REAL-TIME DESIGN OF ARCHITECTURAL STRUCTURES WITH DIFFERENTIABLE MECHANICS AND NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Designing mechanically efficient geometry for architectural structures like shells, towers, and bridges is an expensive iterative process. Existing techniques for solving such inverse mechanical problems rely on traditional direct optimization methods, which are slow and computationally expensive, limiting iteration speed and design exploration. Neural networks would seem to offer a solution, via data-driven amortized optimization, but they often require extensive fine-tuning and cannot ensure that important design criteria, such as mechanical integrity, are met. In this work, we combine neural networks with a differentiable mechanics simulator to develop a model that accelerates the solution of shape approximation problems for architectural structures modeled as bar systems. As a result, our model offers explicit guarantees to satisfy mechanical constraints while generating designs that match target geometries. We validate our model in two tasks, the design of masonry shells and cable-net towers. Our model achieves better accuracy and generalization than fully neural alternatives, and comparable accuracy to direct optimization but in real time, enabling fast and sound design exploration. We further demonstrate the real-world potential of our trained model by deploying it in 3D modeling software and by fabricating a physical prototype. Our work opens up new opportunities for accelerated physical design enhanced by neural networks for the built environment.

1 INTRODUCTION

Mechanical efficiency is required for architectural structures to span hundreds of meters under extreme loads safely with low material volume. An efficient structure sustains loads with small physical element sizes, such as thin bars or slender plates, thus reducing its material footprint. Additionally, shells, towers, and bridges—examples of such systems—must comply with geometric constraints arising from architecture and fabrication requirements to become feasible structures in the built environment. Designing shapes for such long-span structures, which must fulfill mechanical efficiency and geometric constraints, is a complex task requiring substantial domain expertise and human effort. Our goal is to use machine learning to accelerate this challenging task without compromising safety-critical aspects of the design.

One way to approach this problem is to start from the mechanical standpoint, employing a specialized mechanical model that directly computes efficient geometry for structures modeled as a bar systems (Bletzinger and Ramm, 2001; Bletzinger et al., 2005). Unlike standard, finite-element-based mechanical analysis, where one first defines the structure's geometry and then obtains its internal forces, these specialized models – known as *form-finding methods* in structural engineering (Veenendaal and Block, 2012; Adriaenssens et al., 2014)– reverse the relationship between geometry and force to produce mechanically efficient shapes in a forward solve (Shin et al., 2016). As a result, these methods have been successfully applied to design landmark structures with thickness-to-span ratios up to 1:70 (less than that of an eggshell), across a wide palette of materials, including stainless steel (Schlaich, 2018), reinforced concrete (Isler, 1994), and stone (Block et al., 2017).

While utilizing a specialized model can lead to efficient designs, it is difficult to guide solutions toward particular geometries, as the designer only has explicit control of the mechanical behavior and not the shape. To solve this inverse problem, form-finding is complemented with optimization algorithms to

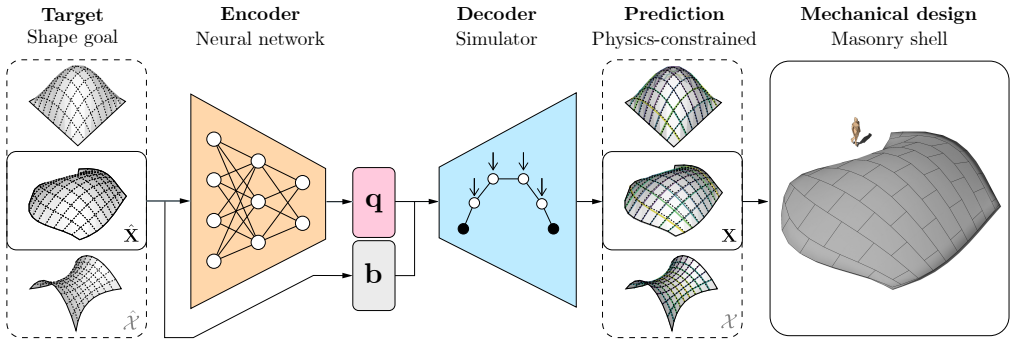


Figure 1: Architecture of our model to amortize the generation of mechanically efficient geometry. Given an input target shape $\hat{\mathbf{X}}$ sampled from a family of shapes $\hat{\mathcal{X}}$, a neural network maps it to a stiffness space $\mathbf{q}$. The stiffnesses, in tandem with boundary conditions $\mathbf{b}$, are then decoded by a mechanical simulator into a physics-constrained shape $\mathbf{X}$ that matches the input. The prediced shape can then be used as the base geometry to design mechanically efficient structures like masonry shells.

find internal force states that satisfy geometric goals (Panozzo et al., 2013; Maia Avelino et al., 2021). For a designer, however, it is cumbersome to perform a time-consuming and computationally intensive optimization when exploring shapes. Practical design requires the evaluation of multiple target shapes to align with geometric desiderata, multiplying computational effort as each shape requires its own optimization. Therefore, using specialized mechanical models and optimization together is effective but inefficient in practice, where shape variety and real-time feedback are essential.

Neural networks (NNs) have shown the potential to accelerate physical design with data-driven surrogate models that amortize inverse problems for more responsive tools. Recent applications include fluid-structure control (Allen et al., 2022) and additive manufacturing (Sun et al., 2021), the design of truss lattices (Bastek et al., 2022), tall buildings (Chang and Cheng, 2020), reticulated shells (Tam et al., 2022) and cable-nets (Mai et al., 2024). However, these purely data-driven approaches require the representation of both the inverse problem and the underlying physics. Even with physics-informed neural networks (PINNs) (Raissi et al., 2019b; Karniadakis et al., 2021) that have specialized architectures (Bastek and Kochmann, 2023; Lu et al., 2021), or that are trained with sophisticated loss balancing schemes (Bischof and Kraus, 2021; Wang et al., 2022), there is **no guarantee of mechanical integrity** in their predictions. Here, we define integrity as the accuracy in predicting the mechanical response of a structure by respecting physical laws. Assurance of mechanical integrity is a foundational tenet in structural design, where poor neural predictions might lead to catastrophic collapse and the loss of human lives. In contrast, mechanical simulators in structural engineering have been developed for decades and offer a principled and interpretable way to model the physics of long-span structures. These models capture the physics by construction. A hybrid solution seems ideal, in which neural network amortization is integrated with differentiable physics models (Belbute-Peres et al., 2020; Thuerey et al., 2022; Um et al., 2021; Oktay et al., 2023; Yang et al., 2022) to construct a class of machine learning models that shift the current paradigm from a physics-informed to a physics-in-the-loop approach in safety-critical applications.

In this paper, we develop a neural surrogate model that couples a neural network with a differentiable mechanics simulator to enable the solution of shape approximation problems for architectural structures in real time (Fig. 1). The coupled model offers advantages over direct gradient-based optimization and current fully neural alternatives for interactive mechanical design. We evaluate our method in two design problems of increasing complexity: masonry shells and cable-net towers. Our contributions are threefold. First, we demonstrate that **our model generates mechanically sound predictions** at higher accuracy than NNs and PINNs of similar architecture. The model exhibits better generalization performance than an equivalent PINN in the masonry shells task. Second, we show that our model reaches comparable accuracy to optimization, but our model is up to four orders of magnitude faster. In the cable-net task, our model provides robust initialization for direct optimization, outperforming the designs generated by optimization initialized with human domain expertise. Third, we showcase the application of a maturing machine learning technique (i.e., coupling learnable and analytical components in the same architecture) to a new, high-impact domain for physical design (i.e., architectural structures). To illustrate its practical impact, we deploy our trained model in a 3D modeling program to design a shell and then fabricate a physical prototype of the predicted geometry.

2 PHYSICS-CONSTRAINED NEURAL FORM DISCOVERY

Our goal is to generate mechanically efficient shapes for architectural structures that approximate target geometries in real time while maintaining mechanical integrity. The challenge is that, because mechanical efficiency is key in architectural structures, it is necessary to reason about designs from the point of view of force balance; but the resulting geometries are a nontrivial function of their mechanical behavior. Thus we seek to use machine learning to efficiently invert this function to generate target designs without compromising their integrity.

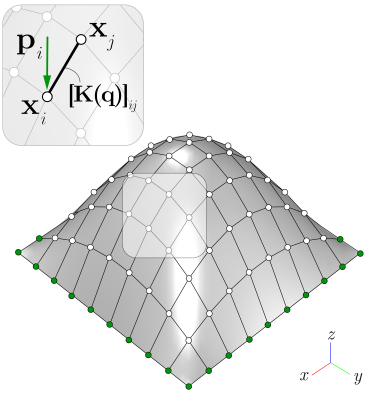


Figure 2: A bar system. In the callout, the stiffness component $[\mathbf{K}(\mathbf{q})]_{ij}$ is indicated at a bar connecting nodes with positions $\mathbf{x}_i$ and $\mathbf{x}_j$. A load $\mathbf{p}_i$ is applied at $\mathbf{x}_i$. In this system, the anchor nodes on the perimeter are fixed.

2.1 COMPUTING EFFICIENT GEOMETRY

We focus on structures modeled as pin-jointed bar systems of N nodes connected by M bars (Figure 2). Each node experiences an external load vector (e.g., self-weight or wind load) and some nodes are constrained to fixed positions (e.g., terrain and anchors). These are the structure's boundary conditions $\mathbf{b} \in \mathbb{R}^L$. After picking bar stiffnesses $\mathbf{q} \in \mathbb{R}^M$, the goal of form-finding is to identify positions of the nodes $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N) \in \mathbb{R}^{N\times 3}$ such that there is no net residual force on the structure.

An efficient structure is one whose loaded configuration is bending and torsion-free, therefore reducing the material volume required to resist applied loads. This configuration minimizes the structure's total strain energy by reducing the contribution of such components, letting a structure sustain applied loads mainly under tensile and compressive axial forces. We can satisfy this property if the shape of a structure modeled as a bar system is in equilibrium. To arrive at equilibrium, the residual force vector $\mathbf{r}_i \in \mathbb{R}^3$ for a free node with position $\mathbf{x}_i$ must be zero. The function $\mathbf{r}_i(\mathbf{X}; \mathbf{q})$ quantifies the difference between the load $\mathbf{p}_i \in \mathbb{R}^3$ applied to node i and the sum of the internal forces of the bars incident to the node, for given node positions, bar stiffnesses and boundary conditions:

$$\mathbf{r}_i(\mathbf{X}; \mathbf{q}) := \sum_{j\in\mathcal{N}(i)} [\mathbf{K}(\mathbf{q})]_{ij}(\mathbf{x}_i - \mathbf{x}_j) - \mathbf{p}_i \tag{1}$$

where $\mathcal{N}(i)$ are the neighbors of node i, and $\mathbf{K}(\mathbf{q}) \in \mathbb{R}^{N\times N}$ is the stiffness matrix as a function of $\mathbf{q}$. The restriction that there is no net residual force can be framed as a constraint in which all of the $\mathbf{r}_i(\mathbf{X}; \mathbf{q}) = \mathbf{0}$. Although this constraint can be solved using direct mechanical simulators, such as form-finding methods (Adriaenssens et al., 2014), the resulting map from the stiffnesses $\mathbf{q}$ to the positions, which we denote $\mathbf{X}(\mathbf{q})$ is implicit and nonlinear, so, difficult to reason about directly.

2.2 DIRECT OPTIMIZATION FOR TARGET SHAPES

Although form-finding methods have the appealing property that they guarantee mechanical efficiency, they do not allow a designer to directly target particular geometries. Moreover, not all geometries are even compatible with mechanical efficiency. If a designer has a target shape $\hat{\mathbf{X}}$, they wish to solve the following optimization problem with respect to $\mathbf{q}$ to approximate $\hat{\mathbf{X}}$ with $\mathbf{X}$:

$$\mathbf{q}^\star = \underset{\mathbf{q}\in\mathbb{R}^M}{\arg\min}\, \mathcal{L}_{\text{shape}}(\mathbf{q}) \qquad \text{where} \qquad \mathcal{L}_{\text{shape}}(\mathbf{q}) := \sum_{i=1}^{N}\sum_{d=1}^{3} |[\mathbf{X}(\mathbf{q})]_{i,d} - [\hat{\mathbf{X}}]_{i,d}|^p \tag{2}$$

and $p > 0$. We call $\mathcal{L}_{\text{shape}}$ the *shape loss*. This objective, which we refer to as *direct optimization*, tries to identify stiffnesses $\mathbf{q}$ which are close to the designer's intent in an ℓ_p sense, while maintaining net zero force balance. Note the optimization setup does not contain any information about the set of physically valid forms and it is simply driven by the minimization of the pointwise difference between shapes. Conventionally, this nonlinear optimization problem needs to be solved numerically in the inner loop of a design process, but that is slow and computationally costly.

