing an implicit Runge Kutta solver of fourth order based on the pioneer work of Baraff and Witkin (Baraff & Witkin [date unknown]). This library has been also used in the development of more simulation tools like the KiteSim, a kitesurf simulator which incorporates bending deformations on inflatable beams (Kitesim 2016) and the second version of the CADenary tool developed by Killian (Killian & Ochsendorf 2005).

Equally relevant has been the development of the ShapeOp library (Deuss et al. 2015) which has been the basis for further contributions of the Royal Danish Academy of Arts (Holden Deleuran et al. 2015) and the new release of Kangaroo2. In general, Kangaroo (Piker 2013) has become the most popular tool among the architectural community for form finding where the major contribution for topological dynamization is attributed to the development of Mesh Machine (Piker & Pearson 2016), a grasshopper component for dynamic remeshing. Lastly, we need to consider efforts conducted at the Vrije Universiteit Brussel in collaboration with the Block Research Group regarding the implementation of a framework for form finding textile hybrid structures in Rhinoceros (Gengnagel et al. 2013). A comparative review of such developments and available physic libraries are presented in Figure 2.
Figure 2. Chart of current efforts for interactive form finding

**An Evolving Network Model**

*Initial Conditions as a Graph Related Problem*

Data structures are a transcendental piece for increasing interactivity in numerical simulations. However, the main problem of designing a data structure relays on the necessity to work on two different layers. The construction of a concrete data structure and the design of an abstract model governing data content and organization. Because of this, abstract network models have become an important research area for numerical methods where first efforts are associated to Fenves and Branin (Fenves & Branin 1963) and the use of branch-node matrices. Most relevant work on the use of network models for dynamic methods are associated to the development of the Force Density Method by Schek (Schek 1974) and Linkwitz (Linkwitz 1999). In the following, the term node is used in the context of graph’s theory for representing the concept of a particle in DR.

In a network formulation, initial conditions can be treated separately in relation to topological (i.e. connectivity), geometrical (i.e. spatial position) and mechanical
properties (i.e. load relationships). Connectivity is then stored through a branch node matrix $C$ separated into two sub-matrices $C_f$ for constrained nodes and $C_i$ for free nodes as in Figure 3. The great benefit is the straightforward modification of stored data regarding geometrical and mechanical properties during the execution of the algorithm. Nevertheless, altering the number of particles and connections is usually not as direct due to the risk of inconsistent information affecting solver’s stability. This is because, network models has been conceived for efficient access and storage of information without further specifications for data dynamization.

![Figure 3. Branch-node matrix of a general network](image)

<table>
<thead>
<tr>
<th></th>
<th>$C_f$</th>
<th></th>
<th>$C_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 2 3</td>
<td>4</td>
<td>5 6 7</td>
</tr>
<tr>
<td>a</td>
<td>0 -1 0</td>
<td>0</td>
<td>0 1 0</td>
</tr>
<tr>
<td>b</td>
<td>0 0 -1</td>
<td>0</td>
<td>0 0 1</td>
</tr>
<tr>
<td>c</td>
<td>0 0 0</td>
<td>-1</td>
<td>0 0 0</td>
</tr>
<tr>
<td>d</td>
<td>-1 0 0</td>
<td>0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>e</td>
<td>0 0 0</td>
<td>0</td>
<td>1 0 0</td>
</tr>
<tr>
<td>f</td>
<td>0 0 0</td>
<td>0</td>
<td>0 1 -1</td>
</tr>
<tr>
<td>g</td>
<td>0 0 0</td>
<td>0</td>
<td>0 0 1</td>
</tr>
<tr>
<td>h</td>
<td>0 0 0</td>
<td>0</td>
<td>-1 0 0</td>
</tr>
</tbody>
</table>

**From a General Network to an Evolving Network Formulation**

To describe complex topological relationships that regularly change under certain conditions, an evolving network formulation is proposed. However, before treating with the dynamization problem, specific and recurrent characteristics on the abstract network model need to be defined.

Non-uniform hypergraphs are best suited for describing network topologies when bending stiffness is calculated through the spline beam element formulation (see section 4.1). Hence, two types of hyperedges are design for describing connections between ordered sets of two (i.e. normal graph’s edge) and three nodes respectively.
Furthermore, nodes and hyperedges need to be labelled for facilitating data access, and coloured for describing states. Connectivity is guaranteeing by not allowing isolated nodes (i.e. nodes with degree zero) and loop hyperedges (i.e. edge that connects a node to itself).

