*Proceedings of the IASS Annual Symposium 2016*
"Spatial Structures in the 21st Century"
26–30 September, 2016, Tokyo, Japan
K. Kawaguchi, M. Ohsaki, T. Takeuchi (eds.)

# Gradient-based optimization of closest-fit funicular structures

Pierre CUVILLIERS*, Renaud DANHAIVE[a], Caitlin MUELLER[a]

*, [a] Building Technology Program, Massachusetts Institute of Technology
77 Massachusetts Avenue, Room 5-418, Cambridge, MA, USA
pcuvil@mit.edu, danhaive@mit.edu, caitlinm@mit.edu

## Abstract

The aim of this research is to solve an inverse form-finding problem: construct funicular, axial-only structures as close as possible to a target surface defined by the designer. The scope is limited to grid-like, node-and-branch only networks, which can be solved efficiently using the force density method (FDM). This problem is formulated as a least-squares nonlinear optimization problem, and is solved using the constrained nonlinear solvers implemented in MATLAB. Two nonlinear constrained solving methods, interior-point and trust-reflective region, are found to be the fastest with a large convergence domain. The first and second-order derivatives of the objective function are found analytically. Lastly, the problem is shown to have several degeneracies; one in particular cannot be removed with physical arguments and leads to new insights in the numerical form finding of funicular surfaces.

**Keywords**: form finding, funicular structures, optimization, closest fit, force density method.

## 1. Introduction

Compression-only structures are much more efficient than structures where bending occurs. For example, in a linear structural element, slender by definition, axial loads are straight forces resisted uniformly by the whole section, while bending is a force with a large moment arm resisted by the section where only small moment arms exists. Thus, by focusing on funicular structures for a given load, we guarantee that all the material will be used with the most efficiency. This explains the significant interest in such structures historically and today.

The compression-only state in a structure is similar to a perfectly flexible chain with no self-weight, fixed at both ends with some slack, hanging under the action of weights attached to it. The chain undergoes tension only, and if its shape were inverted, it would resist the same loads in a compression-only state. This principle is at the root of many physical experiments for the form-finding of funicular structures, such as Antoni Gaudí's (1852-1926) hanging models of the Sagrada Família.

In general, designing compression-only structures is challenging. Physical experiments of hanging chains provide a good way to interact and quickly iterate on design options, but lack the precision of a CAD model. Computationally, finding the shape of a set of hanging chains requires accounting for nonlinear behavior due to large displacements; this problem can be alleviated using the force density method (FDM) presented in Schek [1]. However, this only solves a *direct problem*: finding one discrete funicular shape under given loads and grid properties. Often, this goes against the intuitive design process where the designer has a shape in mind and wants to minimize bending. That describes an *inverse problem* where the closest possible funicular structure to a target surface is found.

The aim of this study is to solve one such inverse problem: construct funicular structures as close as possible to a target surface. The scope is limited to grid-like, node-and-branch only networks, for which an efficient calculation procedure exists in large displacements. In the particular case of grid-like funicular structures, this inverse problem is solved in Block and Lachauer [2] using a genetic algorithm.

This has the advantage of accepting even badly formatted problems, but remains slow and lacks a guarantee of finding a global optimum. The problem receives a more formal treatment in van Mele and Block [3] but the scope is limited to pre-tensioned cable nets, for which a good initial estimate of the solution is known. This paper expands on this previous work by using gradient-based optimization methods to gain more insight into the solutions for such closest-fit inverse problems.

## 2. Background

### 2.1. Funicular bar networks

We consider networks of nodes connected by bars, with free rotations at the nodes. This guarantees that there are only constant axial forces in the bars. The equilibrium at each node is then only a consequence of the position of the nodes. Then for a network of bars intersecting at the nodal positions $\boldsymbol{x}$, under the loads $\vec{p}$ at the nodes and with the objective surface $\mathcal{S}$, the problem has the form:

$$(P): \min_{\mathrm{x}} d(\boldsymbol{x}, \mathcal{S})^2 \text{, such that at all nodes: } \sum \vec{F} = \vec{p},$$

where $d$ is a distance measuring the fitness of the points on the surface. This is a convex objective function with nonlinear constraints. This is the general closest-fit problem formulation for funicular bar networks; the specifics of the distance function $d$ used in this research are given in Section 3.

### 2.2. Force-Density Method

The force-density method, introduced in Schek [1], is a method well-suited to explore a large number of funicular structures resulting from the same initial bar network. Three assumptions are made: (i) every bar is elastically stretched proportionally to the force it carries following Hooke's law (constitutive equation), (ii) the length of a bar is equal to the distance between the nodes that it connects (compatibility equation) and (iii) each node is in equilibrium (equilibrium equation). Mathematically, this becomes:

$$\begin{cases} \boldsymbol{C_N}^T \boldsymbol{Q} \boldsymbol{C_N} \boldsymbol{x_N} + \boldsymbol{C_N}^T \boldsymbol{Q} \boldsymbol{C_F} \boldsymbol{x_F} - \boldsymbol{p_x} = 0 \\ \boldsymbol{C_N}^T \boldsymbol{Q} \boldsymbol{C_N} \boldsymbol{y_N} + \boldsymbol{C_N}^T \boldsymbol{Q} \boldsymbol{C_F} \boldsymbol{y_F} - \boldsymbol{p_y} = 0 \\ \boldsymbol{C_N}^T \boldsymbol{Q} \boldsymbol{C_N} \boldsymbol{z_N} + \boldsymbol{C_N}^T \boldsymbol{Q} \boldsymbol{C_F} \boldsymbol{z_F} - \boldsymbol{p_z} = 0 \end{cases} \tag{1}$$