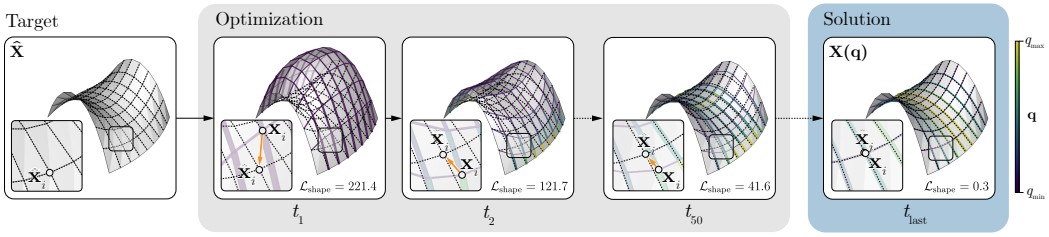


Figure 3: Inverse form-finding. To generate a mechanically congruent shape $\mathbf{X}(\mathbf{q})$ that approximates an arbitrary target $\hat{\mathbf{X}}$, traditional methods like direct optimization find bar stiffnesses $\mathbf{q}$ that minimize the shape loss $\mathcal{L}_{\text{shape}}$, but only after several iterations t. Our model amortizes this computationally taxing process during inference while guaranteeing physics, enabling real-time and sound design.

2.3 Amortized shape matching

Rather than performing many costly optimizations within a design loop, we use a machine learning model to amortize the solution over a family of target shapes $\hat{\mathcal{X}}$. A neural network takes as input the designer's intent $\hat{\mathbf{X}}$, and outputs a set of stiffnesses $\mathbf{q}$ such that $\mathbf{X}(\mathbf{q}) \approx \hat{\mathbf{X}}$. Our model architecture resembles an autoencoder (Figure 1). First, a neural encoder $\mathcal{E}_\phi$ maps the target shape $\hat{\mathbf{X}}$ into $\mathbf{q}$. Then, a mechanical simulator *decodes* the associated shape $\mathbf{X}(\mathbf{q})$, subject to the boundary conditions $\mathbf{b}$. The key property of this construction is that **the resulting shape is mechanically sound even if the neural network is inaccurate**; the failure mode is not a lack of structural integrity, but a shape that does not match the targets very well.

Neural encoder The encoder is a neural network with learnable parameters ϕ that ingests the targets and projects them into stiffness space, $\mathcal{E}_\phi : \mathbb{R}^{N\times 3} \to \mathbb{R}^M$. For simplicity, we cast our problem as a point-wise matching task and use a multilayer perceptron (MLP) as the encoder, although we are not restricted to that. Regardless of the neural network specification, the output representation must be strictly positive. This is necessary for compatibility with our mechanical simulator to avoid null stiffness values that bear limited physical meaning in our representation of an architectural structure. We satisfy this requirement by applying a strictly positive nonlinearity to the last layer of the encoder.

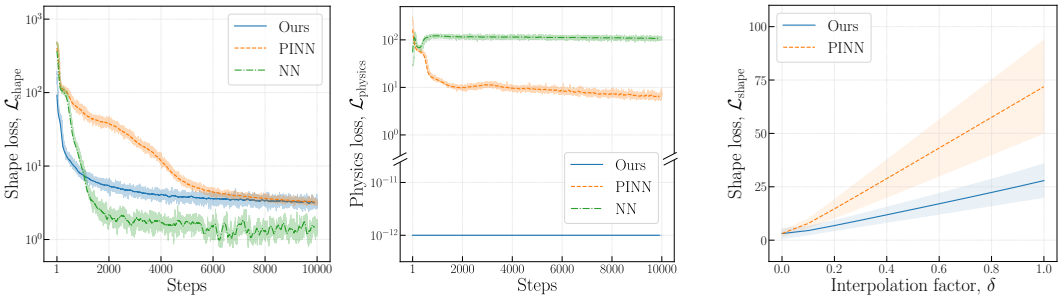
One of the advantages of our mechanical simulator is that it enables us to prescribe tensile or compressive bar forces *a priori*. As a result, rather than making these force directions a learnable feature, we build this bias into the encoder architecture by scaling the strictly positive embedding of the last layer by a force direction vector $\mathbf{s} \in \mathbb{R}^M$. The scaling factors $s \in \{-1, 1\}$ indicate the direction of the internal axial force of every bar: a negative factor s prescribes a compressive force, and a positive factor, a tensile force. Our encoder thus calculates the stiffness vector $\mathbf{q}$ as:

$$\mathbf{q} = \mathbf{s} \odot (\sigma(\mathbf{h}) + \tau) \tag{3}$$

where σ is the strictly positive nonlinearity, $\mathbf{h}$ is the encoder's last layer embedding, $\tau \geq 0$ is a fixed scalar shift that specifies a minimum absolute stiffness value for the entries of $\mathbf{q}$ (akin to a box constraint in numerical optimization), and $\odot$ indicates element-wise product. Setting a lower bound on the stiffnesses $\mathbf{q}$ with τ guides the learning process towards particular solutions since the map from $\mathbf{q}$ to $\mathbf{X}$ is not unique (Van Mele et al., 2012), and provides numerical stability when amortizing over structures with complex force distributions (Section 4.2).

Mechanical decoder The decoder gives the latent space of our model a physical meaning since it represents the inputs of a mechanical simulator. To fulfill equilibrium in a pin-jointed bar system, the relationship between the stiffnesses $\mathbf{q}$, the shape $\mathbf{X}$, and the loads $\mathbf{P} \in \mathbb{R}^{N\times 3}$ must satisfy $\mathbf{K}(\mathbf{q})\,\mathbf{X} - \mathbf{P} = \mathbf{0}$. Solving this equation with standard mechanical simulators (Xue et al., 2023; Wu, 2023) is possible, but generally requires second-order methods. Controlling the force signs adds numerical complexity, but it is a desirable property to design structures built from tailored materials that are strong only in tension or compression (e.g., masonry blocks or steel cables). Here, we utilize the force density method (Schek, 1974) as our simulator. Appendix D offers an extended description of this simulator, but at a high level, it is a specialized form-finding method that linearizes the equilibrium constraint (Equation 1) by assuming independence between stiffnesses and geometry. This reduces the computation of $\mathbf{X}$ with target force signs to a linear solve $\mathbf{X}(\mathbf{q}) : \mathbb{R}^M \to \mathbb{R}^{N\times 3}$:

$$\mathbf{X}(\mathbf{q}) = \mathbf{K}(\mathbf{q})^{-1}\mathbf{P} \tag{4}$$



(a) Train curves for $\mathcal{L}_{\text{shape}}$ and $\mathcal{L}_{\text{physics}}$

(b) $\mathcal{L}_{\text{shape}}$ on interpolated data

Figure 4: Loss curves of the shell design task. (a) Our model learns a meaningful representation that minimizes the shape loss $\mathcal{L}_{\text{shape}}$ while fully satisfying the mechanics of compression-only shells, as $\mathcal{L}_{\text{physics}}$ is zero within numerical precision throughout training. (b) Shape loss of our model and the PINN baseline on test data interpolated between doubly-symmetric ($\delta = 0$) and asymmetric ($\delta = 1$) shapes. Our model's accuracy decays at a lower rate than the PINN's.

Training To amortize the shape-matching problem, we look for model parameters $\phi^\star$ that minimize the expected value of the shape loss over a family of target shapes $\hat{\mathcal{X}}$:

$$\phi^\star = \arg\min_{\phi} \mathbb{E}_{\hat{\mathbf{X}} \sim \hat{\mathcal{X}}} \left[\sum_{i=1}^{N} \sum_{d=1}^{3} |[\mathbf{X}(\mathcal{E}_\phi(\hat{\mathbf{X}}))]_{i,d} - [\hat{\mathbf{X}}]_{i,d}|^p \right] \tag{5}$$

We train our model via first-order stochastic gradient descent, averaging the loss values over batches at each training step. Appendix E provides training specifications. We generate training data by sampling batches of target shapes $\hat{\mathbf{X}}$ from a task-specific family of shapes $\hat{\mathcal{X}}$ parametrized by a probability distribution (Section 4). Our model can be trained end-to-end because the encoder and decoder are both implemented in a differentiable programming environment (Bradbury et al., 2018). As a result, reverse-mode automatic differentiation can seamlessly backpropagate the physics-based gradients that tune the neural network parameters.

3 Evaluation

We evaluate model performance by measuring the inference wall time of the trained model in addition to the value of the shape loss $\mathcal{L}_{\text{shape}}$ over a test batch of shapes of size B. We compare the performance of our model to three other baselines: a fully neural approach (NN), a fully neural model augmented with a physics-informed loss (PINN), and traditional direct optimization. The fully neural approach replaces the differentiable simulator in our model with a learnable decoder mirroring the encoder's architecture, with the inclusion of the boundary conditions $\mathbf{b}$ as inputs, as is the case with the differentiable simulator. The fully neural model is then trained to minimize the shape loss $\mathcal{L}_{\text{shape}}$ without any additional regularization, highlighting that an information bottleneck is insufficient to automatically guarantee physically plausible designs. The second baseline extends the fully neural approach by adding an explicit physics loss:

$$\mathcal{L}_{\text{physics}} = \mathbb{E}_{\hat{\mathbf{X}} \sim \hat{\mathcal{X}}} \left[\sum_{i=1}^{N} ||\mathbf{r}_i(\mathbf{X}(\mathcal{E}_\phi(\hat{\mathbf{X}})))||_2 \right] \tag{6}$$

The physics loss $\mathcal{L}_{\text{physics}}$ is the governing equation of our problem, and it measures the N residual forces $\mathbf{r}_i$ in the nodes. For a shape to be mechanically sound in our setup, $\mathcal{L}_{\text{physics}}$ must be zero within numerical precision (i.e., 1×10^{-12}) as per our physics constraint in Equation 1. We reason that the additional term should provide a training signal to the encoder and decoder such that they learn how to solve the shape-matching tasks and the physics concurrently. The third baseline takes advantage of the differentiable physics simulator and directly optimizes the parameter space $\mathbf{q}$ input to the decoder to minimize the shape loss via deterministic gradient-based optimization on a per-shape basis.

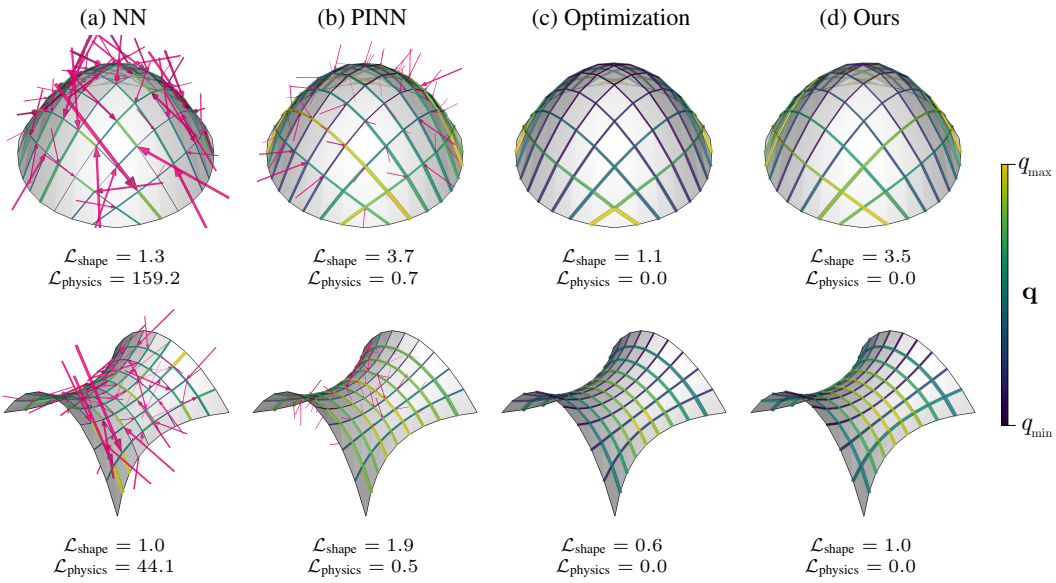


Figure 5: Shape matching for shell design. While the NN and PINN models approximate the targets, they cannot suppress the residual forces (pink arrows). The stiffnesses $\mathbf{q}$ predicted by our model are similar to direct optimization's, indicating our model learns a good neural representation of the task.

