A form-finding method for membrane shells with radial basis functions

Chiang, Y.-C.; Borgart, Andrew

DOI
10.1016/j.engstruct.2021.113514

Publication date
2022

Document Version
Final published version

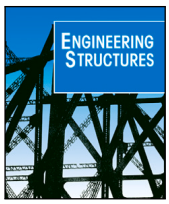
Published in
Engineering Structures

Engineering Structures 251 (2022) 113514

Contents lists available at ScienceDirect

Engineering Structures

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



A form-finding method for membrane shells with radial basis functions

Yu-Chou Chiang *, Andrew Borgart

Faculty of Architecture and the Built Environment, Delft University of Technology, Julianalaan 134, 2628 BL Delft, The Netherlands

ARTICLE INFO

Keywords:
Form-finding
Membrane shell
Radial basis functions
Pucher's equation
Airy stress function
Horizontal loads

ABSTRACT

The equilibrium of a membrane shell is governed by Pucher's equation that is described in terms of the relations among the external load, the shape of the shell, and the Airy stress function. Most of the existing funicular form-finding algorithms take a discretized stress network as the input and find the shape. When the resulting shape does not meet the user's expectation, there is no direct clue on how to revise the input. The paper utilizes the method of radial basis functions, which is typically used to smoothly approximate arbitrary scalar functions, to represent C^{∞} smooth shapes and stress functions of shells. Thus, the boundary value problem of solving Pucher's equation can be converted into a least-squares regression problem, without the need of discretizing the governing equation. When the provided shape or stress function admits no solution, the algorithm recommends users how to tweak the input in order to find an approximate solution. The external load in this method can easily incorporate vertical and horizontal components. The latter part might not always be negligible, especially for the seismic hazard zones. This paper identifies that the peripheral walls are preferable to allow the membrane shells to carry horizontal loads in various directions without deviating from their original shapes. When there are no sufficient supports, the algorithm can also suggest the potential stress eccentricities, which could inform the design of reinforcing beams.

1. Introduction

A membrane shell supports loads by in-plane stresses and its curvatures. The absence of bending moments allows this type of structure to span a considerable distance with minimal material. Due to its structural efficiency and aesthetic elegance, the membrane shells can be found in various structures, from medieval masonry vaults to contemporary glass roofs (Fig. 1).

Most arbitrarily curved surfaces do not have such a bending-free property. The process of finding the special bending-free forms is called form-finding. The existing numerical form-finding algorithms are largely based on funicular networks [1]. The form-found results are affected by networks' connectivity and parameters (e.g., stiffness coefficients in the dynamic relaxation [2] and the force densities in the force density method [3]). But it is often not clear how to update the connectivity or parameters to control the resulting shapes.

Giving users greater control, thrust network analysis [4,5] allows users to interactively manipulate reciprocal diagrams to control the vertically projected thrust network and the values of the force densities. Then, the instant feedbacks are provided by computing the vertical elevation of the thrust network through the force density method. The method of designing self-supporting surfaces in [6] goes one step further; it can take a 3D planar-facet mesh as the input and derive the best-fit polyhedral Airy stress function, which is equivalent to the reciprocal diagrams (see [7]) in the thrust network analysis.

Unfortunately, those two interactive methods are incompatible with horizontal loads, which are crucial for shell structures in seismic hazard zones [8]. The dynamic response under a seismic load should be analyzed through dynamic analysis [9]. However, quasi-static horizontal loading analyses can still provide basic information in a preliminary design stage [10,11], and it is also useful to evaluate the safety of built vault heritage [10].

Horizontal loads are also often left out in the closed-form solutions, even though the analytical equations: Pucher's equation [12] (governing vertical equilibrium) and the Airy stress function (expressing horizontal equilibrium) have adequately incorporated horizontal loads. The analytical discussions often set horizontal loads as zeros [13,14].

To develop a more fundamental form-finding method, this paper revisits the governing Pucher's equation, which describes the relation among the external loads (in vertical and horizontal directions), the shape of the shell, and the Airy stress function. Furthermore, the method of radial basis functions (RBFs) is used, which is a numerical method underutilized in the form-finding application [15]. RBFs can convert boundary value problems into least-squares fitting problems, without the need of discretizing governing equations or boundary conditions [16,17]. This paper uses RBFs to smoothly represent the shapes

* Corresponding author.
E-mail address: Chiang.YuChou@gmail.com (Y.-C. Chiang).

https://doi.org/10.1016/j.engstruct.2021.113514
Received 18 May 2021; Received in revised form 18 October 2021; Accepted 26 October 2021
Available online 11 November 2021

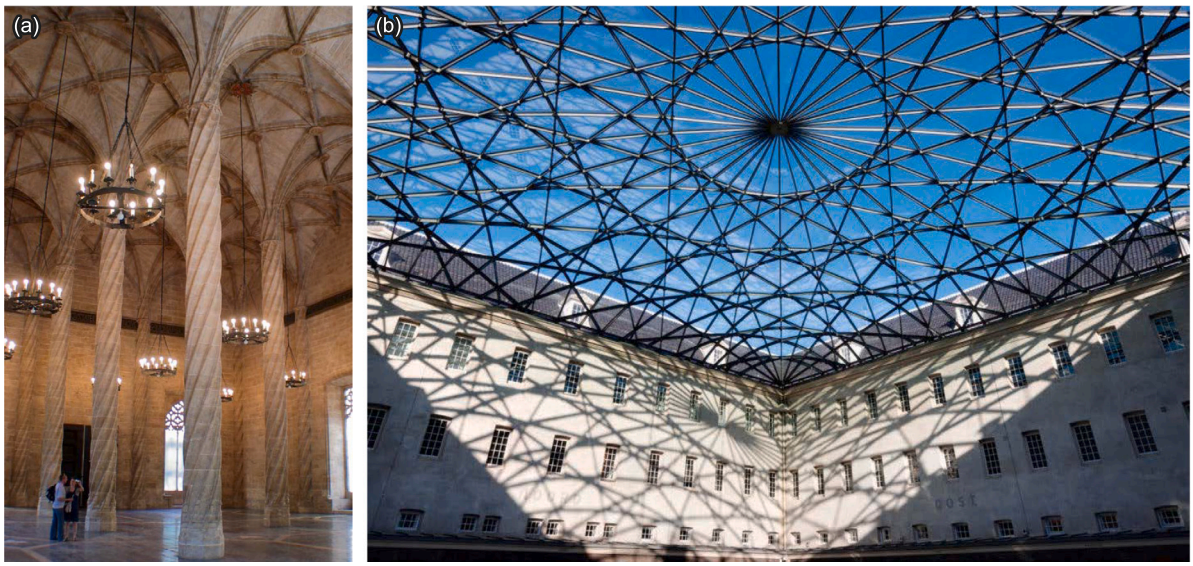


Fig. 1. Examples of shell structures. (a) Masonry vault of *Llotja de la Seda* [Silk Exchange], Valencia, Spain. (b) Glass roof of Dutch Maritime Museum, Amsterdam, Netherlands. *Source:* (a) TMaschler@commons.wikimedia.org. (b) © design Ney & Partners - www.photo-daylight.com.

of shells and the Airy stress functions and applies the least-squares method to best fit the equilibrium equations. The resulting algorithm can take an Airy stress function as the input, and then generates the corresponding shape, or the other way around. Free-edge conditions and horizontal loads are also compatible with the proposed algorithm.

The article is organized as follows. Section 2 gives an overview of form-finding methods and positions the proposed method among them. Section 3 revisits the essential statics theory of membrane shells. Section 4 introduces the method of RBFs for solving generic differential equations. Section 5 applies the RBFs specifically to find membrane shells under vertical loads. Section 6 further applies the algorithm to shells subjected to horizontal loads. Finally, Section 7 summarizes the current finding and projects future investigations.

2. Form-finding methods

This section discusses the features and limitations of various form-finding methods, including analytical approaches, numerical methods, and the proposed RBFs algorithm. The features of discussed methods are organized in Table 1, which shows the proposed RBFs algorithm can unify various features of different form-finding methods.

Although the analytical equations, Pucher's equation and the Airy stress function, can adequately incorporate horizontal loads, horizontal loads are often left out for simplification. Even only focusing on vertical loads, analytical solutions are often limited to certain boundary conditions of simple geometries, due to the limitation of the analytic expressions.

To expand the geometric freedom, numerical algorithms are developed to analyze the statics of the shells. Most of the existing algorithms discretize the continuous surfaces into networks with finite funicular bars [1], or parameterized the surface as smooth Non-Uniform Rational Basis Splines (NURBS) surfaces [18]. These computational form-finding methods can be broken down into two categories: simulating the physical hanging models and numerically solving statics equations [19] (Fig. 2).

Physics-based. In the category of simulating physical models, the most representative method is *dynamic relaxation* [2]. The computational models contain parameters such as the mass of each node, stiffness of each bar, support positions, etc. The model computes the elongation of bars and residual forces on nodes, and subsequently updates the velocity and position of each node. Then, the model iteratively re-calculates the elongation and residual forces based on the updated position until all nodes rest at an equilibrium state. Methods in this category are intuitive, but they can only provide physically stable equilibrium solutions.

Mathematics-based. The methods of this category, in contrast, do not calculate velocity or acceleration. They solve statics equations numerically, and sometimes are called geometric stiffness methods [1]. The most representative method is *force density method* [3], in which the bars have prescribed forces proportional to their lengths. The ratio of the force to the length is called "force density". The algorithm computes the residual force of each node according to the "force densities" and seeks the alternative position of each node to minimize its residual force. The method of *thrust network analysis* [5] and methods in [6,10,18] fall into this category. Methods in this category are slightly abstract, but they can generate both physically stable and unstable equilibrium solutions.

