Parametrizing Convex Sets Using A Sublinear Layer

Anonymous authors
Paper under double-blind review

Abstract

We propose a neural parameterization of convex sets by learning sublinear (positively homogeneous and convex) functions. Our networks implicitly represent both the support and gauge functions of a convex body. We prove a universal approximation theorem for convex sets under this parametrization. Empirically, we demonstrate the method on shape optimization and inverse design tasks, achieving accurate reconstruction of target shapes.

1 Introduction

In geometric machine learning and generative modeling, parameterizing physical domain geometries as the image of a canonical reference domain $B \subset \mathbb{R}^d$ by a neural network $\phi : B \to \Omega$ has become a classical tool (Yang et al., 2018; Paschalidou et al., 2021; Deprelle et al., 2019). This paradigm enables direct computation of geometric metrics, uniform boundary sampling, and seamless integration into end-to-end learning systems. In parallel, downstream tasks such as optimal design, shape reconstruction and inverse problems can often be formulated as a *shape optimization* problem

$$\inf_{\Omega \in \mathcal{A}} L(\Omega)$$

where $\mathcal{A}$ is a class of admissible shapes and L is some optimization criterion. Combined with modern neural differential equation solvers like Physics-Informed Neural Networks (PINNs) (Shin et al., 2023) and Neural Operators (Zhao et al., 2025; Kovachki et al., 2023), these frameworks could offer powerful alternatives to traditional grid- or mesh-based discretization methods (Allaire et al., 2021), that can suffer from the need of a mesh that is computationally expensive to generate and prevents accurate higer-order geometric computations (e.g. curvature).

A central challenge in learning-based geometric modeling is enforcing hard structural constraints directly within the network architecture (Bélières Frendo et al., 2025). In this work, we will be interested in architecturally enforcing convexity of the parametrized sets. Convex sets play a fundamental role in mathematics, optimization, and scientific computing. On the practical side, they arise naturally in applications such as engineering design (Chakib et al., 2024), inverse problems (Colton and Kress, 2019) or computer vision (Liu et al., 2010). In theoretical shape optimization, convexity is often assumed to ensure existence of an optimizer (Buttazzo and Guasoni, 1997) which often makes convexity the first constraint to be considered on a new shape optimization problem. Consequently, the ability to represent and optimize convex sets efficiently is of broad interest across several disciplines, especially in theoretical shape optimization where numerical experiments are regularly used to help generate conjectures on the optimality of certain shapes (Martinet, 2024).

Designing flexible neural representations of convex sets, however, remains challenging. While modern neural networks can model highly complex geometries, they typically do not preserve convexity by construction. Conversely, classical parametrizations based on support or gauge functions can in some cases guarantee convexity but often rely on constrained optimization procedures and are not naturally integrated into modern automatic differentiation frameworks. This raises the following question: can one design a neural architecture whose outputs are convex sets by construction, while remaining expressive enough to approximate arbitrary convex bodies and enabling efficient computation of geometric and PDE-dependent quantities? In this work, we answer this question positively by introducing a neural parametrization based on a sublinear layer, which exactly encodes convexity while remaining fully differentiable and amenable to gradient-based optimization.

We first review existing approaches from convex shape optimization and neural geometric representations, highlighting the limitations that motivate our construction.

1.1 Related works

Early numerical approaches for convex shape optimization enforced convexity through penalization, notably via the distance to the convex hull (Oudet, 2004). Alternative formulations characterized convex sets as intersections of half-spaces (Lachand-Robert and Oudet, 2006), which guarantee convexity but limit the representation of smooth geometries. Both of the approaches requires some reconstruction of the convex hull at every optimization step, which becomes prohibitive in higher dimensions.

A widely adopted paradigm that we will also follow relies on functional parametrization using support or gauge functions, which plays a central role in convex geometry (Bonnesen and Fenchel, 1934; Gruber and Wills, 1993; Rockafellar, 1997; Schneider, 2013). One can then discretize those function on Fourier/spherical harmonic bases, or on a boundary mesh. These approaches, that goes back at least to (Groemer, 1993), enable compact representations and have been successfully applied in various contexts like geometric computing (Ghosh and Kumar, 1998) and later shape optimization (Oudet, 2013; Bayen and Henrion, 2012; Bogosel, 2023; Bogosel et al., 2024; Ftouhi, 2025). However, convexity is encoded through second-order differential inequalities, leading to constrained optimization problems that are difficult to handle, especially in higher dimensions. While exact enforcement is possible in two dimensions (Bayen and Henrion, 2012; Bogosel, 2023), extensions to three dimensions always rely on relaxations that may compromise convexity (Antunes and Bogosel, 2022; Lamberg and Kaasalainen, 2001).

Geometric approaches provide an alternative by directly constraining admissible deformations. For instance, (Bartels and Wachsmuth, 2020) proposes a triangulation-based framework ensuring convexity preservation through restrictions on the deformation field, while (Chakib et al., 2024) makes use of Minkowski deformations.

Few work have been devoted to neural parametrization of convex sets. In (Martinet and Bungert, 2025a) convexity is enforced on the level sets of a neural network, using for instance input-convex neural networks (ICNNs) (Amos et al., 2017). Convex shape priors have been used as an inductive bias in image segmentation (Liu et al., 2025) or for convex decomposition through diffuse interface formulations such as phase-field models (Deng et al., 2020a). In (Tětková et al., 2025), it is shown that convexity is an important property of the latent representation of decision regions.

Some recent works were also interested in parametrizing support functions using neural networks, although not for shape representations purposes. For instance, (Haddad and Halder, 2023) uses an ICNN-type network to represent sublinear functions, while (Olausson et al., 2026) used an homogeneous extension of an ICNN for Maximum Inner Product Search. Those architecture are however not usable in our context, as will be discussed in Section 2.

1.2 Main contributions

Despite these advances, classical approaches face significant challenges, like exact enforcement of constraints (especially in three dimensions or more) or flexibility when it comes to the computation of shape-related quantities. On the other hand, neural approaches often rely on implicit parametrization that makes the computation of boundary quantities expensive, since one needs to reconstruct the boundary at every optimization step.

In contrast, our approach aims to combine the strengths of these paradigms by providing a representation based on the gauge and support functions of convex sets that:

1. ensures exact convexity without constraints,
2. can represent any convex set,
3. allows for simple and precise computation of higher-order geometric terms,
4. is able to seamlessly treat different dimensions.

Representation	Any dimension	Exact convexity	Smooth param.	Unconstrained.	High-order geometry	End-to-end diff.	Explicit	Universal approx.
Convex hull penalization (Oudet, 2004)	≈	✗	✗	✓	✗	✗	✓	N/A
Half-space intersection (Lachand-Robert and Oudet, 2006)	≈	✓	✗	✓	✗	✓	✗	✓
Gauge (Fourier basis) (Bayen and Henrion, 2012)	✗	✓	✓	✗	≈	✓	✓	✓
Support (Spherical basis) (Antunes and Bogosel, 2022) (Lamberg and Kaasalainen, 2001)	≈	✗	✓	✗	≈	✗	✓	N/A
Gauge (mesh discretization) (Bogosel, 2023)	✗	✓	✗	✗	✗	✗	✓	✓
Support (mesh discretization) (Bogosel, 2023)	✗	✓	✗	✗	✗	✗	✓	✓
CvxNet (Deng et al., 2020b)	✓	≈	✓	✓	≈	✓	✗	≈
ICNN level-set (Martinet and Bungert, 2025b)	✓	✓	✓	✓	≈	≈	✗	≈
Neural representation (ours)	✓	✓	✓	✓	✓	✓	✓	✓

Table 1: Comparison of different parametrizations of convex sets. ✓ Yes, ✗ No, ≈ Possible but not considered in the paper, N/A means "Not Applicable". Classical support/gauge parametrizations rely on constrained optimization to enforce convexity, while our sublinear representation guarantees convexity by construction while remaining smooth and fully differentiable. The top group are the classical methods, while the bottom one represents neural network based approachs.

We demonstrate these claims on various inverse and shape optimization problems, like shape reconstruction, the maximization of the torsional rigidity of a rod and classical problems from convex geometry such as the Minkowski and Mahler problems. The different advantages and drawbacks of each methods are represented in Table 1.

From a machine learning perspective, our approach can be interpreted as defining a hypothesis class of neural networks that encode convexity exactly by construction. In contrast to ICNNs, which enforce convexity of scalar functions with respect to inputs, our architecture directly parametrizes convex bodies. This provides an inductive bias tailored to geometric learning problems, enabling both expressive representations and efficient computation of shape-dependent quantities.

1.3 Structure of the paper

The paper will be stuctured as follows:

- In Section 2, we explain how to represent convex sets *via* a discretization of their gauge or support functions using a sublinear layer.
- In Section 3, we show how to leverage this representation to compute shape quantities that are often used in shape optimization, like the volume, perimeter, curvature, or PDE-related quantities.

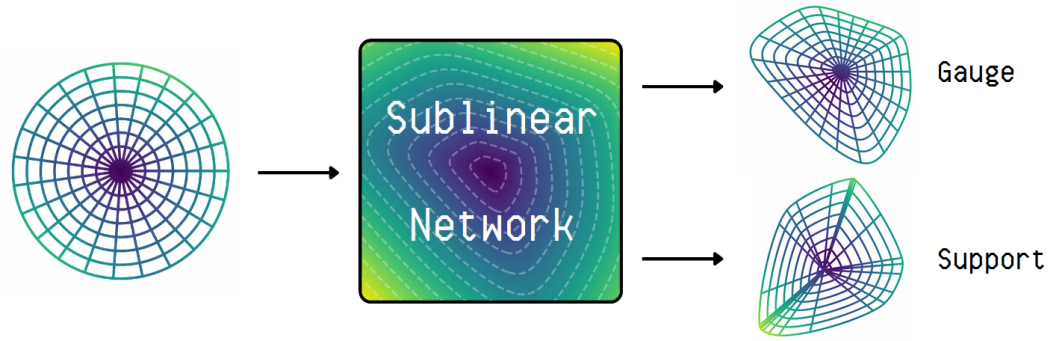

Figure 1: A sublinear layer can represent both the gauge and support functions of convex sets, and allow to parameterize smooth bijections that sends the ball on convex sets.

- In Section 4, we consider different problems that we reformulate as shape optimization problems. We make use of the representation of Section 2 and the computations of Section 3 to minimize non-trivial shape functionals.
- Section 5 and 6 are respectively the Discussion and Conclusion sections.
- An appendix gives additional mathematical and implementational details as well as additional experiments.

2 Parametrization of convex sets using neural networks

2.1 Gauge and support functions

In what follows, $\mathcal{K}$ represents the set of convex bodies (i.e. compact convex sets with non-empty interior) in $\mathbb{R}^d$ (endowed with the usual scalar product and Euclidean norm $\|.\|$) which contains 0 in their interior. The gauge and support functions associated to a convex set $\Omega \in \mathcal{K}$ are respectively defined as

$$g_\Omega(u) := \inf\{\lambda \geq 0 : u \in \lambda\Omega\} \qquad \text{and} \qquad h_\Omega(u) := \sup_{x\in\Omega} x \cdot u$$

for $u \in \mathbb{R}^d$. It can readily be seen that both of these functions are *sublinear*. A function $f : \mathbb{R}^d \to \mathbb{R}$ is said to be sublinear if it satisfies the two following properties:

Subadditivity: for all $x, y \in \mathbb{R}^d$, $f(x + y) \leq f(x) + f(y)$

Positive homogeneity: for all $\lambda > 0$ and $x \in \mathbb{R}^d$, $f(\lambda x) = \lambda f(x)$.

Reciprocally, if $f : \mathbb{R}^d \to \mathbb{R}$ is a sublinear function, then it is respectively the gauge and support function of the convex sets (see Theorem A.1)

$$\left\{x \in \mathbb{R}^d : f(x) \leq 1\right\} \qquad \text{and} \qquad \left\{x \in \mathbb{R}^d : x \cdot y \leq f(y) \text{ for all } y \in \mathbb{R}^d\right\} \tag{1}$$

We can use the gauge function of a convex Ω to give an explicit parametrization of Ω as the image of the unit ball $B \subset \mathbb{R}^d$ in the following way:

Proposition 2.1. *Let g be a positive and sublinear function. The function $\phi : x \longmapsto \frac{\|x\|}{g(x)} x$ (extended by 0 at the origin) is a homeomorphism from B to $\phi(B)$ and the image set $\phi(B)$ is convex with gauge function g.*

Proof. Let $\Omega := \left\{y \in \mathbb{R}^d : g(y) \leq 1\right\}$. Ω is convex as the sub-level set of a convex function. For $x \in B$, we have $g(\phi(x)) = \frac{\|x\|}{g(x)} g(x) = \|x\| \leq 1$. Hence, $\phi(B) \subset \Omega$. On the other hand, if $y \in \Omega$, then $x := \phi^{-1}(y) = \frac{g(y)}{\|y\|} y \in B$ and $y = \phi(x) \in \phi(B)$. Therefore, $\phi(B) = \Omega$ is convex and ϕ is a homeomorphism. □

Using Proposition A.4 we know that if $\Omega \in \mathcal{K}$ is such that h_Ω is differentiable on $\mathbb{R}^d \setminus \{0\}$, then $\nabla h_\Omega : \mathbb{S}^{n-1} \to \partial\Omega$ (where $\partial\Omega$ is the boundary of Ω) is a homeomorphism. This leads to the following proposition:

Proposition 2.2. *Let h be a positive and sublinear function, differentiable on $\mathbb{R}^d \setminus \{0\}$. The function $\phi : x \longmapsto \|x\|\nabla h(x)$ (extended by 0 at the origin) is bijective from B to $\phi(B)$ and the image set $\phi(B)$ is convex with support function h.*

Proof. According to the assumptions, there exists $\Omega \in \mathcal{K}$ such that $h = h_\Omega$. We will show that $\Omega = \phi(B)$. Let $y \in \phi(B)$. Then there exists $x \in B$ such that $y = \phi(x)$. If $x = 0$ then $y = 0$ by definition hence $y \in \Omega$ since h_Ω is positive. If $x \neq 0$ then $y = \|x\|\nabla h_\Omega(x) \in \mathrm{Conv}(\partial\Omega \cup \{0\})$; indeed, since $0 = \phi(0) \in \phi(B)$ and $\|x\| \leq 1$, y is a convex combination of 0 and $\nabla h_\Omega(x) \in \partial\Omega$ (here, we used that ∇h_Ω is 0-homogeneous). Since Ω is a compact convex set, $\mathrm{Conv}(\partial\Omega \cup \{0\}) = \Omega$ and hence $\phi(B) \subset \Omega$. On the other hand, let $y \in \Omega$. Since 0 is in the interior for Ω, there exists $\lambda \geq 1$ such that $\tilde{y} := \lambda y \in \partial\Omega$. Hence, there exists $\tilde{x} \in \partial B$ s.t. $\tilde{y} = \nabla h_\Omega(\tilde{x})$. By taking $x = \frac{1}{\lambda}\tilde{x} \in B$, we have $y = \|x\|\nabla h(x)$ and $\Omega \subset \phi(B)$. Hence $\phi(B) = \Omega$, meaning that $\phi(B)$ is convex.

