CHAPTER THIRTEEN

Best-fit thrust network analysis

Rationalization of freeform meshes

Tom Van Mele, Daniele Panozzo, Olga Sorkine-Hornung and Philippe Block

LEARNING OBJECTIVES

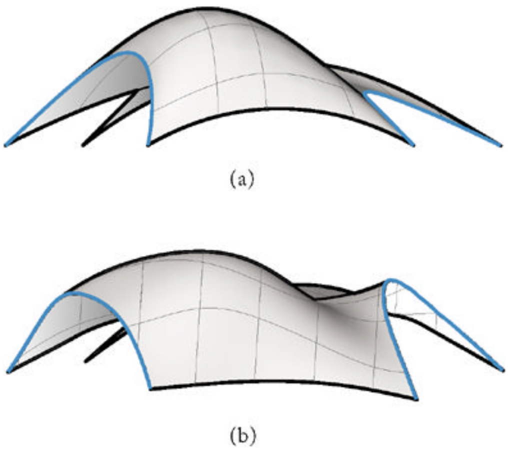
- Formulate the search for a funicular network in compression that is as close as possible, in a least-squares sense, to a given target surface as a series of optimization problems.
- Implement solvers for the different types of optimization problems.
- Implement this method to optimize freeform shapes; and apply this method to evaluate the stability of structures under asymmetrical loading.

PREREQUISITES

- Chapter 7 on thrust network analysis.

Chapter 7 introduced Thrust Network Analysis (TNA) as a method for designing three-dimensional, compression-only equilibrium networks (thrust networks) for vertical loads using planar, reciprocal form and force diagrams. These diagrams allow the high degree of indeterminacy of three-dimensional force networks to be controlled such that possible funicular solutions for a set of loads can be explored. By manipulating the force diagram (through simple geometric operations), the distribution of horizontal thrusts throughout the network is changed and different three-dimensional configurations are obtained.

There are infinitely many possible variations of the force diagram, each corresponding to a different three-dimensional solution for given loads and boundary conditions. This provides virtually limitless freedom in


(a)

(b)

Figure 13.1 (a) The compression-only design for the pavilion as form found in Chapter 7, and (b) the new (geometrical) proposal the design of three-dimensional equilibrium networks, but it makes it almost impossible to find the specific distribution of forces that corresponds to a specific solution, with a specific shape. For example, the required distribution of forces to achieve the upwards flaring edge of the design proposal depicted in Figure 13.1b is not obvious, and finding it is by no means straightforward.

Therefore, in this chapter, we describe how TNA can be extended to find a thrust network that for a given set of loads best fits a specific target shape. We set this up as an optimization problem and discuss the implementation of an efficient solving strategy.

The brief

Chapter 7 described the design of a vaulted, unreinforced cut-stone masonry pavilion for a park in Austin, Texas, USA, that covers the stage and spectator area of a small performance area of 20m × 15m. The client has requested modifications to the shape developed in Chapter 7 to improve the integration of the pavilion into the surrounding landscape and allow access to its top surface to provide visitors with alternative views of the site and the vault. Although the dramatic asymmetry between the two sides of the vault is a key feature of the form, the client would prefer a deeper opening on the side of the shallow main arch to let in more light and make that side of the pavilion look more open and inviting.

The original design and the new proposal are shown in Figure 13.1. Key features of the new design are thus the smoother transition between the landscape and the structure on one side, and the flaring edge on the other.

We have been asked to determine whether the new, geometrically constructed shape is feasible for an unreinforced, masonry stone structure.

13.1 TNA preliminaries

Since this chapter describes an extension of TNA, we assume the reader to be familiar with its fundamental principles as presented in Chapter 7. Here, we briefly summarize those mathematical elements, notations and conventions of TNA that are required for the optimization algorithm.

Let Γ and Γ^* be two planar graphs with an equal number of edges, m. If Γ is a *proper cell decomposition of the plane*, and Γ^* is its *convex, parallel dual*, then Γ and Γ^* are the form and force diagram of a (three-dimensional) thrust network $\mathbf{G}$ that is in equilibrium with a set of vertical loads applied to its nodes, and has Γ as its horizontal projection. Two graphs are parallel if all corresponding edges are parallel, and convex if all their faces are convex. We call two graphs or diagrams reciprocal if one is the parallel dual of the other. The force diagram of a thrust network $\mathbf{G}$ is thus the convex reciprocal of the form diagram of $\mathbf{G}$. A proper cell decomposition of the plane divides the plane into cells formed by (unbounded) convex polygons such that:

- every point in the plane belongs to at least one cell;
- the cells have disjoint (i.e. non-overlapping) interiors;
- any two cells are separated by exactly one edge.

We can describe Γ as a pair of matrices $\mathbf{V}$ and $\mathbf{C}$. $\mathbf{V} = [\mathbf{x}|\mathbf{y}]$ is an $n \times 2$ matrix, which contains the coordinates in the horizontal plane of the i-th node in its i-th row. n is the number of nodes in Γ. $\mathbf{C}$ is the branch-node matrix: an $m \times n$ matrix that contains the connectivity information of the graph of Γ (see Section 7.3.2). Note that $\mathbf{C}$ is the transpose of the incidence matrix of Γ. The edges of Γ, represented as vectors, can be extracted from $\mathbf{V}$ and $\mathbf{C}$ by computing the $m \times 2$ matrix $\mathbf{E} = \mathbf{CV} = [\mathbf{u}|\mathbf{v}]$, which contains the coordinate differences of the i-th branch in its i-th row. Therefore, the length of the i-th edge, l_i, can be computed by taking the norm of the i-th row of $\mathbf{E}$. $\mathbf{L}$ is the $m \times m$ diagonal matrix of the vector of edge lengths $\mathbf{l}$. $\mathbf{V}^*$, $\mathbf{C}^*$, $\mathbf{E}^*$, $\mathbf{L}^*$ are defined equivalently for the reciprocal diagram.

The force densities $\mathbf{q}$ of the network are the ratios of the lengths of corresponding edges of Γ^* and Γ:

$$\mathbf{Q} = \mathbf{L}^{-1}\mathbf{L}^*, \tag{13.1}$$

with $\mathbf{Q}$ the diagonal $m \times m$ matrix of $\mathbf{q}$.