Most of these methods are based on networks of funicular bars connected at nodes. The results are affected by the connectivity of the network and the parameters of the edges, such as the spring constants in dynamic relaxation [2] and the force densities in the force density method [3]. These methods can find funicular forms. However, most methods do not indicate how to alter the connectivity and the parameters to approximate a predefined shape.

User interaction. To give users greater control, more interactive form-finding methods have been proposed. Such as thrust network analysis [5] enable users to manipulate the reciprocal diagrams to *design* the connectivity and the force densities on the funicular networks. Then, the algorithm provides real-time results, so that users can interactively revise the reciprocal diagram until they are happy with the results. Vouga et al. [6] provide a more direct control, their method of designing self-supporting surfaces can take a planar mesh as the input and reversely unveil the underlying polyhedral Airy stress function, which is informationally equivalent to the reciprocal diagrams [7]. When the shapes that users provided are not fully compatible with the statics equations, Vouga's method can tweak and thus *correct* the shape to meet the conditions. The triangular meshes can also be replaced by NURBSs as shown by Miki et al. [18].

Unfortunately, those interactive method is not compatible with horizontal loads. To address this drawback, Marmo et al. [10] have reformulated the thrust network analysis but at the expanse of giving up the interactive reciprocal diagrams that allow users to indirectly shaping the resulting funicular network.

Table 1
Features of various form-finding methods.

Feature	Analytical[a]	DR[b]	FDM[c]	TNA[d]	RTNA[e]	SSS[f]	PSSS[g]	Paper
Basis function	Analytic expressions	Network				Planar mesh	NURBSs	RBFs
Geometric freedom	–	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Unstable solution	Yes	–	Yes	Yes	Yes	Yes	Yes	Yes
Shape tweaking	Yes	–	–	–	–	Yes	Yes	Yes
Horizontal loads	Yes[h]	Yes	Yes	–	Yes	–	–	Yes

[a]See [13,14].
[b]Dynamic Relaxation [2].
[c]Force Density Method [3].
[d]Thrust Network Analysis [5].
[e]Reformulated Thrust Network Analysis [10].
[f]Self-Supporting Surfaces [6].
[g]Parametric Self-Supporting Surfaces [18].
[h]Included in the governing equation but rarely in solutions.

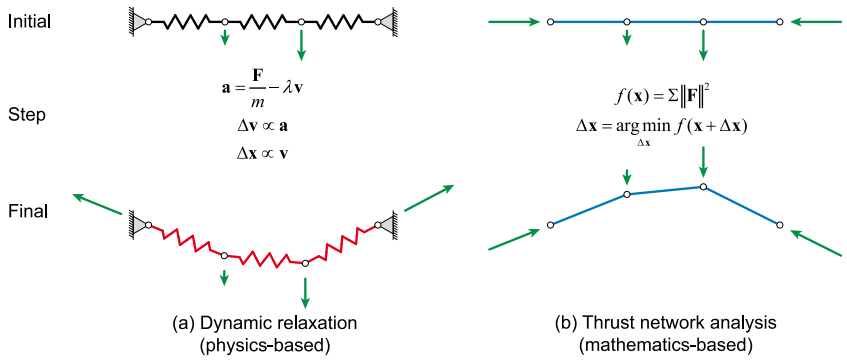


Fig. 2. Schemes of finding funicular structures. (a) The method of dynamic relaxation starts from a set of springs, nodes, and loads. The residual force **F** accelerates the node toward a stable equilibrium. (b) The method of thrust network analysis initializes the process with pre-stressed bars, nodes, and loads. The nodes actively search for alternative positions to minimize all the residual forces $\Sigma\|\mathbf{F}\|^2$.

Motivation of utilizing rbfs. This paper is set to develop a method to collect the positive features mentioned above: having the geometric freedom to meet various boundary conditions, deriving unstable equilibrium configurations, tweaking and revising given shapes to meet the statics conditions, and finally being compatible with horizontal loads. The method of radial basis functions (RBFs) provides a convenient approach, RBFs can rapidly convert a boundary value problem into a least-squares fitting problem, without the processes of discretizing governing equations. The governing equation of membrane shells – Pucher's equation [12] – is readily available, which sufficiently relates shapes, stresses (expressed by the Airy stress functions), and loads in vertical and horizontal directions. In the presented RBFs algorithm, the loads are externally given, while shapes and the Airy stress functions will be represented by RBFs. The least-squares method is used to find the best-fit shapes and the Airy stress functions satisfying the governing equation. The resulting surfaces and stress functions are continuously differentiable similar to analytical solutions. This feature allows convenient calculation of curvatures and stresses, which also benefit the visualization of stress trajectories in this paper (e.g., Fig. 8). The presented algorithm is deemed to be the most versatile form-finding method among the others shown in Table 1.

3. Statics of membrane shells

This section briefly revisits the essential formulations that govern the equilibrium of membrane shells. The equilibrium equations connect three items: load, shape, and stress distribution (or an Airy stress function). The resulting equilibrium configuration, which can set a lower bound in the plastic limit theorems, is useful in the design of shells [13]. Material properties such as density and elastic modulus are not involved in these expressions. In a way, the form-found shapes are regarded as material independent, and the resulting shapes can be found in a wide range of built structures, for instance, steel tessellated roofs, masonry vaults, concrete shells.

Pucher's equation. The governing differential equation – Pucher's equation – is expressed as [12, p. 461]

$$N_{xx} \cdot \partial_{xx} Z + 2N_{xy} \cdot \partial_{xy} Z + N_{yy} \cdot \partial_{yy} Z = p_x \cdot \partial_x Z + p_y \cdot \partial_y Z - p_z, \quad (1)$$

where is N_{ij} $(i,j \in \{x,y\})$ are the horizontal stress resultants, ∂_{ij} $(i,j \in \{x,y\})$ are Euler's notation for the derivatives such that $\partial_{ij} = \frac{\partial^2}{\partial i\, \partial j}$, $Z(x,y)$ is the elevation of the shell, p_x, p_y are horizontal body forces (per unit horizontal area), and p_z is the vertical load (per unit horizontal area). The stress resultants N_{ij} can be expressed with an Airy stress function $F(x,y)$ as [20]

$$N_{xx} = \partial_{yy} F - \int p_x\, dx, \quad N_{yy} = \partial_{xx} F - \int p_y\, dy, \quad N_{xy} = -\partial_{xy} F. \quad (2)$$

These expressions guarantee that N_{ij} satisfy the horizontal equilibrium conditions $\sum_{j\in\{x,y\}} \partial_j N_{ij} + p_i = 0$.

From expressions (2), Pucher's equation (1) becomes

$$\left(\partial_{yy} F - \int p_x\, dx\right) \cdot \partial_{xx} Z - 2\,\partial_{xy} F \cdot \partial_{xy} Z + \left(\partial_{xx} F - \int p_y\, dy\right) \cdot \partial_{yy} Z = p_x \cdot \partial_x Z + p_y \cdot \partial_y Z - p_z. \quad (3)$$

For a form-finding problem, the external loads p_x, p_y, p_z are usually predefined while stress $F(x,y)$ and shape $Z(x,y)$ are unknowns. With two unknown functions and only one governing equation, a form-finding problem is mostly underdetermined.

Two perspectives. One of the two unknown functions shall better be prescribed to turn the form-finding problem well-posed. For instance, in the thrust network analysis [5], reciprocal diagrams are prescribed. Then, the method provides the elevation of the thrust network. Equivalently, in the proposed method, the Airy stress function $F(x,y)$ can be

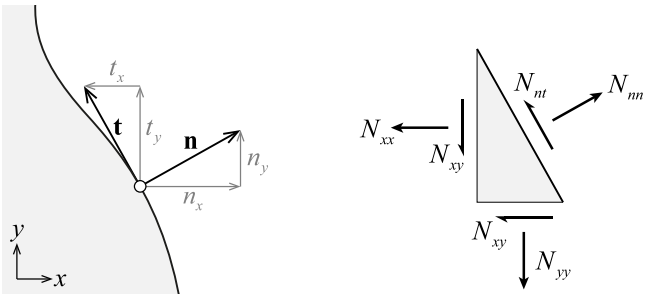


Fig. 3. Local directions on an edge (left) and the stresses of an infinitesimal triangle at the edge (right).

prescribed first, then the shape function $Z(x, y)$ to be derived subsequently.

When stress function $F(x, y)$, as well as the loads $p_i(x, y)$, is predefined, Pucher's equation (3) becomes a second-order partial differential equation:

$$\mathcal{D}_F^{Pucher}[Z(x,\ y)] = -p_z, \tag{4}$$

where $\mathcal{D}_F^{Pucher}[\cdot]$ is the differential operator with subscript F indicates that the stress function $F(x, y)$ is contained in the operator. The operator is defined as

$$\mathcal{D}_F^{Pucher}[\cdot] = \left(\partial_{yy}F - \int p_x\, dx\right) \cdot \partial_{xx}[\cdot] - 2\,\partial_{xy}F \cdot \partial_{xy}[\cdot] + \left(\partial_{xx}F - \int p_y\, dy\right) \cdot \partial_{yy}[\cdot] - p_x \cdot \partial_x[\cdot] - p_y \cdot \partial_y[\cdot].$$

In the other way around, when $Z(x, y)$ is treated as a predefined function, Pucher's equation (3) can also be reorganized as

$$\mathcal{D}_Z^{Pucher}[F(x,\ y)] = -p_z + p_h'. \tag{5}$$

In this perspective, the corresponding differential operator goes as

$$\mathcal{D}_Z^{Pucher}[\cdot] = \partial_{xx}Z \cdot \partial_{yy}[\cdot] - 2\,\partial_{xy}Z \cdot \partial_{xy}[\cdot] + \partial_{yy}Z \cdot \partial_{xx}[\cdot].$$

Meanwhile, the term p_h' at the right-hand side of Eq. (4) is given by

$$p_h' = \partial_{xx}Z \cdot \int p_x\, dx + \partial_{yy}Z \cdot \int p_y\, dy + \partial_x Z \cdot p_x + \partial_y Z \cdot p_y.$$

The expressions (4) and (5) are convenient for formulating the algorithm in Section 5.