One must now prove the injectivity of ϕ. Let $x_1, x_2 \in \mathbb{R}^d \setminus \{0\}$ be such that $\phi(x_1) = \phi(x_2)$. Hence, $\nabla h(x_1)$ and $\nabla h(x_2)$ are positively colinear. Moreover, $\nabla h(x_1), \nabla h(x_2) \in \partial\Omega$. However, since $h > 0$, we have that $0 \in \mathit{int}(\Omega)$. Hence, any half line originating at 0 must intersect $\partial\Omega$ exactly once (by convexity), implying that $\nabla h(x_1) = \nabla h(x_2)$ and further $\|x_1\| = \|x_2\|$. Using the 0-homogeneity of ∇h, we have $\nabla h\left(\frac{x_1}{\|x_1\|}\right) = \nabla h\left(\frac{x_2}{\|x_2\|}\right)$ and using that $\nabla h : \mathbb{S}^{n-1} \to \partial\Omega$ is a homeomorphism, this means that $\frac{x_1}{\|x_1\|} = \frac{x_2}{\|x_2\|}$: therefore $x_1 = x_2$ and ϕ is injective. □

2.2 Sublinear layer

In what follows, we parametrize convex sets *via* their gauge or support function. To this end, we will construct a neural network $p_\theta : \mathbb{R}^d \to \mathbb{R}$, with parameters θ, that is sublinear by design.

A natural starting point is the MaxOut layer (Goodfellow et al., 2013), defined as $p_\theta(x) = \max_{1 \leq i \leq N}(w_i \cdot x)$, where $w_i \in \mathbb{R}^d$. This choice is motivated by the classical characterization of sublinear functions as point-wise supremum of linear forms, and has already been used to parameterizing sublinear functions in (Haddad and Halder, 2023), which also extended this construction to an ICNN-like architecture with ReLU activation function. However, this representation is non-smooth, which can be limiting in applications requiring differential quantities such as normals or curvature, or even for the application of Proposition 2.2. To address this, we could consider to replace the maximum by a smooth approximation, like the LogSumExp (LSE):

$$p_\theta(x) = \beta \mathrm{LSE}\left(\frac{w_1 \cdot x}{\beta}, \ldots, \frac{w_1 \cdot x}{\beta}\right) = \beta \mathrm{LSE}\left(\frac{1}{\beta} W^T x\right)$$

where $\beta > 0$ is a temperature parameter and $W = (w_1, \ldots, w_n)$.

Unfortunately, the obtained function would not be sublinear anymore, as the LSE is not homogeneous. Nevertheless, we are able to recover sublinearity while preserving the trace on the sphere as we will see hereafter. This requires classical tools from convex analysis, mostly taken from (Schneider, 2013), that we provide in Section A for completeness. The results can also be found in the classical reference (Rockafellar, 1997). We first recall the definition of the convex conjugate of a function:

Definition 2.1 (Convex conjugate)**.** *Let $f : \mathbb{R}^d \to (-\infty, +\infty]$. The convex conjugate of $f^* : \mathbb{R}^d \to (-\infty, +\infty]$ is defined as*

$$f^*(y) := \sup_{x \in dom(f)} \{y \cdot x - f(x)\} \tag{2}$$

where $dom(f)$ is the set of $x \in \mathbb{R}^d$ such that $f(x) < +\infty$.

Proposition 2.3. *Let $f : \mathbb{R}^d \to (-\infty, +\infty]$ be a proper convex and closed function such that $f^* \leq 0$ on its domain. Let $x \in \mathbb{R}^d$ and define*

$$h(x) := \|x\| f\left(\frac{x}{\|x\|}\right)$$

with $h(0) := 0$. Then h is sublinear.

Proof. First, h is positively homogeneous by construction. Hence, sublinearity is equivalent to convexity, and we will show the latter. According Proposition A.1, we can write

$$f(x) = f^{**}(x) = \sup_{y \in \mathrm{dom}(f^*)} \{y \cdot x - f^*(y)\},$$

which leads to $h(x) = \sup_{y \in \mathrm{dom}(f^*)} \{y \cdot x - \|x\| f^*(y)\}$. Since $f^* \leq 0$ on its domain, the function $x \mapsto y \cdot x - \|x\| f(y)$ is convex, implying that h is convex as the supremum of a family of convex functions. □

Using the previous proposition, we can define our neural network as

$$p_\theta(x) := \beta \|x\| \mathrm{LSE}\left(W^T \frac{x}{\|x\|}\right) \qquad \text{where} \qquad \mathrm{LSE}(y_1, \ldots, y_n) = \log\left(\sum_{i=1}^n e^{y_i}\right) \tag{3}$$

and $\theta = (\beta, W) \in \mathbb{R} \times \mathbb{R}^{d \times n}$. In what follows, we assume that $n > d$ and W has full rank d.

Proposition 2.4. *p_θ as defined in Eq. (3) is a sublinear function which is C^∞ on $\mathbb{R}^d \setminus \{0\}$.*

Proof. The regularity assumption is obvious. Since the LSE is a convex function with values in $\mathbb{R}$, it is proper and closed. The sublinearity can be shown as follows: let $f(x) = \mathrm{LSE}(W^T x)$. According to Proposition 2.3, it is enough to show that f^* is non positive. According to Proposition A.3 and Proposition A.2, we have that

$$f^*(y) \leq W \triangleright (-S)(y) = \inf_{x \in \mathbb{R}^n, Wx = y} -S(x) \leq 0$$

where S is the entropy function defined in Proposition A.3. The infimum is taken over the set $Wx = y$ which is non-empty due to the assumption that W is of full rank. □

Remark 2.1. *According to the previous derivations, one does not need to restrict to the LSE activation function. Indeed, any convex function such that $f^* \leq 0$ on its domain would also give rise to a sublinear layer. The proof of the universal approximation property (given hereafter) however heavily depends on the fact that the LSE approximates the maximum function.*

Note that the "extension by homogeneity" of Proposition 2.3 has also been considered in the recent paper (Olausson et al., 2026). However, in their case, the function f is given by an ICNN which does not necessarily guarantees that $f^* \leq 0$ and hence can break the convexity of the extension h, as shown by taking $f(x) \equiv -1$.

2.3 Universal approximation

According to Proposition 2.1 and Proposition 2.2, given the the previously defined neural network p_θ, we can define the auxiliary neural networks

$$\phi^g_\theta(x) := \frac{\|x\|}{p_\theta(x)} x \tag{4}$$

and

$$\phi^s_\theta(x) := \|x\| \nabla p_\theta(x) \tag{5}$$

which define the convex sets $\Omega^g_\theta := \phi^g_\theta(B)$ and $\Omega^s_\theta := \phi^s_\theta(B)$ (see Fig. 1 for an illustration of the action of these maps on the unit ball). In the case of the parametrization given by Eq. (4) (resp. Eq. (5)), p_θ is the gauge function of Ω^g_θ (resp. the support function of Ω^s_θ). We will simply write $\Omega_\theta = \phi_\theta(B)$ when the precise representation (gauge or support) does not matter. In what follows,

$$\mathcal{K}^g := \left\{\Omega^g_\theta \subset \mathbb{R}^d : \beta > 0, W \in \mathbb{R}^{d \times m}, m > d, \mathrm{rank}(M) = d\right\}$$

will denote the set of convex bodies that are represented by the gauge parameterization Eq. (4) and

$$\mathcal{K}^s := \left\{\Omega^s_\theta \subset \mathbb{R}^d : \beta > 0, W \in \mathbb{R}^{d \times m}, m > d, \mathrm{rank}(M) = d\right\}$$

the set of convex bodies represented by the support parameterization Eq. (5).

It is of core interest to know whether every convex set can be approximated using this architecture. To this purpose, we will use the Hausdorff distance between two convex sets $K, L \in \mathcal{K}$ which is defined as

$$d_H(K, L) := \sup \left\{ \sup_{x \in K} d(x, L), \sup_{x \in L} d(x, K) \right\}$$

It turns out that we have the following quantitative universal approximation properties in the Hausdorff sense:

Theorem 2.1. *$\mathcal{K}^s$ is dense in $\mathcal{K}$ with respect to the Hausdorff distance. More precisely, there exists a dimensional constant c_d such that for all $\Omega \in \mathcal{K}$, all $\varepsilon > 0$ small enough and all $m \geq c_d \left(\frac{diam(\Omega)}{\varepsilon} \right)^{\frac{d-1}{2}}$, there exists $\theta = (\beta, W) \in \mathbb{R} \times \mathbb{R}^{d \times m}$ such that $d_H(\Omega_\theta^s, \Omega) < \varepsilon$.*

Theorem 2.2. *$\mathcal{K}^g$ is dense in $\mathcal{K}$ with respect to the Hausdorff distance. More precisely, there exists a dimensional constant c_d such that for all $\Omega \in \mathcal{K}$, all $\varepsilon > 0$ small enough and all $m \geq c_d \left(\frac{diam(\Omega)}{\varepsilon} \right)^{\frac{d-1}{2}}$, there exists $\theta = (\beta, W) \in \mathbb{R} \times \mathbb{R}^{d \times m}$ such that $d_H(\Omega_\theta^g, \Omega) < \varepsilon$.*

In particular, those theorems implies that every convex set can be approximated at the precision of ε by a network of a size of $O\left(\varepsilon^{-\frac{d-1}{2}}\right)$ neurons. The proof of Theorem 2.1 is based on the following theorem by Bronshteyn and Ivanov (1975):

Theorem 2.3. *There exists a constant C_d such that for all $\Omega \in \mathcal{K}$, there exists $w_1, \ldots, w_m \in \mathbb{R}^d$ such that*

$$d_H(\Omega, P_m) \leq C_d \frac{diam(\Omega)}{m^{\frac{2}{d-1}}}$$

where $P_m = Conv(w_1, \ldots, w_m)$.

Moreover, we will use the classical fact that for $z_1, \ldots, z_m \in \mathbb{R}$ and $\beta > 0$, we have

$$\max_i z_i \leq \beta \mathrm{LSE}\left(\frac{z_1}{\beta}, \ldots, \frac{z_m}{\beta} \right) \leq \max_i z_i + \beta \log(m). \tag{6}$$

Proof of Theorem 2.1. Let $\Omega \in \mathcal{K}$, $\varepsilon > 0$. According to Theorem 2.3, there exist a polytope $P_m = \mathrm{Conv}(w_1, \ldots, w_m)$ such that $d_H(\Omega, P_m) \leq C_d \frac{\mathrm{diam}(\Omega)}{m^{\frac{2}{m-1}}}$. By chosing $m \geq \left(\frac{2C_d \mathrm{diam}(\Omega)}{\varepsilon} \right)^{\frac{d-1}{2}}$, we get that $d_H(\Omega, P_m) \leq \varepsilon/2$.

It is straightforward to show that the support function of P_m is given by $h_{P_m}(x) := \sup_{1 \leq i \leq m} w_i \cdot x$. Now, let p_θ be as defined in Eq. (3) with $\theta = (\beta, W_\beta)$ where $W_\beta := \beta^{-1}(w_1 \ldots w_m)^T$. Let Ω_θ^s be the associated convex set, defined by Eq. (5). This implies in particular that $h_{\Omega_\theta^s} = p_\theta$.