Table 1: Model evaluation on masonry shells task. We report mean loss values and the standard deviation per shape on the test set of 100 target shapes, in addition to the test inference run time.

	NN	PINN	Optimization	Ours
$\mathcal{L}_{\text{shape}} \downarrow$	1.5 ± 0.4	3.1 ± 1.2	0.8 ± 1.2	3.0 ± 2.0
$\mathcal{L}_{\text{physics}} \downarrow$	104.3 ± 48.6	0.6 ± 0.3	0.0 ± 0.0	0.0 ± 0.0
Time [ms] $\downarrow$	0.3 ± 0.0	0.3 ± 0.0	5810.1 ± 1870.3	0.6 ± 0.1

4 Experiments

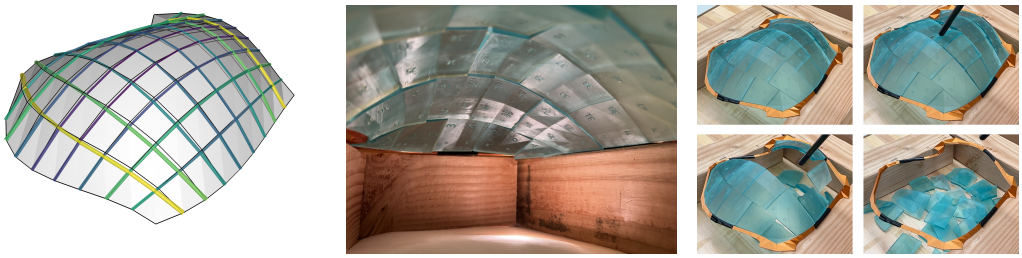
We test our model to amortize shape approximation tasks for masonry shells and cable-net towers. These two tasks represent a broad class of structural typologies dealt with by designers in practice.

4.1 Masonry shells

Our first experiment identifies suitable bar stiffness values for unreinforced masonry shells. Masonry shells sustain external loads with span-to-thickness ratios as low as 1:50 despite being built from materials that are strong in compression and weak in other loading conditions (Block et al., 2017). Shapes that maximize internal compressive axial forces enable this efficient behavior.

An expressive class of masonry shells can be constructed by surfaces parameterized by a Bezier patch. The shape of the patch is in turn described by the position of a grid of C control points $\mathbf{c}$ in Cartesian space (see Appendix F). We apply limit state analysis to model masonry shells as pin-jointed bar systems (Maia Avelino et al., 2021). For this task, we restrict the space of target shapes to a square grid of width $w = 10$ and $C = 16$ control points. In particular, we consider doubly-symmetric shapes of constant discretization, where $N = 100$ and $M = 180$. We apply a constant area load of 0.5 per unit area, representing the self-weight of the shell. The nodes on the perimeter are anchored.

To solve this task, we look for bar stiffnesses that yield shapes that best fit the target geometries, and whose internal axial forces are compressive ($\mathbf{q} < 0$). We satisfy the compression-only requirement by construction with our encoder by setting $\mathbf{s} = -\mathbf{1}$. We use $\tau = 0$ and $p = 1$. Figure 4a illustrates the stochastic loss curves during training for our model and the two neural baselines (NN and PINN). The fully neural approaches achieve a low shape loss but are unable to converge w.r.t the physics



$\mathcal{L}_{\text{shape}} = 16.6$ Interior view Collapse sequence

(a) Model prediction (b) Physical prototype

Figure 6: (a) Our model accurately predicts asymmetric shapes despite being trained exclusively on doubly symmetric geometries. (b) We build a predicted shape as a tabletop prototype of a masonry shell. The bricks stand in equilibrium due to the appropriate shape—they are not mechanically attached as shown by the snapshots of the collapse sequence caused by an external perturbation.

loss. These approaches fail to learn a meaningful intermediary representation $\mathbf{q}$ that satisfies the task physics, unlike our physics-in-the-loop model where this requirement is satisfied by construction. In Figure 5, we plot two representative target shapes for each model, the predicted stiffnesses, the resulting geometries, and the residual forces.

The trends we observe in training are repeated during inference. Table 1 reports the average test loss and the standard deviation for one design in a test group of 100 different structures. Direct optimization achieves the lowest shape loss. The baselines and our model all offer significant speedup w.r.t. optimization. While the NN and the PINN generate accurate shape approximations, the designs predicted by these fully neural baselines are mechanically unfeasible because the residual forces fail to vanish. The magnitude of the physics loss indicates that the structure is missing balancing forces to achieve equilibrium. If we prescribe typical values for a masonry shell constructed out of bricks with an average area of 1m^2 and thickness of 0.05 m (a $1:20$ aspect ratio), density of approximately 2000 kg/m^3 then a total residual force of 1 kN, representative of a unit value of $\mathcal{L}_{\text{physics}}$ in Table 1, would destabilize the structure with an acceleration of 10 m/s^2 in multiple directions. Consequently, the NN and the PINN shapes are unstable, and building masonry shells guided by these predictions can lead to collapse. In contrast, our model and direct optimization satisfy the physical constraints a priori, but our model generates accurate predictions up to four orders of magnitude faster, and offers significant speedup compared to optimization for geometries with an equivalent shape loss value (Appendix B).

Next, we investigate the out-of-distribution generalization of the PINN and our model, a desirable property to build robust neural surrogates for physical design. To this end, we first generate a set of 100 asymmetric Bezier surfaces. We create this new set by sampling control points from the same design space and discretization as in the doubly-symmetric case (Appendix F). Then, we produce input data by interpolating between the shapes in the asymmetric and symmetric sets. We evaluate the shape loss of the model predictions at increasing interpolation factors δ, where $\delta = 0$ denotes double symmetry and $\delta = 1$ indicates full asymmetry. We do an equivalent study for $\mathcal{L}_{\text{physics}}$ in Appendix B. Our model consistently possesses better out-of-distribution performance. Figure 4b demonstrates that the loss of our model predictions decays at least at half of the PINN's rate, and with a lower spread, as we increase the data asymmetry. At $\delta = 1$, the loss of our model is 2.5 times lower. The performance disparity at this point between our model and the PINN is evidenced by the example in the third column of Figure 13b. The PINN prediction is not only a worse fit to the target, but also the residuals are fifteen times higher than at $\delta = 0$, demonstrating that the physics-in-the-loop model offers enhanced generalization over the physics-informed approach.

To further demonstrate the applicability of our model under real-world conditions, Figure 10 in Appendix A shows screen captures of our trained model in action, assisting a designer in exploring different shapes for a masonry shell in an industry-grade 3D modeling software. We also validate our model's generalization and transfer to physical applications by fabricating a tabletop masonry shell based on one of our model predictions on asymmetric targets displayed in Figure 6a. The shell has a thickness-to-span ratio of 1:50 on its longest span. After tessellating the predicted shape, we manufacture the individual bricks and assemble them. The prototype is stable and can resist

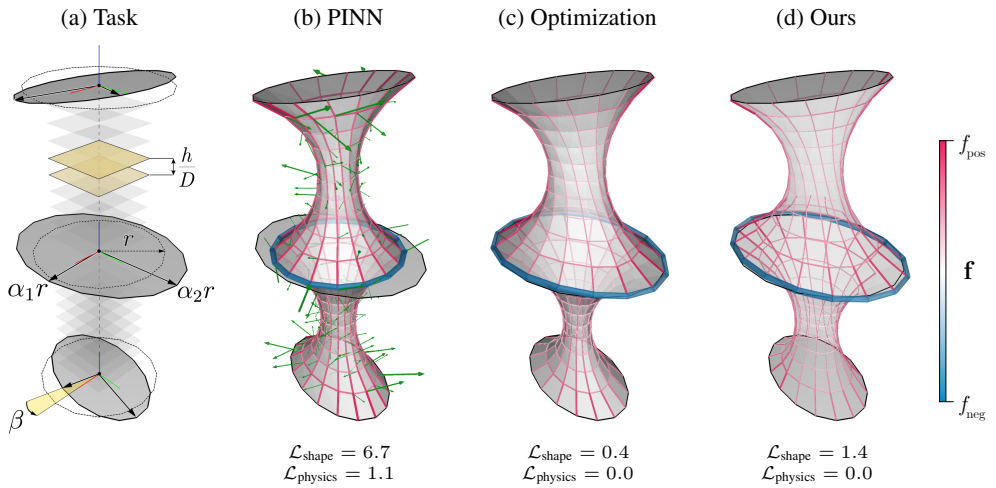


Figure 7: Predictions for cable-net structures. (a) Schematic depicting the design space. (b) - (d) Reconstructions of target shapes, showing internal tensile (red) and compressive (blue) forces $\mathbf{f}$. The PINN model can neither reconstruct the target surface nor ensure a net zero force balance, while our method closely approximates the target shape akin to the solution output by direct optimization.

gravity without glue or mechanical connectors, indicating that the structure can work predominantly under internal compressive forces as required by masonry shells, and proving the feasibility of our predictions under real-world constraints. See Figure 6b for details.

4.2 Cable-net towers

Guided by the success of our initial experiment, we turn to a more complex problem: the design of cable-net towers. Such systems are extremely lightweight, with low structural mass-to-occupied volume ratios, often being found as observation or cooling towers (Adriaenssens et al., 2014). To explore the design space of these towers, we focus on optimizing the shape and orientation of horizontal compression rings encircled by vertically spanning tensile cable-nets.

Figure 7a gives an overview of the task setup. The cable-net towers consist of two tensile nets that interface at a compressive ring at mid-height. Each cable-net tower comprises $D = 21$ rings with 16 points each, spaced at equal h/D intervals over a total of height $h = 10$. The discretization of the structure is $N = 335$ and $M = 656$. We parametrize the geometry of the bottom, middle, and top rings with an ellipse of radii $\alpha_1 r$ and $\alpha_2 r$ and rotation angle on the plane β (Appendix C). The nodes on the top and bottom rings are anchors. The shape approximation task here consists of finding valid cable-net shapes that match the target shape of the middle compression ring. We are interested in geometries where the tension rings are planar. The shape loss incorporates these requirements, measuring the squared ℓ_2 norm (i.e., $p = 2$) between predictions and targets. We set the entries of $\mathbf{s}$ to -1 and 1 to enforce the force direction in the middle ring and the net bars, respectively.

The mechanical behavior of cable-net towers introduces modeling difficulties because the resulting problem comprises members that are either under tensile or compressive forces. This results in an ill-conditioned system due to the interaction between bars of different force signs and can lead to singular systems and other instabilities at force levels near zero (Cai et al., 2018). While the ill-conditioning can be overcome in the forward pass, in the backward pass, it can hinder our model from learning meaningful representations as a result of poorly scaled gradients (Figure 8). To alleviate these numerical instabilities, we clip the global gradient norm to 0.01 and shift the outputs of our last layer to establish a lower bound of $\tau = 1$ on the stiffness space. Additionally, we add a regularization term to the total loss

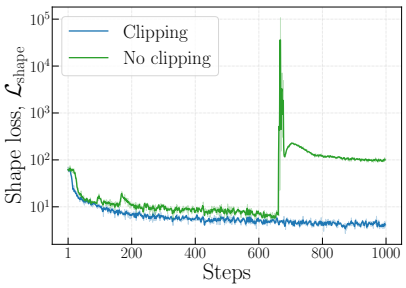


Figure 8: Train loss curve of the cable-net task, with and without clipping.

Table 2: Model evaluation on the cable-net towers task. We report mean loss values and the standard deviation per shape on the test set of 100 towers, in addition to the test inference run time.