Boundary conditions. So far, only the equilibrium within the domain has been discussed. We shall also pay attention to the boundaries. The boundaries are the edges of shells. At an edge, there are membrane stresses on the curvy shell from one side. From the other side, there is no shell element providing membrane stresses. Nonetheless, all elements on an edge need to stay in equilibrium, with or without stresses transmitting across the edge.

One can first transform the stress resultants in x- and y- directions into normal and tangential directions. The normal vector $\mathbf{n} = \begin{bmatrix} n_x & n_y \end{bmatrix}^{\mathrm{T}}$ is defined to be pointing outward while the tangential vector $\mathbf{t} = \begin{bmatrix} t_x & t_y \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} -n_y & n_x \end{bmatrix}^{\mathrm{T}}$ is circling the domain counterclockwise (see Fig. 3). The normal stress N_{nn} normal to the edge, the normal stress N_{tt} parallel to the edge, and the shear stress N_{nt} can be derived from [21, p. 208]

$$\begin{bmatrix} N_{nn} & N_{nt} \\ N_{nt} & N_{tt} \end{bmatrix} = \begin{bmatrix} n_x & n_y \\ -n_y & n_x \end{bmatrix} \begin{bmatrix} N_{xx} & N_{xy} \\ N_{xy} & N_{yy} \end{bmatrix} \begin{bmatrix} n_x & -n_y \\ n_y & n_x \end{bmatrix}. \tag{6}$$

If all elements on an edge segment can reach equilibrium without external stresses, then one can say the segment is a *free edge*, which is free from external supports. On a free edge, one should have

$$N_{nn} = 0, \tag{7a}$$

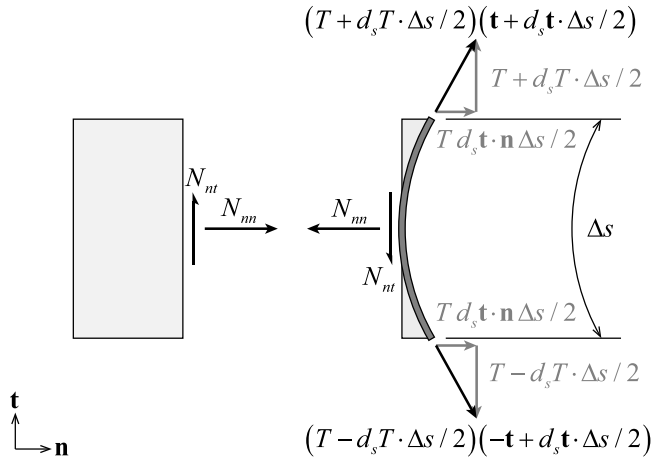


Fig. 4. Forces and stress resultants on a segment of the reinforcing funicular rib.

$$N_{nt} = 0, \tag{7b}$$

$$N_{tt} \cdot \partial_{tt}Z = p_n \cdot \partial_n Z + p_t \cdot \partial_t Z - p_z. \tag{7c}$$

Condition (7a) means there is zero normal stress across the boundary, condition (7b) describes there is zero shear against the boundary, while condition (7c) is a degenerated version of Pucher's equation suggesting the boundary is balanced in the vertical direction. Horizontal body forces p_x and p_y are represented by $p_n = n_x p_x + n_y p_y$ and $p_t = -n_y p_x + n_x p_y$.

Whenever any condition of (7a)–(7c) is not satisfied, external support or a special reinforcement will be needed to achieve an equilibrium. For instance, a designer can add a funicular rib and a vertical wall at the edge to bypass conditions (7b)–(7c). Subsequently, condition (7a) turns into

$$N_{nn} = T(\partial_s \boldsymbol{t} \cdot \boldsymbol{n}), \tag{8}$$

and the corresponding supporting forces become

$$T = \int N_{nt}\, ds, \tag{9}$$

$$p_{ez} = N_{nn} \cdot \partial_n Z + N_{nt} \cdot \partial_t Z - T \cdot \partial_{tt} Z, \tag{10}$$

where T denotes the tension force of the funicular rib (Fig. 4) and p_{ez} denotes the vertical supporting force from the wall.

If the designer would like to bypass the condition (8), then a rigid support shall be provided. Since membrane shells transmit little bending moment, the support is needed to take only translational forces. Once the support is capable of carrying forces in all directions, there is no condition for such a fully supported edge.

4. Introduction to radial basis functions (RBFs)

The method of RBFs is often used to represent a smooth multivariate function. The method has been developed for more than 40 years [16], but it is rarely used in solving form-finding problems [15]. Unlike finite element methods requiring governing equations being discretized and integrated into weak forms over elements, RBFs can rapidly transform a boundary value problem into a least-squares regression fitting prescribed values of the function and its derivatives [16,17]. It is instructive to first introduce how this method is used to numerically solve a generic boundary value problem. Later, Section 5 will apply RBFs to solve the form-finding problems.

An arbitrary function $f(\mathbf{x})$ can be approximated by a set of radial basis functions $\phi(\mathbf{x}) = \phi\left(\left\|\mathbf{x} - \boldsymbol{\mu}_i\right\|; \rho_i\right)$ and augment polynomial terms $h(\mathbf{x})$ [16,22,23]:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \lambda_i \phi\left(\left\|\mathbf{x} - \boldsymbol{\mu}_i\right\|; \rho_i\right) + h(\mathbf{x}) + \varepsilon, \tag{11}$$

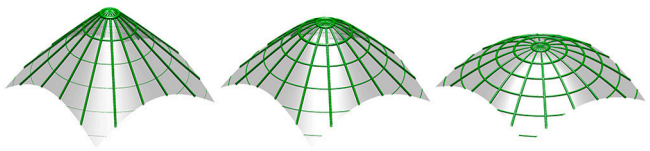


Fig. 5. The multiquadrics is an RBF that has a shape parameter ρ that allows the RBF to take a shape between a cone and a paraboloid as ρ takes a value between 0 and ∞ (left to right).

in which $\mathbf{x}$ is the position of evaluation, λ_i are the magnitude coefficients of the RBFs, $\boldsymbol{\mu}_i$ are the source points, ρ_i are the shape parameters of the RBFs, and ε is the approximate error.

Let $r = \|\mathbf{x} - \boldsymbol{\mu}\|$ denotes the distance. The radial basis functions can be as simple as the Euclidean distance $\phi(r;\rho) = r$, thin-plate splines $\phi(r;\rho) = r^2 \log r$, Gaussian $\phi(r;\rho) = e^{-\rho^2 r^2}$, or *multiquadrics*

$$\phi(\|\mathbf{x} - \boldsymbol{\mu}\|;\rho) = \sqrt{\|\mathbf{x} - \boldsymbol{\mu}\|^2 + \rho^2}. \tag{12}$$

The last one is used in all the cases in this paper because its shape parameter ρ allows the user to adjust the local intensity of the curvature (Fig. 5). This benefit shall be discussed in detail in the upcoming Section 5. Furthermore, there is no necessity to augment multiquadrics with polynomial terms [22]. Therefore, Eq. (11) can be more specific and simplistic as

$$f(\mathbf{x}) = \sum_{i=1}^{n} \lambda_i \sqrt{\|\mathbf{x} - \boldsymbol{\mu}_i\|^2 + \rho_i^2} + \varepsilon. \tag{13}$$

By a sufficient number n of multiquadrics and adequately arranged $\boldsymbol{\mu}_i$ and ρ_i, the representable space [16]

$$S = \left\{ \sum_{i=1}^{n} \lambda_i \sqrt{\|\mathbf{x} - \boldsymbol{\mu}_i\|^2 + \rho_i^2} \,\middle|\, \{\lambda_i\} \in \mathbb{R}^n \right\} \tag{14}$$

shall be flexible enough to minimize the approximate error ε. Let $\mathbf{x}_j$ denote m calibration points, and f_j denote the target values. The errors ε_j shall be

$$\varepsilon_j = f_j - \phi_{ji} \cdot \lambda_i, \tag{15}$$

where $\phi_{ji} = \phi\left(\left\|\mathbf{x}_j - \boldsymbol{\mu}_i\right\|;\rho_i\right) = \sqrt{\left\|\mathbf{x}_j - \boldsymbol{\mu}_i\right\|^2 + \rho_i^2}$. In matrix form, it can be expressed as

$$\boldsymbol{\varepsilon} = \mathbf{f} - \boldsymbol{\Phi}\boldsymbol{\lambda}, \tag{16}$$

in which $\boldsymbol{\varepsilon}$ and $\mathbf{f}$ are two $m \times 1$ column vectors contenting ε_j and f_j; $\boldsymbol{\lambda}$ is an $n \times 1$ column vector containing λ_i; while $\boldsymbol{\Phi}$ is an $m \times n$ rectangular matrix with elements as ϕ_{ji}. Then, the best-fit coefficient vector $\boldsymbol{\lambda}$ can be provided by minimizing the approximate errors with the ordinary *least-squares method (LSM)*:

$$\boldsymbol{\lambda} = \arg\min_{\boldsymbol{\lambda}} \boldsymbol{\varepsilon}^{\mathrm{T}}\boldsymbol{\varepsilon} = \left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{f}. \tag{17}$$

Eq. (17) is often applied to approximation problems.