The nodes of Γ are divided into two sets, N and F, that denote the (non-fixed) free nodes and the (fixed) support nodes, respectively. The heights of the free nodes of the thrust network $\mathbf{G}$ described by Γ and Γ^* are computed as:

$$\mathbf{z}_{\mathrm{N}} = \mathbf{D}_{\mathrm{N}}^{-1}(\mathbf{p} - \mathbf{D}_{\mathrm{F}}\,\mathbf{z}_{\mathrm{F}}), \qquad (13.2)$$

with $\mathbf{D}_{\mathrm{N}}$ and $\mathbf{D}_{\mathrm{F}}$ the columns of $\mathbf{D} = \mathbf{C}_{\mathrm{N}}^{\mathrm{T}}\mathbf{QC}$ corresponding to the n_{N} free and n_{F} fixed nodes, respectively. $\mathbf{p}$ is the vector of external loads applied at the free nodes and $\mathbf{z}_{\mathrm{F}}$ are the heights of the fixed support nodes.

13.2 Formulation of the problem

Let $\mathbf{G}$ be a thrust network generated from a pair of reciprocal diagrams Γ and Γ^*, and S a target surface. Keeping the form diagram Γ fixed, our objective is to optimize the force diagram Γ^* such that the network $\mathbf{G}$ is as close as possible, in the least-squares sense, to the target S. The variables we are optimizing for are the nodes $\mathbf{V}^*$ of the force diagram. Formally:

$$\underset{\mathbf{V}^*}{\operatorname{argmin}} \sum_i (z_i - s_i)^2 \qquad (13.3)$$

$$\text{subject to } \Gamma^* \text{ is the convex reciprocal of } \Gamma, \quad (13.4)$$

where i runs over the nodes of Γ and z_i and s_i are, respectively, the height of the network and the surface at the i-th node.

Note that the heights z_i do not directly depend on the variables $\mathbf{V}^*$. However, we can compute the heights z_i from the force densities $\mathbf{q}$ using equation (13.2). The energy is thus a function of $\mathbf{q}$:

$$f(\mathbf{q}) = (\mathbf{z}_{\mathrm{N}}(\mathbf{q}) - \mathbf{s}_{\mathrm{N}})^2. \qquad (13.5)$$

Therefore, to find the best-fit solution, we must search for the force densities $\mathbf{q}$ that minimize the energy according to equation (13.5) and allow for a force diagram Γ^* that satisfies constraint, expressed in equation (13.4):

$$\underset{\mathbf{q}}{\operatorname{argmin}}\ (\mathbf{z}_{\mathrm{N}}(\mathbf{q}) - \mathbf{s}_{\mathrm{N}})^2 \qquad (13.6)$$

$$\text{subject to } \Gamma^* \text{ is the convex reciprocal of } \Gamma.$$

In the following sections, we describe the strategy for solving this problem.

13.3 Overview

Starting from a given target surface S, the solving procedure consists of two main steps.

1. Generate a starting point:
   a. choose a form diagram;
   b. generate an initial force diagram;
   c. optimize the scale of the initial force diagram.
2. Find a best-fit solution by repeating the following two-step procedure until convergence:
   a. find the force densities $\mathbf{q}$ that minimize energy according to equation (13.6), ignoring the equilibrium constraints;
   b. for the current force densities, find the force diagram Γ^* that is as parallel as possible to the form diagram.

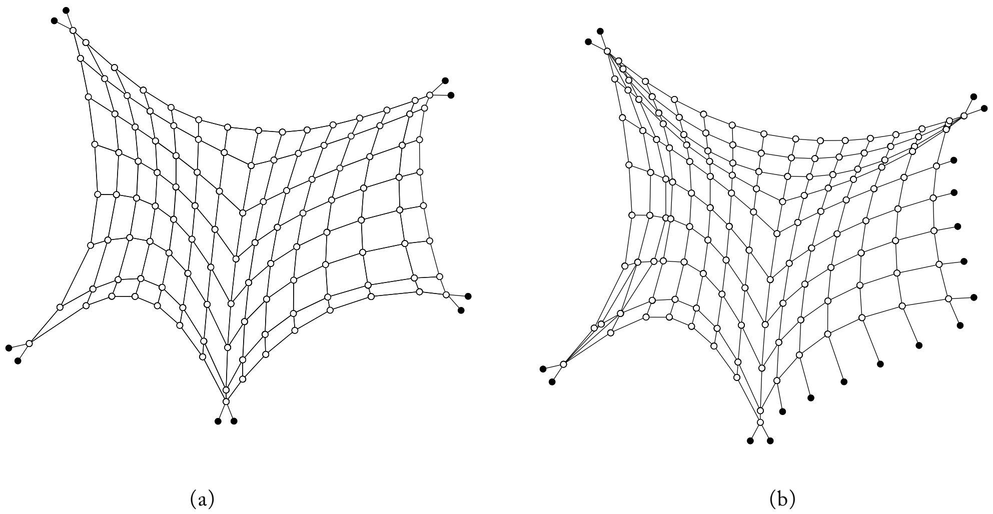
We discuss each of the steps and substeps in detail in the following sections.

13.4 Generate a starting point

In this section, we discuss the generation of a starting point for the iterative part of the optimization process. First, we choose a form diagram and generate an initial force diagram, and then optimize the scale of this force diagram.

13.4.1 The form diagram

In order to be able to obtain a well-fitting thrust network for a given target, a force diagram must be chosen that is based on the target's features. Our choice of form diagram for the target surface described in the brief is depicted in Figure 13.2b. Note that in comparison with the original diagram of Chapter 7, we have added force paths that gradually divert horizontal forces to the supports before they hit the open edges. This provides finer control over the equilibrium of these edges and will, for example, allow the upward flaring edge to develop.



(a)

(b)

Figure 13.2 (a) The form diagram of the original design, and (b) the modified form diagram used here.