	NN	PINN	Optimization			Ours
			Randomized	Expert	with Ours	
$\mathcal{L}_{\text{shape}} \downarrow$	7.5 ± 3.1	7.4 ± 3.4	14.2 ± 28.1	0.3 ± 0.3	0.2 ± 0.3	0.6 ± 0.7
$\mathcal{L}_{\text{physics}} \downarrow$	45.8 ± 0.4	1.2 ± 0.1	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Time [ms] ↓	0.4 ± 0.0	0.4 ± 0.0	1902.0 ± 1297.3	1876.2 ± 821.5	1430.1 ± 838.9	4.6 ± 0.1

to steer our model towards a uniform stiffness distribution:

$$\mathcal{L}_{\text{reg}} = \text{Var}\left(\mathbf{q}_{\text{pos}}\right) + \text{Var}\left(\mathbf{q}_{\text{neg}}\right) \tag{7}$$

The regularizer measures the variance, $\text{Var}(\mathbf{q})$, of the structure's bar stiffnesses, for tensile $\mathbf{q}_{\text{pos}}$ and compressive $\mathbf{q}_{\text{neg}}$ values, over all the B samples in a batch. We scale $\mathcal{L}_{\text{reg}}$ by a constant factor λ. We employ $\lambda = 10$ to train our model and the baselines.

Figures 7b-7d show an example of the predicted cable-net shapes. In Appendix C, we show that our model predictions cover the task space satisfactorily, generating accurate and mechanically congruent cable-net shapes whose rings radii and in-plane rotation vary within a range of r and $\pi/6$, respectively. Table 2 demonstrates that our model generates shapes that match the targets with a 3 times tighter fit than the PINN and the NN models. These purely neural baselines make faster predictions because they do not run an explicit physics simulator. However, the baselines are unable to learn the cable-net physics because $\mathcal{L}_{\text{physics}}$ is nonzero within numerical precision, like we identified in the shell task.

Next, we compare the two approaches that guarantee physics: ours and direct optimization. Converging to a good local optimum with optimization is contingent on adequate initial stiffnesses $\mathbf{q}$. In Figure 9 we thus analyze the expected convergence rate of direct optimization on the test set with four different initializations. Random initial guesses converge to poor local optima as the shape loss is the highest among our experiments, highlighting the relevance of picking adequate $\mathbf{q}$ values to optimize the geometry of structures with complex mechanical behavior. If the initial $\mathbf{q}$ values are handpicked by a human expert, optimization is more accurate than our model and the other baselines. Optimization with expert initialization achieves an equivalent shape loss to our model's at about $1/5$ of the total convergence time (350 ms) given by Table 2. However, our trained model matches optimization in only 1 inference step that is on average three orders of magnitude faster and is free of potentially expensive human intervention—key attributes for fast and automated design tools.

Lastly, we investigate the effect of using our model and the PINN predictions as the initial stiffness input to direct optimization to further refine the tower designs. This combination results in the most accurate matches in the cable-net task, converging faster and consistently achieving a lower shape loss than direct optimization with expertly initialized parameters (Figure 9). However, the PINN initialization is, on average, slower to converge and only matches our model towards the end of the curve. In both cases, the neural models provide a better initial guess than the human expert for optimization in this task, highlighting the potential of utilizing neural networks and standard optimization techniques in tandem to arrive at better-performing designs.

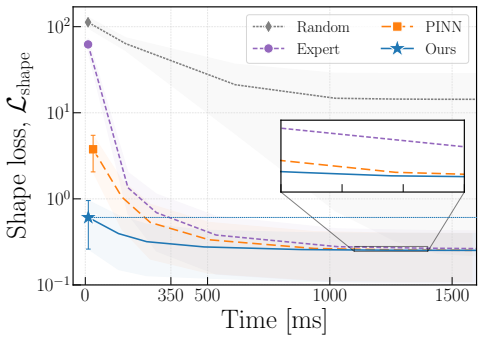


Figure 9: Convergence curves of direct optimization with four distinct initializations.

5 Related Works

Differentiable mechanical simulators Machine learning and automatic differentiation have successfully obtained derivatives of complex forward mechanical models by either learning a differentiable surrogate with a neural network, or by implementing the analytical physics model in a

differentiable programming environment. For neural surrogates, Xue et al. (2020) used an autoencoder to simulate 2D metamaterials, while Zheng et al. (2020) learned the forward solution of 3D graphic statics on compression-dominant shells. Differentiable simulator implementations for mechanics problems include the finite element method for 3D solids (Xue et al., 2023) and isogeometric analysis of 3D membranes (Oberbichler et al., 2021). Specifically for form-finding, differentiable simulators exist for matrix structural analysis (Wu, 2023), and for combinatorial equilibrium modeling (Pastrana et al., 2023a). Our work utilizes one such form-finding simulator and connects it with a neural network to amortize an inverse problem.

Amortization models for mechanical design Neural surrogates that amortize inverse problems have garnered attention for their ability to approximate nonlinear relationships in mechanical systems between target properties and input parameters. At the centimeter scale, Bastek et al. (2022) address the inversion of the structure-property map in truss metamaterials, enabling the discovery of optimal configurations. In Oktay et al. (2023), amortized models were used to generate actuation policies to deform 2D cellular metamaterials to several target configurations. Focusing on architectural structures at the meter scale, Hoyer et al. (2019) proposed a neural basis for computing material distributions that minimize compliance for buildings in 2D; while Chang and Cheng (2020) amortized with graph neural networks the design of the cross sections of the beams and columns of buildings in 3D against multiple load cases. Favilli et al. (2024) applied geometric deep learning and a differentiable simulator to gridshell structures for low-strain energy shapes. Unlike their work, which trains a neural network for a single problem, we amortize over multiple inverse problems. Tam et al. (2022) also amortize a shape-matching problem with form-finding like in our work, but they do so using a fully neural approach that must capture both physics and solve the matching task simultaneously.

6 Conclusion

We presented a physics-in-the-loop neural model that expedites the solution of shape-matching problems to design mechanically efficient architectural structures. By embedding prior physics knowledge in a neural network and training end-to-end, our model learns representations that solve a family of inverse problems with precision while enforcing mechanical constraints by construction; which is where current neural approaches fall short. While we do not discard that physics-informed networks could push the physics loss closer to zero in the limit of hyperparameter tuning and network size, the physics guarantees and the stronger generalization performance of our model set it apart as a more robust approach to support design in practical settings. Our method generates mechanically sound geometries in milliseconds, with accuracy comparable to gradient-based optimization algorithms. The speed and reliability of our model enable real-time design exploration of long-span structures modeled as pin-jointed bar systems, such as masonry shells and cable-net towers.

6.1 Limitations and future work

Although not a cure-all for mechanical design problems, our work evidences that domain expertise in structural engineering and machine learning is necessary to address numerical pathologies that can stem from infusing physics into neural networks (Wang et al., 2021; 2022; Metz et al., 2022). In our case, the simulator can yield a stiff ill-conditioned system which affects training in the presence of structures with complex stress distributions. Tackling these instabilities requires knowledge of the physics of the problem being modeled, while also utilizing gradient stabilization techniques common in machine learning, such as gradient clipping.

Like past methods at this intersection, our model requires devising specialized parameter spaces for learning (Allen et al., 2022). Consequently, one of our trained models may not necessarily generalize to different types of architectural structures. Another limitation is that our model is currently restricted to a fixed topology and would require retraining if the discretization changed. In the future, however, we expect that applying methods such as graph networks (Battaglia et al., 2018; Pfaff et al., 2020) to our encoder will enable moving beyond one bar connectivity. We additionally note that the choice of bar stiffnesses for a given system is not unique and it is potentially appealing to present to the designer a diversity of possible solutions by reformulating our model in a variational setting (Kingma and Welling, 2014; Salamanca et al., 2023). That is another exciting avenue for future research.

REFERENCES

Sigrid Adriaenssens, Philippe Block, Diederik Veenendaal, and Chris Williams, editors. *Shell Structures for Architecture: Form Finding and Optimization*. Routledge/ Taylor & Francis Group, London ; New York, 2014. ISBN 978-0-415-84060-6.

Kelsey R. Allen, Tatiana Lopez-Guevara, Kimberly Stachenfeld, Alvaro Sanchez-Gonzalez, Peter Battaglia, Jessica Hamrick, and Tobias Pfaff. Physical Design using Differentiable Learned Simulators. *arXiv:2202.00728 [cs]*, February 2022. URL http://arxiv.org/abs/2202.00728.

Jan-Hendrik Bastek and Dennis M. Kochmann. Physics-Informed Neural Networks for shell structures. *European Journal of Mechanics - A/Solids*, 97:104849, January 2023. ISSN 0997-7538. doi: 10.1016/j.euromechsol.2022.104849. URL https://www.sciencedirect.com/science/article/pii/S0997753822002790.

Jan-Hendrik Bastek, Siddhant Kumar, Bastian Telgen, Raphaël N. Glaesener, and Dennis M. Kochmann. Inverting the structure–property map of truss metamaterials by deep learning. *Proceedings of the National Academy of Sciences*, 119(1):e2111505119, January 2022. doi: 10.1073/pnas.2111505119. URL http://www.pnas.org/doi/10.1073/pnas.2111505119.

Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. June 2018. URL https://arxiv.org/abs/1806.01261v3.

Filipe De Avila Belbute-Peres, Thomas Economon, and Zico Kolter. Combining Differentiable PDE Solvers and Graph Neural Networks for Fluid Flow Prediction. In *Proceedings of the 37th International Conference on Machine Learning*, pages 2402–2411. PMLR, November 2020. URL https://proceedings.mlr.press/v119/de-avila-belbute-peres20a.html.

Rafael Bischof and Michael Kraus. Multi-objective loss balancing for physics-informed deep learning. 2021. doi: 10.13140/RG.2.2.20057.24169. URL http://rgdoi.net/10.13140/RG.2.2.20057.24169.

Kai-Uwe Bletzinger and Ekkehard Ramm. Structural optimization and form finding of light weight structures. *Computers and Structures*, 2001.

Kai-Uwe Bletzinger, Roland Wüchner, Fernaß Daoud, and Natalia Camprubí. Computational methods for form finding and optimization of shells and membranes. *Computer Methods in Applied Mechanics and Engineering*, 194(30-33):3438–3452, August 2005. ISSN 00457825. doi: 10.1016/j.cma.2004.12.026.

Philippe Block, Tom van Mele, Matthias Rippmann, and Noelle C. Paulson. *Beyond Bending: Reimagining Compression Shells*. Edition Detail, Munich [Germany], 2017. ISBN 978-3-95553-390-8.

Mathieu Blondel, Quentin Berthet, Marco Cuturi, Roy Frostig, Stephan Hoyer, Felipe Llinares-López, Fabian Pedregosa, and Jean-Philippe Vert. Efficient and Modular Implicit Differentiation, October 2022. URL http://arxiv.org/abs/2105.15183.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, and Skye Wanderman-Milne. JAX: Composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Jianguo Cai, Xinyu Wang, Xiaowei Deng, and Jian Feng. Form-finding method for multi-mode tensegrity structures using extended force density method by grouping elements. *Composite Structures*, 187:1–9, March 2018. ISSN 0263-8223. doi: 10.1016/j.compstruct.2017.12.010. URL http://www.sciencedirect.com/science/article/pii/S026382231733009X.

Kai-Hung Chang and Chin-Yi Cheng. Learning to simulate and design for structural engineering. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1426–1436. PMLR, 2020. URL http://proceedings.mlr.press/v119/chang20a.html.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs), February 2016. URL http://arxiv.org/abs/1511.07289.

DeepMind, Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Antoine Dedieu, Claudio Fantacci, Jonathan Godwin, Chris Jones, Ross Hemsley, Tom Hennigan, Matteo Hessel, Shaobo Hou, Steven Kapturowski, Thomas Keck, Iurii Kemaev, Michael King, Markus Kunesch, Lena Martens, Hamza Merzic, Vladimir Mikulik, Tamara Norman, George Papamakarios, John Quan, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Laurent Sartran, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Miloš Stanojević, Wojciech Stokowiec, Luyu Wang, Guangyao Zhou, and Fabio Viola. The DeepMind JAX Ecosystem, 2020. URL http://github.com/google-deepmind.