Here, $\boldsymbol{C_N}$ is the edge matrix for the free nodes, taking value -1 for the start node of a bar and +1 at its end node; $\boldsymbol{C_F}$ is the edge matrix for the fixed nodes; $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{z}$ are the positions of the nodes; $\boldsymbol{Q}$ is the diagonal matrix of the force densities; and $\boldsymbol{p}$ is the vector of all external loads. Recall that the force density $q$ is the ratio of the force in a bar, $s$, to its length, $l$:

$$q = s/l \tag{2}$$

We set for clarity $\boldsymbol{D_N} = \boldsymbol{C_N}^T \boldsymbol{Q} \boldsymbol{C_N}$ and $\boldsymbol{D_F} = \boldsymbol{C_N}^T \boldsymbol{Q} \boldsymbol{C_F}$. The positions of the nodes solutions to a direct problem can then be found using only linear algebra:

$$\begin{cases} \boldsymbol{x_N} = \boldsymbol{D_N}^{-1}(\boldsymbol{p_x} - \boldsymbol{D_F} \boldsymbol{x_F}) \\ \boldsymbol{y_N} = \boldsymbol{D_N}^{-1}(\boldsymbol{p_y} - \boldsymbol{D_F} \boldsymbol{y_F}) \\ \boldsymbol{z_N} = \boldsymbol{D_N}^{-1}(\boldsymbol{p_z} - \boldsymbol{D_F} \boldsymbol{z_F}) \end{cases} \tag{3}$$

However, finding the solution to an inverse problem, where the target shape is known and the force densities are the unknown, is not evident, as there is no clear link between the two.

### 2.3. Rationalization of freeform surfaces with funicularity constraints

As shown above, different choices of force densities naturally lead to different funicular solutions. This is a consequence of the linearization of the system of equilibrium equations where, physically, different stiffness is assigned to each bar of the network. If we recall that the initial problem is to solve the equilibrium of a bar network, it is cogent that different distributions of stiffness yield different equilibrium shapes. We can use these force densities to obtain a funicular network that fits a target surface as close as possible. The strategy is illustrated in Figure 1 and formalized in Section 3.

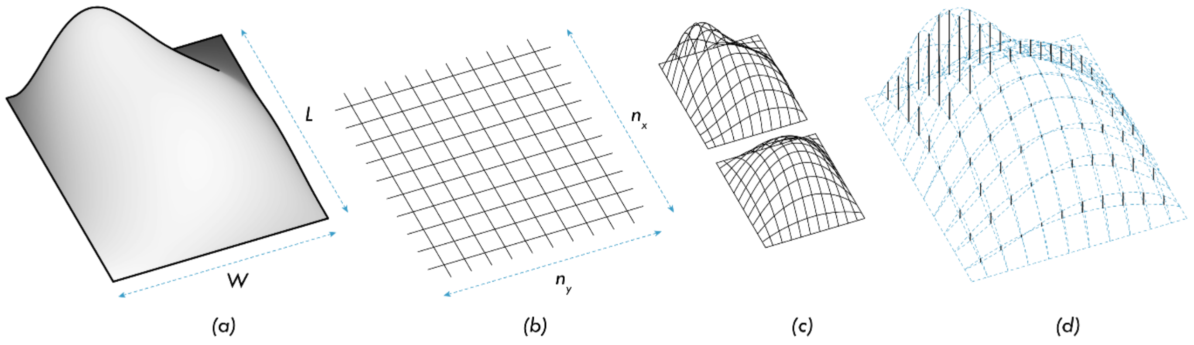


Figure 1: Rationalizing a freeform surface (a): (b) choose numbers of branches $n_x$ and $n_y$ in the x and y directions, (c) approximate the target surface by projecting the grid vertically and (d) compare to the funicular network found using FDM. Then, optimize the force densities to reduce the distance.

## 3. Problem Formulation

To keep the problem tractable, we constrain it from the outset by imposing that the nodes will only move vertically from their initial positions. This can be achieved by reducing the number of free parameters, i.e. by imposing that each bar of a branch has the same force density. This constraint is derived by analyzing the horizontal equilibrium of a single node in a quadrilateral grid. In particular, this means that two bars initially aligned must carry the same horizontal force, related to their force density and length by eq. (2). Given that we restrict our problem to regular rectangular grid, the lengths of two bars belonging to the same branch must thus be equal and so must be their force densities. The matrix $\boldsymbol{K}$, whose entries are all 0 or 1, is introduced to link the bar force densities $q_i$'s to the $n_x + n_y$ independent branch force densities:

$$\boldsymbol{q} = \boldsymbol{K}\boldsymbol{b} \tag{4}$$

Since we have restricted our problem to vertical displacements only, we are able to devise a simple metric for the distance between a bar network and our target surface. This metric is defined as follows:

$$d(\boldsymbol{z}_N) = \|\boldsymbol{z}_N - \boldsymbol{z}_T\|^2, \tag{5}$$

where $\boldsymbol{z}_N$ is the vector of the z-coordinates of the free nodes of the funicular bar network and $\boldsymbol{z}_T$ is the vector of the target z-coordinates, i.e. the z-coordinates of the grid nodes projected on the target surface.