13.4.2 An initial force diagram

To generate an initial force diagram, that is, a convex reciprocal of the form diagram, we use an iterative procedure. We start with the *centroidal dual* of the form diagram, rotated 90° as depicted in Figure 13.3a. The form diagram's centroidal dual is the dual of which the vertices or nodes coincide with the centroids of the faces of the form diagram. The corresponding edges of the form diagram and this rotated dual are generally not parallel. Therefore, at each iteration of the procedure we perform the following calculations. First, we compute a set of target directions $\mathbf{t}_i$ for the edges of the new force diagram by averaging the directions of the (fixed) form diagram and the current force diagram:

$$\mathbf{t}_i^* = \left(\frac{\mathbf{e}_i}{l_i} + \frac{\mathbf{e}_i^*}{l_i^*}\right) \Big/ \left|\frac{\mathbf{e}_i}{l_i} + \frac{\mathbf{e}_i^*}{l_i^*}\right|, \qquad (13.7)$$

with $\mathbf{e}_i$ the i-th row of $\mathbf{E}$, representing the i-th edge of the form diagram, and l_i its length; and, similarly, $\mathbf{e}_i^*$ the i-th row of $\mathbf{E}^*$, representing the i-th edge of the current force diagram, and l_i^* its length. Note that $\mathbf{e}_i/l_i$ is constant, since the form diagram is fixed, and thus does not need to be recalculated at each iteration. Using these target vectors, the edges of the new, 'ideal' (i.e. parallel) force diagram are thus:

$$\mathbf{e}_i^* = l_i^* \mathbf{t}_i^*. \qquad (13.8)$$

Note that this new diagram cannot be properly connected, since its edges have the same lengths but different directions than before. Therefore, we search for a diagram that is similar to the ideal one, but connected, by solving the following minimization problem:

$$\underset{\mathbf{V}^*}{\operatorname{argmin}} \sum (\mathbf{e}_i^* - l_i^* \mathbf{t}_i^*)^2 \qquad (13.9)$$

$$\text{subject to } \mathbf{V}_0^* = \mathbf{0} \qquad (13.10)$$

Note that without the constraint in equation (13.10), there are infinite graphs that minimize energy in equation (13.9), all identical up to a translation. Fixing a single node ($\mathbf{V}_0^*$) to an arbitrary value makes the solution unique.

We repeat these steps until a convex reciprocal of the form diagram is found. The centroidal dual of the form diagram and the initial force diagram derived from it are depicted in Figure 13.3.

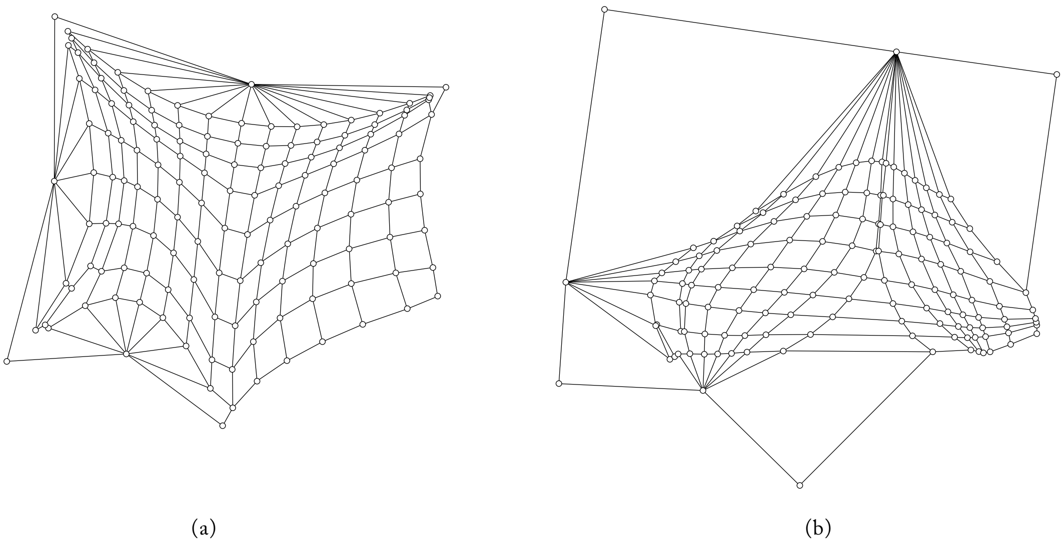


Figure 13.3 (a) The centroidal dual of the form diagram, and (b) an initial force diagram, based on the centroidal dual

13.4.3 Scale optimization

In TNA, we can change the depth of a funicular network simply by uniformly scaling all horizontal forces, which is equivalent to uniformly scaling the force diagram. Higher and lower thrusts result in shallower and deeper solutions, respectively. Therefore, before starting the optimization process, we can reduce energy according to equation (13.5), without changing the distribution of forces, simply by changing the scale of the force diagram. The optimal scaling factor r is obtained by minimizing:

$$\underset{r}{\operatorname{argmin}}\ (\mathbf{z} - \mathbf{s})^2 \qquad (13.11)$$

$$\text{subject to } \mathbf{Dz} - r\mathbf{p} = \mathbf{0}, \qquad (13.12)$$

this is a linear least-squares problem subject to linear equality constraints and can solved using the method of Lagrange multipliers. We rewrite the problem introducing additional variables, one for every equality constraint, obtaining the following Lagrange function:

$$\begin{aligned}\Lambda(\mathbf{z}, r, \boldsymbol{\lambda}) &= (\mathbf{z} - \mathbf{s})^2 + \boldsymbol{\lambda}^{\mathrm{T}}(\mathbf{Dz} - r\mathbf{p}) \\ &= \mathbf{z}^{\mathrm{T}}\mathbf{z} - 2\mathbf{z}^{\mathrm{T}}\mathbf{s} - \mathbf{s}^{\mathrm{T}}\mathbf{s} + \boldsymbol{\lambda}^{\mathrm{T}}(\mathbf{Dz} - r\mathbf{p}) \qquad (13.13)\end{aligned}$$

with $\boldsymbol{\lambda}$ the Lagrange multipliers. The unique minimum of the Lagrange function is the solution we are looking for. Setting the partial derivatives of Λ equal to zero leads to the following linear system, the solution of which is the scaled thrust network and the scaling factor r:

$$\begin{bmatrix} 2 & \mathbf{0} & \mathbf{D}^{\mathrm{T}} \\ \mathbf{0} & -\boldsymbol{\lambda}^{\mathrm{T}} & \mathbf{0} \\ \mathbf{D} & -\mathbf{p} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{z} \\ r \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} 2\mathbf{s} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \qquad (13.14)$$

Figure 13.4 shows the scaled force diagram and corresponding thrust network in comparison with the target surface.

