Transferable Foundation Models for Geometric Tasks on Point Cloud Representations: Geometric Neural Operators

B. Quackenbush[1], P. J. Atzberger[1,2]
[1] Department of Mathematics, University of California Santa Barbara (UCSB).
[2] (corresponding author), Department of Mathematics, Department of Mechanical Engineering, University of California Santa Barbara (UCSB); atzberg@gmail.com; http://atzberger.org/

We introduce methods for obtaining pretrained Geometric Neural Operators (GNPs) that can serve as basal foundation models for use in obtaining geometric features. These can be used within data processing pipelines for machine learning tasks and numerical methods. We show how our GNPs can be trained to learn robust latent representations for the differential geometry of point-clouds to provide estimates of metric, curvature, and other shape-related features. We demonstrate how our pre-trained GNPs can be used (i) to estimate the geometric properties of surfaces of arbitrary shape and topologies with robustness in the presence of noise, (ii) to approximate solutions of geometric partial differential equations (PDEs) on manifolds, and (iii) to solve equations for shape deformations such as curvature driven flows. We release codes and weights for using GNPs in the package geo_neural_op. This allows for incorporating our pre-trained GNPs as components for reuse within existing and new data processing pipelines. The GNPs also can be used as part of numerical solvers involving geometry or as part of methods for performing inference and other geometric tasks.

Introduction

A recent development in machine learning has been increasing efforts to formulate and train models for reuse across a broad range of related tasks. These are often referred to as *foundation models* to indicate they are to be built upon in order to perform further tasks [1, 2]. Prominent recent examples include large language models (LLMs) [3–5], image models [6–8], and object detection models [9–11]. Such approaches highlight the utility of having off-line training protocols and models with capabilities that are transferable to facilitate a wide range of tasks. Further specialized developments of shared off-line models even at smaller computational scales hold promise for impacting fields that include inference in scientific machine learning, physics-based feature extraction, and developing numerical solvers and simulations [12–15].

For geometric tasks, we introduce methods for developing pre-trained basal foundation models that are transferable for tasks involving point-clouds and other types of representations. Our approaches allow for discovering geometric structures and processing methods without the need for meshing or retraining to obtain information for down-stream tasks. We build on our work on Geometric Neural Operators (GNPs) in [14]. Our methods can be used to obtain estimates that include different types of curvatures, metrics, and other geometric features. Our methods also can be used as part of evaluation of differential operators on manifold surfaces, exterior calculus operations, and other procedures that arise in geometric analysis and numerical solvers for geometric PDEs [16–19]. Our training methods also incorporate approaches for obtaining estimates that are robust to noise or designed to deal with artifacts that can arise in point-cloud representations and

other datasets. Our methods allow for learning pre-trained GNP models that can then be deployed within other computational methods and data-processing pipelines to perform geometric tasks.

Geometric problems arise in many inference settings and numerical methods [20–23]. Related challenges include approaches for handling point-cloud representations in shape classification, segmentation, and [24–30], developing PDE solvers on manifolds [16, 25, 31–36], and geometric processing of meshes, lidar measurements, and other types of data [29, 37–41]. Related prior work includes development of neural operators for processing point-clouds in [42, 43]. In this work, graph neural networks are trained to mimic the action of the Laplace-Beltrami operator for use in estimation of local geometric properties, geodesic distances, deformations, and spectral reductions [42, 43]. Related work on approximating solutions of Partial Differential Equations (PDEs) on manifolds based on machine learning methods and meshfree methods include [25, 31, 32, 34, 44–50]. In these methods, estimators are developed for meshes or point-cloud representations of the manifolds to obtain geometric quantities used in evaluation of the differential operators. In [32, 44, 46, 47, 51, 52], more general neural operators are trained to learn surrogate models for parametric PDEs as part of solving inverse and design problems. The methods are trained primarily for processing data globally to obtain operators that learn implicit latent representations for making predictions for solutions over a parameterized class of shapes or PDEs. These methods have been shown to be useful for geometries and functions close to the training datasets, and to help in improving the efficiency of methods used in inverse and design problems. However, since the learned representations are implicit, this poses some limits to transferability to other geometries and problem settings. Another important issue that arises in practice for geometries obtained from data-driven methods or empirical measurements is the need for robustness to sampling noise and other artifacts.