Solving a differential equation. When the least-squares scheme include prescribing not only the values of the function but also its derivatives, the RBFs can also provide solutions of differential equations. Suppose there is a linear partial differential equation problem as

$$\mathcal{L}[f(\mathbf{x})] = g(\mathbf{x}), \quad \mathbf{x} \in \Omega, \tag{18}$$

$$\mathcal{B}[f(\mathbf{x})] = q(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega, \tag{19}$$

where Ω is the domain, $\partial\Omega$ is the boundary, $\mathcal{L}[\cdot]$ and $\mathcal{B}[\cdot]$ are the differential operators of the governing equation and the boundary condition respectively, while g and q are prescribed functions. When $\mathbf{x}_\Omega$ and $\mathbf{x}_{\partial\Omega}$ are selected as calibration points, radial basis functions can be used to approximate the problem as

$$g(\mathbf{x}_\Omega) = \sum_{i=1}^{n} \mathcal{L}\left[\phi_i(\mathbf{x}_\Omega)\right] + \varepsilon_\Omega, \qquad \mathbf{x}_\Omega \in \Omega, \tag{20}$$

$$q(\mathbf{x}_{\partial\Omega}) = \sum_{i=1}^{n} \mathcal{B}\left[\phi_i(\mathbf{x}_{\partial\Omega})\right] + \varepsilon_{\partial\Omega}, \quad \mathbf{x}_{\partial\Omega} \in \partial\Omega, \tag{21}$$

or in a matrix form as

$$\begin{bmatrix}\mathbf{g}\\ \mathbf{q}\end{bmatrix} = \begin{bmatrix}\boldsymbol{\Phi}_{\mathcal{L}}\\ \boldsymbol{\Phi}_{\mathcal{B}}\end{bmatrix}\lambda + \begin{bmatrix}\boldsymbol{\varepsilon}_\Omega\\ \boldsymbol{\varepsilon}_{\partial\Omega}\end{bmatrix}. \tag{22}$$

The weighted least-squares method can provide the solution:

$$\lambda = \arg\min_{\lambda} \begin{bmatrix} w_{\mathcal{L}}\,\boldsymbol{\varepsilon}_\Omega \\ w_{\mathcal{B}}\,\boldsymbol{\varepsilon}_{\partial\Omega}\end{bmatrix}^{\mathrm{T}} \begin{bmatrix} w_{\mathcal{L}}\,\boldsymbol{\varepsilon}_\Omega \\ w_{\mathcal{B}}\,\boldsymbol{\varepsilon}_{\partial\Omega}\end{bmatrix}, \tag{23}$$

where $w_{\mathcal{L}}$ and $w_{\mathcal{B}}$ are the weights. The weights are crucial when $\mathcal{L}[\cdot]$ and $\mathcal{B}[\cdot]$ have different dimensions.

The generic theory of differential and least-squares operations has been clarified. The upcoming sections employ the method to form-finding problems.

5. Algorithm and shells subjected to vertical loads

This section first builds generic RBF models to represent arbitrary C^∞ smooth shapes $Z(x,y)$ and Airy stress functions $F(x,y)$ (Section 5.1), then introduces the algorithm that solves form-finding problems when the loads are given (Section 5.2), and presents examples of finding membrane shells on a triangular floor plan with free edges or wall-supported edges (Sections 5.3–5.5).

With two unknown functions ($Z(x,y)$, $F(x,y)$) and only one governing Eq. (3), the form-finding problem is under-determined. However, for a membrane that has free edges, its stress function $F(x,y)$ faces two boundary conditions (7a)–(7b), which make the free-edges over-determined. Overall speaking, finding a membrane shell with free edges is a problem with countless solutions yet in certain subsets. The proposed algorithm can identify if an arbitrarily given shape or stress function is one of the countless solutions. When the input function permits no solution, the algorithm will tweak the input in order to reach the solution subsets.

5.1. Models of the shape and the Airy stress function

In a problem of finding a membrane shell, the radial basis functions are used to model two unknown functions of stress $F(\mathbf{x})$ and shape $Z(\mathbf{x})$:

$$F(\mathbf{x}) = \sum_{i=1}^{n} \lambda_{F,i}\phi\left(\left\|\mathbf{x} - \boldsymbol{\mu}_i\right\|;\rho_i\right) + \varepsilon_F,$$

$$Z(\mathbf{x}) = \sum_{i=1}^{n} \lambda_{Z,i}\phi\left(\left\|\mathbf{x} - \boldsymbol{\mu}_i\right\|;\rho_i\right) + \varepsilon_Z.$$

At points $\mathbf{x}_\Omega$ within the domain Ω, the functions $F(\mathbf{x})$ and $Z(\mathbf{x})$ should satisfy Pucher's equation (3). At points $\mathbf{x}_{\partial\Omega,e}$ on the edges, the functions should meet boundary conditions, such as Eqs. (7) for free edges or Eq. (8) for wall-supported edges.

For the shape function, the relevant conditions can be translated into a fitting problem:

$$-p_z(\mathbf{x}_\Omega) = \sum_{i=1}^{n} \lambda_{Z,i} D_F^{Pucher}\left[\phi\left(\mathbf{x}_\Omega\right)\right] + \varepsilon_Z^{Pucher}, \tag{24a}$$

$$Z_{\partial\Omega,s} = \sum_{i=1}^{n} \lambda_{Z,i}\phi\left(\mathbf{x}_{\partial\Omega,s}\right) + \varepsilon_Z^{Support}, \tag{24b}$$

where $Z_{\partial\Omega,s}$ are prescribed elevations at the positions $\mathbf{x}_{\partial\Omega,s}$ of the supports.

On the other hand, for the stress function, the constraints include

$$-p_z(\mathbf{x}_\Omega) + p_h'(\mathbf{x}_\Omega) = \sum_{i=1}^{n} \lambda_{F,i} D_Z^{Pucher}\left[\phi\left(\mathbf{x}_\Omega\right)\right] + \varepsilon_F^{Pucher}, \tag{25a}$$

$$q^{Normal}(\mathbf{x}_{\partial\Omega,e}) = \sum_{i=1}^{n} \lambda_{F,i} \mathcal{D}^{Normal}\left[\phi\left(\mathbf{x}_{\partial\Omega,e}\right)\right] + \epsilon_F^{Normal}, \tag{25b}$$

$$q^{Shear}(\mathbf{x}_{\partial\Omega,e}) = \sum_{i=1}^{n} \lambda_{F,i} \mathcal{D}^{Shear}\left[\phi\left(\mathbf{x}_{\partial\Omega,e}\right)\right] + \epsilon_F^{Shear}, \tag{25c}$$

$$F_{\partial\Omega,s} = \sum_{i=1}^{n} \lambda_{F,i} \phi\left(\mathbf{x}_{\partial\Omega,s}\right) + \epsilon_F^{Support}. \tag{25d}$$

Since the conditions (25a)–(25c) only refer to the second derivatives of the stress function, extra conditions to fix the three degrees of freedom on the overall value and the first-order derivatives are necessary. Therefore, condition (25d) is added.

The prescribed functions $q^{Normal}(\cdot)$, $q^{Shear}(\cdot)$ and the differential operators $\mathcal{D}_Z^{Normal}[\cdot]$, $\mathcal{D}_Z^{shear}[\cdot]$ can be provided by

$$q^{Normal}(\mathbf{x}) = n_x^2 \int p_x\, dx + n_y^2 \int p_y\, dy, \tag{26a}$$

$$\mathcal{D}^{Normal}[\cdot] = n_x^2 \frac{d^2}{dy^2}[\cdot] - n_x n_y \frac{d^2}{dxdy}[\cdot] + n_y^2 \frac{d^2}{dx^2}[\cdot], \tag{26b}$$

$$q^{Shear}(\mathbf{x}) = -n_x n_y \int p_x\, dx + n_x n_y \int p_y\, dy, \tag{26c}$$

$$\mathcal{D}^{Shear}[\cdot] = -n_x n_y \frac{d^2}{dy^2}[\cdot] - (n_x^2 - n_y^2)\frac{d^2}{dxdy}[\cdot] + n_x n_y \frac{d^2}{dx^2}[\cdot]. \tag{26d}$$

5.2. Steps of the algorithm

The basic workflow of the algorithm is outlined in Fig. 6, which consists of 7 critical steps:

1. Determine the ground plan. List the boundary conditions of the shell.
2. Describe load distributions (p_x, p_y, and p_z).
3. Identify potential *singular points* and arrange the basis functions accordingly.
4. The designer provides either (a) an initial stress function or (b) an initial shape function.
5. (a) When a stress function is provided, find the corresponding stress function.
   (b) When a shape function is provided, find the corresponding shape function.
6. Revise the stress function according to the updated shape, and revise the shape function according to the updated stress until two of them converged.
7. Export the stress and shape functions.

The singular points mentioned in step 3 refer to points having stress concentration (for stress function) or dramatic curvature (for shape function). In the following demonstration, the support points are such singular points.