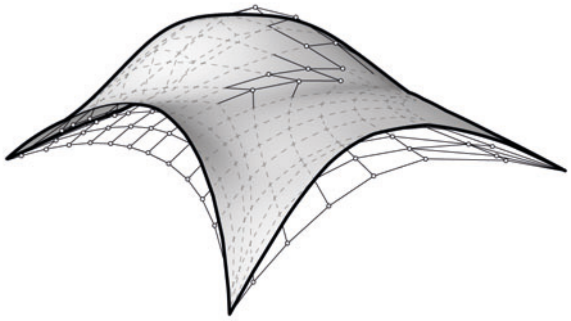


Figure 13.4 By uniformly scaling the force diagram we obtain a funicular network that is closer to the target surface

13.5 Iterative procedure

In the previous section, we have generated an initial pair of reciprocal diagrams, and rescaled the force diagram such that the corresponding thrust network is, for that distribution of thrusts, as close as possible to the target. Rescaling the force diagram has changed the depth of the funicular solution, but the overall shape has stayed the same, because the proportional distribution of thrusts remained unchanged.

During the following iterative procedure, we redistribute the thrust forces and thus change the shape of the thrust network until a better fit of the target is found. Each iteration of this procedure consists of two steps. In the first step, we optimize the force densities without taking into account the reciprocity constraint in equation (13.6) on the force diagram. In the second step of each iteration, we search for a force diagram that generates these optimized force densities and is as parallel as possible to the form diagram. We repeat these steps until a solution is found with optimal force densities and parallel diagrams.

13.5.1 Force densities optimization

To optimize the force densities, we minimize energy according to equation (13.5) using a gradient descent algorithm (Nocedal and Wright, 2000). In short, this means that we move from the current force densities to the next using

$$\mathbf{q}^{t+1} = \mathbf{q}^{t} - \lambda \nabla f(\mathbf{q}^{t}) \tag{13.15}$$

with $\nabla f(\mathbf{q})$ the direction of maximum increase or decrease of f at $\mathbf{q}$ (i.e. the gradient) and λ a step length that satisfies the strong Wolfe conditions (Nocedal and Wright, 2000).

The gradient of f can be efficiently evaluated in closed form:

$$\frac{\partial f(\mathbf{q})}{\partial \mathbf{q}} = \frac{\partial}{\partial \mathbf{q}}\left((\mathbf{Z}_{\mathrm{N}} - \mathbf{S}_{\mathrm{N}})^{2}\right) = 2(\mathbf{Z}_{\mathrm{N}} - \mathbf{S}_{\mathrm{N}})\frac{\partial \mathbf{z}_{\mathrm{N}}}{\partial \mathbf{q}}, \tag{13.16}$$

where $\mathbf{Z}_{\mathrm{N}}$ and $\mathbf{S}_{\mathrm{N}}$ are diagonal matrices corresponding to $\mathbf{z}_{\mathrm{N}}$ and $\mathbf{s}_{\mathrm{N}}$ respectively.

Using equation (13.2), the gradient of $\mathbf{z}_{\mathrm{N}}$ can be written as

$$\begin{aligned}\frac{\partial \mathbf{z}_{\mathrm{N}}}{\partial \mathbf{q}} &= \frac{\partial\left(\mathbf{D}_{\mathrm{N}}^{-1}(\mathbf{p} - \mathbf{D}_{\mathrm{F}}\mathbf{z}_{\mathrm{F}})\right)}{\partial \mathbf{q}} \\ &= \frac{\partial \mathbf{D}_{\mathrm{N}}^{-1}}{\partial \mathbf{q}}(\mathbf{p} - \mathbf{D}_{\mathrm{F}}\mathbf{z}_{\mathrm{F}}) - \mathbf{D}_{\mathrm{N}}^{-1}\frac{\partial(\mathbf{p} - \mathbf{D}_{\mathrm{F}}\mathbf{z}_{\mathrm{F}})}{\partial \mathbf{q}} \\ &= \frac{\partial \mathbf{D}_{\mathrm{N}}^{-1}}{\partial \mathbf{q}}(\mathbf{p} - \mathbf{D}_{\mathrm{F}}\mathbf{z}_{\mathrm{F}}) + \mathbf{D}_{\mathrm{N}}^{-1}\left(\mathbf{C}_{\mathrm{N}}^{\mathrm{T}}\mathbf{C}\begin{bmatrix}\mathbf{0}\\ \mathbf{z}_{\mathrm{F}}\end{bmatrix}\right),\end{aligned} \tag{13.17}$$

where we used

$$\mathbf{D}_{\mathrm{F}}\mathbf{z}_{\mathrm{F}} = \mathbf{C}_{\mathrm{N}}^{\mathrm{T}}\mathbf{Q}\mathbf{C}\begin{bmatrix}\mathbf{0}\\ z_{\mathrm{F}}\end{bmatrix}. \tag{13.18}$$

Finally, $\partial \mathbf{D}_{\mathrm{N}}^{-1}/\partial \mathbf{q}$ can be rewritten using the identity (Petersen and Pedersen, 2008)

$$\frac{\partial \mathbf{A}^{-1}}{\partial \mathrm{x}} = -\mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial x}\mathbf{A}^{-1}. \tag{13.19}$$