We address these and other challenges by developing methods building on Geometric Neural Operators (GNPs) [14]. This provides the basis for obtaining robust pre-trained transferable foundation models for diverse geometric tasks. We demonstrate use cases for our GNPs showing their robustness, transferability, and characterize their other properties. This includes (i) performing validation studies against test sets involving topologies and geometric shapes not used in training, (ii) computing deformations of shapes driven by mean-curvature flows, and (iii) developing geometric PDE solvers based on the GNP models. The results indicate the pre-trained GNPs can be used as data-driven alternatives to development of hand-crafted geometric estimators, which are often technical to formulate analytically and implement in practice. Our data-driven approaches also provide ways to enhance the robustness of estimators by allowing for noise and other artifacts to be taken into account during training. We release our codes and model weights in a package `https://github.com/atzberg/geo_neural_op`. Our package can be used both for training new models or for incorporating our pre-trained GNP models into other data processing pipelines and computational methods. Our training approaches provide ways to learn data-driven filtering and adjustments for handling noise and other artifacts in point-clouds. The methods can be used to obtain transferable pre-trained models that can be used for performing diverse geometric tasks.

We organize our paper as follows. We discuss how geometric features can be learned from point cloud representations using Geometric Neural Operators (GNPs) in Section 1. We discuss our approach to obtain transferable pre-trained GNPs for geometric tasks in Section 2. We discuss results using our transferable GNP models in Section 3. This includes demonstrating the methods for (i) estimating metrics and curvatures of surfaces, (ii) shape changes driven by mean-curvature

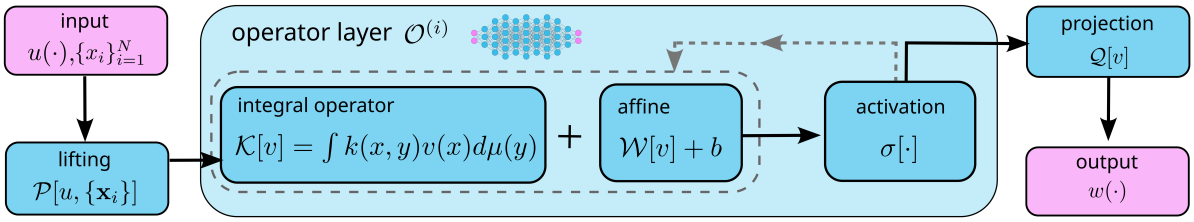

Figure 1: ***Geometric Neural Operators (GNPs).*** *We learn features from point-clouds using geometric neural operators GNPs [14]. These neural operators consist of three learnable components (i) a lifting operation $\mathcal{P}[u,\{\mathbf{x}_i\}]$ that provides initial features for the input geometry data $\{\mathbf{x}_i\}_{i=1}^N$ and input function $u(\cdot)$, (ii) layers of operators $\mathcal{O}^{(i)}$ that consist of kernel integral operations $\mathcal{K}[v](x)$ and affine operations $(\mathcal{W}v)(x)+b(x)$ each of which are passed through a non-linear activation operation $\sigma[\cdot]$, and (iii) a projection operation $\mathcal{Q}$ for constructing the final output function $w(x)=w(\xi^1,\xi^2)$ and local parameterization $\tilde{\boldsymbol{\sigma}}(\xi^1,\xi^2)=\bar{\mathbf{x}}+\xi^1\psi_1+\xi^2\psi_2$. We use geometric neural operators (GNPs) to map a collection of points $\{x_i\}_{i=1}^N$ sampled from the geometry to local parameterizations and functions $w=w(\xi^1,\xi^2)$.*

flows, and (iii) development of numerical methods for solving geometric PDEs.