5.3. Steps 1–3: Calibration points and source points

The first 3 steps are eventually arranging calibration points ($\mathbf{x}_\Omega$, $\mathbf{x}_{\partial\Omega,e}$, $\mathbf{x}_{\partial\Omega,s}$) and the prescribed values on these points ($p_z(\mathbf{x}_\Omega)$, $Z(\mathbf{x}_{\partial\Omega,s})$) as well as the basis functions' source points ($\boldsymbol{\mu}_i$) and the shape parameters (ρ_i). In the later steps, calibration points would be used to construct the prescribing conditions, while the source points will provide degrees of freedom of the representable space (Eq. (14)). The number of source points shall be sufficient so that the optimization algorithm has sufficient space to search the best fit magnitude coefficients. The number of calibration points shall be larger than the number of source points, so that the overall optimization problem stays well-posed.

Once the ground plan and boundary conditions are determined, one can scatter the calibration points evenly within the domain and on the boundary segments. Source points can be placed within and around the domain. Extra source points are needed for where the resulting function may have intense changes (e.g., the corners of the following triangular shell).

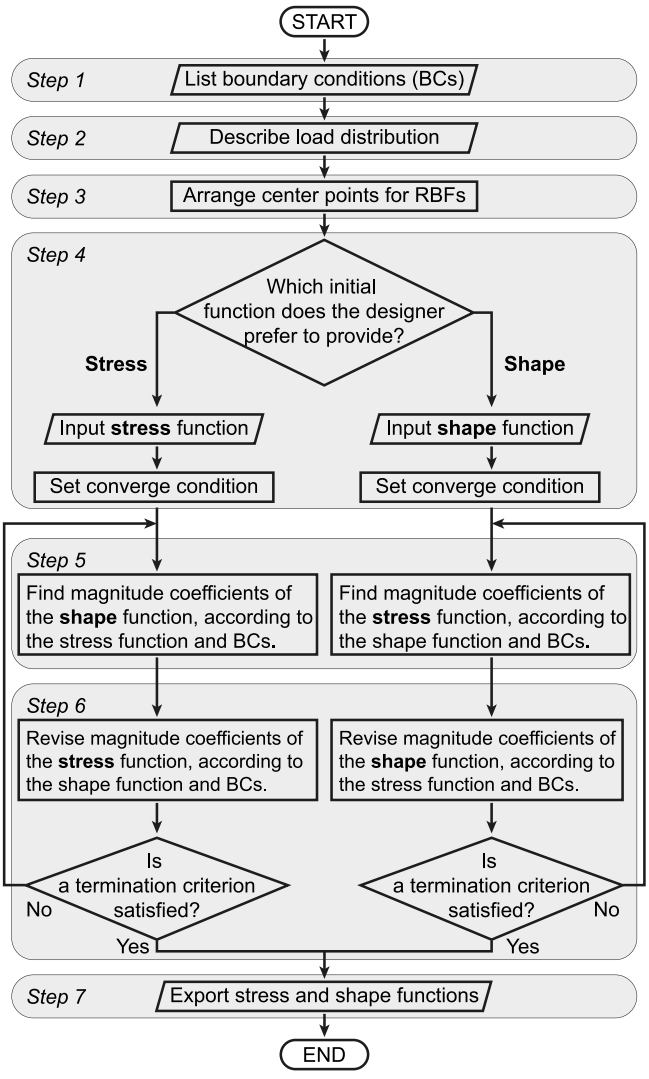


Fig. 6. Flowchart of the algorithm.

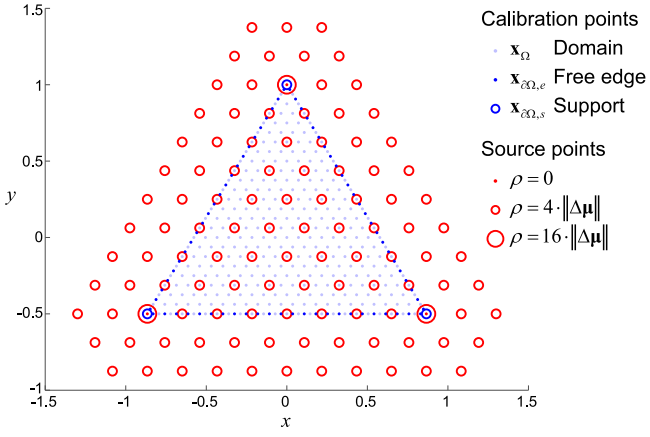


Fig. 7. Distribution of calibration points and source points for the triangular shell (color figure in web version.

Fig. 7 shows the arrangement of the calibration points and source points for the demonstration shell, which has a triangular domain, three anchorages, and a uniform vertical load. Triangular grid points are arranged to be the domain type calibration points $\mathbf{x}_\Omega$. Regarding

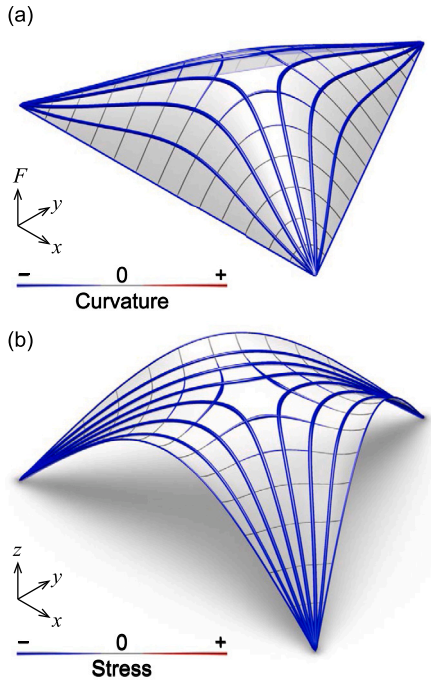


Fig. 8. The stress function (a) and the form-found shape function (b). The curvature network of the stress function is the principal stress network of the resulting shell. (color figure in web version).

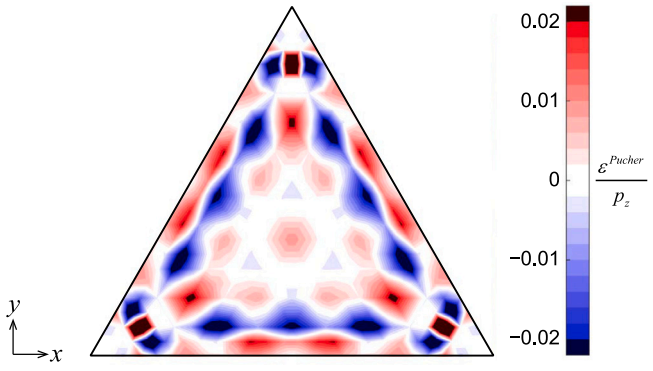


Fig. 9. Distribution of the residual error (color figure in web version).

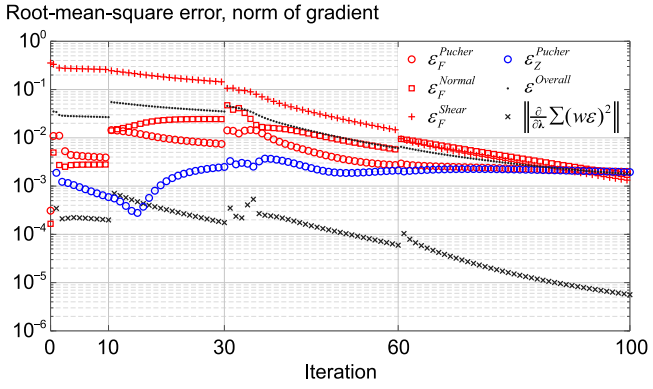


Fig. 10. Root-mean-square errors decaying with the iteration (color figure in web version).

calibration points of the boundary type, the first group $\mathbf{x}_{\partial\Omega,e}$ contains points on the three edges, while the second group $\mathbf{x}_{\partial\Omega,s}$ includes three vertices of the triangle.

Regarding the source points $\boldsymbol{\mu}_i$ of the basis functions, grid points within and around the domain Ω are used. The shape parameters are set to be 4 times the spacing of the grid ($\rho = 4 \cdot \|\Delta\boldsymbol{\mu}\|$) for most of the source points. Considering there will be stress concentration around the corners, each corner is assigned to two source points. One of them has $\rho = 0$ and the other $\rho = 16 \cdot \|\Delta\boldsymbol{\mu}\|$. Since the least-squares method is going to be used, the source points $\boldsymbol{\mu}$ can have less density than domain calibration points $\mathbf{x}_\Omega$ (see Fig. 7).

5.4. Steps 4–5: Two paths of seeking the balance

After the common setup is built in the first three steps, the remaining demonstration splits into two paths. The first path starts with an initial stress function, and the second path starts with an initial shape function.

Path A: Stress initiation. In this path, the demonstration initiates the process from an analytical stress function provided by Csonka [14, p. 587]. The initial input of the stress function goes as

$$F(x,\,y) = \frac{2(L_0 \cdot L_1 \cdot L_2)}{15(L_0 \cdot L_1 \cdot L_2) + 6(x^2 + y^2 - 1)}, \tag{27}$$

where L_i $(i \in \{1, 2, 3\})$ are linear functions that $L_i(x,\,y) = \cos(2\pi i/3)\,x + \sin(2\pi i/3)\,y - 1/2$. This stress function automatically satisfies the free edge conditions (25b)–(25c). Fig. 8a displays this stress function and its principal curvature network.

The only part left unknown is the shape function $Z(x,\,y)$, or $\lambda_{Z,i}$ which can be answered by minimizing the errors in Eqs. (24a)–(24b). One can derive the coefficients for the differential operator $D_F^{Pucher}[\cdot]$ in the governing Eq. (24a) by evaluating the second deviates of the stress function $F(x,\,y)$ at the calibration points $\mathbf{x}_\Omega$. Along with the boundary condition (24b), $\lambda_{Z,i}$ can be provided by the method of least squares (see Eq. (17)). The shape solution provided by the radial basis functions, along with the network of stress trajectories (or isostatic lines), is illustrated in Fig. 8b. The visualization of the stress network is adapted from [24] for smooth stress trajectories.