Andrea Favilli, Francesco Laccone, Paolo Cignoni, Luigi Malomo, and Daniela Giorgi. Geometric deep learning for statics-aware grid shells. *Computers & Structures*, 292:107238, February 2024. ISSN 0045-7949. doi: 10.1016/j.compstruc.2023.107238. URL https://www.sciencedirect.com/science/article/pii/S0045794923002687.

Ehsan Haghighat, Maziar Raissi, Adrian Moure, Hector Gomez, and Ruben Juanes. A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. *Computer Methods in Applied Mechanics and Engineering*, 379:113741, June 2021. ISSN 0045-7825. doi: 10.1016/j.cma.2021.113741. URL https://www.sciencedirect.com/science/article/pii/S0045782521000773.

Stephan Hoyer, Jascha Sohl-Dickstein, and Sam Greydanus. Neural reparameterization improves structural optimization. September 2019. URL https://arxiv.org/abs/1909.04240v2.

Heinz Isler. Concrete Shells Derived from Experimental Shapes. *Structural Engineering International*, 4(3):142–147, August 1994. ISSN 1016-8664. doi: 10.2749/101686694780601935. URL https://doi.org/10.2749/101686694780601935.

George Em Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, May 2021. ISSN 2522-5820. doi: 10.1038/s42254-021-00314-5. URL https://www.nature.com/articles/s42254-021-00314-5.

Patrick Kidger and Cristian Garcia. Equinox: Neural networks in JAX via callable PyTrees and filtered transformations, October 2021. URL http://arxiv.org/abs/2111.00254.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv:1312.6114 [cs, stat]*, May 2014. URL http://arxiv.org/abs/1312.6114.

Dieter Kraft. Algorithm 733: TOMP–Fortran modules for optimal control calculations. *ACM Transactions on Mathematical Software*, 20(3):262–281, September 1994. ISSN 0098-3500, 1557-7295. doi: 10.1145/192115.192124. URL https://dl.acm.org/doi/10.1145/192115.192124.

Lu Lu, Raphaël Pestourie, Wenjie Yao, Zhicheng Wang, Francesc Verdugo, and Steven G. Johnson. Physics-Informed Neural Networks with Hard Constraints for Inverse Design. *SIAM Journal on Scientific Computing*, 43(6):B1105–B1132, January 2021. ISSN 1064-8275. doi: 10.1137/21M1397908. URL https://epubs.siam.org/doi/10.1137/21M1397908.

Dai D. Mai, Tri Diep Bao, Thanh-Danh Lam, and Hau T. Mai. Physics-informed neural network for nonlinear analysis of cable net structures. *Advances in Engineering Software*, 196:103717, October 2024. ISSN 0965-9978. doi: 10.1016/j.advengsoft.2024.103717. URL https://www.sciencedirect.com/science/article/pii/S0965997824001248.

R. Maia Avelino, A. Iannuzzo, T. Van Mele, and P. Block. Assessing the safety of vaulted masonry structures using thrust network analysis. *Computers & Structures*, 257:106647, December 2021. ISSN 0045-7949. doi: 10.1016/j.compstruc.2021.106647. URL https://www.sciencedirect.com/science/article/pii/S0045794921001693.

Luke Metz, C. Daniel Freeman, Samuel S. Schoenholz, and Tal Kachman. Gradients are Not All You Need, January 2022. URL http://arxiv.org/abs/2111.05803.

T. Oberbichler, R. Wüchner, and K.-U. Bletzinger. Efficient computation of nonlinear isogeometric elements using the adjoint method and algorithmic differentiation. *Computer Methods in Applied Mechanics and Engineering*, 381:113817, August 2021. ISSN 00457825. doi: 10.1016/j.cma.2021.113817. URL https://linkinghub.elsevier.com/retrieve/pii/S0045782521001535.

Deniz Oktay, Mehran Mirramezani, Eder Medina, and Ryan P. Adams. Neuromechanical Autoencoders: Learning to Couple Elastic and Neural Network Nonlinearity, January 2023. URL http://arxiv.org/abs/2302.00032.

D. Panozzo, P. Block, and O. Sorkine-Hornung. Designing Unreinforced Masonry Models. *ACM Transactions on Graphics - SIGGRAPH 2013*, 32(4):91:1–91:12, July 2013. doi: 10.1145/2461912.2461958.

Rafael Pastrana, Patrick Ole Ohlbrock, Thomas Oberbichler, Pierluigi D'Acunto, and Stefana Parascho. Constrained Form-Finding of Tension–Compression Structures using Automatic Differentiation. *Computer-Aided Design*, 155:103435, February 2023a. ISSN 0010-4485. doi: 10.1016/j.cad.2022.103435. URL https://www.sciencedirect.com/science/article/pii/S0010448522001683.

Rafael Pastrana, Deniz Oktay, Ryan P. Adams, and Sigrid Adriaenssens. JAX FDM: A differentiable solver for inverse form-finding. In *Differentiable Almost Everything Workshop of the 40th International Conference on Machine Learning*, Hawaii, USA, 2023b. URL https://openreview.net/forum?id=Uu9OPgh24d.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. Learning Mesh-Based Simulation with Graph Networks. October 2020. URL https://arxiv.org/abs/2010.03409v4.

M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, February 2019a. ISSN 0021-9991. doi: 10.1016/j.jcp.2018.10.045. URL https://www.sciencedirect.com/science/article/pii/S0021999118307125.

M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, February 2019b. ISSN 0021-9991. doi: 10.1016/j.jcp.2018.10.045. URL https://www.sciencedirect.com/science/article/pii/S0021999118307125.

Robert McNeel & Associates. Rhinoceros 3D, 2024. URL https://rhino3d.com.

Luis Salamanca, Aleksandra Anna Apolinarska, Fernando Pérez-Cruz, and Matthias Kohler. Augmented Intelligence for Architectural Design with Conditional Autoencoders: Semiramis Case Study. In Christoph Gengnagel, Olivier Baverel, Giovanni Betti, Mariana Popescu, Mette Ramsgaard Thomsen, and Jan Wurm, editors, *Towards Radical Regeneration*, pages 108–121, Cham, 2023. Springer International Publishing. ISBN 978-3-031-13249-0. doi: 10.1007/978-3-031-13249-0_10.

H.-J. Schek. The force density method for form finding and computation of general networks. *Computer Methods in Applied Mechanics and Engineering*, 3(1):115–134, January 1974. ISSN 0045-7825. doi: 10.1016/0045-7825(74)90045-0. URL https://www.sciencedirect.com/science/article/pii/0045782574900450.

Mike Schlaich. Shell Bridges - and a New Specimen Made of Stainless Steel. *Journal of the International Association for Shell and Spatial Structures*, 59(3):215–224, September 2018. ISSN 1028-365X. doi: 10.20898/j.iass.2018.197.027.

Hijung V. Shin, Christopher F. Porst, Etienne Vouga, John Ochsendorf, and Frédo Durand. Reconciling Elastic and Equilibrium Methods for Static Analysis. *ACM Transactions on Graphics*, 35(2):1–16, May 2016. ISSN 0730-0301, 1557-7368. doi: 10.1145/2835173.

Xingyuan Sun, Tianju Xue, Szymon M. Rusinkiewicz, and Ryan P. Adams. Amortized Synthesis of Constrained Configurations Using a Differentiable Surrogate. *arXiv:2106.09019 [cs]*, June 2021. URL http://arxiv.org/abs/2106.09019.

Kam-Ming Mark Tam, Tom Mele, and Philippe Block. Trans-topological learning and optimisation of reticulated equilibrium shell structures with Automatic Differentiation and CW Complexes Message Passing. In *Proceedings of the IASS 2022 Symposium Affiliated with APCS 2022 Conference*, Beijing, China, June 2022. International Association for Shell and Spatial Structures (IASS) and Asian-Pacific Conference on Shell and Spatial Structures (APCS).

Nils Thuerey, Philipp Holl, Maximilian Mueller, Patrick Schnell, Felix Trost, and Kiwon Um. Physics-based Deep Learning, April 2022. URL http://arxiv.org/abs/2109.05237.

Kiwon Um, Robert Brand, Yun, Fei, Philipp Holl, and Nils Thuerey. Solver-in-the-Loop: Learning from Differentiable Physics to Interact with Iterative PDE-Solvers. *arXiv:2007.00016 [physics]*, January 2021. URL http://arxiv.org/abs/2007.00016.

Tom Van Mele, Lorenz Lachauer, Matthias Rippmann, and Philippe Block. Geometry-Based Understanding of Structures. *Journal of the International Association for Shell and Spatial Structures*, 53(4):285–295, December 2012.

D. Veenendaal and P. Block. An overview and comparison of structural form finding methods for general networks. *International Journal of Solids and Structures*, 49(26):3741–3753, December 2012. ISSN 0020-7683. doi: 10.1016/j.ijsolstr.2012.08.008. URL http://www.sciencedirect.com/science/article/pii/S002076831200337X.

Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and Mitigating Gradient Flow Pathologies in Physics-Informed Neural Networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, January 2021. ISSN 1064-8275. doi: 10.1137/20M1318043. URL https://epubs.siam.org/doi/abs/10.1137/20M1318043.

Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why PINNs fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, January 2022. ISSN 0021-9991. doi: 10.1016/j.jcp.2021.110768. URL https://www.sciencedirect.com/science/article/pii/S002199912100663X.

Gaoyuan Wu. A framework for structural shape optimization based on automatic differentiation, the adjoint method and accelerated linear algebra. *Structural and Multidisciplinary Optimization*, 66(7):151, June 2023. ISSN 1615-1488. doi: 10.1007/s00158-023-03601-0. URL https://doi.org/10.1007/s00158-023-03601-0.

Tianju Xue, Thomas J. Wallin, Yigit Menguc, Sigrid Adriaenssens, and Maurizio Chiaramonte. Machine learning generative models for automatic design of multi-material 3D printed composite solids. *Extreme Mechanics Letters*, 41:100992, November 2020. ISSN 23524316. doi: 10.1016/j.eml.2020.100992. URL https://linkinghub.elsevier.com/retrieve/pii/S2352431620302182.

Tianju Xue, Shuheng Liao, Zhengtao Gan, Chanwook Park, Xiaoyu Xie, Wing Kam Liu, and Jian Cao. JAX-FEM: A differentiable GPU-accelerated 3D finite element solver for automatic inverse design and mechanistic data science. *Computer Physics Communications*, 291:108802, October 2023. ISSN 0010-4655. doi: 10.1016/j.cpc.2023.108802. URL https://www.sciencedirect.com/science/article/pii/S0010465523001479.

Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J. Ramadge. Learning Physics Constrained Dynamics Using Autoencoders. *Advances in Neural Information Processing Systems*, 35:17157–17172, December 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/6d5e035724687454549b97d6c805dc84-Abstract-Conference.html.

Hao Zheng, Vahid Moosavi, and Masoud Akbarzadeh. Machine learning assisted evaluations in structural design and construction. *Automation in Construction*, 119:103346, November 2020. ISSN 0926-5805. doi: 10.1016/j.autcon.2020.103346. URL http://www.sciencedirect.com/science/article/pii/S0926580520309262.

Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw.*, 23(4): 550–560, December 1997. ISSN 0098-3500. doi: 10.1145/279232.279236. URL https://dl.acm.org/doi/10.1145/279232.279236.

A REAL-TIME DESIGN IN CAD SOFTWARE

We deploy our trained model in a 3D modeling software called Rhino3D (Robert McNeel & Associates, 2024) as illustrated in Figure 10. Rhino3D supports traditional computer-aided design workflows and Grasshopper (Figure 10 top left), its visual programming extension, allows the creation of new software features via custom Python scripts. We load our model as a Grasshopper plugin via Python and test it to support the real-time exploration of shapes for masonry shells.

We describe a design exploration session next. A designer models a Bezier surface in Rhino3D by hand and imports it into Grasshopper. Then, as the designer moves the control points of the Bezier, the geometry automatically updates. Our model generates new compression-only shapes (see the bar systems rendered in Figure 10). Note that as we detail in Section 4, we trained our model on fully symmetric surfaces but the adequate generalization it exhibits to asymmetric geometries makes it possible to support the designer during their exploratory session.