### 3.1. Unconstrained Problem

The inverse form finding problem is the nonlinear, unconstrained optimization problem formulated as follows:

$$(P_1)\colon \min_{\boldsymbol{b}} \; F(\boldsymbol{b}) = \left\| \left( \boldsymbol{D}_N{}^{-1}(\boldsymbol{p}_z - \boldsymbol{D}_F \boldsymbol{z}_F) \right) - \boldsymbol{z}_T \right\|^2, \tag{6}$$

where $\boldsymbol{D}_N = \boldsymbol{C}_N{}^T diag(\boldsymbol{q}) \boldsymbol{C}_N$, $\boldsymbol{D}_F = \boldsymbol{C}_N{}^T diag(\boldsymbol{q}) \boldsymbol{C}_F$, and $\boldsymbol{q} = \boldsymbol{K}\boldsymbol{b}$. Even though $(P_1)$ is not convex for all values of $\boldsymbol{b}$, convexity can be found and gradient-based methods for solving it are still available if the problem is constrained to a smaller domain.

### *3.1.1. Gradient*

To find the gradient of the objective function, we first look for the Jacobian $\boldsymbol{J}(\boldsymbol{b})$ of the function $\boldsymbol{f}(\boldsymbol{b}) = \boldsymbol{z}_N(\boldsymbol{b}) - \boldsymbol{z}_T$. Using the chain rule, we get:

$$\boldsymbol{J}(\boldsymbol{b}) = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}_N} \frac{\partial \boldsymbol{z}_N(\boldsymbol{q})}{\partial \boldsymbol{q}} \frac{\partial \boldsymbol{q}(\boldsymbol{b})}{\partial \boldsymbol{b}} \tag{7}$$

With $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}_N} = 1$ , $\frac{\partial \boldsymbol{z}_N(\boldsymbol{q})}{\partial \boldsymbol{q}} = -\boldsymbol{D}_N{}^{-1}\boldsymbol{C}_N{}^T\boldsymbol{W}$ (from Schek [1]), and $\frac{\partial \boldsymbol{q}(\boldsymbol{b})}{\partial \boldsymbol{b}} = \boldsymbol{K}$ , we get: $\boldsymbol{J}(\boldsymbol{b}) = -\boldsymbol{D}_N{}^{-1}\boldsymbol{C}_N{}^T\boldsymbol{W}$. $\boldsymbol{W}$ is the diagonal matrix of the lengths of the bars projected on the $z$ axis: $\boldsymbol{W} = diag(\boldsymbol{C}_N\boldsymbol{z}_N + \boldsymbol{C}_F\boldsymbol{z}_F)$. Finally, the gradient of the objective function is:

$$\nabla_{\boldsymbol{b}}\|\boldsymbol{z}_N - \boldsymbol{z}_T\|^2 = -2\,\boldsymbol{K}^T.\boldsymbol{W}.\boldsymbol{C}_N.\boldsymbol{D}_N{}^{-\mathrm{T}}.(\boldsymbol{z}_N - \boldsymbol{z}_T) \tag{8}$$

### *3.1.2. Hessian*

It is also possible to obtain the Hessian of $F(\boldsymbol{b})$ in a similar fashion. This derivation, or a similar one for equivalent problems, was not found in the literature by the authors.

$$\begin{aligned}\boldsymbol{H}(\boldsymbol{b}) = \frac{\partial^2\|\boldsymbol{z}-\boldsymbol{z}_T\|}{\partial \boldsymbol{b}^2} &= 2 * \frac{\partial}{\partial \boldsymbol{b}}\left((\boldsymbol{z}-\boldsymbol{z}_T)^T.\frac{\partial(\boldsymbol{z}-\boldsymbol{z}_T)}{\partial \boldsymbol{b}}\right)\\ &= 2 * \left(\boldsymbol{J}(\boldsymbol{b})^T.\boldsymbol{J}(\boldsymbol{b}) + \sum\nolimits_i (\boldsymbol{z}-\boldsymbol{z}_T)_i.\frac{\partial^2 z_i(\boldsymbol{b})}{\partial \boldsymbol{b}^2}\right)\end{aligned} \tag{9}$$

We used an explicit summation to lift any ambiguity on the third order tensors contraction. To get $\frac{\partial^2 z_i(\boldsymbol{b})}{\partial \boldsymbol{b}^2}$, we apply the chain rule twice on $\boldsymbol{b} = \boldsymbol{b}(\boldsymbol{q})$ and note that $\boldsymbol{K}$ is constant in $\boldsymbol{b}$, to reuse the expression of $\boldsymbol{J}(\boldsymbol{b})$:

$$\begin{aligned}\frac{\partial^2 z_i(\boldsymbol{b})}{\partial \boldsymbol{b}^2} &= \frac{\partial}{\partial \boldsymbol{b}}\left(\frac{\partial z_i(\boldsymbol{b})}{\partial \boldsymbol{q}}.\frac{\partial \boldsymbol{q}}{\partial \boldsymbol{b}}\right) = \frac{\partial}{\partial \boldsymbol{b}}\left(\frac{\partial z_i(\boldsymbol{b})}{\partial \boldsymbol{q}}.\boldsymbol{K}\right) = \frac{\partial^2 z_i(\boldsymbol{b})}{\partial \boldsymbol{b}\partial \boldsymbol{q}}.\boldsymbol{K} = \boldsymbol{K}^T.\frac{\partial^2 z_i(\boldsymbol{b})}{\partial \boldsymbol{q}^2}.\boldsymbol{K}\\ &= \boldsymbol{K}^T.\frac{\partial}{\partial \boldsymbol{q}}(-\boldsymbol{D}^{-1}\boldsymbol{C}^T\boldsymbol{W})_{i,.}.\boldsymbol{K}\end{aligned} \tag{10}$$

where $(-\boldsymbol{D}^{-1}\boldsymbol{C}^T\boldsymbol{W})_{i,.}$ is the $i$th row of $\boldsymbol{J}(\boldsymbol{b})$. Then, for each component, using eq. (59) from Petersen and Pedersen [4] for the one variable derivative of the inverse of a matrix:

$$\left\{\frac{\partial^2 z_\alpha(\boldsymbol{b})}{\partial \boldsymbol{q}^2}\right\}_{i,j} = \left\{\frac{\partial}{\partial q_i}(-\boldsymbol{D}^{-1}\boldsymbol{C}^T\boldsymbol{W})_{\alpha,j}\right\}_{i,j} = \left\{\left[\boldsymbol{D}^{-1}\boldsymbol{C}^T\frac{\partial \boldsymbol{Q}}{\partial q_i}\boldsymbol{C}\boldsymbol{D}^{-1}\boldsymbol{C}^T\boldsymbol{W}\right]_{\alpha,j}\right\}_{i,j} \tag{11}$$

Since $\boldsymbol{Q} = \mathrm{diag}(\boldsymbol{q})$, $\frac{\partial \boldsymbol{Q}}{\partial q_i}$ is a matrix of zeros with only one 1 on the diagonal in row $i$. Then, we have $\boldsymbol{C}^T\frac{\partial \boldsymbol{Q}}{\partial q_i}\boldsymbol{C} = C_{i,i}^2 = 1$ since $\boldsymbol{C}$ is made entirely of 1 and $-1$. We get:

$$\left\{\frac{\partial^2 z_\alpha(\boldsymbol{b})}{\partial \boldsymbol{q}^2}\right\}_{i,j} = \left\{[\boldsymbol{D}^{-2}\boldsymbol{C}^T\boldsymbol{W}]_{\alpha,j}\right\}_{i,j} = \begin{pmatrix}[\boldsymbol{D}^{-2}\boldsymbol{C}^T\boldsymbol{W}]_{\alpha,1}\\ \vdots \\ [\boldsymbol{D}^{-2}\boldsymbol{C}^T\boldsymbol{W}]_{\alpha,n_q}\end{pmatrix}_{i,j} \tag{12}$$

Finally by replacing $\frac{\partial^2 z_i(\boldsymbol{b})}{\partial \boldsymbol{b}^2}$ with its value, reorganizing the sum and compacting it to a matrix product:

$$\boldsymbol{H}(\boldsymbol{b}) = 2 * \left( \boldsymbol{J}(\boldsymbol{b})^T \boldsymbol{J}(\boldsymbol{b}) + \boldsymbol{K} \begin{pmatrix} (\boldsymbol{z} - \boldsymbol{z}_T) \, . \, [\boldsymbol{D}^{-2} \boldsymbol{C}^T \boldsymbol{W}] \\ \vdots \\ (\boldsymbol{z} - \boldsymbol{z}_T) \, . \, [\boldsymbol{D}^{-2} \boldsymbol{C}^T \boldsymbol{W}] \end{pmatrix} \boldsymbol{K}^T \right) \tag{13}$$

This matrix has a rank deficiency of 1, the consequences of which are discussed in the example problem of Section 4.1.1 and in general in Section 3.3.

## 3.2. Structure of the unconstrained problem

In order to gain insight about the problem, the simplest example comprised of four bars and one node is analyzed. Although the bar network has four bars, it only has two branches, hence two independent force parameters, which will allow us to visualize the objective function. The physical problem is presented visually in Figure 2. The goal of the optimization is to have $\boldsymbol{z}_N$ reach $\boldsymbol{z}_T$ by modifying the independent force densities of the network, namely $b_1$ and $b_2$. In this particular problem, it is obvious that it is possible to fit a funicular solution to the target since the target is itself funicular. This simple problem is thus well-suited not only for understanding the problem but also to test out algorithms. In the numerical applications, $L$, $W$, $\boldsymbol{z}_T$, and $\boldsymbol{p}_z$ are respectively equal to 10, 10, 4, and 10.

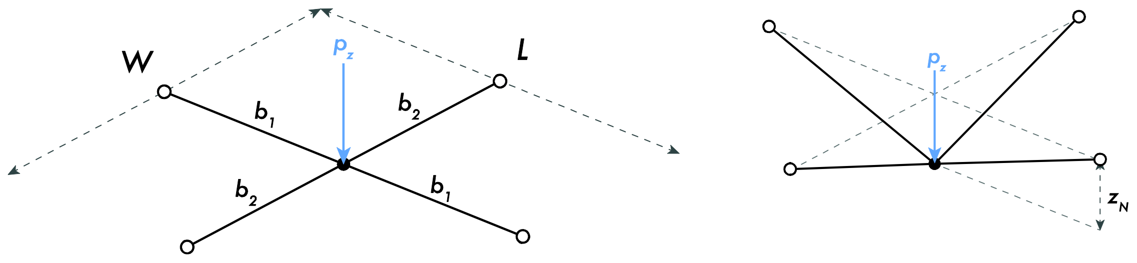


Figure 2 : The four-bar problem.