Regarding the residual errors ε_F^{Pucher} and ε_Z^{Pucher}, they are eventually the same. Both of them represent the difference between prescribed p_z and the supporting stress that $F(\mathbf{x})$ and $Z(\mathbf{x})$ are actually providing. The error is within $\pm 0.02 p_z$ at most of the surface as depicted in Fig. 9. Near the corners, some spots have larger errors ($\|\varepsilon^{Pucher}\| > 0.2p_z$). These spikes can be attributed to the short distance between the free edges, which have calibration points on them. One may opt to place more source points at those regions to curb the residual errors.

Path B: Shape initiation. For most ground plans, there may not be an analytical free-edge stress function available. The designer can choose path B to input an initial shape, then the algorithm can tweak the shape to meet the prescribed boundary conditions. For instance, the designer can provide a paraboloid as the initial shape:

$$Z(x,\,y) = \frac{7}{10}(-x^2 - y^2 + 1).$$

The initial shape function provided crucial coefficients to the differential operator D_Z^{Pucher} in the governing equation (25a). As suggested by Csonka [14], the paraboloid cannot simultaneously be compatible with the two free-edge conditions (25b)–(25c). To bypass condition (25c), the designer is obliged to place funicular ribs at the edges and walls below them to reinforce and support the paraboloid shell. The tension forces in the ribs can be provided by Eq. (9), and the vertical supporting force can be evaluated from Eq. (10). The resulting stress function and the initial shape function are displayed in Fig. 11a. In practice, the vertical supporting forces are often provided structural mullions [25].

The designer can also impose condition (25c), which will result in a different solution of the stress function. Then, based on the updated

stress function, the algorithm will find the *tweaked* shape. The algorithm will alternately tweak or modify the stress and shape functions before the converge condition is satisfied. When the iteration does not converge well, the designer can first impose the condition (25c) with lower weight then gradually increase it. In this demonstration, there are four stages in the process. Each stage has a different value of the weight, which is set to be 0.25 in the first 10 iterations (1–10), 0.5 in the following 20 iterations (11–30), 0.75 in another group of 30 iterations (31–60), and eventually to be 1 in the last 40 iterations (61–100). Fig. 10 shows the residual errors evolve along with the iterations. The overall root-mean-square error $\epsilon^{Overall} = \sqrt{\sum(w\varepsilon)^2 / \sum w^2}$ monotonically decreases, and jumps only when the weights are changed.

Fig. 11b shows intermediate stress and shape functions after the first 30 iterations. Then, the stress and shape functions are turned into a free-edged shell after 100 iterations as illustrated in Fig. 11c.

5.5. Steps 6–7: Convergence and export

Pucher's equation (3) suggests the vertical load equals the derivatives of the stress function $\partial_{ij} F$ $(i, j \in x, y)$ time the derivatives of the shape function $\partial_{ij} Z$ $(i, j \in x, y)$ (plus some other terms). When both stress function and shape function are unknown (or both λ_F and λ_Z are unknown), finding the solution set of stress function and shape function is a non-linear problem.

The proposed algorithm adjusts the stress function and shape function alternately. In an individual adjustment, either λ_F or λ_Z is treated as unknown and the other is treated as a given constant vector. In this perspective, the error vector has a linear relation to the unknown coefficient vector (λ_F or λ_Z). The method of linear least square can provide best fit magnitude coefficients without further trials, unlike the method of gradient descent [26].

However, the overall path of the proposed algorithm still shows striking similarities to the gradient descent (Fig. 12). In the gradient descent, there are three features. First, the overall path is monotonically descent. Secondly, each step ends at the tangent point to a contour of the minimizing function. Thirdly, any subsequent step is perpendicular to the previous step. In the proposed algorithm, each adjustment ends at a conditional global minimum, thus the overall path is monotonic and each endpoint tangents to the minimizing function. Moreover, in each iteration step, the updated coefficient vector is either $(\Delta\lambda_Z, \mathbf{0})$ or $(\mathbf{0}, \Delta\lambda_F)$, which is always perpendicular to the previous step.

Furthermore, given that all calibration points are in a non-singular region, thus the smoothness of minimizing function $\sum[w\varepsilon(\lambda)]^2$ is guaranteed. These features ensure that the proposed algorithm shall converge at stationary points where the gradient of the minimizing function is zero: $\frac{\partial}{\partial\lambda}\sum(w\varepsilon)^2 = \mathbf{0}$. Fig. 10 shows that the norm of gradient $\left\|\frac{\partial}{\partial\lambda}\sum(w\varepsilon)^2\right\|$ decays slowly. The slow convergence is due to the problem has strong eccentricity (e.g., the minimum is at the bottom of a narrow valley) [27]. Another optimization scheme might be preferable to accelerate the converge, but the inquiry is outside the scope of this paper.

Practically, the sequence can be terminated if the minimizing function $\sum(w\varepsilon)^2$ is smaller than a prescribed standard, says $(\epsilon^{standard})^2$. If the sequence reaches $\frac{\partial}{\partial\lambda}\sum(w\varepsilon)^2 = \mathbf{0}$ or the descend of $\sum(w\varepsilon)^2$ stagnates before $\sum(w\varepsilon)^2 < (\epsilon^{standard})^2$, then it is necessary to add or rearrange $\boldsymbol{\mu}_i$ and ρ_i in order to enlarge the representable space (14).

Regarding the export, the algorithm shall export the centers $\boldsymbol{\mu}_i$ and the shape coefficients ρ_i as well as the magnitude coefficients $\lambda_{F,i}$ and $\lambda_{Z,i}$. For some users, a graphical output may be more relevant than the numbers. This paper uses sets of stress trajectories, or *principal stress networks*, to visualize the results. The widths of the trajectories are proportional to the stress and are inversely proportional to the distances to the adjacent trajectories. The visualization of the stress network is adapted from [24] for smooth stress trajectories.

6. Shells subjected to both vertical and horizontal loads

This section shows how membrane stresses are affected by the horizontal loads. If the original shape, or the thrust surface, cannot carry the horizontal loads, this section also shows how the thrust surface shall deviate from the original position.

The shape in Fig. 8 is put under this horizontal loading analysis. The shape is regarded as the initial shape in the proposed form-finding algorithm introduced in Section 5. The load cases contain horizontal loads, which are set to be 30% of the p_z in various directions:

$$(p_x, p_y) = 0.3\,\|p_z\|\,(\cos\theta, \sin\theta), \qquad \theta \in \mathbb{R}.$$

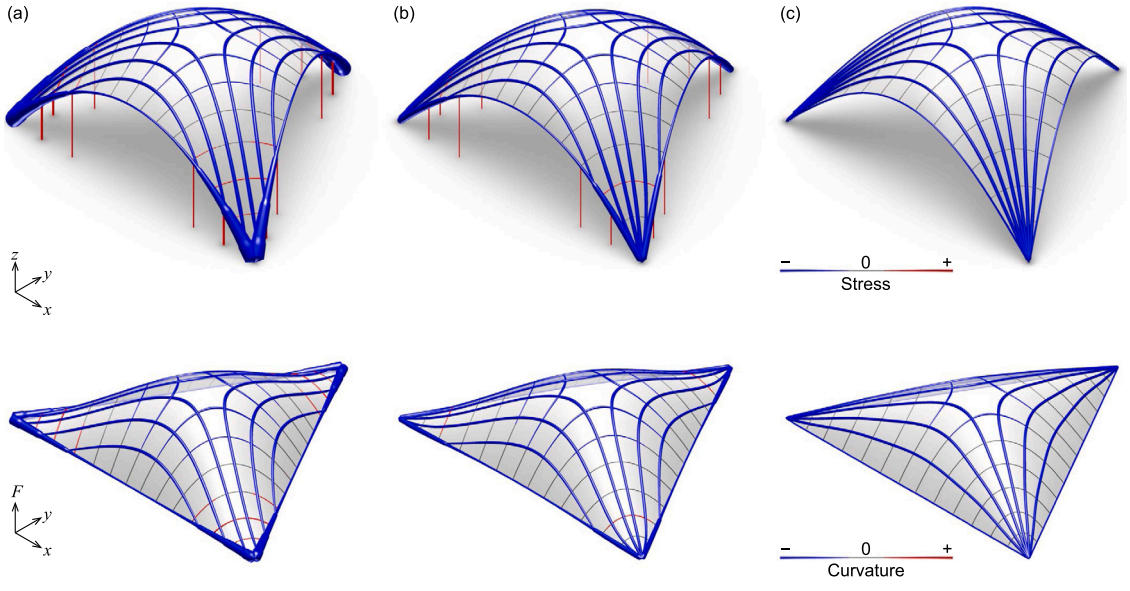


Fig. 11. Shape functions (upper row) and stress functions (lower row) of the shell gradually evolve from a walls-supported paraboloid shape to a free-edge shell. (a) When the paraboloid shell is supported on walls. The shell is pure compressed around the center and has tension stress near the corners. Correspondingly, the stress function is elliptic around the center and hyperbolic near the corners. (b) Evolution toward satisfying free-edges conditions. (c) Result of the last iteration, which has stress concentration at the corners. Correspondingly, the stress function is parabolic near the corners. (color figure in web version).