Now, using the triangle inequality, $d_H(\Omega, \Omega_\theta) \leq d_H(\Omega, P_m) + d_H(P_m, \Omega_\theta)$. According to (Schneider, 2013, Lemma 1.8.14), $d_H(P, \Omega_\theta^s) = \|h_P - p_\theta\|_{C(\mathbb{S}^{n-1})}$. Using Eq. (6) we deduce that

$$\|h_{P_m} - p_\theta\|_{C(\mathbb{S}^{n-1})} \leq \beta \log(m).$$

By taking $\beta \leq \frac{\varepsilon}{2 \log(m)}$, we have that

$$d_H(K, \Omega_\theta^s) \leq d_H(K, P) + d_H(P, \Omega_\theta^s) \leq \varepsilon$$

and the result follows. □

Now let us turn to the proof of Theorem 2.2. While Theorem 2.3 controls the number of vertices of a polytope approximating a convex set, we will need an analogous result to control the number of faces. This is the object of a theorem of Dudley (1974):

Theorem 2.4. *There exists a constant C_d such that for all $\Omega \in \mathcal{K}$, there exists $w_1, \dots, w_m \in \mathbb{R}^d$ such that*

$$d_H(\Omega, P_m) \leq C_d \frac{diam(\Omega)}{m^{\frac{2}{d-1}}}$$

where

$$P_m = \bigcap_{1 \leq i \leq m} \left\{ x \in \mathbb{R}^d : w_i \cdot x \leq 1 \right\}.$$

We can now show the proof of Theorem 2.2, which uses the same ideas than Theorem 2.1 but involves a few additional technicalities.

Proof of Theorem 2.2. We first prove the following fact: for $K, L \in \mathcal{K}$, we have $d_H(K, L) \leq \|\rho_K - \rho_L\|_{C(\mathbb{S}^{n-1})}$ where

$$\begin{aligned} \rho_K \colon \mathbb{S}^{n-1} &\longrightarrow \mathbb{R}^+ \\ x &\longmapsto \sup\{r > 0 : rx \in K\} \end{aligned}$$

is the *radial function* of K. In particular, it is immediate that the radial function is the inverse of the gauge function on $\mathbb{S}^{n-1}$. For $x \in K$, let us define

$$y = \begin{cases} \rho_L\left(\frac{x}{\|x\|}\right)\frac{x}{\|x\|} & \text{if } x \notin L \\ x & \text{if } x \in L \end{cases}.$$

Then $y \in L$ and we have

$$d(x, L) \leq \max\left\{0, \|x\| - \rho_L\left(\frac{x}{\|x\|}\right)\right\} \leq \left|\rho_K\left(\frac{x}{\|x\|}\right) - \rho_L\left(\frac{x}{\|x\|}\right)\right| \leq \|\rho_K - \rho_L\|_{C(\mathbb{S}^{n-1})}$$

Hence $\sup_{x \in K} d(x, L) \leq \|\rho_K - \rho_L\|_{C(\mathbb{S}^{n-1})}$. By symmetry, we deduce the claim.

Now, let $\Omega \in \mathcal{K}$, $\varepsilon > 0$. According to Theorem 2.4, there exist a polytope $P_m = \bigcap_{1 \leq i \leq m} \left\{ x \in \mathbb{R}^d : w_i \cdot x \leq 1 \right\}$ such that $d_H(\Omega, P_m) \leq C_d \frac{\text{diam}(\Omega)}{m^{\frac{2}{m-1}}}$. By chosing $m \geq \left(\frac{2C_d \text{diam}(\Omega)}{\varepsilon}\right)^{\frac{d-1}{2}}$, we get that $d_H(\Omega, P_m) \leq \varepsilon/2$.

In particular, we have $g_{P_m}(x) = \max_{1 \leq i \leq m} w_i \cdot x$. For ε small enough, $d_H(\Omega, P_m) < \varepsilon/2$ implies that $\min_{u \in \mathbb{S}^{n-1}} g_{P_m}(u) \geq \frac{\alpha_\Omega}{2}$ where $\alpha_\Omega := \min_{u \in \mathbb{S}^{n-1}} g_\Omega(u) > 0$. Let $\theta = (\beta, W_\beta)$ with $W_\beta = (\frac{w_1}{\beta} \dots \frac{w_m}{\beta})^T$. Since $p_\theta \xrightarrow[\beta \to 0]{} g_{P_m}$ uniformly on $\mathbb{S}^{d-1}$ thanks to Eq. (6), we can take $\beta \leq \frac{\alpha_\Omega^2}{16 \log(m)} \varepsilon$ small enough so that $\min_{u \in \mathbb{S}^{n-1}} p_\theta(u) \geq \frac{\alpha_\Omega}{4}$. Hence, for Ω_θ^g defined as in Eq. (4), we have

$$\begin{aligned} \|\rho_{\Omega_\theta^g} - \rho_{P_m}\|_{C(\mathbb{S}^{n-1})} &= \left\| \frac{1}{p_\theta} - \frac{1}{g_{P_m}} \right\|_{C(\mathbb{S}^{n-1})} \\ &\leq \frac{\|g_{P_m} - p_\theta\|_{C(\mathbb{S}^{n-1})}}{\min_{\mathbb{S}^{n-1}} p_\theta \min_{\mathbb{S}^{n-1}} g_{P_m}} \\ &\leq \frac{\beta \log(m)}{\min_{\mathbb{S}^{n-1}} p_\theta \min_{\mathbb{S}^{n-1}} g_{P_m}} \qquad \text{using Eq. (6)} \\ &\leq \frac{8\beta \log(m)}{\alpha_\Omega^2} \leq \frac{\varepsilon}{2} \end{aligned}$$

leading to $d_H(\Omega, \Omega_\theta^g) \leq d_H(\Omega, P_m) + d_H(P_m, \Omega_\theta^g) \leq d_H(\Omega, P_m) + \|\rho_{\Omega_\theta^g} - \rho_{P_m}\|_{C(\mathbb{S}^{n-1})} \leq \varepsilon$. □

Remark 2.2. *In order to complete the theoretical analysis, we perform in Section 4.6 an ablation study, showing how the amount of neurons impacts the approximation power of our sublinear layer. We show that we are able, in practice, to recover the theoretical rates given by Theorem 2.1.*

2.4 Symmetries

A natural question is to know whether we can enforce symmetries on the parametrized shapes. In term of group actions, it amounts at asking if, for a certain group of symmetries G we can make Ω_θ *invariant* with respect to the action of G, i.e. $g.\Omega_\theta = \Omega_\theta$ for all $g \in G$. In our case, we achieve invariance by *frame averaging* (Puny et al., 2021). Given G a finite subgroup of isometries of $\mathbb{R}^d$ and p_θ a sublinear layer, we define $p_\theta^G(x) := \frac{1}{|G|}\sum_{\tilde{g}\in G} p_\theta(\tilde{g}.x)$. we have the following proposition:

Proposition 2.5. *Let p_θ^G be either the gauge or support function of a convex set Ω_θ. Then Ω_θ is invariant with respect to G.*

Proof. We will use Eq. (1). Let p_θ be a sublinear layer.

Case 1 (gauge): Let $\Omega_\theta := \left\{x \in \mathbb{R}^d : p_\theta^G(x) \leq 1\right\}$. For $x \in \Omega_\theta$ and $g \in G$, we have that $p_\theta^G(g.x) = \frac{1}{|G|}\sum_{\tilde{g}\in G} p_\theta(g.(\tilde{g}.x)) = \frac{1}{|G|}\sum_{\tilde{g}\in G} p_\theta((g\tilde{g}).x) = \frac{1}{|G|}\sum_{\tilde{g}\in G} p_\theta(\tilde{g}.x) = p_\theta^G(x) \leq 1$ (where we used the change of variable $\tilde{g} \leftarrow \tilde{g}g$) and hence $g.\Omega_\theta \subset \Omega_\theta$. We can then deduce the reverse inclusion by $x \in \Omega_\theta \implies g^{-1}.x \in \Omega_\theta \implies x \in g.\Omega_\theta$.

Case 2 (support): Let $\Omega_\theta := \left\{x \in \mathbb{R}^d : x \cdot y \leq p_\theta^G(y) \text{ for all } y \in \mathbb{R}^d\right\}$. For $x \in \Omega_\theta$ and $g \in G$, we have for all $y \in \mathbb{R}^d$:

$$(g.x) \cdot y = x \cdot (g^{-1}.y) \leq p_\theta^G(g^{-1}.y) = p_\theta^G(y)$$

hence $g.\Omega_\theta \subset \Omega_\theta$. The reverse inclusion follows. □

An experiment making use of the symmetries of both the gauge and support functions can be found in Section 4.5.

Remark 2.3 (Which representation should be used?)**.** *In this work, the gauge and support functions representations are most often interchangeable, since the computation of shape-related quantites are made agnostic with respect to the choice of representation (see Section 3). However, there is certain cases where one of the parametrization is preferable, like the gauge parametrization in Section 4.1 or the support parametrization in Section 4.4. The gauge parametrization may also be preferred when one needs an explicit inverse, as illustrated in Section D. Moreover, certain geometric quantities that we do not discuss here are easier to compute with a certain representation (for instance, the width is easily computed in the support function parametrization).*

3 Computation of shape quantities

Shape optimization problems, as well as certain inverse problems, require to optimize a certain criterium involving quantities related to the shape, like the volume, perimeter, curvature, etc, for which we can benefit from automatic differentiation. Other geometric quantities have also been explored in the literature (Martinet and Ftouhi, 2026).

Let p_θ be a sublinear layer as defined in Eq. (3). In what follows, ϕ_θ will denote either the neural network defined in Eq. (4) or in Eq. (5) *via* p_θ. We will reformulate shape optimization problems into finite-dimensional optimization problems over the parameters θ of $\Omega_\theta = \phi_\theta(B)$. Using the previous propositions, we have the guarantee that Ω_θ will stay convex in the course of the optimization. Moreover, since ϕ_θ is a smooth homeomorphism, we will be able to compute integral quantities (like the volume or the surface area) by using the change of variable formula.

3.1 Integral quantities

We pull back the computations to the reference domain B, using change of variables (Evans and Gariepy, 2025):

$$\int_{\Omega_\theta} f dx = \int_B (f \circ \phi_\theta)\mathrm{Jac}(\phi_\theta) dx \quad \text{and} \quad \int_{\partial\Omega_\theta} g d\sigma = \int_{\partial B} (g \circ \phi_\theta)\mathrm{Jac}_{\partial B}(\phi_\theta) d\sigma,$$

where $\mathrm{Jac}(\phi_\theta) = |\det(D\phi_\theta)|$, $\mathrm{Jac}_{\partial B}(\phi_\theta) = \mathrm{Jac}(\phi_\theta)\|\left(D\phi_\theta\right)^{-T} n_B\|$ and $n_B(x) := x$ on ∂B is the unit outward normal vector. For instance, we can compute the volume and perimeter (i.e. surface area) of Ω_θ as

$$\mathrm{Vol}(\Omega_\theta) = \int_B \mathrm{Jac}\phi_\theta dx \qquad \text{and} \qquad \mathrm{Per}(\Omega_\theta) = \int_{\partial B} \mathrm{Jac}_{\partial B}(\phi_\theta) d\sigma$$

All the differential quantities are automatically computed using PyTorch (Paszke et al., 2019). The integrals can be discretized using either Monte-Carlo or fixed quadrature points. We chose the latter approach, as we will mainly use L–BFGS as an optimizer, which is known for being sensitive to noise. In particular, we use a Fibonacci lattice approach for the discretizations on the 2-sphere (González, 2010).

3.2 Geometric-differential quantities

Normal vector: Let $y = \phi_\theta(x)$, $x \in \partial B$. We can express the normal vector at $y \in \partial\Omega_\theta$ by

$$n_\theta(y) = \frac{\left(D\phi_\theta\right)^{-T}(x) n_B(x)}{\left\|\left(D\phi_\theta\right)^{-T}(x) n_B(x)\right\|}.$$

Indeed, if φ is a level set function associated to a smooth set Ω (i.e. $\Omega = \{\varphi \leq 0\}$), then for $y \in \partial\Omega$, the normal vector can be computed as

$$n(y) = \frac{\nabla\varphi(y)}{\|\nabla\varphi(y)\|}.$$

In the case of Ω_θ, the function $\varphi(y) = \|\phi_\theta^{-1}(y)\|^2 - 1$ is an admissible level set function since $\varphi(y) \leq 0 \iff y \in \Omega_\theta$. Computing the gradient leads to $\nabla\varphi(y) = D\left(\phi_\theta^{-1}\right)^T(y).2\phi_\theta^{-1}(y) = 2\left(D\phi_\theta\right)^{-T}(x)x$, from which we deduce the formula by normalization. Notice that we never need the expression of ϕ_θ^{-1} to compute $n_\theta(y)$ when $y = \phi_\theta(x)$.

Curvature terms: The *Weingarten map* at y is defined as

$$\begin{aligned} S_y \colon T_y\partial\Omega_\theta &\longrightarrow T_y\partial\Omega_\theta \\ v &\longmapsto D_\Gamma n(y).v \end{aligned}$$

where $D_\Gamma n(y)$ is the tangential derivative of n, i.e. the restriction of $Dn(y) \in \mathbb{R}^{d\times d}$ to the tangent space $T_y\partial\Omega_\theta$. Using this map, we can respectively define the mean curvature and Gaussian curvature as

$$H_\theta(y) = (d-1)^{-1}\operatorname{tr} S_y \qquad \text{and} \qquad \kappa_\theta(y) = \det S_y$$

In order to compute the Weingarten map numerically, one introduces the Housholder matrix $H_y = I - vv^T$ where

$$v = \frac{n(y) - e_d}{\|n(y) - e_d\|},$$

$e_1, \ldots, e_d$ being the canonical basis of $\mathbb{R}^d$. Hence, $H_y n = e_d$ and in particular, the first $d-1$ lines of H_y are orthogonal to $n(y)$ and hence forms an orthonormal basis of $T_y\partial\Omega_\theta$. One can then compute $D_\Gamma n(y)$ in this basis as

$$D_\Gamma n(y) = \tilde{H}_y Dn(y) \tilde{H}_y^T \in \mathbb{R}^{(d-1)\times(d-1)}$$

where $\tilde{H} = (H_{ij})_{\substack{1\leq i\leq d-1 \\ 1\leq j\leq d}}$.