The incidence matrix $C$ formulated as in Figure 4, is separated into six sub-matrices, two of them for hyperedges and the rest for nodes. Nodes ’sub-matrices are categorized as $C_1$ for nodes on form-active components (i.e. light blue), $C_2$ for nodes on bending-active components (i.e. green), $C_3$ for shared nodes laying at the intersection of multiple components (i.e. purple) and $C_4$ for constrained nodes (i.e. red). On the other hand, hyperedges ’sub-matrices are categorized as $C_t$ when describing a form-active component (i.e. grey) and $C_b$ when describing a bending-active component (i.e. black). Additional sub-matrices are also defined for describing local components. In the latter case, the number of sub-matrices is proportional to the number of components shaping the textile hybrid entire system.

![Figure 4. Modified branch-node matrix of an evolving network](image)

**Hypergraph’s grammars**

User-defined grammars are then incorporated for driven dynamic data transformations.
In general terms, grammars are used to specify all allowed transformation on a graph by combining a vocabulary with rules and axioms (Ehrig et al. op. 1999). In our model, integers and colour’s values are considered as the vocabulary, whereas axioms are structured combinations of the vocabulary describing basic network configurations of tensile form- and bending active components.

Rules are grouped into rulesets and consist of operations for union/deletion of hypergraphs, hyperedges and nodes. A detailed information of rules and rulesets are presented in Figure 5. Taking ruleset I, the deletion of a node in all cases (e.g. rules 1, 5) implies the removal of incident hyperedges. The insertion of a node (e.g. rules 2, 6) within a pair of pre-stored nodes is executed by removing all incident hyperedges and creating all necessary hyperedges for the new configuration. Moreover, the deletion of hyperedges can be a rather simple task as in the case of rules 3 or a complex operation as in the case of rules 4, 7 and 8. On the other hand, production rules for operations between hypergraphs are grouped in ruleset II. In some union operations (e.g. rules 5, 10, 11, 17 and 18), a host hypergraph $G_1$ where all transformations are cumulated is required in addition to the hypergraph $G_2$ to be added. In other cases (e.g. rules 14, 15 and 16), operations are more straightforward since no host hypergraph is required.
In this section, we will discuss the dynamic relaxation schema adopted for calculating particle’s forces. DR uses the concept of forces applied to particles for approximating the physical behaviour of rigid and non-rigid objects. One particle is defined as an object with its own data structure mainly consisting of a fictitious scalar mass $M$ and a

**Dynamic Relaxation Scheme**

In this section, we will discuss the dynamic relaxation schema adopted for calculating particle’s forces. DR uses the concept of forces applied to particles for approximating the physical behaviour of rigid and non-rigid objects. One particle is defined as an object with its own data structure mainly consisting of a fictitious scalar mass $M$ and a

**Figure 5. Hypergraph grammars for data dynamization**
set of three-dimensional vectors representing position $\mathbf{x}$, velocity $\mathbf{v}$ and acceleration $\dot{\mathbf{v}}$.

The motion of any particle $i$ at time $t$ is governed by Newton’s second law of motion reorganized to solve acceleration as in eq. (1), where $R_i$ denote the vector sum of all internal forces.

$$
\dot{\mathbf{v}}_i^t = \frac{R_i^t}{M_i}
$$

Three different options for numerical integrators scheme have been adopted where recurrent equations for updating particle’s velocity and position following a symplectic Euler scheme are presented in eq. (2), Velocity Verlet in eq. (3) and Runge Kutta 4 in eq. (4).

$$
\mathbf{v}_{i}^{t+\Delta t} = \mathbf{v}_{i}^{t} + \Delta t \cdot \dot{\mathbf{v}}_{i}^{t}
$$

$$
\mathbf{x}_{i}^{t+\Delta t} = \mathbf{x}_{i}^{t} + \Delta t \cdot \mathbf{v}_{i}^{t+\Delta t}
$$

$$
\mathbf{v}_{i}^{t+\Delta t} = \mathbf{v}_{i}^{t} + \frac{\mathbf{v}_{i}^{t} + \mathbf{v}_{i}^{t+\Delta t}}{2} \Delta t
$$