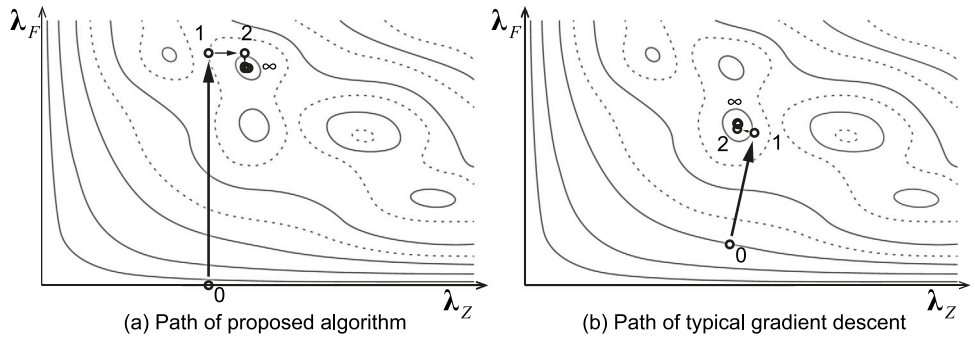


Fig. 12. Converging path to a local minimum $\sum(w\epsilon)^2$.

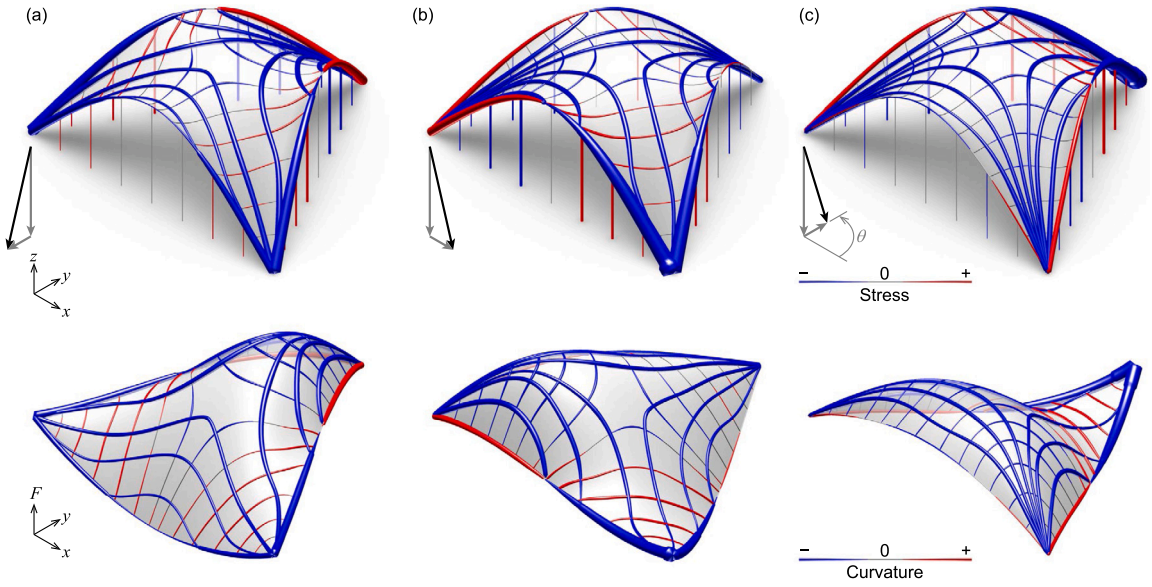


Fig. 13. Shape functions (upper row) and stress functions (lower row) of the walls-supported shell under horizontal loads of different directions. (a) $\theta = -90°$. (b) $\theta = 0°$. (c) $\theta = 90°$. (color figure in web version).

For these uniform horizontal loads, a potential function $V(x, y)$ can be introduced [28]:

$$V(x, y) = p_x x + p_y y,$$

which satisfies $\partial_x V = p_x$ and $\partial_y V = p_y$, and thus the potential function can replace $\int p_x\, dx$ and $\int p_y\, dy$ in Eqs. (26a), (26c).

The membrane shell is supported in two ways. In Section 6.1, the shell's corners are anchored to the foundation and the edges are reinforced by funicular ribs and set on vertical walls, which only provide vertical reactions. In Section 6.2, the ribs and walls are removed. The shell is only supported at the three corners. Two supporting conditions generate quite distinct results. In the first case, the initial shape can sufficiently carry different load cases. In the second case, the shape function has to deviate from the initial geometry.

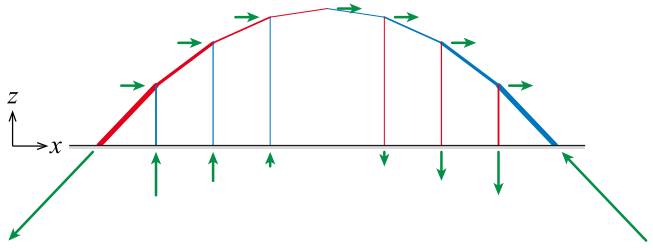


Fig. 14. A wall-supported edge under horizontal loads. The external forces are represented by the green arrows. Meanwhile, the red and blue lines represent tension and compression forces, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

6.1. Walls-supported shell

As previously discussed and shown in Section 5, wall supports can help a non-form-found shape carry vertical loads with membrane stresses. The section further shows how wall supports can help a vertically form-found shape carry horizontal loads.

At the edges, the funicular ribs and supporting walls make the stress function only need to comply with Eqs. (25a), (25b), (25d). Eq. (25a) governs the stress function in the domain, condition (25b) asks the stress function to curve at the edges, and condition (25d) determines the overall elevation and slopes of the stress function. The resulting stress function changes dramatically in response to the horizontal loads. Fig. 13 displays the stress functions and the stress flow one the shape. The shear stress of the shell next to a funicular rib accumulates as the axial force in the rib. The axial force, which is also curved in the vertical plane, requires vertical stress in order to strike the balance. Such vertical stress can be provided by the supporting wall. The stresses can be drawn as concentrated forces as in Fig. 14, which shows the forces induced by a uniform horizontal load.

6.2. Points-supported shell

Another way to set the boundary condition is to remove the ribs and the walls. As a result, when horizontal loads appear, the shape function has to deviate from the initial input. Even if the input is the form-found shape under the vertical load, the deviation is still necessary in order to recover the balance from the altered load condition. The algorithm proposed in Section 5 can provide the solution sets of stress and shape functions.

Fig. 15 displays the solutions of the stress functions in response to the horizontal loads. In the presence of horizontal body forces, the Airy stress functions have to curve following condition (25b), which is shared with the stress functions in Fig. 13. Therefore, the edges of the stress functions in these two figures have the same edges.

However, the stress functions for free edges have to exclusively follow condition (25c), which asks the stress functions to twist specifically at the edges. As the result, the principal orientation of stresses will align with the edges. One can see the curvature lines of stress functions are parallel or normal to the edges. The orientation of the stress functions' principal curvatures is the same as the orientation of the principal stresses, since the potential function $V(x, y)$ is adopted to represent the integrals $\int p_x\,dx$ and $\int p_y\,dy$. Therefore, the parallelism between edges and curvature lines also implies the parallelism between the edges and principal stresses, which is required by condition (25c).

A side effect of condition (25c) is that the initial shape function may no longer satisfy Pucher's equation. The proposed algorithm can be used to find an alternative shape function. After the iterations in the algorithm, one can get the shape functions in Fig. 15. This shape functions have only three pinned supports and no supporting forces from the walls. The shapes have to deviate from the original symmetry geometry and lean into horizontal loads as a cyclist leans into a strong crosswind.

The shape functions of various horizontal loads collectively define a pair of upper and lower envelopes (see Fig. 16). The distance between the envelopes suggests the potential eccentricity of the membrane stress in the shell. For this points-supported shell, the eccentricity at the boundary is larger than that in the center. Meanwhile, the eccentricity at the midpoints of the edges is larger than that around the supports. These envelopes can inform the design of the thickness of a masonry vault [13], the reinforcement arrangement of a concrete shell [29], or the depths of reinforcing beams [30].

7. Concluding remarks

This paper has presented the first form-finding algorithm which can take an initial shape as the input and adjust it under loads in various directions. The algorithm uses the numerical method of radial basis functions to represent Airy stress and shape functions of shells, solves Pucher's equation by the least-squares method, and finds the equilibrium states of membrane shells.

Like other funicular form-finding methods, only the equilibrium equations are concerned in this proposed method. The resulting equilibrium configuration can set a lower bound in the plastic limit theorems, which is useful in the preliminary design of shells. More sophisticated modeling can later be applied to analyze other aspects, such as displacement, strains, buckling, creep, fatigue, etc.

The proposed method can start the form-finding process from either an initial stress or shape function of the shell. The stress functions denote the horizontal components of the membrane stresses, which are equilibrium to the force density in the force density method or the reciprocal diagrams in the thrust network analysis. In contrast to the conventional force-density-like methods, which require the force density to be provided, the proposed method allows users to directly start the form-finding method with a preliminary shape without prior knowledge on stress distribution. If the provided preliminary shape is not compatible with the desired boundary conditions, the algorithm can tweak the shape to meet the conditions.

Horizontal loads have also been integrated into the form-finding algorithm. The analyses of the triangular shell acknowledge that peripheral walls are preferable to keep the shells having minimum stress eccentricities. If no wall is supporting any edge, the algorithm can also suggest the potential stress eccentricities, which can inform the design of reinforcing beams.