3.3 PDE-related quantities

It is common in shape optimization to consider quantities that depends on the solution of a PDE. For instance, optimal design of structures often minimize for the compliance, which is a by-product of the linear elasticity equation; aerodynamic shape optimization needs to solve Navier-Stokes. In this section, we show how it is possible to seamlessly bridge precise and robust mesh free methods with the auto-differentiation of PyTorch in order to easily compute derivatives of PDE-dependent quantities in the simple case of the Poisson equation.

Mesh free Galerkin method: For $f \in L^2(\Omega_\theta)$, the Poisson problem with Dirichlet boundary condition aims at finding $u \in H_0^1(\Omega_\theta)$ which satisfies

$$\begin{cases} -\Delta u &= f \quad \text{in } \Omega_\theta, \\ u &= 0 \quad \text{on } \partial\Omega_\theta. \end{cases} \tag{7}$$

Passing to the *weak formulation* (Evans, 2022) and changing variables, Eq. (7) can be express in a weak sense as

$$\int_B A_\theta \nabla u \cdot \nabla v = \int_B (\mathrm{Jac}\phi_\theta)(f \circ \phi_\theta) v,$$

for $u, v \in H_0^1(B)$, where $A_\theta := (\mathrm{Jac}\phi_\theta)(D\phi_\theta)^{-1}(D\phi_\theta)^{-T}$. The Dirichlet boundary condition is weakly enforced by adding a penalization term with penalty parameter $\alpha > 0$. The resulting problem is then discretized on a subspace spanned by radial basis functions (RBFs) $\varphi_i(x) := \psi(|x - x_i|)$, $x_i \in B$, possibly augmented by a polynomial basis (Wendland, 1999) leading to a system $K\bar{u} = \bar{f}$, where

$$K_{ij} = \int_B A_\theta \nabla\varphi_i \cdot \nabla\varphi_j + \alpha \int_{\partial B} \varphi_i \varphi_j \qquad \text{and} \qquad \bar{f}_i = \int_B (\mathrm{Jac}\phi_\theta)(f \circ \phi_\theta)\varphi_i.$$

The integrals are discretized as previously described. We can then solve the previous linear system and recover the solution $u(x) = \sum_i \bar{u}_i \varphi(x)$.

Remark 3.1. *The RBF-Galerkin method is chosen because of its theoretical convergence guarantees (unlike other meshless methods relying on the strong formulation of the PDE like the Kansa method (Fasshauer, 2007) or PINN-based methods (Dissanayake and Phan-Thien, 1994)) and is easy to implement in PyTorch.*

Method of fundamental solutions An important particular case of Eq. (7) is for $f = 1$, in which case the equation can be solved thanks to a method exhibiting spectral convergence. The details are provided in Section B.

A note on the use of PINNs One may wonder why we do not use PINNs to solve the PDEs, as in related work (Bélières Frendo et al., 2025). Our choice is guided by both practical and structural considerations. On the practical side, recent studies indicate that PINNs do not match the efficiency of classical solvers on several low-dimensional PDEs (Grossmann et al., 2024; McGreivy and Hakim, 2024). On the structural side, while (Bélières Frendo et al., 2025) relies on a min–min formulation enabling joint optimization, most shape optimization problems are naturally min–max, requiring the inner problem to be (approximately) solved at each step, which increases computational cost.

For completeness, we include a comparison with PINNs in Section 4.7, where classical methods remain way more efficient in our setting.

On shape derivatives and the FEM Classical approaches to minimizing a shape functional rely on shape derivatives evaluated on a mesh. Their numerical implementation typically involves additional components such as adjoint states and extension–regularization procedures, which require solving additional PDEs on the domain. In contrast, our approach avoids these steps by leveraging automatic differentiation.

That said, mesh-based methods might remain desirable in a context where high accuracy is needed. However, there is currently no standard PyTorch-based framework for finite element computations. We discuss a possible interface with external FEM libraries in Section D.

Remark 3.2. *Notably, the arguments in this section do not rely on the convexity of Ω_θ or on any special structure of ϕ_θ, except than being a diffeomorphism. This suggests that the framework can be extended beyond the present setting to arbitrary invertible neural networks ϕ_θ, which we leave to future work.*

4 Application to shape optimization problems

In this section, we present the performance of our method on a range of problems arising from geometry and shape optimization. For fairness with respect to classical methods and easy reproducibility, all experiments

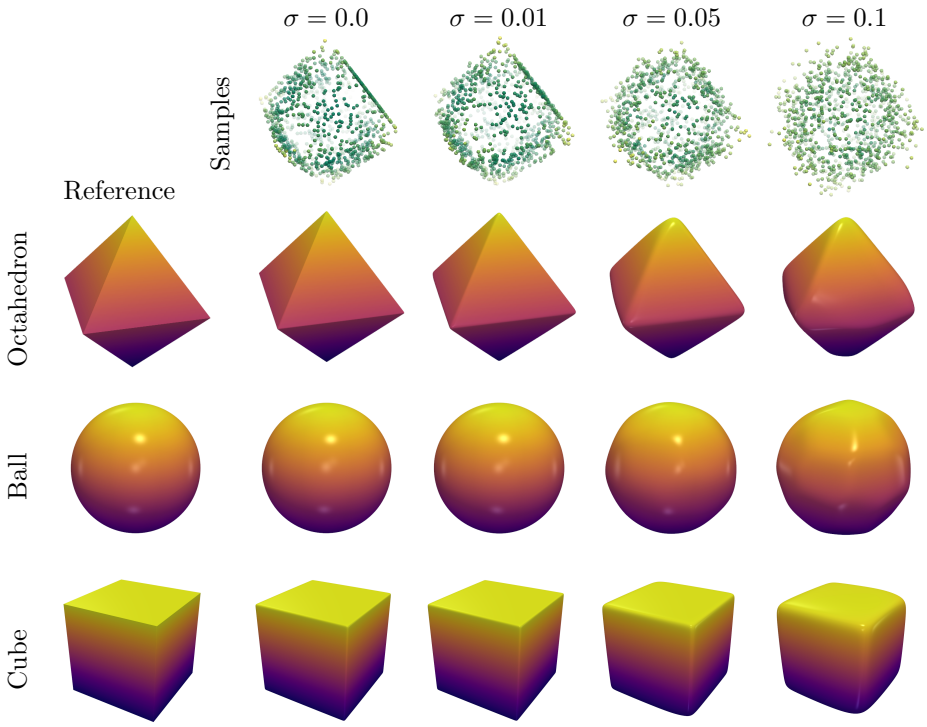

Figure 2: Reconstruction of convex shapes from noisy point clouds. First column: reference shapes. Top row: input point clouds sampled from the octahedron under increasing Gaussian noise levels σ. Remaining rows: reconstructed shapes for different target geometries.

are conducted on a single CPU (AMD Ryzen 7 Pro) with 28 GB of RAM. The computational time for all two-dimensional cases is on the order of one minute, while in three dimensions it is typically on the order of ten minutes. The implementation is also CUDA-compatible and can therefore benefit from standard GPU acceleration. Contrary to classical numerical convex shape optimization, every shape optimization problems considered here reduces to an unconstrained optimization problem.

Note that a direct quantitative comparison with classical methods of convex shape optimization is not possible in practice, as no publicly available implementations seems to exist in a form that would allow a consistent and reproducible evaluation.

4.1 Learning convex sets with noisy boundary samples

The first problem we consider is to reconstruct a convex shape from noisy observations of its boundary. More precisely, given samples $y_1, \dots, y_n \in \mathbb{R}^d$, we minimize the loss function

$$L(\theta) = \sum_{i=1}^{n} |p_\theta(y_i) - 1|^2$$

where p_θ is the gauge function of the convex set Ω_θ. This loss is motivated by the fact that $\partial\Omega_\theta = \{p_\theta = 1\}$.

In practice, the observations y_i are generated synthetically as follows: first, $x_i \sim \mathcal{U}(\partial B)$. Then, $y_i = \phi_{\text{target}}(x_i) + \varepsilon_i$ where $\varepsilon_i \sim \mathcal{N}(0, \sigma)$, and ϕ_{target} denotes a maps from B to Ω_{target}. The results, shown in Fig. 2, are obtained with $n = 1000$ samples and varying noise levels across three different shapes. We observe that the convex inductive bias enables an accurate reconstruction of the shapes, even with a relatively small number of samples and substantial noise. Additional statistical experiments assessing sensitivity to noise and to the amount of samples over multiple runs are reported in Section C.

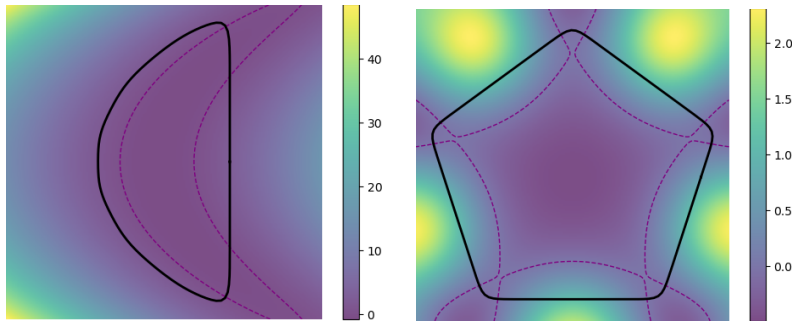

Figure 3: Optimal shape for the Poisson-based shape optimization problem (in black) along with the function f. The dotted line is the 0 level set.

Table 2: Optimal values for L_{Vol} and L_{Per} in several dimensions.

Dimension	L^*_{Vol}	L^*_{Per}
2	0.35809	0.09886
3	0.30918	0.13842
4	0.28397	0.15532

4.2 Optimization of a Poisson problem

One of the simplest PDE-constrained shape optimization problem is the following: considering a function $f : \mathbb{R}^d \to \mathbb{R}$ and u_θ to be the solution of Eq. (7) on a domain $\Omega_\theta \in \mathcal{K}$, what is the minimum of $J(\Omega_\theta) := \int_{\Omega_\theta} u_\theta$? As described previously, the solution u_θ is computed by a mesh free Galerkin method, while the integral is evaluated by change of variables. The loss function that we minimize is then

$$L(\theta) = \int_B (u_\theta \circ \phi_\theta)\text{Jac}(\phi_\theta)dx$$

We performs our experiments in the same settings as in (Bogosel, 2023), i.e. in dimension 2 for two different functions f. The optimal shapes are given in Fig. 3. They can be compared to the ones in (Bogosel, 2023).

4.3 Maximization of the gradient of the torsion function

In linear elasticity, the quantity $\|\nabla u_\Omega\|_{L^\infty}$ (where u_Ω is the torsion function Eq. (13)) represents the maximal shear stress of a rod with section Ω. In the case of a planar convex domain $\Omega \subset \mathbb{R}^d$, many authors are interested in knowing what shape maximizes this quantity when prescribing the area or the perimeter (see (Burdzy et al., 2025) and references therein). Using homogeneity arguments, it amounts at maximizing respectively $\frac{\|\nabla u_\Omega\|_{L^\infty}}{\text{Vol}(\Omega)^{1/d}}$ and $\frac{\|\nabla u_\Omega\|_{L^\infty}}{\text{Per}(\Omega)^{1/(d-1)}}$. Assuming enough regularity, the maximum principle applied on $|\nabla u|^2$ ensures that the infinity norm is attained on the boundary, i.e. $\|\nabla u_\Omega\|_{L^\infty(\Omega)} = \|\partial_n u\|_{L^\infty(\partial\Omega)}$. For a parametrized convex set Ω_θ, maximizing the previous quantities is hence equivalent to maximizing respectively the two loss functions

$$L_{\text{Vol}}(\theta) := \frac{|\partial_n u_{\Omega_\theta}(\phi_\theta(x))|}{\text{Vol}(\Omega_\theta)^{1/d}} \qquad \text{and} \qquad L_{\text{Per}}(\theta) := \frac{|\partial_n u_{\Omega_\theta}(\phi_\theta(x))|}{\text{Per}(\Omega)^{1/(d-1)}}.$$

for a fixed $x \in \partial B$. The optimal shapes in dimension 2 and 3 are given in Fig. 4. In order to show that our algorithm easily adapts to higher dimensions, we give in Table 2 the optimal values obtained with our method for each functionals in dimension up to 4. In dimension 2, the values that are obtained matches previous numerical experiments found in the literature (Burdzy et al., 2025).

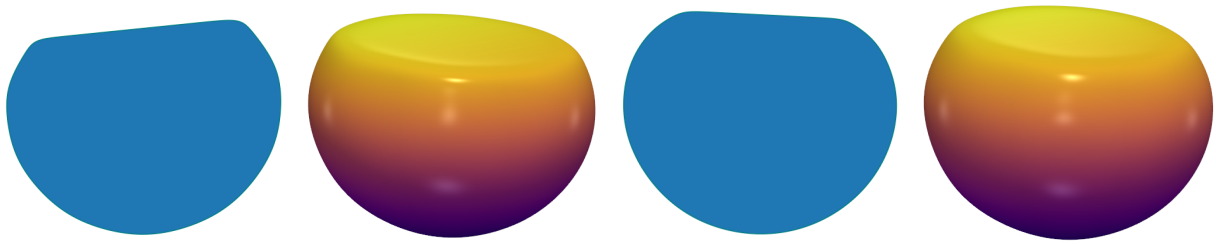

Figure 4: Optimal shapes for maximizing the torsion gradient under two constraints: fixed volume (left) and fixed perimeter (right). While the resulting shapes are similar, the choice of normalization produces distinct geometries.