$$
\mathbf{x}_{i}^{t+\Delta t} = \mathbf{x}_{i}^{t} + \mathbf{v}_{i}^{t} \Delta t + \frac{\Delta t}{2} \dot{\mathbf{v}}_{i}^{t}
$$

$$
\mathbf{v}_{i}^{t+\Delta t} = \mathbf{v}_{i}^{t} + \frac{1}{6} \Delta t(k_1 + 2k_2 + 2k_3 + k_4)
$$

$$
\mathbf{x}_{i}^{t+\Delta t} = \mathbf{x}_{i}^{t} + \frac{1}{6} \Delta t(k_1 + 2k_2 + 2k_3 + k_4)
$$
Particle’s Forces

Particle’s vector forces due to bending and shear are calculated through the spline beam element formulation proposed by Adriaenssens et al. (Adriaenssens & Barnes 2001), where sets of three consecutive particles lying in the entire beam element are required. Taking one set of three consecutive particles $i, j$ and $k$, with $a$ the vector from $j$ to $i$, $b$ the vector from $j$ to $k$ and with $c$ the vector from $i$ to $k$. Then the curvature vector $u$ and bending moment vector $m$ at $j$ can be defined as in eq. (1) and (2), where $E$ is the elastic modulus and $I$ is the moment of inertia.

\[
\begin{align*}
  u_j &= \frac{(\|a\|^2b - \|b\|^2a) \times (a \times b)}{2\|a \times b\|^2} \\
  m_j &= \frac{EI}{\|u_j\|} \hat{u}_j
\end{align*}
\]  

Shear forces $S_a$ and $S_b$ for segments $ij$ and $jk$ are computed as in eq. (3) and applied through vectors acting normal to segments $ij$ and $jk$ within the plane define by $ijk$ as in eq. (4) and (5).

\[
\begin{align*}
  S_a &= \frac{2EIsin \propto}{\|a\|\|c\|} \quad \text{and} \quad S_b = \frac{2EIsin \propto}{\|b\|\|c\|}
\end{align*}
\]

\[
\begin{align*}
  s_{a,i} &= S_a \left[ \frac{(a \times b) \times a}{\|(a \times b) \times a\|} \right] \quad \text{and} \quad s_{a,j} = -s_{a,i} \\
  s_{b,i} &= S_b \left[ \frac{(a \times b) \times b}{\|(a \times b) \times b\|} \right] \quad \text{and} \quad s_{b,j} = -s_{b,i}
\end{align*}
\]

Axial vector forces $t$ for all elements connecting two particles $i$ and $j$ are calculated as
in eq. (6), where \((EA/L_0)\) is the elastic stiffness, \(T\) is the initial specified tension, \(L\) is the natural length of the element and \(\Delta L\) is the extension of the element from the initial length.

\[
t_i = T + \frac{(EA)}{L_0} \cdot \Delta L \text{ and } t_j = -t_i
\]  

\((6)\)

**ElasticSpace**

ElasticSpace is the name of the form finding tool implementing our network model and numerical scheme. The Java programming language was selected for implementation given the flexibility of the object oriented paradigm to deal with highest levels of abstractions in comparison to procedural paradigm. The tool is characterized by a highly interactive workflow that closely approximate shape exploration in analogue form-finding. A concrete data structure from our abstract model is constructed in a class Hypergraph where grammars can be easily implemented via class’s methods. Hash tables with a map interface, are used to store instantiations of nodes (i.e. particles) and hyperedges (i.e. connections) classes by means of unique identifiers of type long integer (i.e. labels). The designer can dynamically control topological transformations in addition to material and geometric properties, and solver’s setups. Therefore, form finding can start without specifying an input model since this one can be progressively built by using precomputed descriptions of basic components (i.e. axioms). Three families of constructors are available for component’s initialization. In all cases, the GUI displays auxiliary geometries helping the construction.
Figure 6ElasticSpace a) insertion bending-active components, b) insertion form-active components, c) Catenoid-like membrane, d) Multi-dimensional membrane, e) Bending-active component with external references, f) Bundling

Bending-active components are built (Figure 5a) by defining material and metric properties, number of particles in the rod and, optionally, external references (i.e. particles that will be shared with other components). On the absence of external references, the component requires to specify the type of support condition (e.g. hinged, clamped), in close or open linear configuration. When built through external references, the bundling characteristic is activated by defining the type of connection condition (Figure 5e-5f). On the other hand, form-active components are always externally referenced. A form-active component can be constructed from one or two rods (Figure
5b) while another possibility is to build an open/close catenoid-like component between two prebuilt form-active components (Figure 5d). The construction also involves selecting a seed particle and defining a neighbourhood range (Figure 5c). Finally, a hybrid component can be directly built by specifying external references. In this case, material and metric properties for both, bending- and form-active components are required.