Further valuable extensions would be to integrate a fast optimization scheme, develop a double-membrane model for non-membrane shells, and limit membrane stresses to compression for masonry vaults. This paper uses gradient descent as the optimization scheme, but it converges slowly. Another scheme might be preferable to accelerate the process. Regarding the double-membrane model, previous research indicates that the eccentricity of the stress resultants in a shell is dependent on the orientation [9,31]. This observation suggests that, for a general shell, the stress resultants should not be projected onto a single thrust surface. Conversely, a double-membrane model can

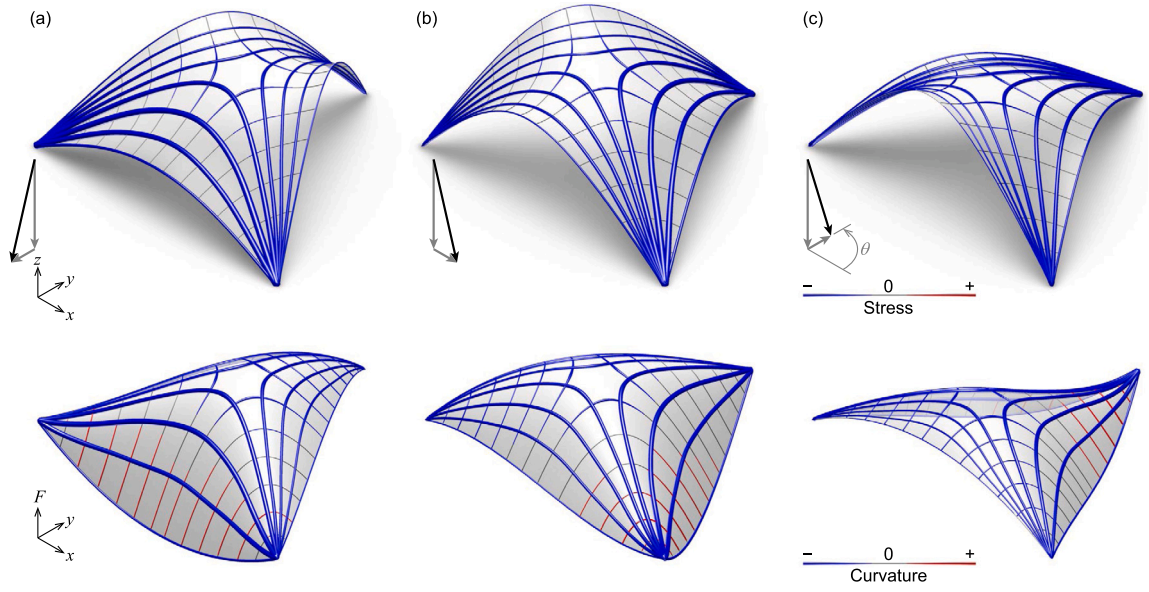


Fig. 15. Shape functions (upper row) and stress functions (lower row) of the points-supported shell under horizontal loads of different directions. (a) $\theta = -90°$. (b) $\theta = 0°$. (c) $\theta = 90°$. (color figure in web version).

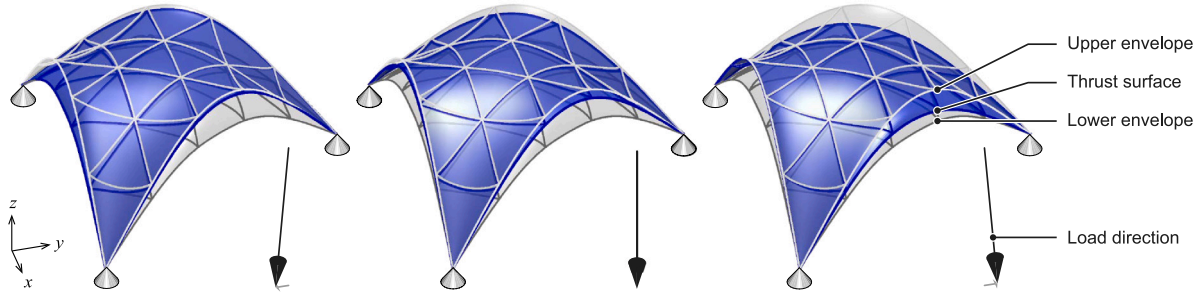


Fig. 16. The envelopes defined by the thrust surfaces under various horizontal loads (color figure in web version).

capture the above observation. A general shell can be conceptually decomposed into two membrane shells sandwiching a middle layer of shear stress. The shear force can be treated as horizontal loads acting on the membranes. Thus, the model can represent non-membrane shells faithfully. Additionally, when both the membrane surfaces are limited to compression, the model can also analyze the equilibrium states of thick vaults.

CRediT authorship contribution statement

Yu-Chou Chiang: Conceptualization, Methodology, Writing – original draft. **Andrew Borgart:** Supervision.

Declaration of competing interest

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

References

[1] Veenendaal D, Block P. An overview and comparison of structural form finding methods for general networks. Int J Solids Struct 2012;49(26):3741–53. http://dx.doi.org/10.1016/j.ijsolstr.2012.08.008.
[2] Barnes MR. Form-finding and analysis of prestressed nets and membranes. Comput Struct 1988;30(3):685–95.
[3] Schek H-J. The force density method for form finding and computation of general networks. Comput Methods Appl Mech Engrg 1974;3(1):115–34. http://dx.doi.org/10.1016/0045-7825(74)90045-0.
[4] O'Dwyer D. Funicular analysis of masonry vaults. Comput Struct 1999;73(1–5):187–97.
[5] Block P, Ochsendorf J. Thrust network analysis: A new methodology for three-dimensional equilibrium. J Int Assoc Shell Spat Struct 2007;48(155):167–73.
[6] Vouga E, Höbinger M, Wallner J, Pottmann H. Design of self-supporting surfaces. ACM Trans Graph 2012;31(4). http://dx.doi.org/10.1145/2185520.2185583.
[7] Maxwell C. On reciprocal figures, frames, and diagrams of forces. Earth Environ Sci Trans R Soc Edinb 1870;26(1):1–40.
[8] Michiels T, Adriaenssens S. Identification of key design parameters for earthquake resistance of reinforced concrete shell structures. Eng Struct 2017;153:411–20.
[9] Tomasello G, Adriaenssens S, Gabriele S. Dynamic behavior of form-found shell structures according to modal and dynamic funicularity. Eng Struct 2019;198:109521.
[10] Marmo F, Rosati L. Reformulation and extension of the thrust network analysis. Comput Struct 2017;182:104–18.
[11] DeJong MJ. Seismic assessment strategies for masonry structures (Ph.D. thesis), Massachusetts Institute of Technology; 2009.
[12] Timoshenko SP, Woinowsky-Krieger S. Theory of plates and shells. McGraw-Hill; 1959.
[13] Heyman J. Equilibrium of shell structures. Oxford: Oxford University Press; 1977.
[14] Csonka P. Theory and practice of membrane shells. Düsseldorf: VDI Verlag; 1987.
[15] Chiang Y-C, Borgart A, Li Q. Finding membrane shells subjected to horizontal body forces with radial basis functions. In *Proceedings of the IASS annual symposium 2019 and structural membranes 2019* Barcelona, 2019, p. 1556–1563.
[16] Buhamann MD. Radial basis functions. Cambridge; 2003.
[17] Chen C, Hon Y, Schaback R. Scientific computing with radial basis functions. 2007, URL http://num.math.uni-goettingen.de/schaback/SCwRBF.pdf.
[18] Miki M, Igarashi T, Block P. Parametric self-supporting surfaces via direct computation of Airy stress functions. ACM Trans Graph 2015;34(4). http://dx.doi.org/10.1145/2766888.
[19] Veenendaal D. Design and form finding of flexibly formed concrete shell structures (Ph.D. thesis), Swiss Federal Institute of Technology in Zürich; 2017.
[20] Airy GB. On the strains in the interior of beams. Philos Trans R Soc Lond 1863;153:49–79.
[21] Arfken GB, Weber HJ, Harris FE. Mathematical methods for physicists. 7th ed.. Boston: Academic Press; 2013.
[22] Boyd JP, Gildersleeve KW. Numerical experiments on the condition number of the interpolation matrices for radial basis functions. Appl Numer Math 2011;61(4):443–59.
[23] Biancolini ME. Fast radial basis functions for engineering applications. Cham: Springer; 2017, http://dx.doi.org/10.1007/978-3-319-75011-8.
[24] Chiang Y-C, Buskermolen P, Borgart A. Discretised Airy stress functions and body forces. In: Advances in architectural geometry 2020. Champs-sur-Marne; 2021, p. 62–83.
[25] Chilton J. The engineer's contribution to contemporary architecture: Heinz Isler. Thomas Telford; 2000.
[26] Curry HB. The method of steepest descent for non-linear minimization problems. Quart Appl Math 1944;2(3):258–61.
[27] Forst W, Hoffmann D. Optimization: theory and practice. Springer Science & Business Media; 2010.
[28] Sadd MH. Elasticity: Theory, applications, and numerics. second ed.. Burlington: Academic Press; 2009.
[29] López López D, Roca P, Liew A, Van Mele T, Block P. Tile vaults as integrated formwork for reinforced concrete: Construction, experimental testing and a method for the design and analysis of two-dimensional structures. Eng Struct 2019;188:233–48. http://dx.doi.org/10.1016/j.engstruct.2019.03.034.
[30] Liew A, López López D, Van Mele T, Block P. Design, fabrication and testing of a prototype, thin-vaulted, unreinforced concrete floor. Eng Struct 2017;137:323–35. http://dx.doi.org/10.1016/j.engstruct.2017.01.075.
[31] Gabriele S, Varano V, Tomasello G, Alfonsi D. R-funicularity of form found shell structures. Eng Struct 2018;157:157–69. http://dx.doi.org/10.1016/j.engstruct.2017.12.014.