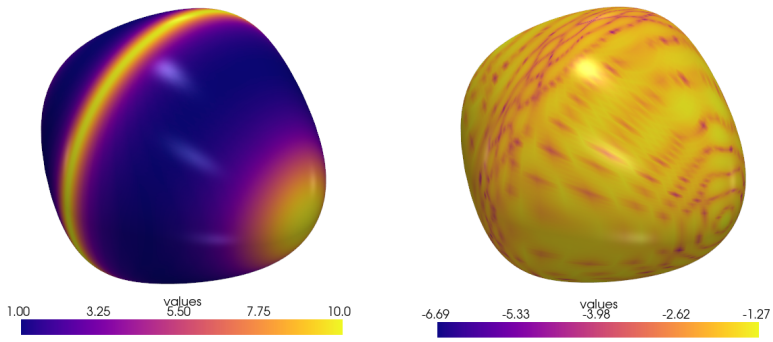

Figure 5: Solution to the Minkowski problem. Left: prescribed curvature. Right: $\log_{10}$ relative error. The method accurately recovers the target curvature distribution.

4.4 Minkowski problem

The Minkowski problem Minkowski (1989) (see (Huang et al., 2025) for a modern exposition) is a foundational problem in convex and differential geometry, connected to the famous Monge–Ampère equation. Informally speaking, it is concerned with the question of existence of a convex set with prescribed Gaussian curvature. More precisely, let $g : \mathbb{S}^{n-1} \to \mathbb{R}$ be a positive, continuous function. Does there exists a convex set Ω such that $\kappa_\Omega \circ n_\Omega^{-1} = g$ on ∂B? It is known that such Ω exists if and only if g verifies $\int_{\partial B} \frac{u}{g(u)} du = 0$. We can approximate this problem by formulating it as a regression problem, by minimizing the relative mean squared error between the actual and target curvature:

$$L(\theta) = \int_{\partial B} \left| \frac{\kappa_\theta(n_\theta^{-1}(u)) - g(u)}{g(u)} \right|^2 du.$$

Except for n_θ^{-1}, everything is easily computable in the present framework. However, as it is pointed out in (Schneider, 2013), if p_θ is the support function of Ω_θ (i.e., choosing the parametrization Eq. (5)) we have $n_\theta^{-1} = \nabla p_\theta = \phi_\theta$ on ∂B. Hence, the previous loss is readily computable. We show that our method successfully applies to the Minkowski problem in Fig. 5. The final L^2 relative error is of the order of 10^{-2}.

4.5 Minimization of the Mahler volume

Gauge and support functions are related to each other through the notion of *polar body*. The polar body of a convex set $K \in \mathcal{K}$ is defined as $K^\circ := \{x \in \mathbb{R}^d : x \cdot y \leq 1 \text{ for all } y \in K\}$. Then, the gauge function of K is the support function of K° (see (Schneider, 2013, Lemma 1.7.13)), i.e. $g_K = h_{K^\circ}$.

In this section, we are interested in the minimization of the *Mahler volume* which is the product of the volumes of a convex Ω and its polar body Ω°, i.e. $\mathrm{Vol}_M(\Omega) := \mathrm{Vol}(\Omega)\mathrm{Vol}(\Omega^\circ)$. Our framework allows us to easily compute both volumes using simultaneously the support and the gauge parametrization. The loss

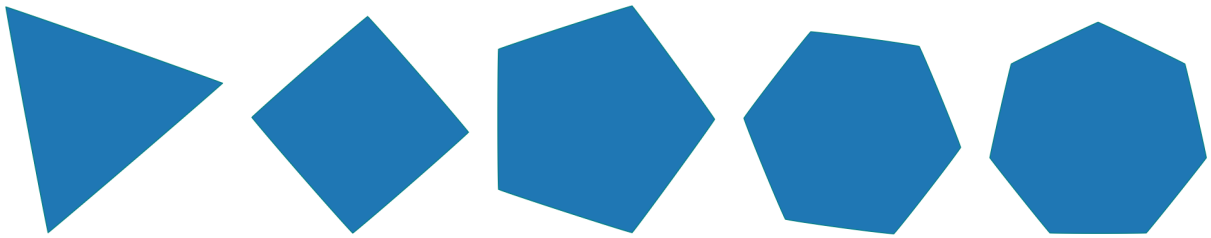

Figure 6: Minimizers of the Mahler volume under n-fold symmetry constraints. The method recovers polygonal structures, highlighting its ability to represent non-smooth convex shapes.

function that we minimize is

$$L(\theta) = \left(\int_B \mathrm{Jac}(\phi_\theta^s)dx\right)\left(\int_B \mathrm{Jac}(\phi_\theta^g)dx\right)$$

where ϕ_θ^s and ϕ_θ^g are defined using the same sublinear layer p_θ.

In $\mathbb{R}^2$, it is known that the convex set with n-fold rotational symmetry that minimizes the Mahler volume is the regular polygon, and the optimal value is $n^2 \sin(\pi/n)^2$ (Böröczky et al., 2013). Using our parametrization of symmetric convex sets, we get the results presented in Fig. 6. As you can see, our parametrization is flexible enough to recover polygons.

4.6 Ablation study

In order to shed some light on the practical approximation power of the proposed sublinear layer, we perform an ablation study on the number of neurons. Here, we assume that our parametrized sets $\Omega_\theta \subset \mathbb{R}^d$ are parametrized by the support function p_θ. The purpose will be to approximate random ellipses Ω_{target} with support function $p_{\text{target}}(x, y) = ((ax)^2 + (by)^2)^{\frac{1}{2}}$, where $a, b \sim \mathcal{U}([1, 20])$ are the axes of the ellipse. We chose to focus on ellipses since their support function can not be as easily described as, e.g., random polygons (since a m-sided polygon can be approximated almost exactly with a sublinear layer of m neurons and very large β). To asses the approximation power of our parametrization, we define the relative error

$$E\left(\Omega_\theta, \Omega_{\text{target}}\right) = \frac{d_H(\Omega_\theta, \Omega_{\text{target}})}{\mathrm{diam}(\Omega_{\text{target}})}.$$

According to Theorem 2.1, there exists $C > 0$ such that for all $m \in \mathbb{N}$ there exists $\theta = (\beta, W) \in \mathbb{R} \times \mathbb{R}^{2\times m}$ such that

$$E(\Omega_\theta, \Omega_{\text{target}}) \leq \frac{C}{m^2}. \tag{8}$$

Since Ω_θ and Ω_{target} are represented using their support function, we can write the Hausdorff distance as $d_H(\Omega_\theta, \Omega_{\text{target}}) = \|p_\theta - p_{\text{target}}\|_{C(\mathbb{S}^1)} = \sup_{x\in\mathbb{S}^1} |p_\theta(x) - p_{\text{target}}(x)|$. The diameter can be expressed as $\mathrm{diam}(\Omega_{\text{target}}) = \sup_{x\in\mathbb{S}^1} |p_{\text{target}}(x) + p_{\text{target}}(-x)|$.

In order to compute those quantities numerically, the suprema are approximated by suprema over $n = 2000$ samples $x_1, \dots x_n$ equidistributed on $\mathbb{S}^1$. Moreover, since we will rely on $L-BFGS$ for optimization, we smooth the supremum in the Hausdorff distance using a LSE with temperature $\tau = \frac{\varepsilon}{\log(m)}$, where $\varepsilon = 10^{-5}$. This leads to the loss

$$L(\theta) = \frac{\tau \mathrm{LSE}\left(\frac{|p_\theta(x_1) - p_{\text{target}}(x_1)|}{\tau}, \dots, \frac{|p_\theta(x_n) - p_{\text{target}}(x_n)|}{\tau}\right)}{\max_{1\leq i\leq n} |p_{\text{target}}(x_i) + p_{\text{target}}(-x_i)|}.$$

The ablation study works as follows: we first sample m on a logarithmic scale ranging from $m = 3$ to $m \approx 300$ neurons. Then, for a given m, we sample 20 random ellipses Ω_{target}. For each ellipse, we minimize the

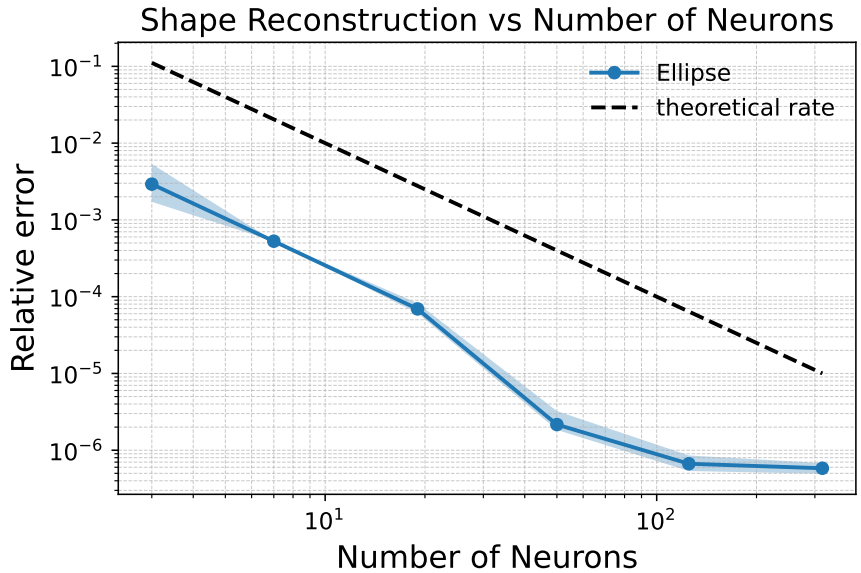

Figure 7: Result of the ablation study. The method recovers the theoretical rates.

loss $L(\theta)$ using a sublinear layer p_θ with m neurons. We then evaluate the relative error $E(\Omega_\theta, \Omega_{\text{target}})$ by approximating it on 10^5 points sampled on $\mathbb{S}^1$. We then compute, for a given m, the median and interquartile range of $E(\Omega_\theta, \Omega_{\text{target}})$ for the 20 different runs.

The result is displayed in Fig. 7. The resulting values are plotted in log-log scale, along with the theoretical rate Eq. (8). We see that the method allows to roughly recover the theoretical rate predicted by Theorem 2.1. Around $m = 300$ neurons, the empirical error tends to stall a bit, which could be explained by the fact that the loss becomes increasingly non-convex, making it difficult for the optimizer to find the optimal θ.

4.7 Comparison with PINNs

As it has already been stressed, PINNs are not particularly well suited in general for the computation of PDE-constrained shape optimization. However, there is some very specific problems for which they can easily be applied, namely when the objective function is the *Dirichlet energy* of the shape. This is the type of energy that has been considered in (Bélières Frendo et al., 2025). In this section, we will compare the use of PINNs with the other methods previously introduced on a similar problem for which we know the analytical solution.

Consider the following problem:

$$\max_{\substack{\Omega \in \mathcal{K} \\ \text{Vol}(\Omega)=1}} T(\Omega) \tag{9}$$

where T is the torsional rigidity defined in Eq. (14). A foundational result in shape optimization, called the Saint–Venant inequality, states that the solution of this problem is the ball (Brasco and De Philippis, 2016).

The Dirichlet energy associated with Eq. (13) is defined as

$$E(\Omega, v) := \frac{1}{2} \int_\Omega |\nabla v|^2 - \int_\Omega v \tag{10}$$

for $v \in H_0^1(\Omega)$. A fundamental property of the Dirichlet energy is that the minimizer u_Ω of $E(\Omega, \dot{)}$ is the solution of Eq. (13). Using the fact that $\int_\Omega |\nabla u_\Omega|^2 = \int_\Omega u_\Omega = T(\Omega)$, this implies that $E(\Omega, u_\Omega) = -\frac{T(\Omega)}{2}$. Hence:

$$\min_{\substack{\Omega \in \mathcal{K}, v \in H_0^1(\Omega) \\ \text{Vol}(\Omega)=1}} E(\Omega, v) = \min_{\substack{\Omega \in \mathcal{K} \\ \text{Vol}(\Omega)=1}} \min_{v \in H_0^1(\Omega)} E(\Omega, v) = \min_{\substack{\Omega \in \mathcal{K} \\ \text{Vol}(\Omega)=1}} E(\Omega, u_\Omega) = -\frac{1}{2} \max_{\substack{\Omega \in \mathcal{K} \\ \text{Vol}(\Omega)=1}} T(\Omega),$$

which means that minimizing E in both Ω and v is actually equivalent to Eq. (9). Using the following proposition, we can further reformulate the problem:

Proposition 4.1. *Any minimizer* (Ω, v) *of*

$$\min_{\substack{\Omega\in\mathcal{K}, v\in H_0^1(\Omega)\\ Vol(\Omega)=1}} E(\Omega, v) \tag{11}$$

is a minimizer of

$$\min_{\Omega\in\mathcal{K}, v\in H_0^1(\Omega)} \frac{E(\Omega, v)}{Vol(\Omega)^{\frac{d+2}{d}}}. \tag{12}$$

Reciprocally, if (Ω, v) *is a minimizer of Eq.* (12) *then* $\left(\alpha_\Omega\Omega, \alpha_\Omega^2 v(\cdot/\alpha_\Omega)\right)$ *where* $\alpha_\Omega = Vol(\Omega)^{-1/d}$ *is a minimizer of Eq.* (11).