As seen in Figure 3 (a), the problem is nonconvex and has a line of global minima. The level sets of the objective function are lines with the following form:

$$F_\alpha(b_1, b_2) = \begin{cases} b_1 + b_2 = C_\alpha \\ z = \alpha \end{cases}, \tag{14}$$

where $C_\alpha$ is a constant depending on $\alpha$. From the form of the level sets, we see that the gradient always has the same direction (1,1); only its scale and sign will change. However, depending on the starting point, a computational optimization scheme based on the gradient will not necessarily converge. This is particularly clear when looking at the section of the objective surface by the plane $b_1 = b_2$. For a starting point $(b_0^1, b_0^2)$, a gradient-based algorithm will converge only if at all steps:

$$(b_k^1, b_k^2) \in \{(b_1, b_2) \in \mathbb{R}^2 | \, b_1 + b_2 > 0\} \tag{15}$$

A partial solution to the non-convexity issue is to reformulate our problem and constrain it to the positive orthant. This will solve the problem of the choice of the starting point. Moreover, in the four-bar example, the plane $b_1+b_2 = 0$ is particular because it corresponds to a state of physical instability: with these force densities, the bars cannot equilibrate the vertical force. In general, this is a problem that can arise if we allow for force densities of different signs. By restricting the problem to non-negative densities, the issue is simply avoided.

## 3.3. Structure of the constrained problem

We constrain the base set of problem $(P_1)$ to the non-negative orthant, to obtain $(P_2)$:

$$(P_2): \min_{b} \; F(\boldsymbol{b}) = \left\| \left( \boldsymbol{D}_N{}^{-1}(\boldsymbol{p}_z - \boldsymbol{D}_F \boldsymbol{z}_F) \right) - \boldsymbol{z}_T \right\|^2 \qquad \text{such that } \boldsymbol{b} \geq \boldsymbol{0} \tag{16}$$

Two details are of importance in the structure of $(P_2)$. First, the objective shape is in general not a funicular shape so the optimum value will be an unknown positive number. This means that solvers cannot be stopped based on the current function value being closed to zero, but only based on the improvement in the objective function or the size step.

Second, as mentioned in section 3.1.2, the Hessian matrix of $(P_1)$ has a rank deficiency of 1. The same goes for the matrices $\boldsymbol{C}$ and $\boldsymbol{A} = \text{abs}(\boldsymbol{C})$ of the absolute values of the components of $\boldsymbol{C}$. $\boldsymbol{A}$ represents the indices of the bars connected to each node, or equivalently the indices of the forces acting on each node, and so is locally representative of the structure of the possible equilibrium positions. Reordering the rows of $\boldsymbol{C}$, the structure of $\boldsymbol{A}$ can be written:

$$\boldsymbol{A} = \left( \begin{array}{cccccc} & \multicolumn{5}{c}{\overbrace{\hphantom{1 \quad 0 \quad 0 \quad \dots \quad 0}}^{n_x}} \\ & 1 & 0 & 0 & \dots & 0 \\ \boldsymbol{I}_{n_y} & \vdots & \vdots & \vdots & \dots & \vdots \\ & 1 & 0 & 0 & \dots & 0 \\ & 0 & 1 & 0 & \dots & 0 \\ \boldsymbol{I}_{n_y} & \vdots & \vdots & \vdots & \dots & \vdots \\ & 0 & 1 & 0 & \dots & 0 \\ \vdots & & & \vdots & & \\ & 0 & 0 & \dots & 0 & 1 \\ \boldsymbol{I}_{n_y} & \vdots & \vdots & \dots & \vdots & \vdots \\ & 0 & 0 & \dots & 0 & 1 \end{array} \right), \tag{17}$$

with $n_x$ and $n_y$ the number of independent force density values in the $x$ and $y$ directions, respectively, and supposing $n_x < n_y$. This matrix has a rank deficiency of 1, giving an indeterminacy in the FDM problem: a linear combination of the independent force densities will give the same result in the final shape. Given the structure of $A$, one could believe that those equivalent values are in a vector space of dimension 1; however, the positions of the nodes at equilibrium are related to $\boldsymbol{A}$ and $\boldsymbol{b}$ only after multiplications by the lengths of the bars to get to forces acting on the nodes. There is no clear structure to the set of equivalent independent force densities values.

This indeterminacy is detrimental to the quality of the optimization processes. At every point one direction of the problem will always be flat, so the solver will never search in that direction even though it might be on a shorter path to the optimum. Also, it means that solutions exist with unbalanced values of the independent force densities. This is also generally unwanted physically – it leads to high concentration of forces in certain bars.