Applied to $\mathbf{D}_{\mathrm{N}}^{-1}$ this gives

$$\begin{aligned}\frac{\partial \mathbf{D}_{\mathrm{N}}^{-1}}{\partial \mathbf{q}} &= -\mathbf{D}_{\mathrm{N}}^{-1}\frac{\partial\left(\mathbf{C}_{\mathrm{N}}^{\mathrm{T}}\mathbf{Q}\mathbf{C}\begin{bmatrix}\mathbf{I}\\ \mathbf{0}\end{bmatrix}\right)}{\partial \mathbf{q}}\mathbf{D}_{\mathrm{N}}^{-1} \\ &= -\mathbf{D}_{\mathrm{N}}^{-1}\mathbf{C}_{\mathrm{N}}^{\mathrm{T}}\mathbf{C}\begin{bmatrix}\mathbf{I}\\ \mathbf{0}\end{bmatrix}\mathbf{D}_{\mathrm{N}}^{-1},\end{aligned} \tag{13.20}$$

where we used

$$\mathbf{D}_{\mathrm{N}} = \mathbf{C}_{\mathrm{N}}^{\mathrm{T}}\mathbf{Q}\mathbf{C}\begin{bmatrix}\mathbf{I}\\ \mathbf{0}\end{bmatrix}. \tag{13.21}$$

Substituting equations 13.20 and 13.17 in 13.16, and using equation 13.2, we get

$$\begin{aligned}\nabla f(\mathbf{q}) &= -2\,(\mathbf{z}_{\mathrm{N}} - \mathbf{s})^{\mathrm{T}} \\ &\left(\mathbf{D}_{\mathrm{N}}^{-1}\mathbf{C}_{\mathrm{N}}^{\mathrm{T}}\mathbf{C}\begin{bmatrix}\mathbf{I}\\ \mathbf{0}\end{bmatrix}\mathbf{z}_{\mathrm{N}} + \mathbf{D}_{\mathrm{N}}^{-1}\mathbf{C}_{\mathrm{N}}^{\mathrm{T}}\mathbf{C}\begin{bmatrix}\mathbf{0}\\ \mathbf{z}_{\mathrm{F}}\end{bmatrix}\right).\end{aligned} \tag{13.22}$$

This gives the final expression of $\nabla f(\mathbf{q})$,

$$\nabla f(\mathbf{q}) = 2(\mathbf{Z}_{\mathrm{N}} - \mathbf{S}_{\mathrm{N}})\mathbf{D}_{\mathrm{N}}^{-1}\mathbf{C}_{\mathrm{N}}^{\mathrm{T}}\mathbf{C}\mathbf{z}. \tag{13.23}$$

In the evaluation of $\nabla f(\mathbf{q})$, we need to compute $\mathbf{D}_{\mathrm{N}}^{-1}$. To avoid computing the dense inverse explicitly, we can compute $\mathbf{D}_{\mathrm{N}}^{-1}\mathbf{C}_{\mathrm{N}}^{\mathrm{T}}\mathbf{C}\mathbf{z}$ indirectly by solving the equivalent sparse linear system:

$$\mathbf{D}_{\mathrm{N}}\mathbf{x} = \mathbf{C}_{\mathrm{N}}^{\mathrm{T}}\mathbf{C}\mathbf{z}. \tag{13.24}$$

Since $\mathbf{D}_{\mathrm{N}}$ is symmetric and positive definite, we can

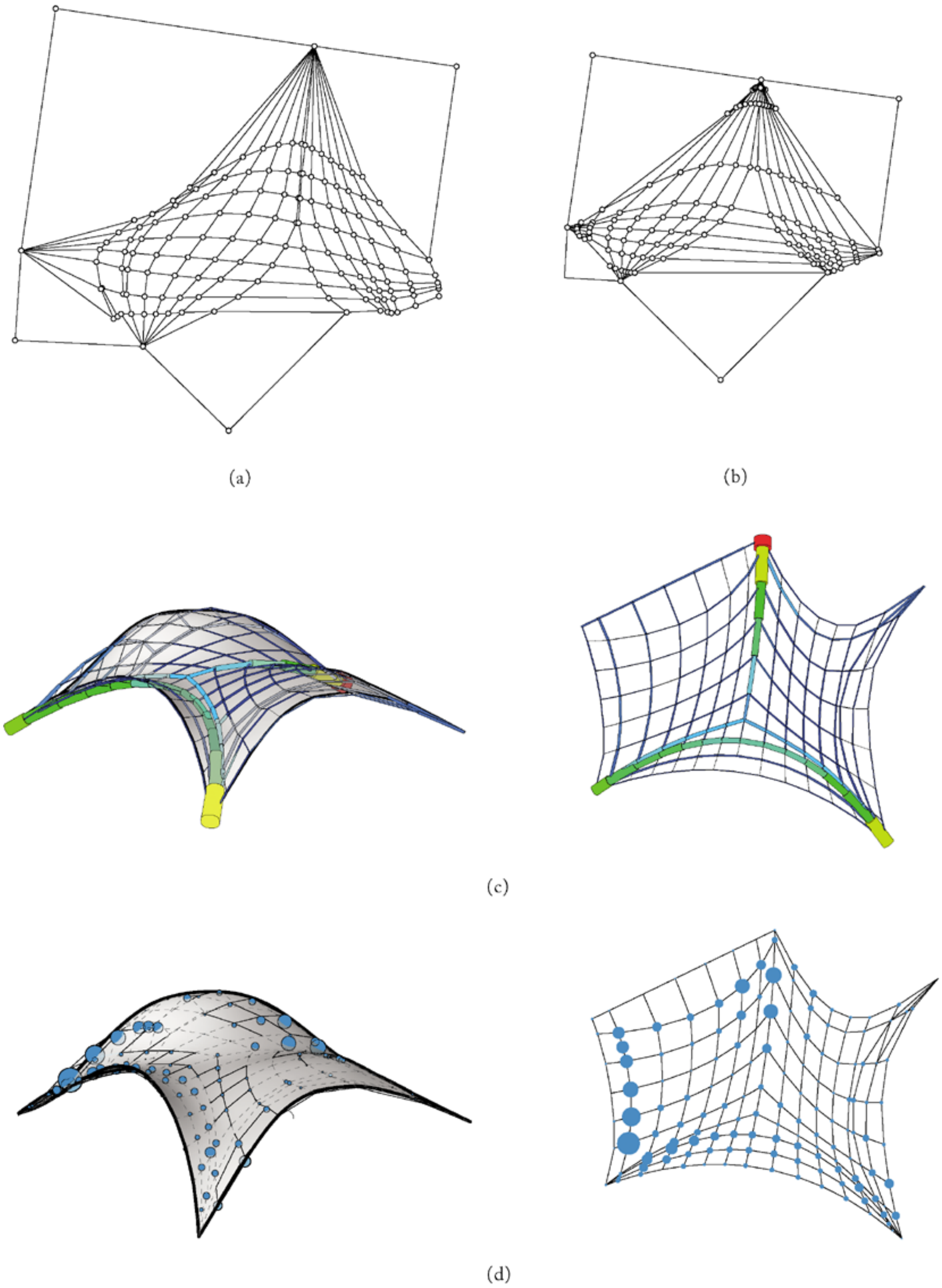


Figure 13.5 Result of the optimization process: the best-fit funicular network for the given target surface and loads