Proof. Using the change of variable formula, we can show that for $\alpha > 0$,

$$E\left(\alpha\Omega, \alpha^2 v(\cdot/\alpha)\right) = \alpha^{d+2} E(\Omega, v).$$

The proposition follows. □

Therefore, we can actually drop the volume constraint and instead solve Eq. (12). For a parametrized set Ω_θ and $v \in H_0^1(\Omega_\theta)$, we have

$$E(\Omega_\theta, v) = \frac{1}{2}\int_B A_\theta \nabla(v\circ\phi_\theta^{-1})\cdot\nabla(v\circ\phi_\theta^{-1}) - \int_B (\mathrm{Jac}\phi_\theta)(v\circ\phi^{-1});$$

moreover, minimizing this last expression with respect to $v \in H_0^1(\Omega_\theta)$ is equivalent to minimizing

$$F(\Omega_\theta, v) = \frac{1}{2}\int_B A_\theta \nabla v\cdot\nabla v - \int_B (\mathrm{Jac}\phi_\theta) v.$$

for $v \in H_0^1(B)$. Finally, by taking v to be a neural network v_η where η are its parameters, we can approximate Eq. (9) by equivalently minimizing the loss

$$L(\theta, \eta) := \frac{F(\Omega_\theta, v_\eta)}{\mathrm{Vol}(\Omega)^{\frac{d+2}{d}}}.$$

Remark 4.1. *In order to exactly impose the Dirichlet boundary condition on* ∂B*, we chose* v_η *of the form* $v_\eta(x) = dist_B(x) MLP_\eta(x)$ *where* $dist_B$ *is the signed distance function of* B*.*

This approach follows the line of the so-called *Deep Ritz Method* (Yu et al., 2018). In opposition to what was previously described in this paper and to classical shape optimization algorithms, one does not need to fully solve the state equation before updating the shape; the state and the shape are jointly optimized, and one can expect v_θ to be close to the solution of the state equation only at convergence. Since we do not have to solve a PDE at each iteration, one could expect a certain speedup of this method compared to classical ones; it is however not the case, as is shown in the next experiment.

Evaluation of the method: We compare the speed and accuracy of the previously described method with the maximization of $\frac{T(\Omega_\theta)}{\mathrm{Vol}(\Omega_\theta)^{\frac{d+2}{d}}}$ where $T(\Omega_\theta)$ is computed using either the method of fundamental solutions or the mesh free Galerkin method. While the former is expected to be extremely fast and accurate, it doesn't generalize well to other problems with non-constant source term. On the contrary, the latter is expected to be slower due to the fact that it is not taylor to this particular problem; the tradeoff being that it is able to treat a large variety of problems.

Since we know that the optimal shape is the ball, we can measure the discrepency between the shape Ω_θ and B using the *isoperimetric deficit* (Fusco et al., 2008), defined as

$$D(\Omega_\theta) = c_d \frac{\mathrm{Per}(\Omega_\theta)}{\mathrm{Vol}(\Omega_\theta)^{\frac{d-1}{d}}} - 1$$

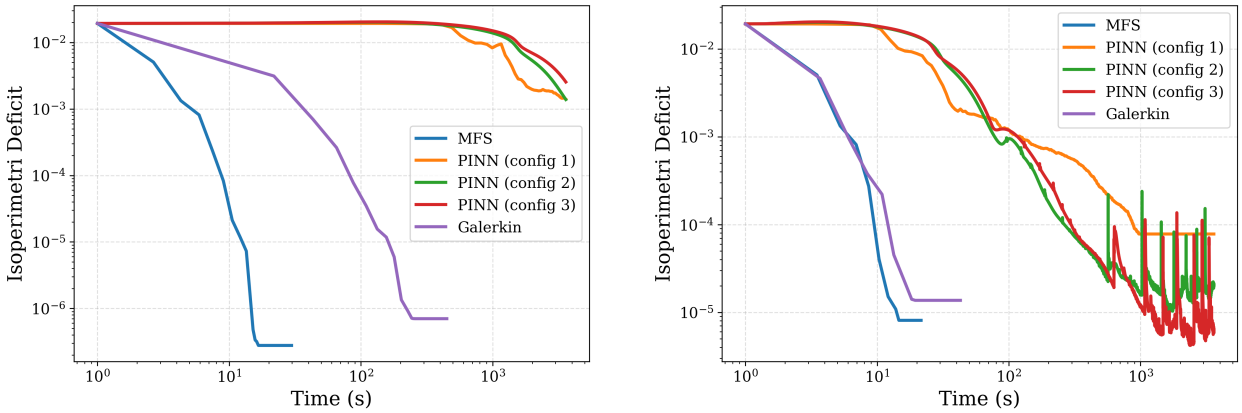

Figure 8: Runtime vs accuracy comparison between classical solvers and PINN-based approaches on CPU (Left) and GPU (Right). Classical methods on CPU achieve higher accuracy at significantly lower computational cost.

with $c_d = (d\text{Vol}(B)^{1/d})^{-1}$. The smallest the isoperimetric deficit, the closest Ω_θ is from the ball.

In order to give the best advantage to the PINN approach, we first perform a hyperparameter search for the PINN-based method. The tested parameters are given in Table 3, along with the two best configurations, that we picked in our final experiment. The number of points used for integral evaluation is the same across all experiments ($n = 20000$). The evaluation of the perimeter and volume for the computation of the isoperimetric deficit is made using $n = 100000$ points. On the 72 possible choices of parameters, only 3 reached the precision of 10^{-5} eventually; we decided to keep the three of them to compare with the classical methods. These configurations are decribed in Table 3.

Table 3: Hyperparameter search space (left) and the best three configurations that are kept for the final experiment.

Hyperparameter	Values	Config 1	Config 2	Config 3
Neurons per layer	$\{32, 64\}$	32	32	32
Depth	$\{2, 3, 4\}$	3	3	2
Activation Function	{ sin, tanh }	tanh	tanh	tanh
Optimizer	{Adam, L–BFGS}	L–BFGS	Adam	Adam
Learning rate	$\{10^{-1}, 10^{-2}, 10^{-3}\}$	10^{-2}	10^{-2}	10^{-2}

In Fig. 8, we show the comparison between the two classical methods and the three PINN-based ones, both on CPU (AMD Ryzen Pro 7) and on GPU (Nvidia L40). We see that the classical methods consistently outperforms the PINN-based one; interestingly enough, both classical methods ran on CPU are still faster and more accurate than the PINN-based methods ran on GPU. This experiment show that, even in the most favorable setting, the PINN approach is outperformed by the classical methods by several orders of magnitude.

Reproducibility The code is fully available on GitHub: `https://anonymous.4open.science/r/SublinearNet-5726/`.

5 Discussion

The proposed approach allows for a provably expressive and unconstrained representation of convex sets. Profiting from the auto-differentiation capabilities of PyTorch, it allows to easily solve a large variety of shape optimization problems without any shape derivative computation.

Limitations However, several limitations remain. The main one is the possible extension from a shallow, one-layer network to a deeper architecture. Our approach relying heavily on convex conjugation, such a generalization is not straightforward, due to its complex interaction with function composition. Indeed, convex conjugation of a convex function f with a function L is only tractable through infimal postcomposition Definition A.1 when L is linear. If L is non-linear, no simple closed-form expression exists for the conjugate of a composition. A deep neural network being a composition of multiple non-linearities, it is not clear how to generalize our approach in this setting.

The reliance on quasi-Newton methods like L–BFGS with fixed discretizations limits scalability due to the curse of dimensionality. This could however be mitigated by first training using Monte–Carlo sampling with a first–order optimizer, then fine–tuning using L–BFGS. Certain geometric quantities (e.g., curvature) may become unstable near non-smooth shapes. Memory usage is also higher than in classical approaches, since PDE solvers must remain in the computational graph. Finally, symmetry enforcement increases computational cost linearly with respect to the size of the symmetry group due to repeated evaluations.

Broader impact We validate the proposed method primarily on mathematical shape optimization problems, where it may help accelerate exploration and conjecture generation. Extending the framework to more realistic engineering settings—such as structural or aerodynamic optimization—could further broaden its applicability and impact.

6 Conclusion

We introduced a neural parametrization of convex sets based on sublinear layer, ensuring exact convexity while retaining strong approximation capabilities. We proved universal approximation in the Hausdorff distance and showed how the parametrization enables efficient evaluation of geometric and PDE-dependent quantities via automatic differentiation.

Numerical experiments demonstrate competitive performance on a range of shape optimization problems, and show that classical solvers combined with our parametrization outperform PINN-based approaches in both accuracy and efficiency.

Future work includes extending the framework to more complex PDEs (e.g., elasticity or fluid dynamics) and designing architectures that preserve additional geometric constraints, like the volume (using measure-preserving neural networks) or the perimeter, for which a new architecture must probably be derived.

References

Yaoqing Yang, Chen Feng, Yiru Shen, and Tianyue Dong. FoldingNet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 206–215, 2018.

Despoina Paschalidou, Angelos Katharopoulos, Antonio Frassetto, Cordelia Schmid, François Fleuret, Juan Nieto, Federico Tombari, Marc Pollefeys, and Peter Gehler. Neural parts: Learning expressive 3d shape abstractions with invertible neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3204–3215, 2021.

Theo Deprelle, Alexis Thibault, Theo Ben-Hamu, Ersin Yumer, Vladimir G. Kim, and Mathieu Aubry. Learning elementary structures for 3d shape generation and matching. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7433–7442, 2019.

Seungyeon Shin, Dongju Shin, and Namwoo Kang. Topology optimization via machine learning and deep learning: a review. *Journal of Computational Design and Engineering*, 10(4):1736–1766, 2023.

Zhiwei Zhao, Changqing Liu, Yingguang Li, Zhibin Chen, and Xu Liu. Diffeomorphism neural operator for various domains and parameters of partial differential equations. *Communications Physics*, 8(1), January 2025. ISSN 2399-3650. doi: 10.1038/s42005-024-01911-3. URL `http://dx.doi.org/10.1038/s42005-024-01911-3`.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to PDEs. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

Grégoire Allaire, Charles Dapogny, and François Jouve. Shape and topology optimization. In *Handbook of numerical analysis*, volume 22, pages 1–132. Elsevier, 2021.

Amaury Bélières Frendo, Emmanuel Franck, Victor Michel-Dansac, and Yannick Privat. Volume-preserving geometric shape optimization of the Dirichlet energy using variational neural networks. *Neural Networks*, 184:106957, 2025.

Abdelkrim Chakib, Ibrahim Khalil, and Azeddine Sadik. An improved numerical approach for solving shape optimization problems on convex domains. *Numerical Algorithms*, 96(2):621–663, 2024.

David Colton and Rainer Kress. *Inverse Acoustic and Electromagnetic Scattering Theory*. Springer International Publishing, 2019. ISBN 9783030303518. doi: 10.1007/978-3-030-30351-8. URL `http://dx.doi.org/10.1007/978-3-030-30351-8`.

Hairong Liu, Wenyu Liu, and Longin Jan Latecki. Convex shape decomposition. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 97–104, 2010. doi: 10.1109/CVPR.2010.5540225.

Giuseppe Buttazzo and Paolo Guasoni. Shape optimization problems over classes of convex domains. *Journal of Convex Analysis*, 4:343–352, 1997.

Eloi Martinet. Numerical optimization of neumann eigenvalues of domains in the sphere. *Journal of Computational Physics*, 508:113002, 2024. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2024.113002. URL `https://www.sciencedirect.com/science/article/pii/S0021999124002511`.

Édouard Oudet. Numerical minimization of eigenmodes of a membrane with respect to the domain. *ESAIM Control Optim. Calc. Var.*, 10(3):315–330, 2004.

Thomas Lachand-Robert and Édouard Oudet. Minimizing within Convex Bodies Using a Convex Hull Method. *SIAM J. Optim.*, 2006.

Tommy Bonnesen and Werner Fenchel. *Theorie der konvexen Körper*, volume 3 of *Ergebnisse der Mathematik und ihrer Grenzgebiete*. Julius Springer, Berlin, 1934.

Peter M. Gruber and Jörg M. Wills, editors. *Handbook of Convex Geometry*. North-Holland, Amsterdam, 1993. ISBN 9780444895974. doi: 10.1016/C2009-0-15706-9.

R Tyrrell Rockafellar. *Convex analysis*, volume 28. Princeton university press, 1997.

R. Schneider. *Convex Bodies: The Brunn-Minkowski Theory*. Cambridge University Press, 2nd expanded edition edition, 2013.

Helmut Groemer. Fourier series and spherical harmonics in convexity. In Peter M. Gruber and Jörg M. Wills, editors, *Handbook of Convex Geometry*, volume 2, pages 1259–1295. North-Holland, Amsterdam, 1993. ISBN 9780444895974.

Pijush K. Ghosh and K. Vinod Kumar. Support function representation of convex bodies, its application in geometric computing, and some related representations. *Computer Vision and Image Understanding*, 72(3):379–403, December 1998. ISSN 1077-3142. doi: 10.1006/cviu.1998.0674.

Édouard Oudet. Shape Optimization Under Width Constraint. *Discrete Comput. Geom.*, 49(2):411–428, 2013.

Térence Bayen and Didier Henrion. Semidefinite programming for optimizing convex bodies under width constraints. *Optim. Methods Softw.*, 2012.

Beniamin Bogosel. Numerical Shape Optimization Among Convex Sets. *Appl. Math. Optim.*, 87(1):1, 2023.

Beniamin Bogosel, Antoine Henrot, and Marco Michetti. Optimization of Neumann Eigenvalues Under Convexity and Geometric Constraints. *SIAM J. Math. Anal.*, November 2024.