![Model built in ElasticSpace](image1)

![Model reconstruction in Grasshopper](image2)

Figure 7. Data exchange between ElasticSpace and Rhinoceros/Grasshopper

Components deletion is rather a straightforward and intuitive task. However, only in the special case of deleting a bending-active component where particles are shared within multiple form-active components, the designer has the possibility to merge all form-active components or keep them separately. The first scenario also implies re-
computing material and geometrical properties. At the lowest level, individual particles can be also added or deleted. Since elements can be freely added and eliminated during the simulation, the equilibrium state vary with each modification. Simulation stop when the program is closed. At any time in the simulation, the database of the model can be exported using an XML file which enables communication with different programs. Data is organized in such a way that the model can be easily reconstructed in other environments like Rhinoceros/Grasshopper as shown in Figure 7.

A set of Grasshopper components have been equally developed for preparing the exchange of data between Rhinoceros and ElasticSpace. Models generated at ElasticSpace can be easily imported and reconstructed in Grasshopper as shown in Figure 6a. Geometrical elements built in Rhinoceros can be assembled into a textile hybrid object and imported to ElasticSpace via an XML file as shown in Figure 6b, where further processes of topological data dynamization and geometrical transformation occur.

![Diagram](image1)

![Diagram](image2)

Figure 8. Grasshopper components for ElasticSpace, a) Data Parsing, b) XML Creation
Conclusions and Further Work

In the context of textile hybrid structures, higher degrees of topological control are desired during form finding due to complex interrelations of local components governing system’s global deformation. To be precise, more flexibility on data structures is required. In this paper, we have presented what has been called a “bottom-up” approach for form-finding based on extending the general network model for defining static topologies to what has been called, an evolving network model for defining dynamic topologies.

Among different challenges, the main problem to overcome was the formulation and the implementation of such network model in order to effectively organize dynamic information of connectivity at the intersection of different components without creating topological inconsistencies, and affecting the stability and the speed of the physical solver. At the abstract level of the data structure, we have defined a set of grammars for driving transformations on hypergraphs describing such evolving networks. Then, we have presented the development of ElasticSpace, an interactive form finding tool for textile hybrid structures implementing the evolving network model. This tool facilitates the exploration of component’s aggregation driven by reciprocal elastic deformations where picturing initial topologies becomes digitally unmanageable.

This so-called “bottom-up” approach opens new research questions regarding the exploration of larger articulation of bending- and form-active components into a unique system. Questions on the subject of what type of component, where and when needs to be introduced, and how is affecting the stability of the entire hybrid system, are arising. What is more, in some cases, the system may need to be aware that the addition of new components could provoke deformations rendering unnecessary the presence of old components. Therefore, decision-making in digital spaces could be driven by
behavioural systems allowing to manage higher degrees of design complexities. Through the integration of weighted behaviours, challenging decisions regarding where particles need to be eliminated or added, and where elements need to be expanded or contracted could be driven on the basis of different optimization criterions.

Figure 9. Extension of ElasticSpace with behavioural logics

Initial studies for such problematic have been started through the development of a second Java program extending ElasticSpace where particles have the capacity to embed conducts of agency. Behaviours are generated from the classification of two types of behavioural particles: parents and children. The workflow start by setting an initial position for parents and some random zones where they are attracted as shown in (Fig. 7a). Parents are flocking in the digital space while dropping children (Fig. 7b). Each parent and its children defines the topology of an elastic rod which is progressively built over time. Contrary to the behaviour of parents, children attract each other by creating a cable connection between closest children of a different family (Fig.
With such basic rules, an emergent textile hybrid system arises (Fig. 7d) which shows the generative potentials of integrating agency during earlier forming processes of textile hybrid-active structures.

Further studies will be focused on the development of new methods for extending behavioural logics into particles driving towards optimization strategies for complex material articulations in textile hybrid-active structures.

References