efficiently solve the system using the sparse Cholesky decomposition (Nocedal and Wright, 2000). We first compute the Cholesky decomposition of $\mathbf{D}_N$:

$$\mathbf{D}_N = \mathbf{L}\mathbf{L}^T \tag{13.25}$$

with $\mathbf{L}$ a lower triangular matrix. Then, we solve the system of equations

$$\mathbf{L}\mathbf{y} = \mathbf{C}_N^T\mathbf{C}\mathbf{z} \tag{13.26}$$

for $\mathbf{y}$. This is done using forward substitution, since $\mathbf{L}$ is lower triangular. Finally, we find $\mathbf{x}$ by solving:

$$\mathbf{L}^T\mathbf{x} = \mathbf{y}. \tag{13.27}$$

13.5.2 Force diagram optimization

Given the current optimized force densities $\mathbf{q}$, we search for the force diagram Γ^* that is as parallel as possible to the form diagram while generating these force densities.

This procedure is similar to the one discussed in Section 13.4. We first compute a set of target directions for the edges of Γ^* using equation 13.7. Then, we generate target lengths for the edges of Γ^* using equation (13.1),

$$l_i^* = q_i l_i \tag{13.28}$$

Now we know the directions and lengths of the edges of the ideal Γ^* that generates the current force densities and is parallel to the form diagram. As before, this graph will generally not be connected. To compute a graph that is similar to the ideal one, but connected, we solve the same minimization problem as in equation (13.9).

The final result of the optimization process is shown in Figure 13.5. The figure depicts the scaled reciprocal diagram (Fig. 13.5a), which was the starting point for the optimization, and the final, optimized diagram (Fig. 13.5b). The thicknesses of the branches (Fig. 13.5c) visualize the distribution of forces in the thrust network, and the spheres (Fig. 13.5d) represent the deviation from the target surface.

13.6 Basic coding

Figure 13.6 is a flowchart that gives an overview of a complete implementation of the algorithm discussed in the previous section.

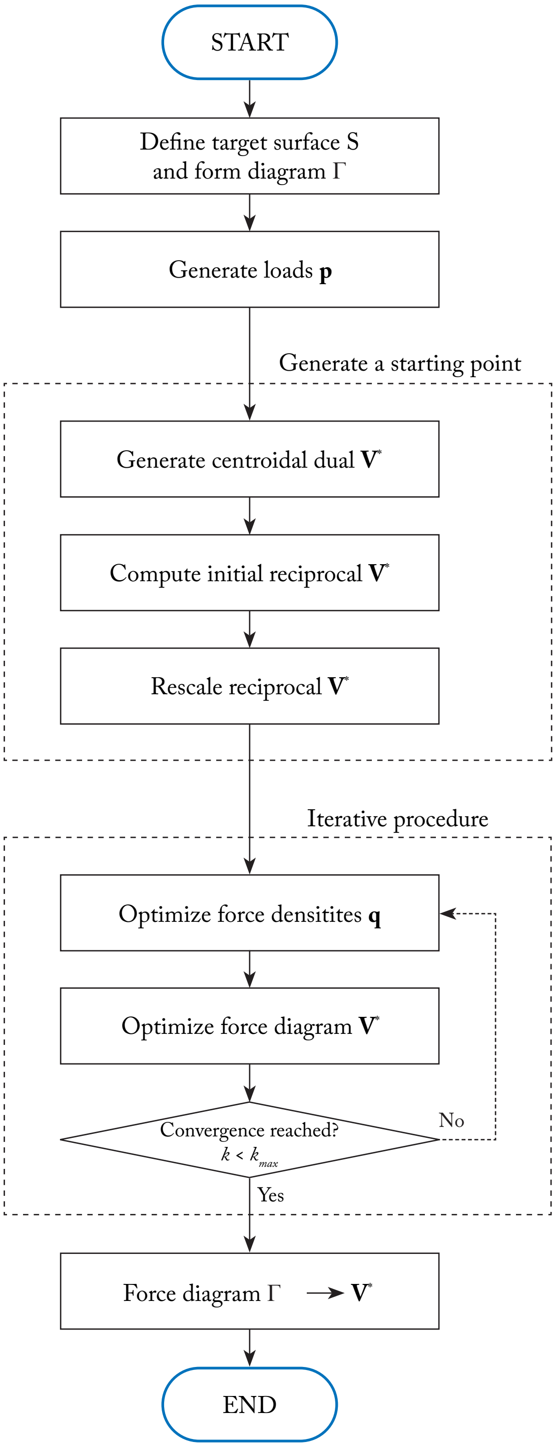

START
Define target surface S and form diagram Γ
Generate loads **p**
Generate a starting point
Generate centroidal dual **V***
Compute initial reciprocal **V***
Rescale reciprocal **V***
Iterative procedure
Optimize force densitites **q**
Optimize force diagram **V***
Convergence reached? k < k_max
No
Yes
Force diagram Γ → **V***
END

Figure 13.6 Flowchart of a complete implementation

13.7 Assessment of the proposed design

A masonry structure is considered safe if a network of compressive forces contained within (the middle third or kern of) the vault's geometry can be found for all possible loading cases (see Chapter 7).