Ilias Ftouhi. Improved Description of Blaschke–Santaló Diagrams via Numerical Shape Optimization. *Appl. Math. Optim.*, 91(3):55, 2025.

Pedro R. S. Antunes and Beniamin Bogosel. Parametric shape optimization using the support function. *Comput. Optim. Appl.*, 82(1):107–138, 2022.

L Lamberg and M Kaasalainen. Numerical solution of the minkowski problem. *Journal of computational and applied mathematics*, 137(2):213–227, 2001.

Sören Bartels and Gerd Wachsmuth. Numerical Approximation of Optimal Convex Shapes. *SIAM J. Sci. Comput.*, April 2020.

Eloi Martinet and Leon Bungert. Meshless Shape Optimization Using Neural Networks and Partial Differential Equations on Graphs. In *Scale Space and Variational Methods in Computer Vision*, pages 285–297. 2025a.

Brandon Amos, Lei Xu, and J. Zico Kolter. Input convex neural networks. In *International Conference on Machine Learning (ICML)*, 2017.

Jun Liu, Kehui Zhang, Xue-Cheng Tai, and Shousheng Luo. Convex shape prior for deep convolution neural network-based image segmentation. *Journal of Mathematical Imaging and Vision*, 67(6):61, 2025.

B. Deng, K. Genova, S. Yazdani, S. Bouaziz, G. Hinton, and A. Tagliasacchi. Cvxnet: Learnable convex decomposition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020a.

Lenka Tětková, Thea Brüsch, Teresa Dorszewski, Fabian Martin Mager, Rasmus Ørtoft Aagaard, Jonathan Foldager, Tommy Sonne Alstrøm, and Lars Kai Hansen. On convex decision regions in deep network representations. *Nature Communications*, 16(1):5419, 2025.

Shadi Haddad and Abhishek Halder. Convex and nonconvex sublinear regression with application to data-driven learning of reach sets. In *2023 American Control Conference (ACC)*, pages 4581–4586. IEEE, 2023.

Theo X. Olausson, João Monteiro, Michal Klein, and Marco Cuturi. Amortized maximum inner product search with learned support functions, 2026. URL `https://arxiv.org/abs/2603.08001`.

Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. Cvxnet: Learnable convex decomposition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 31–44, 2020b.

Eloi Martinet and Leon Bungert. Meshless shape optimization using neural networks and partial differential equations on graphs. In *International Conference on Scale Space and Variational Methods in Computer Vision*, pages 285–297. Springer, 2025b.

Ian Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. In *International conference on machine learning*, pages 1319–1327. PMLR, 2013.

Efim M Bronshteyn and Leonid D Ivanov. The approximation of convex sets by polyhedra. *Siberian Mathematical Journal*, 16(5):852–853, 1975.

Richard M Dudley. Metric entropy of some classes of sets with differentiable boundaries. *Journal of Approximation Theory*, 10(3):227–236, 1974.

Omri Puny, Matan Atzmon, Heli Ben-Hamu, Ishan Misra, Aditya Grover, Edward J Smith, and Yaron Lipman. Frame averaging for invariant and equivariant network design. *arXiv preprint arXiv:2110.03336*, 2021.

Eloi Martinet and Ilias Ftouhi. Numerical exploration of the range of shape functionals using neural networks. *arXiv preprint arXiv:2602.14881*, 2026.

Lawrence C. Evans and Ronald F. Gariepy. *Measure theory and fine properties of functions*. 2025.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Álvaro González. Measurement of Areas on a Sphere Using Fibonacci and Latitude–Longitude Lattices. *Math. Geosci.*, 42(1):49–64, 2010.

Lawrence C Evans. *Partial differential equations*, volume 19. 2022.

Holger Wendland. Meshless galerkin methods using radial basis functions. *Mathematics of Computation*, 68 (228):1521–1531, 1999.

Gregory E Fasshauer. *Meshfree approximation methods with Matlab (With Cd-rom)*, volume 6. World Scientific Publishing Company, 2007.

MWM Gamini Dissanayake and Nhan Phan-Thien. Neural-network-based approximations for solving partial differential equations. *communications in Numerical Methods in Engineering*, 10(3):195–201, 1994.

Tamara G Grossmann, Urszula Julia Komorowska, Jonas Latz, and Carola-Bibiane Schönlieb. Can physics-informed neural networks beat the finite element method? *IMA Journal of Applied Mathematics*, 89(1): 143–174, 2024.

Nick McGreivy and Ammar Hakim. Weak baselines and reporting biases lead to overoptimism in machine learning for fluid-related partial differential equations. *Nature machine intelligence*, 6(10):1256–1269, 2024.

Krzysztof Burdzy, Ilias Ftouhi, and Phanuel Mariano. Geometric properties of optimizers for the maximum gradient of the torsion function. *arXiv preprint arXiv:2512.09400*, 2025.

Hermann Minkowski. Volumen und oberfläche. In *Ausgewählte Arbeiten zur Zahlentheorie und zur Geometrie: Mit D. Hilberts Gedächtnisrede auf H. Minkowski, Göttingen 1909*, pages 146–192. Springer, 1989.

Yong Huang, Deane Yang, and Gaoyong Zhang. Minkowski problems for geometric measures. *Bulletin of the American Mathematical Society*, 62(3):359–425, 2025.

K Böröczky, E Makai, Mathieu Meyer, and Shlomo Reisner. On the volume product of planar polar convex bodies—lower estimates with stability. *Studia Scientiarum Mathematicarum Hungarica*, 50(2):159–198, 2013.

Lorenzo Brasco and Guido De Philippis. Spectral inequalities in quantitative form. *arXiv preprint arXiv:1604.05072*, 2016.

Bing Yu et al. The deep ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018.

Nicola Fusco, Francesco Maggi, and Aldo Pratelli. The sharp quantitative isoperimetric inequality. *Annals of mathematics*, pages 941–980, 2008.

Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Heinz H Bauschke and Patrick L Combettes. Correction to: convex analysis and monotone operator theory in hilbert spaces. In *Convex analysis and monotone operator theory in Hilbert spaces*. Springer, 2020.

Alex H Barnett and Timo Betcke. Stability and convergence of the method of fundamental solutions for helmholtz problems on analytic domains. *Journal of Computational Physics*, 227(14):7003–7026, 2008.

Beniamin Bogosel. The method of fundamental solutions applied to boundary eigenvalue problems. *J. Comput. Appl. Math.*, 306:265–285, 2016.

Liviu Marin and Daniel Lesnic. The method of fundamental solutions for the cauchy problem in two-dimensional linear elasticity. *International journal of solids and structures*, 41(13):3425–3438, 2004.

Carlos JS Alves, Rodrigo G Serrao, and Ana L Silvestre. Fundamental solutions for the stokes equations: Numerical applications for 2d and 3d flows. *Applied Numerical Mathematics*, 170:55–73, 2021.

Antoine Henrot and Michel Pierre. *Shape variation and optimization*, volume 28 of *EMS Tracts in Mathematics*. European Mathematical Society (EMS), Zürich, 2018.

Christophe Geuzaine and Jean-François Remacle. Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities. *International journal for numerical methods in engineering*, 79(11): 1309–1331, 2009.

Tom Gustafsson and Geordie Drummond Mcbain. scikit-fem: A python package for finite element assembly. *Journal of Open Source Software*, 5(52):2369, 2020.

A Reminder of convex analysis

Here we give some useful results of convex analysis. The following proposition can be found in (Boyd and Vandenberghe, 2004, Exercise 3.39):

Proposition A.1. *Let $f : \mathbb{R}^d \to [-\infty, +\infty]$ be a proper convex and closed function. Then $f^{**} = f$.*

The behavior of the convex conjugate with respect to the composition needs the following definition (Bauschke and Combettes, 2020, Definition 12.34):

Definition A.1 (Infimal postcomposition)**.** *Let $f : \mathbb{R}^d \to [-\infty, +\infty]$ and $L : \mathbb{R}^d \to \mathbb{R}^m$. The infimal postcomposition of f by L is*

$$\begin{aligned} L \triangleright f \colon \mathbb{R}^m &\longrightarrow [-\infty, +\infty] \\ y &\longmapsto \inf_{Lx=y} f(x) \end{aligned}$$

In our case, we only need the conjugate of the composition with a linear map, which is given by the following proposition (adapted from (Bauschke and Combettes, 2020, Proposition 13.24)):

Proposition A.2 (Composition with a linear map)**.** *Let $f : \mathbb{R}^d \to (-\infty, +\infty]$ and $A \in \mathbb{R}^{d\times m}$. Then*

$$(f \circ A)^* \leq A^T \triangleright f^*.$$

Finally, we need to know the conjugate of the log-sum-exp function:

Proposition A.3 (Convex conjugate of the log-sum-exp)**.** *The convex conjugate of the log-sum-exp is the negative entropy function $-S$ where*

$$\begin{aligned} S \colon \Delta^N &\longrightarrow (0, +\infty) \\ y &\longmapsto -\sum_i y_i \log y_i \end{aligned}$$

where Δ^N is the N-dimensional simplex.

This can be shown by computing the optimality conditions in equation 2 (see (Boyd and Vandenberghe, 2004, Example 3.25) for more details).

For the next theorem, see (Schneider, 2013, Theorem 1.7.1):

Theorem A.1. *If $f : \mathbb{R}^d \to \mathbb{R}$ is a sublinear function, then there is a unique convex body with support function f.*

For the following property, see (Schneider, 2013, Corolary 1.7.3):

Proposition A.4. *Let Ω be a convex body. Its support function h_Ω is differentiable at $u \in \mathbb{R}^d \setminus \{0\}$ if and only if the support set at u contains only one point x. In this case, $x = \nabla h_\Omega(u)$.*

B Method of fundamental solutions

An important special case of Eq. (7) is the case where f is identically equal to 1, i.e.

$$\begin{cases} -\Delta u &= 1 \quad \text{in } \Omega_\theta, \\ u &= 0 \quad \text{on } \partial\Omega_\theta. \end{cases} \tag{13}$$

In this case, the solution $u \in H_0^1(\Omega_\theta)$ is called the *torsion function*, which is an important function in mechanics that has been extensively studied in mathematics. One important derived quantity is the *torsional rigidity* defined as

$$T(\Omega_\theta) = \int_{\Omega_\theta} u \tag{14}$$

which describes the rigidity of a rod of cross section Ω_θ.

It is well known that Eq. (13) can be solved in an extremely precise and efficient way using the *method of fundamental solutions* (Barnett and Betcke, 2008; Antunes and Bogosel, 2022; Bogosel, 2016). Indeed, by putting $\phi(x) = u(x) + \frac{x_1^2}{2}$, Eq. (13) can be equivalently reformulated as the following Laplace equation:

$$\begin{cases} -\Delta \phi &= 0 \quad \text{in } \Omega_\theta, \\ \phi &= \frac{x_1^2}{2} \quad \text{on } \partial\Omega_\theta. \end{cases} \tag{15}$$

One may then seek an approximate solution $\tilde{\phi}$ expressed as a linear combination of fundamental solutions, namely,

$$\tilde{\phi}(x) := \sum_{i=1}^{n} c_i \psi(x - y_i),$$

where ψ is the fundamental solution to $-\Delta\psi = \delta_0$ in $\mathbb{R}^d - \{0\}$ and $y_1, \ldots, y_n \in \Omega_\theta^c$ (Evans, 2022). Since ϕ is harmonic in Ω, we only have to fit the boundary condition, for instance, in an L^2 sense, which amounts at solving

$$\min_{c_1,\ldots,c_n \in \mathbb{R}} \int_{\partial\Omega} \left| \tilde{\phi}(x) - \frac{x_1^2}{2} \right|^2 d\sigma.$$

Since this integral cannot be analytically computed for a general Ω, it is discretized and the resulting least squares problem is solved using `torch.linalg.lstsq`.

There exists several valid choices for the placement of the sources $y_1, \ldots, y_n$. In our case, we decided to do it the following way: draw samples $x_1, \ldots, x_n \in \partial B$, and set $y_i := \phi_\theta(x_i) + \varepsilon n_\theta(x_i)$ for $\varepsilon > 0$. This way, we have the guarantee that y_i lies outside and at distance ε of Ω_θ. An alternative and more flexible approach in the case where Ω_θ is parametrized by a gauge function Eq. (4) is to take $y_i = \phi_\theta\left((1+\varepsilon)x_i\right)$. This has the advantage that scaling Ω_θ leads to the same scaling of the distance from y_i to Ω_θ.

It is well known that this method is very sensitive to the source placement, and that the conditioning worsens when the number of sources increase. However, this can be mitigated by an *a posteriori* error estimation using the maximum principle: knowing that $\|\phi - \tilde{\phi}\|_\infty := \sup_{\Omega_\theta} |\phi - \tilde{\phi}| = \sup_{\partial\Omega_\theta} |\phi - \tilde{\phi}| = \sup_{x \in \partial\Omega_\theta} \left| \frac{x_1^2}{2} - \tilde{\phi} \right|$, we can estimate the last quantity by sampling a large amount of points on $\partial\Omega_\theta$. If the error is greater than a certain tolerance (taken in our experiments of the order $10^{-4} - 10^{-5}$), we can modify the source placement parameter ε and increase the number of sources until the tolerance is reached.