One possible remedy is to remove the indeterminacy by adding the constraint $\sum_{i=1}^{nx} b_i = \sum_{i=1}^{nx} b_i$, following the structure of the 4-bar problem. While this works well in the unconstrained problem, it stops the algorithm too soon in the constrained problem. In fact, when the algorithm finds an optimum on the frontier of one of the inequality constraints and the equality constraint proposed above, it will not be able to get away from the barrier in the flat direction at that point to then follow a descent direction again. This constraint was not used in the rest of this paper except for part of section 4.2.

## 4. Results

### 4.1. Steepest Descent Algorithm (SDA)

In this section, we apply the steepest descent Algorithm 1 to the unconstrained problem. The gradient of the objective function with respect to the independent force densities was given by eq. (8). The gradient can be used to provide a descent direction; as a matter of fact, it provides the steepest descent direction. The step-size is determined by computing $\alpha^k := arg \min_{\alpha} F(b^k + \alpha\, d^k)$. Hence, in order to find to $\alpha^k$, one has to solve:

$$2\left(\left(D_N{}^{-1}C_N{}^T W\right)\big|_{b=b^k+\alpha d^k} * K * J\left(b^k\right)^T * (z_N - z_T)\big|_{b=b^k}\right)^T * (z_N - z_T)\big|_{b=b^k+\alpha d^k} = 0 \quad (18)$$

No closed-form solution exists in general and we must resort to an approximate line-search by bisection.

```
**Algorithm 1** Steepest Descent Algorithm (SDA)
Initialize at $b^0$, and set k←0
At iteration k:
$d^k := -\nabla F(b^k) = -2\,J(b^k)^T(z_N^k - z_T)$ . If $||d^k|| \le \epsilon$ , then stop ($\epsilon$ is specified tolerance)
Choose step-size $\alpha^k$ (by performing a line-search)
Set $x^{k+1} \leftarrow x^k + \alpha^k d^k, k \leftarrow k+1$
```

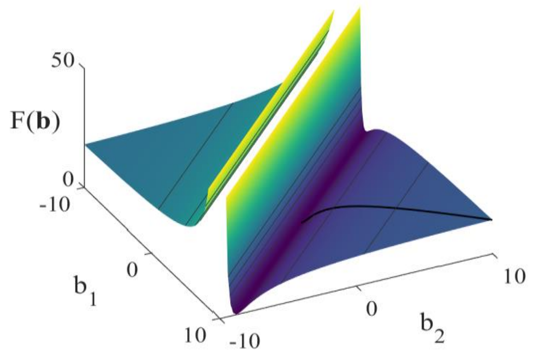
### *4.1.1. Example 1: 4-bar problem*

We use Algorithm 1 on the four-bar problem introduced in section 3.2. We apply the steepest descent algorithm with and without line-search. Our starting point is $\boldsymbol{b}_0 = (10{,}10)$.

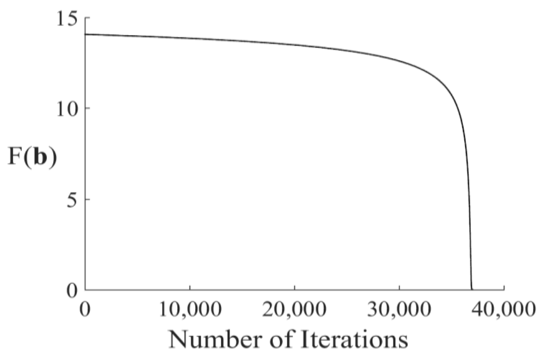
With the line-search, the algorithm converges in one step, which was expected given the structure of the problem. We note however that we had to modify the bisection algorithm as to avoid overshooting past the plane $b_1 + b_2 = 0$, i.e. we had to set an upper bound for the choice of the step-size. Because we knew the objective surface beforehand and knowingly chose an easy starting point, this constraint was easy to implement as follows:

$$\alpha^k \le \min\left(\left|\frac{b^k}{d^k}\right|\right), \quad (19)$$

where $\frac{b^k}{d^k}$ indicates a component-wise division. Generally, this constraint is not necessarily as easy to formulate. This problem is a motivation for constraining the problem to non-negative force densities. Without the line-search (arbitrary step-size of 0.001), the algorithm expectedly converges very slowly as seen in Figure 3 (b).



(a)



(b)

Figure 3 : (a) The objective surface with the optimization path (black curve) for the steepest descent algorithm with a fixed step-size of 0.001. At every iteration, the direction of the gradient remains unchanged because the level sets of the function are parallel lines. (b) Convergence profile of the steepest descent algorithm with a step-size of 0.001, for the 4-bar problem.