For most masonry structures, the dominant loading case governing their design is self-weight. Therefore, to evaluate the feasibility of the proposed design, we first use the algorithm described in Section 13.5 to find the best-fit thrust network for the self-weight of the design, offset the solution with the thickness used to calculate the self-weight, and then use the algorithm to search for thrust networks contained within the kern of the new geometry for other loading cases.

13.7.1 Self-weight

We can calculate the weight per square metre of the proposed design using a chosen thickness and the weight of the stone: 0.3m × 2,400kgm^{-3} = 720kgm^{-2}. The equivalent distribution of point loads on the nodes of the form diagram according to their respective tributary areas on the target is depicted in Figure 13.7. Note that the compression-only solution captures the design intent of the client very well and allows for the realization of the key features of the shape.

13.7.2 Additional live loads

For the further evaluation of all additional load cases, we define the geometry of the vault by taking the best-fit thrust network determined in the previous step and setting it off by 0.15m on both sides (Fig. 13.8). As discussed, the structure can be considered safe if we can find a thrust network within the kern of its geometry for all additional loading cases (see Section 7.1.3).

In a real project, there are many different, additional loading cases and they should all be considered. However, here, we only discuss the case resulting from the allowed public access to the pavilion's surface.

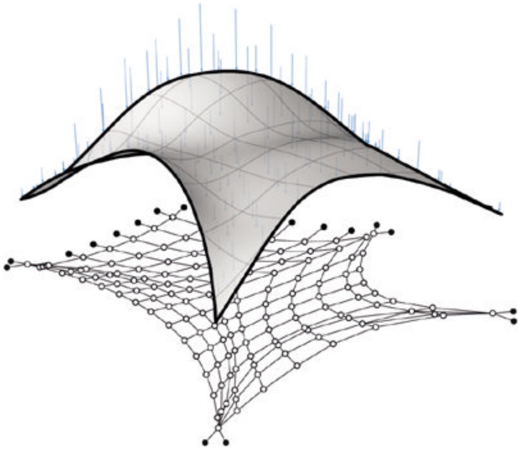


Figure 13.7 The self-weight of the structure distributed over the nodes according to their tributary areas

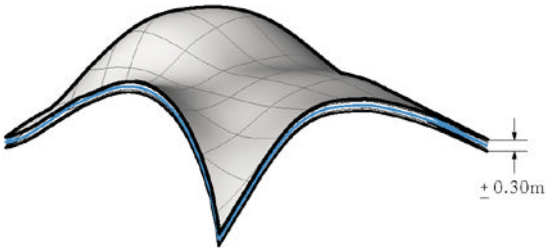


Figure 13.8 The new shell defined as an offset from the best-fit thrust network (blue) for the structure's self-weight

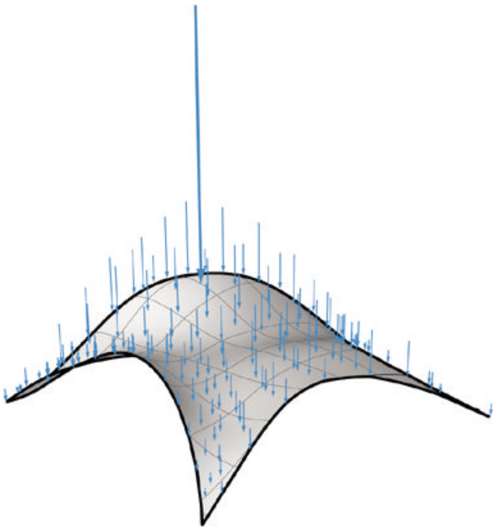


Figure 13.9 The self-weight of the vault combined with a hugely exaggerated additional point load

Typical values are 5.0kNm^{-2} for patch and 7.0kN for point loads. In this example, a point load is applied, because it has a more noticeable effect on the vault. Furthermore, a much higher point load of 100kN is used, to further emphasize the effect. Note that this is roughly equivalent to Godzilla standing on one leg on the viewing platform. The location of the additional load is depicted in Figure 13.9.

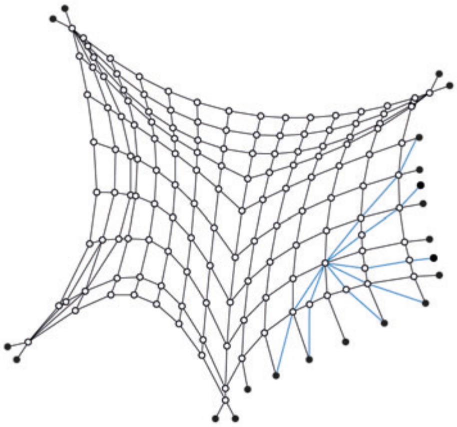


Figure 13.10 To find a best-fit funicular network for the combination of self-weight and additional live load(s) we draw a new form diagram that provides appropriate force paths

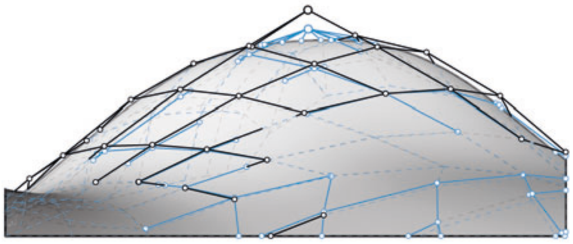


Figure 13.11 Comparison of the best-fit thrust network corresponding to the original force pattern (blue) and to the modified force pattern (black). The modified pattern clearly produces a much better fitting result

In order to find a compression-only force network that fits within the newly determined kern of the vault, we simply run the algorithm as before using point loads that represent the combination of self-weight and the additional loading.

However, as before, it is important that we start with a form diagram that provides force paths along which the loads can 'flow' to the supports. Therefore, we superimpose a force pattern on the previous form diagram that radiates from the point of application of the additional load to the supports (blue in Figure 13.10).

With this new form diagram and combined loads (self-weight and point load), we repeat the best-fit search to find the best-fit funicular network to the target surface. The result of this search is depicted in Figure 13.11. Note that, for such an extreme loading case, it is sufficient that the thrust network stays within the entire section of the vault (not just the middle third), although this would represent an equilibrium state at the onset of collapse. If such a solution cannot be found, the vault's thickness should be modified; for example, by iteratively searching for the bounding box of all loading cases.