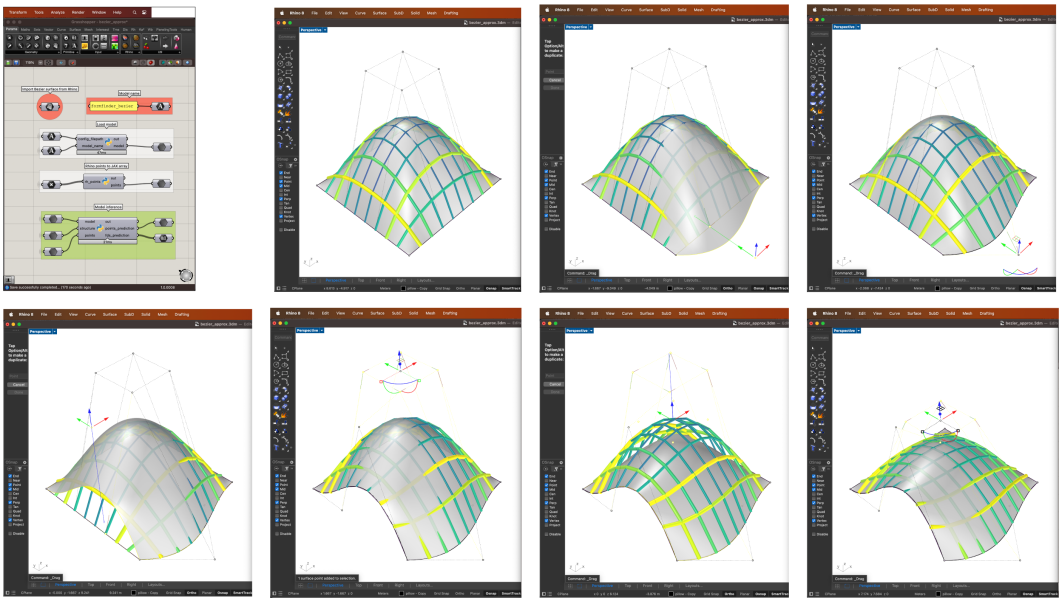


Figure 10: A demonstration of our trained model providing predictions of compression-only shapes in real time. The first screenshot shows the environment running our custom code. The screenshot in the second column and first row shows the initial input Bezier surface with control points. Every subsequent pair of screenshots shows the designer moving Bezier control points and our model reacting to the designer's intent by approximating the shape with one that is compression-only.

B ADDITIONAL STUDIES FOR SHELL DESIGN

B.1 COMPARISON WITH DIRECT OPTIMIZATION

Both our model and direct optimization satisfy the physical constraints a priori, but they generate predictions at drastically different speeds. Since optimization is an iterative approach, we compare the evolution of the shape loss over run time to that of our method, which predicts the bar stiffnesses of a structure in one step. Figure 11 shows that optimization reaches the best predictive performance among the baselines (Table 1), but only after convergence, which incurs run time over 5000 ms (Table 1). Even though optimization matches the shape loss of our model in 40% of the expected convergence time, our approach offers a speedup of above $450\times$ as it generates designs of equivalent accuracy in only a few milliseconds.

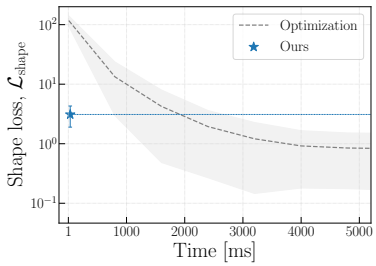


Figure 11: Shape loss evolution of direct optimization for shell design.

B.2 GENERALIZATION ON THE PHYSICS LOSS

In Section 4.1, we discussed the generalization capacity of our model and a PINN —the two approaches that are aware of the inverse problem physics— by observing the rate of change of the shape loss $\mathcal{L}_{\text{shape}}$ as we morph the geometry of the target shapes from doubly-symmetric to asymmetric. Both models were trained solely on the former type.

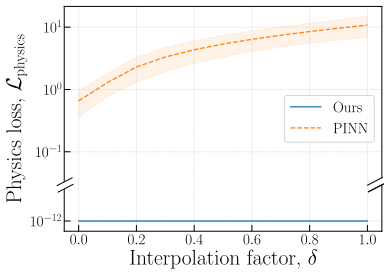


Figure 12: Physics loss changes as the shapes vary between double symmetry ($\delta = 0$) and full asymmetry ($\delta = 1$).

We now quantify the impact of perturbing the test data distribution on the models' capacity to preserve the problem's physics. Figure 12 depicts the evolution of the physics loss $\mathcal{L}_{\text{physics}}$ in logarithmic scale. The performance of the PINN model rapidly deteriorates as the distribution changes from doubly-symmetric to asymmetric shapes, until the loss becomes one order of magnitude higher than at the start.

This finding illustrates that reasonable PINN performance in-distribution is not conducive to good generalization w.r.t. physics information out-of-distribution. In contrast, our model satisfies the physics by construction, so $\mathcal{L}_{\text{physics}}$ remains constant and effectively zero within numerical precision regardless of the geometry of the target shapes. Figure 13 depicts this trend with examples.

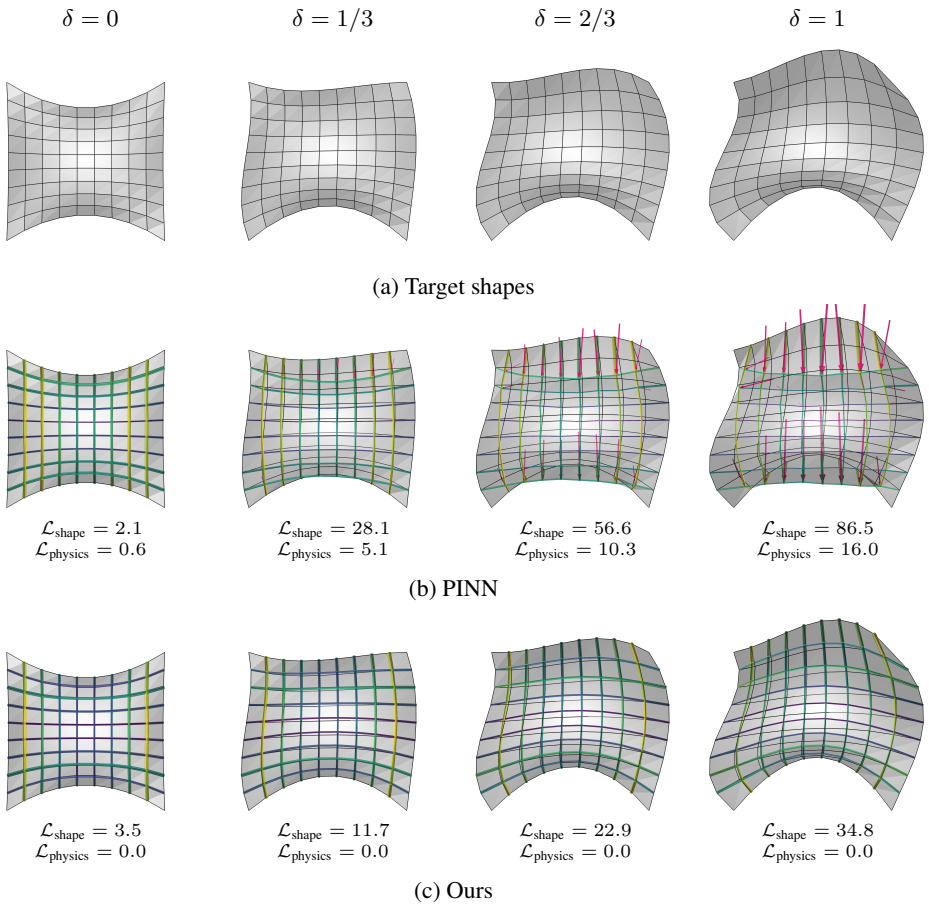


Figure 13: Predicted shapes for shell design. (a) The targets are interpolated between doubly-symmetric ($\delta = 0$) and fully asymmetric ($\delta = 1$) shapes. (b) The accuracy of the PINN decays quickly w.r.t. $\mathcal{L}_{\text{shape}}$ and $\mathcal{L}_{\text{physics}}$ as asymmetry increases. (c) Our model's shape accuracy deteriorates, but at a lower rate, and the problem physics are fulfilled by construction since $\mathcal{L}_{\text{physics}} = 0$. Top view.

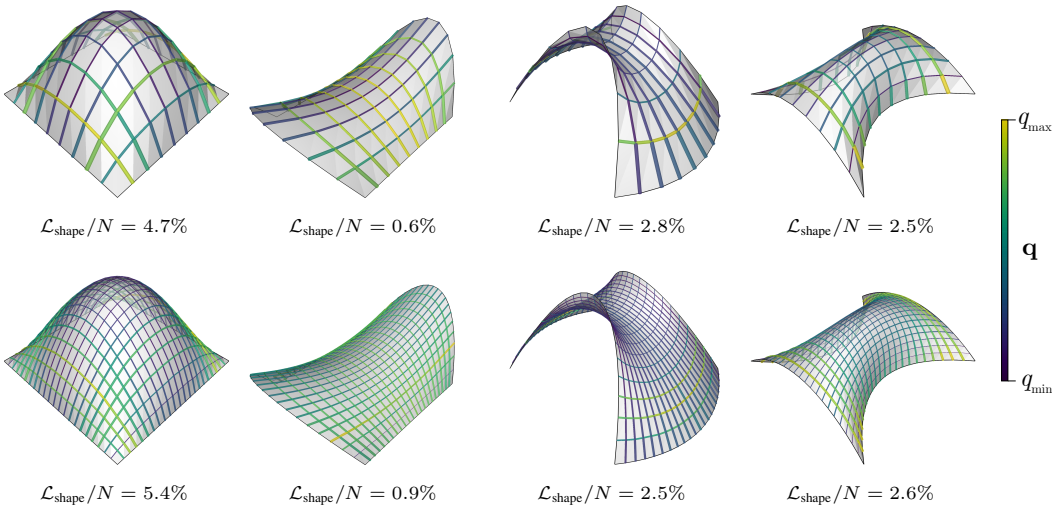


Figure 14: Our model makes accurate predictions across multiple discretizations. Top row: $N = 100$. Bottom row: $N = 529$. The predictions sastisfy the problem physics and are made in milliseconds.

Table 3: Comparison between our model and direct optimization for varying problem sizes in the shells task. We report the mean loss values and standard deviations per shape on a test set normalized by N, and the test inference run time. The size of the test set is $B = 100$, except in optimization for $M = 1012$. There, $B = 10$ due to hardware limitations. The last column on the right lists the number of direction optimization problems required to match or exceed the training time of our model.

N	M	Optimization		Ours			
		$\mathcal{L}_{\text{shape}}/N$ [%] ↓	Time [s] ↓	$\mathcal{L}_{\text{shape}}/N$ [%] ↓	Test time [ms] ↓	Train time [s] ↓	# Opt ↓
100	180	0.8 ± 1.2	5.8 ± 1.9	3.0 ± 2.0	0.6 ± 0.1	140	25
256	480	1.0 ± 1.5	287.8 ± 115.8	3.6 ± 2.3	3.5 ± 0.1	1928	7
529	1012	0.8 ± 0.9	5483.1 ± 1946.0	3.5 ± 2.4	10.2 ± 0.3	6405	2

B.3 Training cost amortization

We measure trade-offs between our model and direct optimization in terms of training versus test inference time in problems up to $N > 500$ and $M > 1000$, where N and M are the number of nodes and bars in a structure, respectively. In these experiments, we keep the architecture of our MLP encoder, except for the input dimension of the first layer, and the output dimension in the last layer, matching the larger values of N and M. The training and the direct optimization settings repeat the configuration in Appendix E. These tests also reflect the overhead of the mechanical simulator in training since enlarging the problem size also increments the size of $\mathbf{K}$, the bottleneck in Equation 4.

Table 3 demonstrates that our model remains feasible to amortize inverse problems on larger structures. While training and inference efforts grow with the problem size, the expected inference time at the finest discretization is only over 10 milliseconds, which is suitable for real-time design exploration. Remarkably, the shape loss $\mathcal{L}_{\text{shape}}$ normalized by N changes marginally as we enlarge the problem size, revealing the ability of our physics-in-the-loop model to generate adequate fits to the target shapes across various discretizations despite the minimal changes in the encoder architecture (Figure 14).