1. Geometric Neural Operators (GNPs) for Point-Cloud Representations

Geometry plays an important role in many machine learning and numerical tasks. This includes classification, segmentation, and of shapes [24–30], approximation of solutions of PDEs on manifolds [16, 31, 32, 34, 36], and other tasks [37–41]. For point-cloud data, a fundamental challenge is to estimate reliably the curvature and other intrinsic geometric quantities from discrete samples of the shape. This can be especially challenging in the presence of noise and other artifacts that can arise in practice.

We develop neural operators for learning representations from the point-cloud data building on [53, 54] and our prior work introducing geometric neural operators (GNPs) in [14]. In contrast to conventional neural networks, neural operators provide a distinct class of models that can learn mappings between function spaces that are independent of the specific underlying discretizations. The key idea is to model sample evaluations of the input function as a collection of features and to learn various types of integral and linear operations that successively process the features of the input. The methods allow for readily recovering Fourier transform-based methods, kernel integral operator methods, and other non-linear transformations of functions [14, 53, 54]. This provides a powerful framework for learning operations on diverse classes of functions. We utilize this approach to learn mappings from point-cloud samplings of the geometry to functions capturing geometric features [14].

1.1. Learning Latent Geometric Features with GNPs

We develop methods building on Geometric Neural Operators (GNPs) introduced in [14]. GNPs represent operators $\mathcal{G}_\theta : \mathcal{A} \to \mathcal{U}$, where $\mathfrak{a} \in \mathcal{A}$, $\mathfrak{a} = (u(\cdot), \Phi(\cdot))$ with $u(\cdot)$ is a function and $\Phi(\cdot)$ denotes geometric information. Many choices can be used for Φ to provide a description of the

manifold shape $\mathcal{M}$. Here, we consider the collection of points within an embedding in $\mathbb{R}^n$ of $\mathcal{M}$. In practice, we use for $\mathfrak{a}$ a discretization a where $a \in \mathbb{R}^{d_a}$ and $a = \{u(\tilde{x}_i)\}_{i=1}^N, \{x_i\}_{i=1}^N$, where $u(\tilde{x}_i)$ denote function evaluations and x_i denote sample points in $\mathcal{M}$. Here we allow for the case where $\tilde{x}_i \neq x_i$.

We use GNPs that consist of the following three learnable components, (i) a lifting operator $\mathcal{P}$ for $a \in \mathbb{R}^{d_a}$ where $a = \{u(\tilde{x}_i)\}_{i=1}^N, \{x_i\}_{i=1}^N$ samples function evaluations to a higher dimensional set of feature functions $v_0 \in \mathbb{R}^{d_v}$ with $d_v \geq d_a$, (ii) a composition of layers consisting of integral operators $\mathcal{K}[v]$ and affine operators $\mathcal{W}[v](x) + b(x)$ that are processed by a non-linear activation $\sigma[\cdot]$ to obtain $v_{i+1} = \sigma\left(W v_i + \mathcal{K}[v_i] + b\right)$, and (iii) a projection operator $\mathcal{Q}$ to construct an $\mathbb{R}^{d_u}$-valued function w. The trainable sub-components also include within the operator layers the kernel k, bias function b, and local function operator W. We show the parts of the geometric neural operator in Figure 1. This gives a neural operator with T layers of the general form

$$\mathcal{G}_\theta^{(T)} = \mathcal{Q} \circ \sigma_T \left(W_T + \mathcal{K}_T + b_T\right) \circ \cdots \circ \sigma_0 \left(W_0 + \mathcal{K}_0 + b_0\right) \circ \mathcal{P}. \tag{1}$$

We collect all trainable parameters into θ. The activation of the last layer σ_T is typically taken to be the identity.