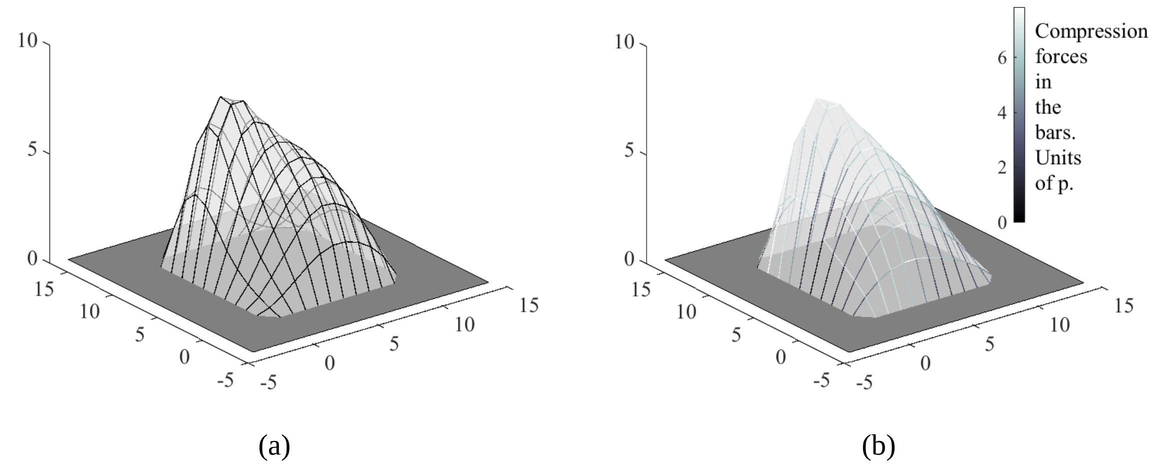


Figure 4: Target surface (a) vs. optimum surface found (b).

#### *4.1.2. Example 2: 10x10-bar problem*

In this example, we deal with a larger dimension problem, i.e. a grid network of size 10 by 10 nodes. The target surface and optimum found are presented in Figure 4. Applying the steepest descent algorithm with the starting point $\boldsymbol{b}^0 = [10 \ldots 10]^T$, where $\boldsymbol{b}^0$ has 20 components, the problem converges to the optimum solution ($F^* = 38.1539$) in 4213 iterations. Convergence is defined by a relative change in the objective value of less than $10^{-6}$. Figure 5 (a) shows that the algorithm converges quickly in the first 20 iterations but considerably slows downs afterwards. If instead, we use a starting point $\boldsymbol{b}^0 = [1 \ldots 1]^T$, the algorithm does not converge. A solution is to set up an arbitrarily low step-size, as in the previous example but it results in an exceptionally slow algorithm. Again, these observations motivate to switch to a constrained problem.

### 4.2. Quasi-Newton Methods

In this section, we focus on finding a performant algorithm for solving $(P_2)$ in the 100 nodes examples of section 4.1.2. Looking at the poor performance of the SDA in a realistic problem, we implemented a Newton Method (NM) with line-search, Algorithm 2. Because the method relies on inverting the Hessian, non-invertible as per section 3.3, it was necessary to enforce the additional constraint $\sum_{i=1}^{nx} b_i = \sum_{i=1}^{nx} b_i$. The constraint is written as an additional line in the Hessian and the gradient is augmented by one component equal to 0 to get a well-formed system of equations in the first step of Algorithm 2.

**Algorithm 2** Newton's method (NM)

```
Initialize at $b^0$, and set $k \leftarrow 0$. Let $\epsilon > 0$ be a given error tolerance.
At iteration $k$:
1. Set $H_{const}(b^k) = \begin{pmatrix} H(b^k) \\ \underbrace{1 \ \cdots \ 1}_{nx} \ \underbrace{-1 \ \cdots \ -1}_{ny} \end{pmatrix}$, $\nabla F_{const}(b^k) = \begin{pmatrix} \nabla F(b^k) \\ 0 \end{pmatrix}$.
2. $d^k := -H_{const}(b^k)^{-1} \nabla F_{const}(b^k)$. If $\|d^k\| \leq \epsilon$ , then stop.
3. Choose step-size $\alpha^k = \mathrm{argmin}_{a \geq 0} F(b^k + \alpha d^k)$.
4. Set $x^{k+1} = x^k + \alpha^k d^k, k = k + 1$
```

However, given the narrow zone in which the objective function of $(P_1)$ is convex, this method only converges with a starting point very close to the optimum. In practice, we were only able to obtain convergence by using a starting point found as a result of the SDA, stopped when the relative change in the objective value was $10^{-3}$.

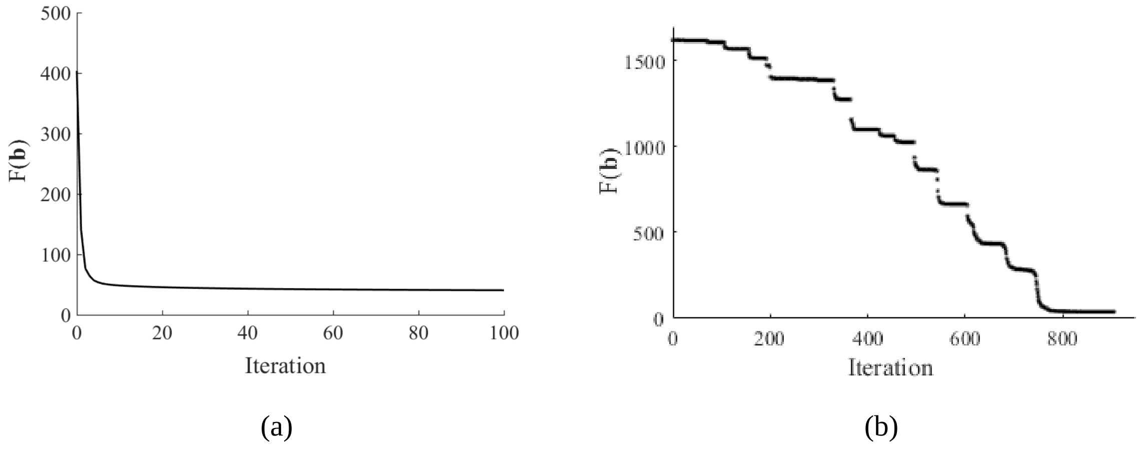


Figure 5: Convergence profile of (a) the steepest descent algorithm with an inexact line-search, and (b) of the interior-point method for the 10x10 bar problem.