The amortization cost of training our model decreases relative to direct optimization as the problem size grows (Table 3). Relative to the reference case in Section 4.1, the optimization runtime to match a single shape surges by two orders of magnitude for $2\times$ more bars, and by three orders of magnitude for $5\times$ more bars. The number of optimization runs to justify training decreases by $3\times$ every time the number of bars doubles.

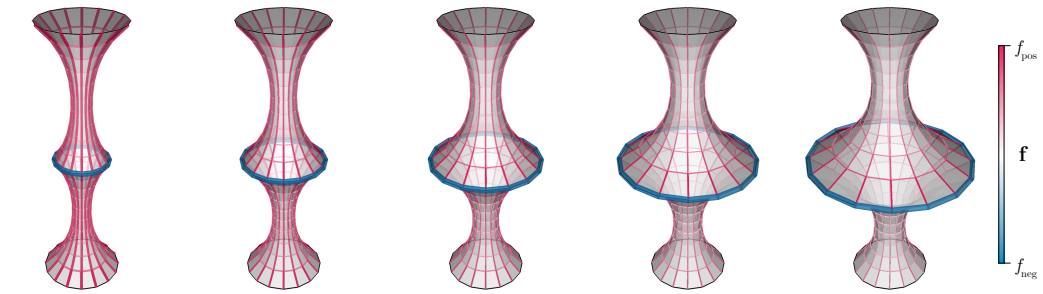
C SHAPE EXPLORATION FOR CABLE-NET TOWER DESIGN

C.1 DATA GENERATION

To generate data for the cable-net task described in Section 4.2, we parametrize the geometry of the bottom, middle, and top rings with an ellipse of radii $\alpha_1 r$ and $\alpha_2 r$ and rotation angle β. After setting the reference radius to $r = h/5$, we generate shape variations by sampling the scale factors $\alpha_i \in [1/2, 3/2]$, and the angles $\beta \in [-\pi/12, \pi/12]$ from a uniform distribution. See Figure 7a for a graphical description.

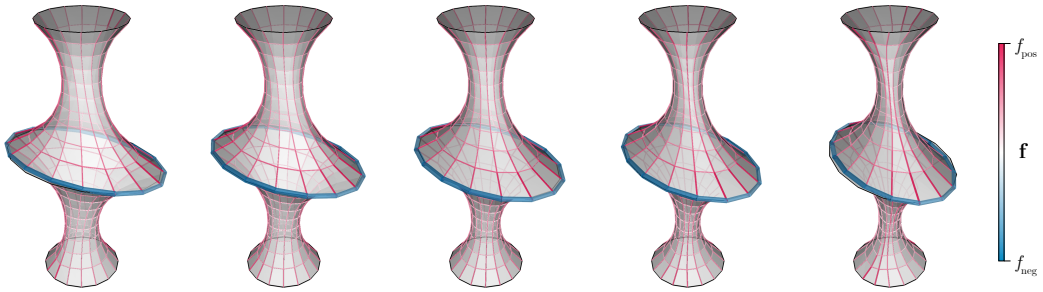
C.2 SHAPE EXPLORATION

We carry out two experiments that modify the size and the rotation of the middle compressive ring in a cable-net tower to illustrate the geometric range of our model predictions. In both experiments, we keep the scale of the radii of the top and bottom disks circular and equal by setting $\alpha_1 = \alpha_2 = 3/4$. In the first experiment, shown in Figure 15, we show our model predictions as we linearly scale the radius α of the middle circular ring from $\alpha = 1/2$ to $\alpha = 3/2$. In Figure 16, the middle ring is elliptical ($\alpha_1 = 1/2$ and $\alpha_2 = 3/2$), and we twist it around the z Cartesian axis in five steps between $-\pi/12$ and $\pi/12$. The predictions in both tests match the target geometry. Note that the distribution of the bar forces $\mathbf{f}$ changes per structure due to the geometric changes on the middle ellipse.



$\alpha = 1/2$ $\alpha = 3/4$ $\alpha = 1$ $\alpha = 5/4$ $\alpha = 3/2$

Figure 15: Our model predictions on the cable-net tower design task. We gradually increase the radius scale factor α of the middle compression circular ring of the tower from $\alpha = 1/2$ to $\alpha = 3/2$.



$\beta = -\pi/12$ $\beta = -\pi/24$ $\beta = 0$ $\beta = \pi/24$ $\beta = \pi/12$

Figure 16: Our model produces accurate predictions for the design of a cable-net tower as we vary the twist angle β of its middle elliptical ring in the range $[-\pi/12, \pi/12]$.

D MECHANICAL SIMULATOR

We employ the force density method (FDM) (Schek, 1974) as our mechanical simulator. The FDM is a form-finding method that computes torsion and bending-free shapes on pin-jointed bar systems with N nodes and M bars, by mapping the bars' stiffness vector $\mathbf{q} \in \mathbb{R}^M$ to node positions $\mathbf{X} \in \mathbb{R}^{N\times 3}$; subject to boundary conditions $\mathbf{b} \in \mathbb{R}^L$. The stiffness q_m –or the *force density*– of bar m sets the ratio between its internal force f_m and its length l_m in the equilibrium configuration:

$$q_m = \frac{f_m}{l_m} \tag{8}$$

Since a length is strictly positive, a negative q_m indicates an internal compression force in the bar and a positive q_m a tensile force. In the FDM, a portion of nodes of size N_s is fixed (i.e., anchors), and another of size N_u is free to displace. The size of the partitions is arbitrary as long as $N = N_s + N_u$. For the geometry of a bar system to be in equilibrium, the residual force vector $\mathbf{r}_i$ at each free node i must be zero (Equation 1). To satisfy this constraint, the FDM calculates the free node positions $\mathbf{X}_u \in \mathbb{R}^{N_u\times 3}$ by solving a linear system of equations:

$$\mathbf{X}_u(\mathbf{q}) = \mathbf{K}(\mathbf{q})_u^{-1}\,[\mathbf{P}_u - \mathbf{K}(\mathbf{q})_s\,\mathbf{X}_s] \tag{9}$$

where the submatrices $\mathbf{K}(\mathbf{q})_u \in \mathbb{R}^{N_u\times N_u}$ and $\mathbf{K}(\mathbf{q})_s \in \mathbb{R}^{N_u\times N_s}$ are created by the rows and the columns in the stiffness matrix $\mathbf{K}(\mathbf{q}) \in \mathbb{R}^{N\times N}$ that correspond to the free and the fixed nodes in the structure. The FDM assembles $\mathbf{K}(\mathbf{q})$ based on the connectivity between bars and the nodes. The value of the ij-th entry in the stiffness matrix is defined as:

$$[\mathbf{K}(\mathbf{q})]_{ij} = \begin{cases} \sum_{m\in\mathcal{N}(i)} q_m & \text{if } i = j \\ -q_m & \text{if nodes } i \text{ and } j \text{ are connected by bar } m \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

where $\mathcal{N}(i)$ denotes the bars connected to node i. The loads applied to the nodes $\mathbf{P} \in \mathbb{R}^{N\times 3}$ and the positions of the node anchors $\mathbf{X}_s \in \mathbb{R}^{N_s\times 3}$ are the boundary conditions $\mathbf{b}$ of the form-finding process, and are inputs of the simulator. To create the boundary conditions vector $\mathbf{b}$, we concatenate $\mathbf{P}$ and $\mathbf{X}_s$, and then flatten the resulting matrix into one vector. Once we know $\mathbf{X}_u$, we concatenate it with $\mathbf{X}_s$ to obtain all the nodal positions in equilibrium $\mathbf{X}$ of the bar structure.

E IMPLEMENTATION DETAILS

We implement our work in JAX (Bradbury et al., 2018) and Equinox (Kidger and Garcia, 2021). We use JAX FDM (Pastrana et al., 2023b) as our differentiable simulator and Optax (DeepMind et al., 2020) for derivatives processing. For our model and the neural baselines (NN and PINN), we train a separate model instance per task to minimize the loss expectation over a batch of samples of size B. For optimization, we directly minimize the loss for every shape in the batch, one shape at a time.

Training and inference for all models is executed on a CPU, on a Macbook Pro laptop with an M2 chip. This choice of hardware is motivated by the type of devices structural designers have access to in practice but also demonstrates the potential for even better model performance on other more powerful hardware, like GPU accelerators. We train all three models using Adam with default parameters, except for the learning rate, for a fixed number of 10,000 steps. All the models' hyperparameters are calibrated to maximize predictive performance on the test set via a random search over 3 seeds.

We employ SLSQP (Kraft, 1994) and L-BFGS-B (Zhu et al., 1997), two deterministic gradient-based optimizers, for direct optimization, as implemented in JAXOPT (Blondel et al., 2022) with default settings. To make a fair comparison, we also perform all the direct optimization experiments on CPU and the same laptop. Additionally, we impose box constraints on the bar stiffnesses $\mathbf{q}$ (the optimization variables) in direct optimization to prescribe an explicit lower bound on the force signs and values as we do for the last layer of our encoder via τ, the scalar shift in the last layer of the encoder (Section 2.3). We use the same values of τ that we report in Section 4. Additionally, we set an explicit upper bound on the individual bar stiffnesses q so that $|q| \leq 20$, where $|\cdot|$ denotes absolute value. We do not enforce an upper bound on the stiffnesses via box constraints in any of the neural models (neither ours nor the baselines). We run the optimizers for at most 5,000 steps and with a convergence tolerance of 1×10^{-6}.

Table 4: Neural architectures. We show the composition of the two types of autoencoders we use in this work: a fully neural model employed for the NN and PINN models, and ours with the differentiable simulator as an analytical decoder. In this table, Linear(in_size, out_size) denotes a linear layer with learnable weights and biases followed by a nonlinearity; whereas Concat(in_size, out_size) denotes the concatenation of two vectors. Our model is more compact than the neural baselines because it has at least half of the number of trainable parameters. However, our model takes longer to train due to the computational cost of our mechanical simulator.

Task	Model	Encoder	Decoder	# Parameters ↓	Train time (s) ↓
Vaults	NN, PINN	[Linear$(3N, H)$; $2\times$ Linear(H, H); Linear(H, M)]	[Linear$(M + L, H)$; $2\times$ Linear(H, H); Linear$(H, 3N_u)$; Concat$(3N_u + 3N_s, 3N)$]	535,412	85
	Ours	[Linear$(3N, H)$; $2\times$Linear(H, H); Linear(H, M)]	Simulator$(M + L, 3N)$	254,900	140
Towers	NN, PINN	[Linear$(3R, H)$; $4\times$ Linear(H, H); Linear(H, M)]	[Linear$(M + L, H)$; $4\times$ Linear(H, H); Linear$(H, 3N_u)$; Concat$(3N_u + 3N_s, 3N)$]	1,245,216	136
	Ours	[Linear$(3R, H)$; $4\times$Linear(H, H); Linear(H, M)]	Simulator$(M + L, 3N)$	468,880	810

Table 4 summarizes the configuration of the neural models studied herein. We utilize MLPs for the encoder and decoder components of the three models. The MLPs are a sequence of linear layers followed by a nonlinear activation function. The weights and biases per layer are initialized at random from a uniform distribution in the interval $[-1/\sqrt{\rho}, 1/\sqrt{\rho})$, where ρ is equal to the layer's input size. We use an ELU (Clevert et al., 2016) as the activation function in every MLP layer except for the last layer of each encoder. The encoders employ Softplus as the last nonlinearity to satisfy the strictly positive requirement of our mechanical simulator established in Section 2.3.

The MLP decoder in the NN and PINN baselines takes as input the bar stiffnesses predicted $\mathbf{q} \in \mathbb{R}^M$ by the encoder, in addition to a vector with the boundary conditions $\mathbf{b} \in \mathbb{R}^L$ input to the mechanical simulator so that this learned decoder sees as much of the information the simulator has access to. This is different from, for example, conventional PINN approaches where the boundary conditions should be learned in addition to the problem physics (Haghighat et al., 2021; Raissi et al., 2019a); but akin to the approach proposed in other works (Lu et al., 2021; Bastek and Kochmann, 2023; Mai et al., 2024) where boundary conditions are treated as either hard constraints or imposed as network inputs. Here, we simplify the decoder's goal by imposing the boundary conditions by construction. This is why the boundaries of the shape targets are matched exactly in all of our examples. For simplicity, the vector $\mathbf{b}$ only comprises the Cartesian coordinates of the N_s fixed nodes and the vertical component of the loads applied to the N nodes (i.e., the horizontal components are zero) per target shape. The vector $\mathbf{b}$ has size $L = 3N_s + N$.