13.8 Conclusion

This chapter has shown how to find a thrust network that best fits a given target surface for a given set of loads, formulate this search as a series of optimization problems, and use appropriate and efficient solving strategies for each of them.

The presented technique was applied to the assessment of the structural feasibility of a vaulted masonry structure with a complex, geometrically designed shape (Fig. 13.4). This entailed the search for a best-fit thrust network for the dominant loading case of self-weight, the derivation of a new geometry from this result, and the assessment of the safety of the new geometry in all other loading cases.

Another important application of the technique described in this chapter is the equilibrium analysis of historic masonry vaults with complex geometry, such as the sophisticated nave vaults of the Church of Santa Maria of Bélem at the Jerónimos monastery, completed in the early sixteenth century, shown on page 156. The approach to such an analysis is very similar to the previously discussed assessment of a design proposal. Provided that sufficient information about the geometry of the structure in its current state is available, the target surface can be taken as the surface that lies at the middle of the structure's section, and an appropriate form diagram can be derived from the structure's rib pattern and stereotomy. Otherwise, the procedure is exactly the same. The results for the nave vaults of the Jerónimos monastery are depicted in Figure 13.12.

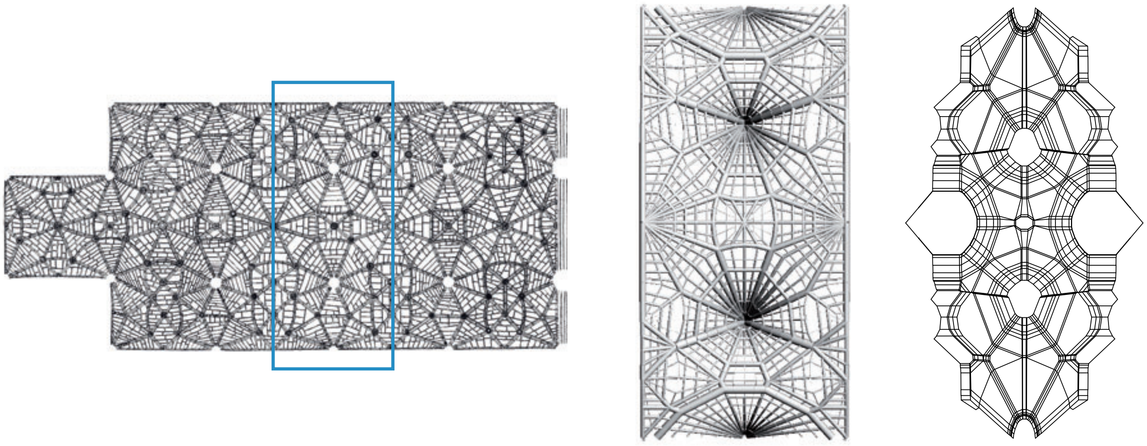


Figure 13.12 (a) Rib and stereotomy pattern of the nave vaults of the Jerónimos monastery. Resulting (b) form diagram with sizing proportional to the forces in (c) the force diagram

Key concepts and terms

A **graph of a network of branches and node** is a drawing that visualizes the connectivity of the network.

A **planar graph** is planar if it can be drawn on a sheet of paper without overlapping edges; in other words, if it can be embedded in the plane.

A **dual graph** is a graph with the same number of edges as the original, but in which the meaning of nodes and faces has been swapped.

The **centroidal dual** of a graph is the dual of which the vertices or nodes coincide with the centroids of the faces of the original graph.

The **convex, parallel dual** is a dual graph with convex faces and edges parallel to the corresponding edges of the original graph.

Reciprocal diagrams are two planar diagrams or graphs that are said to be reciprocal if one is the convex, parallel dual of the other. See Chapter 7 for an alternative definition.

Line search strategy is one of the two basic iterative approaches to finding a local minimum of an objective function; the other is trust region. It first finds a descent direction along which the objective function reduces and then computes a step size that decides how far it should be moved along that direction. The step size can be determined either exactly or inexactly.

A **gradient descent algorithm** is a type of line search in which steps are taken proportional to the negative of the gradient of the objective function at the current point.

Strong Wolfe conditions ensure that the step length reduces the objective function 'sufficiently', when solving an unconstrained minimization problem using an inexact line search algorithm. Strong Wolfe conditions ensure convergence of the gradient to zero.

Closed form means that a mathematical expression can be expressed analytically in terms of a finite number of certain well-known functions.

Cholesky decomposition is used in linear algebra for solving systems of linear equations. It is a decomposition of a Hermitian, positive-definite matrix into the product of a lower triangular matrix and its conjugate transpose.

Exercises

- Define a target surface and draw a form diagram according to the features (e.g. ribs, open edges) of the surface – for instance, within the plan of the standard grid (Fig. 6.12). Make sure to provide force paths that allow those features to develop.
- For a simple target surface, draw a form diagram and an initial force diagram and compute and draw the corresponding thrust network. Try to manually

modify the force diagram such that a better fit of the simple target is obtained.

- Compare the outcome of best-fit optimizations for the same target surface, using different force diagrams (i.e. allowed force flows).
- For a simple target surface and a form diagram corresponding to the standard grid (Fig. 6.12), generate an initial force diagram and corresponding thrust network as explained in Chapter 7 considering only the structure's self-weight.
- Calculate the squared sum of the vertical distances between the nodes of the thrust network and the nodes of the target, as a function of the force densities in the edges of the network. Calculate force densities that make this squared sum smaller or, even better, as small as possible. Attempt to generate a force diagram with edges parallel to the form diagram and the length of the edges equal to the calculated force densities.
- Increase the load on one of the nodes of the network. Draw the thrust network for the current force diagram. Repeat the steps of the previous exercise until a network is found that is close to the target again.
- The architectural program for the Texas shell has changed. The architect now envisages a shell supported on the four corners and one central support. Attempt to generate such a target surface and draw a form diagram according to the features which include ribs and open edges. Hint: make sure to provide force paths that allow those features to develop.

Further reading

- *Numerical Optimization*, Nocedal and Wright (2000). This book describes efficient methods in continuous optimization, including the gradient descent algorithm in Section 13.5.