### *4.2.1. Levenberg-Marquardt algorithm*

To have a better convergence rate for the final iterations along with a large convergence domain, we used the Levenberg-Marquardt algorithm (Levenberg [5], Marquardt [6]) implementation in MATLAB [7] for non-linear least-square problems. This is the method used in van Mele and Block [3].

The Levenberg-Marqurdt algorithm uses a search direction that is intermediate between the one found in the SDA and the one found in NM. A parameter is used to orient the amount by which the problem is similar to the SDA or the NM. At the beginning, this parameter is chosen so that the problem is very close to the SDA (to get a large convergence region) and is progressively updated to match more closely the NM (to get a faster convergence rate). Because the problem is never exactly the NM, the non-invertible Hessian is not a problem anymore and we can eliminate the additional constraint used above.

Using this method, we were able to find a solution with the same optimum value as in Section 4.1.2, in 1585 iterations using the same stopping criterion. However with several values of the force densities found are negative, which is unacceptable if we were to build the solution. In consequence, we look at constrained solvers to find solutions to $(P_2)$ in both reasonable time and large convergence.

## 4.3. Solvers for the constrained problem

Using the solvers implemented in Matlab's *fmincon* package, we looked at their relative performance for solving $(P_2)$ in the 100 nodes example presented in section 4.1.2. The results are compiled in Table 1. The solvers are always stopped when the relative improvement in objective value is less than $10^{-6}$. The gradient is given explicitly to the solvers. The Hessian is computed numerically by the solvers, as some of them do not accept a user-defined analytical expression for the Hessian.

Convergence was always obtained when the initial point was a vector of identical independent force densities within one order of magnitude (above or below) of the uniform load at the nodes. It should be noted that the excellent convergence rate of the Trust region reflective algorithm fades when the stopping criterion is made smaller.

For the interior point method, the solver quickly gets stuck against a bound and, because it relies heavily on the Hessian, does not explore in the flat direction of the problem. When the additional constraint $\sum_{i=1}^{nx} b_i = \sum_{i=1}^{nx} b_i$ is added, we get the convergence as for the other algorithms, this is the result presented in Table 1. This explains the different objective value found at the optimum. A typical evolution of the objective values using the interior-point method is depicted in Figure 5 (b) for a "bad" starting point.

Table 1: Performance comparison of constrained nonlinear solvers in the 100 nodes example.
(*An additional constraint was added to the interior point method.)

| **Algorithm** | **Iterations** | **Function calls** | **Objective value** |
|---|---|---|---|
| Interior point* | 887 | 1070 | 38.2510 |
| Sequential Quadrating Programming | 923 | 1431 | 38.2356 |
| Trust region reflective | 28 | 29 | 38.2356 |

## 5. Conclusions

In this paper, we analyzed the structure of a closest-fit inverse form-finding problem for funicular structure. Using the force density method framework, we optimized the force densities of a bar network to reach a target surface. We formulated the problem first as an unconstrained optimization problem and calculated the analytic expressions of the gradient and Hessian of the problem. Based on this, we applied the steepest descent algorithm on a 4-bar problem. The insight gained from this analysis motivated the introduction of a non-negative force densities constraint. The constraint was dealt with computationally by using nonlinear constrained optimization methods that proved effective, especially the interior point and trust region reflective methods. We note that, for particular surfaces, the constraint will yield solutions that are sub-optimal compared to solutions of the unconstrained problem. Yet, both for physical and for computational stability motives, the constraint introduced is useful.

Future work should be focused on handling realistic construction constraints, for example limits on the forces in the bars or smaller deviations of those forces. Looking at self-weight funicular structures, where the applied load is a function of the geometry of the shell, is also promising. Finally, the optimization strategy presented in this paper could be integrated in a user-guided design tool used by designers to directly create and interact with funicular structures.

## References

[1] Schek H.-J., "The force density method for form finding and computation of general networks," *Comput. Methods Appl. Mech. Eng.*, vol. 3, pp. 115–134, 1974.

[2] Block P. and Lachauer L., "Closest-Fit , Compression-Only Solutions for Freeform Shells," in *Proceedings of the IABSE-IASS Symposium 2011*, 2011.

[3] van Mele T. and Block P., "A novel form finding method for fabric formwork for concrete shells," *J. Int. Assoc. Shell Spat. Struct.*, vol. 52, no. 170, pp. 217–224, 2011.

[4] Petersen K. B. and Pedersen M. S., "The Matrix Cookbook," 2012.

[5] Levenberg K., "A Method for the Solution of Certain Problems in Least-Squares," *Q. Appl. Math. 2,* pp. 164–168, 1944.

[6] Marquardt D., "An Algorithm for Least-squares Estimation of Nonlinear Parameters," *SIAM J. Appl. Math.*, vol. 11, pp. 431–441, 1963.

[7] "MATLAB and Optimization Toolbox Release 2015b." The MathWorks, Inc., Natick, Massachusetts, United States.