Remark B.1. *While we decided to treat the case of the Laplace equation for simplicity, this approach can easily be extended to other, more practical settings where we have access to the fundamental solutions of the PDE, like in linearized elasticity (Marin and Lesnic, 2004) for the optimal design of structures or in the Stokes problem (Alves et al., 2021).*

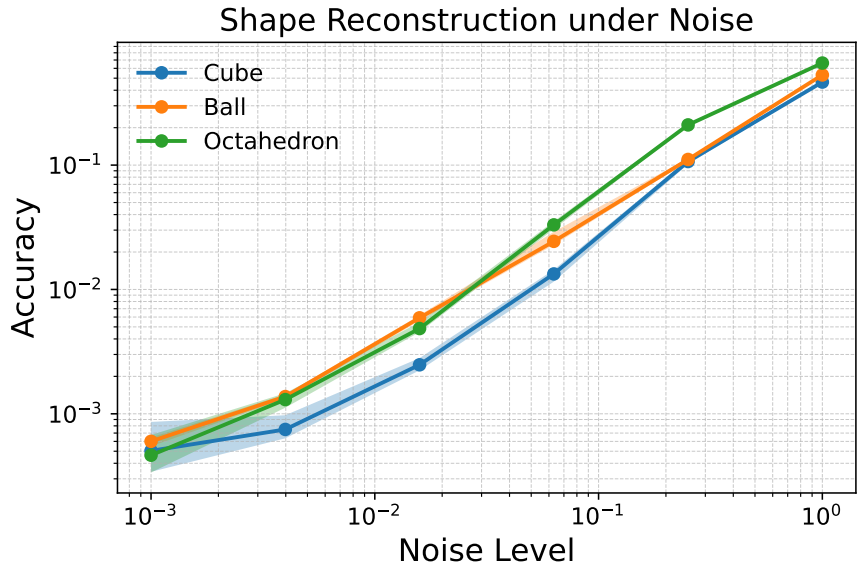

Figure 9: Robustness to noise and network initialization across 100 runs. Median reconstruction error with interquartile range. The method exhibits low variance and stable performance across noise levels.

C Statistical Analysis

We complement the experiments of Section 4.1 with a statistical evaluation of robustness to noise. For each configuration (shape and noise level), we perform the following procedure. We randomly initialize the sublinear layer p_θ, sample $n = 1000$ points $y_{i=1}^n$, and optimize the loss defined in Section 4.1.

To quantify reconstruction quality, we measure the L^2 discrepancy between the learned function p_θ and the gauge function of the target shape g_{target}. Since $g_{\text{target}} = 1$ on $\partial\Omega_{\text{target}}$, this reduces to

$$\text{Acc}(\theta) = \|g_{\text{target}} - p_\theta\|_{L^2(\partial\Omega_{\text{target}})} = \|p_\theta - 1\|_{L^2(\partial\Omega_{\text{target}})}$$

We approximate this quantity by sampling 10^5 points uniformly on ∂B and mapping them via ϕ_{target}.

Each experiment is repeated 100 times with independent random initializations and samples. We report the median performance together with the interquartile range (25th–75th percentiles). The aggregated results are shown in Fig. 9.

In a second experiment, we analyze the influence of the number of samples on the reconstruction. We fix the noise level to $\sigma = 0.01$ and use different amount of points n ranging from 10 to 10^4. The results are reported in Fig. 10.

D Integration of the finite element method

Assume that we want to minimize a certain shape function J, depending on a PDE that we would like to solve using the finite element method. While possible, it requires to compute certain derivatives manually.

Defining $j(\theta) := J(\Omega_\theta)$, we want to relate the derivatives $\partial_{\theta_k} j(\theta)$ with the shape derivative $dJ(\Omega_\theta).V$. According to (Henrot and Pierre, 2018, Chapter 5, Section 5.9), under some mild regularity assumptions, shape derivatives can be put in the form

$$dJ(\Omega).V = \int_{\partial\Omega} f(x)V(x) \cdot n_\Omega(x) \tag{16}$$

where $f : \partial\Omega \to \mathbb{R}$ depends on Ω. This formula is particularly well suited when one needs to compute f *via* a mesh-based. Now, let $\theta = (\theta_1, \ldots, \theta_n) \in \mathbb{R}^d$ and define $\bar{\theta}(t) := (\theta_1, \ldots, \theta_k + t, \ldots, \theta_n)$. Following (Henrot and

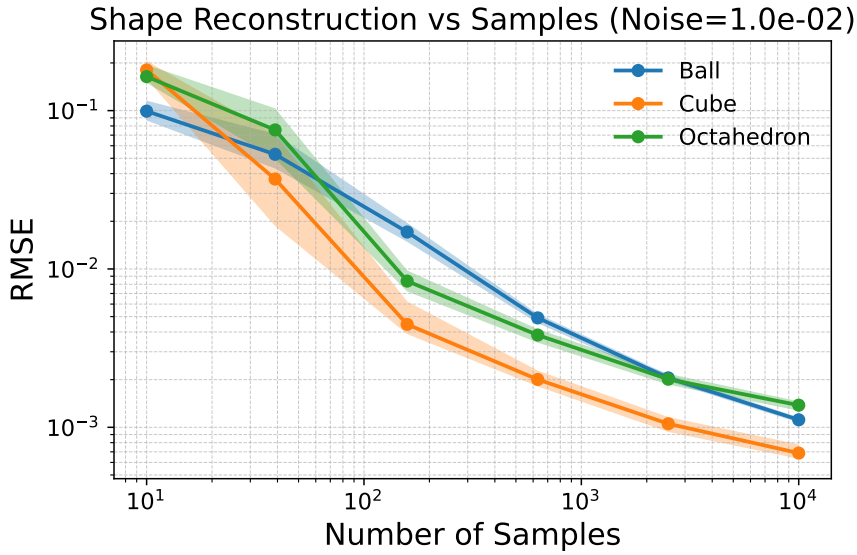

Figure 10: Robustness to amounts of samples and network initialization across 100 runs. Median reconstruction error with interquartile ranges. As expected, the variance is higher when using fewer samples.

Pierre, 2018, Chapter 5, Section 5.2), define

$$\Phi(t)(x) = \phi_{\bar{\theta}(t)} \circ \phi_\theta^{-1}(x)$$

for $t \in \mathbb{R}$ and $x \in \Omega_\theta$. We have $\Phi(t)(\Omega_\theta) = \phi_{\bar{\theta}(t)} \circ \phi_\theta^{-1}(\Omega_\theta) = \phi_{\bar{\theta}(t)} \circ \phi_\theta^{-1} \circ \phi_\theta(B) = \Omega_{\bar{\theta}(t)}$. Hence, by letting $V := \Phi'(0)$, we have by definition of the shape derivative that

$$dJ(\Omega_\theta).V = \lim_{t\to 0} \frac{J(\Omega_{\bar{\theta}(t)}) - J(\Omega_\theta)}{t} = \lim_{t\to 0} \frac{j(\bar{\theta}(t)) - j(\theta)}{t} = \partial_{\theta_k} j(\theta).$$

On the other hand,

$$V(x) = \Phi'(0)(x) = \left.\frac{d\phi_{\bar{\theta}(t)} \circ \phi_\theta^{-1}(x)}{dt}\right|_{t=0} = \partial_{\theta_k}\phi_\theta\left(\phi_\theta^{-1}(x)\right),$$

meaning that formally, we have

$$\partial_{\theta_k} j(\theta) = \int_{\partial\Omega_\theta} f(x)\partial_{\theta_k}\phi_\theta\left(\phi_\theta^{-1}(x)\right) \cdot n_{\Omega_\theta}(x)dx. \tag{17}$$

If ϕ_θ^{-1} is easily computable (which is the case, for instance, for Eq. (4) of inverse $\phi_\theta^{-1}(y) = \frac{p_\theta(y)}{\|y\|}y$) and one can precisely evaluate the boundary integral (for instance, if we have a mesh) then this integral can be easily computed. Otherwise, one can formulate it on the reference domain:

$$\begin{aligned}
\partial_{\theta_k} j(\theta) &= \int_{\partial B} f(\phi_\theta(x))\partial_{\theta_k}\phi_\theta\left(\phi_\theta^{-1}(\phi_\theta(x))\right) \cdot n_{\Omega_\theta}(\phi_\theta(x))\mathrm{Jac}_{\partial B}(\phi_\theta(x))dx \\
&= \int_{\partial B} (f\circ\phi_\theta)\,\partial_{\theta_k}\phi_\theta \cdot \frac{(D\phi_\theta)^{-T} n_B}{\left|(D\phi_\theta)^{-T} n_B\right|}\mathrm{Jac}_{\partial B}(\phi_\theta) \\
&= \int_{\partial B} (f\circ\phi_\theta)\,\partial_{\theta_k}\phi_\theta \cdot (D\phi_\theta)^{-T} n_B \mathrm{Jac}(\phi_\theta)
\end{aligned}$$

An example of optimal sets using this method can be found in Fig. 11, where we minimize the first 6 Dirichlet eigenvalues under volume and convexity constraint. Specifically, we solve:

$$\min_{\mathrm{Vol}(\Omega)=1} \lambda_k(\Omega) \tag{18}$$

where $\Omega \in \mathcal{K}$ and $\lambda_k(\Omega)$ is the k-th Dirichlet eigenvalue, i.e. it solves

$$\begin{cases} -\Delta u = \lambda_k(\Omega) u & \text{in } \Omega \\ u = 0 & \text{on } \partial\Omega \end{cases} \tag{19}$$

for some $u \in H_0^1(\Omega)$. This particular problem has already been considered in (Antunes and Bogosel, 2022).

In order to compute the derivatives Eq. (17) of λ_k, we created a mesh of the domain Ω_θ using `gmsh` (Geuzaine and Remacle, 2009), by giving it a sequence of points $y_i = \phi_\theta(x_i)$ where $x_i = (\cos(2i\pi/n), \sin(2i\pi/n))^T$, $1 \leq i < n$. An eigenpair $\lambda(\Omega_\theta), u(\Omega_\theta)$ is computed *via* finite elements using `scikit-fem` (Gustafsson and Mcbain, 2020). Using the Hadamard expression for the shape derivative $d\lambda(\Omega).V = \int_{\partial\Omega} |\nabla u|^2 (V \cdot n)$, we deduce that

$$\partial_{\theta_k} j(\theta) = \int_{\partial\Omega_\theta} |\nabla u(x)|^2 \partial_{\theta_k} \phi_\theta \left(\phi_\theta^{-1}(x)\right) \cdot n_{\Omega_\theta}(x) dx.$$

The differential quantity $\nabla u(x)$ is computed on the finite element space, while $\partial_{\theta_k}\phi_\theta$ is computed using AD. The integral is computed on the boundary mesh by `scikit-fem`. All of this process is wrapped in `torch.autograd.Function` in order to seamlessly integrate it into PyTorch's AD.

You can see the six optimal shapes found by the algorithm in Fig. 11 along with the corresponding eigenvalue and the relative error $E_{\text{rel}} := \frac{\lambda_k - \lambda_k^*}{\lambda_k^*}$ where λ_k^* is either the optimal value obtained numerically in (Antunes and Bogosel, 2022) or the analytical one for $k = 1$ and $k = 3$ where the optimal shape is known to be the ball. We see that the shapes are in good agreement with the previous results, and that the relative error is of order -3 to -4.

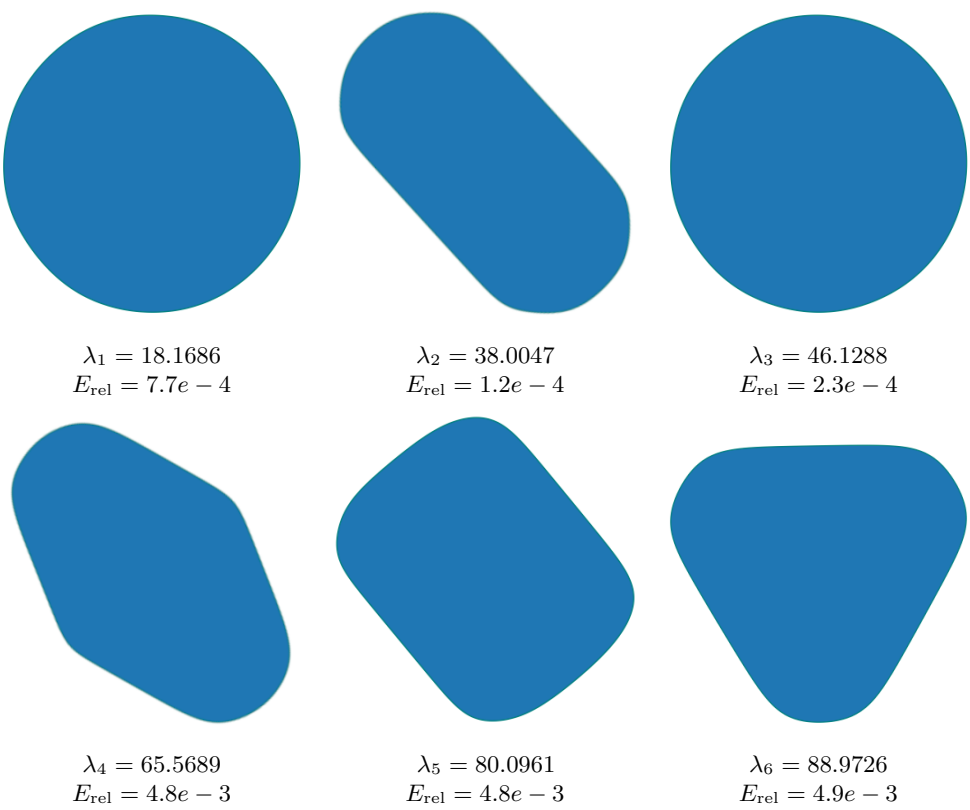

Figure 11: Optimal shapes for the first six Dirichlet eigenvalues