Similarly, the last linear layer in the MLP decoders outputs a vector of size $3N_u$, representing the 3D coordinates of the free nodes of a bar system $\mathbf{X}_u$. This is motivated by the fact that the mechanical simulator solves the form-finding problem only on the free nodes of a bar system with Equation 9, and it then concatenates the known position of the support nodes $\mathbf{X}_s$ to assemble the shape matrix $\mathbf{X}$. Therefore, we replicate this operation with the MLP decoder so that it predicts as much information as the simulator does. We then concatenate the output of the last layer of the neural decoder with the known position of the fixed anchor nodes. We describe task-specific implementation details next.

E.1 MASONRY SHELLS TASK

For the masonry vault amortization task, we train MLPs with 2 hidden layers, with $H = 256$ units each. The size of the input and output layers varies as per Table 4. The batch size is $B = 64$. The learning rate is 3×10^{-5} for the fully neural baselines (NN and PINN), and 5×10^{-5} for our model. Our model takes 2 minutes and 20 seconds to train. The fully neural alternatives, in contrast, take 1 minute and 25 seconds. We utilize SLSQP (Kraft, 1994) as our baseline for direct optimization with uniformly sampled, random values of the bar stiffnesses $\mathbf{q}$ as initial guesses. We train the PINN baseline to minimize a weighted combination of two terms: the shape loss $\mathcal{L}_{\text{shape}}$ (Equation 2) and the physics loss $\mathcal{L}_{\text{physics}}$ (Equation 6):

$$\mathcal{L} = \mathcal{L}_{\text{shape}} + \kappa \mathcal{L}_{\text{physics}} \tag{11}$$

Different values of κ impact the model performance on this bipartite task (i.e., simultaneously matching target shapes and reducing the residual forces $\mathbf{r}_i$). To strengthen the PINN baseline, we tune the value κ in five consecutive increments, from $\kappa = 10^{-2}$ to $\kappa = 10^{2}$, and train a separate instance of the model for each. We then pick the PINN that achieves the lowest unweighted loss on the test set (i.e., we set $\kappa = 10^{0}$ during inference to evaluate performance). Table 5 shows our results. We find that employing $\kappa = 10^{1}$ during training produces the best-performing PINN baseline, and we use this PINN for comparison with our model and the baselines in Section 4.

Table 5: PINN model evaluation on the masonry shells task for five different coefficients κ applied to the physics loss $\mathcal{L}_{\text{physics}}$ during training. The table reports the unweighted loss values generated by the trained PINN model predictions at inference time. $\kappa = 10^{1}$ produces best results.

κ	10^{-2}	10^{-1}	10^{0}	10^{1}	10^{2}
$\mathcal{L}_{\text{shape}} \downarrow$	1.9 ± 0.4	1.5 ± 0.3	1.4 ± 0.3	$\mathbf{3.1 \pm 1.2}$	92.0 ± 40.5
$\mathcal{L}_{\text{physics}} \downarrow$	35.3 ± 16.7	6.3 ± 2.8	2.3 ± 1.0	$\mathbf{0.6 \pm 0.3}$	0.3 ± 0.1

E.2 CABLE-NET TOWERS TASK

In the cable-net task, we train MLPs with 4 hidden layers of size $H = 256$. In all models, we reduce the size of the encoder input from $3N$ to $3R$, where N is the total number of nodes in the structure, and $R = 48$ corresponds to the number of nodes on the three rings (bottom, middle, and top) that parametrize this task. This choice reduces the total number of parameters in the encoders and makes the encoder less computationally intensive. We set the batch size to $B = 16$ for all models. The shape approximation task is underspecified because it only prescribes target height values for the intermediary tensile rings in a given cable-net tower instead of target positions. Therefore, we mask (i.e., multiply by zero) the predicted x and y coordinates of the nodes on these intermediary rings before evaluating the shape loss $\mathcal{L}_{\text{shape}}$.

As for our model, we train it in two stages, reducing the optimizer's learning rate from one stage to the next. In the first stage, we optimize for 5,000 steps with a constant learning rate of 1×10^{-3}. In the second stage, we fine-tune our model for another 5,000 steps with a smaller learning rate of 1×10^{-4}. During both stages, the global gradient clip value is 0.01. The total training time of our model is 13.5 minutes.

We train the MLPs of the NN and the PINN with a constant learning rate of 1×10^{-3} over 10,000 steps. The training time of both models is equal to 2 minutes and 16 seconds. The loss function we utilize to train the PINN baseline includes an explicit regularization term, in addition to the shape and the physics losses:

$$\mathcal{L} = \mathcal{L}_{\text{shape}} + \kappa \mathcal{L}_{\text{physics}} + \lambda \mathcal{L}_{\text{reg}} \tag{12}$$

Like in the masonry task, we tune the value of the weight coefficient κ during training in five distinct steps between $\kappa = 10^{-2}$ and $\kappa = 10^{2}$. The value of λ is reported in Section 4. As shown in Table 6, training the PINN with $\kappa = 10^{0}$ yields the best performance during training and inference. We use this PINN variant for comparison with our model and the other baselines. Lastly, we employ L-BFGS-B (Zhu et al., 1997), a quasi-Newton method, as our baseline for direct optimization.

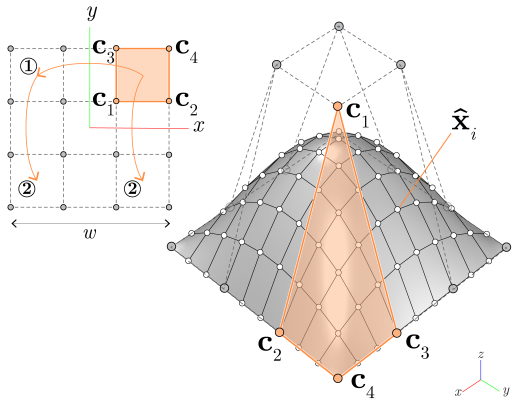


Figure 17: The data generation process of doubly-symmetric shapes for the masonry shells task consists of, first, creating variations of the positions of the points $\mathbf{c}_1$ to $\mathbf{c}_3$ in a corner of the control grid of a Bezier surface, and, then, mirroring twice. We create a target point $\hat{\mathbf{x}}_i$ by evaluating the Bezier at local coordinates (u, v).

We initialize the bar stiffnesses in this task with three different configurations as described in Table 2: randomized, expert, and with our trained model. The randomized initialization samples $\mathbf{q}$ from a uniform distribution bounded between the box constraint values. The expert initialization sets $\mathbf{q} = \mathbf{1}$. Both initialization approaches preset the sign of the bar forces as specified by the task (e.g., only negative values in the shells task; and mixed positive and negative values in the towers task).

Table 6: PINN model evaluation on the cable-net tower task for different coefficients κ on the physics loss $\mathcal{L}_{\text{physics}}$. The table reports unweighted loss values generated by the trained PINN predictions at inference time. Using $\kappa = 10^0$ during training produces the best performing PINN model.

κ	10^{-2}	10^{-1}	10^{0}	10^{1}	10^{2}
$\mathcal{L}_{\text{shape}} \downarrow$	42.8 ± 17.8	9.6 ± 5.6	**7.4 ± 3.4**	4.2 ± 2.6	5.1 ± 3.5
$\mathcal{L}_{\text{physics}} \downarrow$	1.1 ± 0.1	2.1 ± 0.0	**1.2 ± 0.1**	3.8 ± 0.7	4.3 ± 0.6

F Data generation on Bezier surfaces

A Bezier patch $\mathcal{B}$ maps a matrix $\mathbf{C} \in \mathbb{R}^{C\times 3}$ of control points to a smooth surface in $\mathbb{R}^3$, $\mathcal{B} : \mathbb{R}^{C\times 3} \rightarrow \mathcal{S}(u, v)$, parameterized by local coordinates (u, v). This parametrization offers clear control over architectural design intent as it enables the exploration of a wide array of smooth geometries by simply changing the positions of a coarse control grid.

F.1 Data generation for training

A summary of the data generation process is given in Figure 17. To generate a family of shapes for the shells task in Section 4, we focus on doubly-symmetric shapes generated by a square grid $w = 10$ units wide centered on the origin. The grid contains $C = 16$ control points arranged in a 4×4 layout.

We then follow three main steps. First, we vary the 3D coordinates of control points $\mathbf{c}_1$ to $\mathbf{c}_3$ on a quarter of the control grid. This construction assures the double symmetry in the generated data. To vary the position of each of the three control points $\mathbf{c}$, we first sample a translation vector $\mathbf{t}$ at random from a uniform distribution in the interval $[\mathbf{t}_{\text{min}}, \mathbf{t}_{\text{max}})$; and we add it to the control point's reference coordinates $\mathbf{c}_0$, such that $\mathbf{c} = \mathbf{c}_0 + \mathbf{t}$. The position of $\mathbf{c}_4$ is static. We detail the reference coordinates and the intervals for each control point in Table 7.

Next, we mirror these 4 control points on the xz and the yz Cartesian planes to obtain the position of the remaining 12 control points in the grid (see callout in Figure 17). The bounding box of the

resulting design space illustrates the scale of the masonry shell task and has dimensions $[2w, 2w, w]$. In the third and last step, we evaluate $N = 100$ equally spaced, local coordinates u and v on the Bezier, ranging from 0 to 1, to generate an equal number of points on the surface. The evaluated points represent the position $\hat{\mathbf{X}} \in \mathbb{R}^{N\times 3}$ of the vertices of the structure we employ as targets to train our model and the baselines. The coordinates of a point $\hat{\mathbf{x}} \in \mathbb{R}^3$ on the Bezier surface are a function of a (u, v) coordinates pair:

$$\hat{\mathbf{x}}(u, v) = \sum_{e=1}^{E} \sum_{g=1}^{G} \gamma_e(u)\, \gamma_g(v)\, \mathbf{c}_{eg} \tag{13}$$

where γ denotes a Bernstein polynomial of degree 3; $E = 4$ and $G = 4$; and $\mathbf{c}_{eg}$ indicates the position of the Bezier's control points, indexed on a $E \times G$ grid.

F.2 Interpolation of Beziers

We utilize linear interpolation to blend between doubly-symmetric and asymmetric Bezier surfaces. Since the targets $\hat{\mathbf{X}}$ are a function of the control points matrix $\mathbf{C}$, we interpolate between the control points matrix $\mathbf{C}_{\text{sym}}$ of a surface with double symmetry and that of an asymmetric surface $\mathbf{C}_{\text{asym}}$ to create one design

$$\mathbf{C}_{\text{interp}} = (1 - \delta)\mathbf{C}_{\text{sym}} + \delta\mathbf{C}_{\text{asym}} \tag{14}$$

where δ is the interpolation factor and $\mathbf{C}_{\text{interp}}$ is the interpolated control points matrix. We vary all the control points in $\mathbf{C}_{\text{asym}}$ sampling random translation vectors from uniform distributions like in the doubly-symmetric case, except for the four control points at the corners of the Bezier grid whose position also remains fixed. We finally generate target points $\hat{\mathbf{X}}$ from $\mathbf{C}_{\text{interp}}$ with Equation 13. An example of the shapes resulting from interpolating two Bezier surfaces is provided in Figure 13a.

Table 7: Generation parameters to sample random variations of the 3D coordinates of the points on a quarter of the control grid of a Bezier surface.

Control point	Reference position, $\mathbf{c}_0$	Lower bound, $\mathbf{t}_{\text{min}}$	Upper bound, $\mathbf{t}_{\text{max}}$
$\mathbf{c}_1$	$w/6, w/6, 0$	$0, 0, w/10$	$0, 0, w$
$\mathbf{c}_2$	$w/2, w/6, 0$	$-w/2, 0, 0$	$w/2, 0, w/2$
$\mathbf{c}_3$	$w/6, w/2, 0$	$0, -w/2, 0$	$0, w/2, 0$
$\mathbf{c}_4$	$w/2, w/2, 0$	$-$	$-$