We consider linear operators $\mathcal{K}$ of the form

$$\mathcal{K}[v](x) = \int_D k(x, y) v(y)\ d\mu(y). \tag{2}$$

The μ is a measure on $D \subset \mathbb{R}^{d_v}$, $v : D \to \mathbb{R}^{d_v}$ is the input function, and k is a kernel $k(x, y) \in \mathbb{R}^{d_v} \times \mathbb{R}^{d_v}$. In practice, we approximate this integral on a truncated domain $B_r(\boldsymbol{x})$ using

$$\tilde{\mathcal{K}}[v_t](\boldsymbol{x}_j) = \frac{1}{N} \sum_{x_k \in B_r(\boldsymbol{x}_j)} k(x_j, x_k) v_t(x_k). \tag{3}$$

This discretization can be interpreted as message passing on a graph [55, 56]. For each layer t, we use a trainable kernel $k = k(x, y; \theta_t)$ parameterized by θ_t for fully connected neural networks having layer-widths (d_a, n, d_v^2) for $n \in \mathbb{N}$. We increase efficiency by using Nystrom approximation, summing over a subset of the points in $B_r(\boldsymbol{x}_j)$ instead of all points, using a maximum of 32 points in the sum. We also enforce a block-factorized structure in the outputs of the kernel network k, as in [14]. To obtain the output function representation, we apply average pooling to the outputs of $\mathcal{G}_\theta$ and pass this through two fully connected layers to obtain our final output functions. For additional discussions of GNPs and technical details, see [14].

We use GNPs to learn latent representations of the geometry from point cloud representations $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^M$ of a manifold $\mathcal{M}$. In general, we consider samplings where $\mathbf{x}_i = \hat{\mathbf{x}}_i + \zeta_i$ with $\hat{\mathbf{x}}_i \in \mathcal{M}$ and ζ_i is noise obscuring the geometry. For each point $\mathbf{x} \in \mathcal{X}$, we define the neighborhood $\mathcal{N}_\epsilon(\boldsymbol{x}) = \{\mathbf{x}_i \mid \|\mathbf{x}_i - \mathbf{x}\| < \epsilon\}$ as the set of points within distance ϵ of $\mathbf{x}$. To obtain good characteristic scales, we take $\epsilon = r_k \tau$, r_k to be the radius of the kth nearest neighbor of $\boldsymbol{x}$, and τ as a user-specified parameter. Typical values we use are $\tau = 1.1$ and $k \in \{30, 50\}$.

As part of our GNPs for point-cloud processing, we output for a neighborhood $\mathcal{N}_\epsilon$ around each point $\mathbf{x}$ a local coordinate system (ξ^1, ξ^2). We construct this by performing the following computations as part of our GNPs. We center the points using $\mathcal{N}_\epsilon - \bar{\mathbf{x}}$ where $\bar{\mathbf{x}}$ is the mean and we

perform Principal Component Analysis (PCA) [57–59]. This provides a local orthonormal basis $\{\boldsymbol{\psi}_j\}_{j=0}^3$ such that each $\boldsymbol{y} \in \mathcal{N}_\epsilon(\boldsymbol{x})$ can be expressed as

$$\boldsymbol{y} = \bar{\boldsymbol{x}} + \xi^1(\boldsymbol{y})\boldsymbol{\psi}_1 + \xi^2(\boldsymbol{y})\boldsymbol{\psi}_2 + \xi^3(\boldsymbol{y})\boldsymbol{\psi}_3. \tag{4}$$

The $\xi^j(\mathbf{y})$ provide a representation of $\mathbf{y}$ in terms of this basis. In the limit of increasingly dense samplings of $\mathcal{M}$, the vectors $\boldsymbol{\psi}_1$, $\boldsymbol{\psi}_2$ would approach tangent vectors of the manifold and the vector $\boldsymbol{\psi}_3$ approaches the normal vector. For finite samplings these provide approximations for obtaining a local coordinate system. We set these vectors to have unit norm, $\|\boldsymbol{\psi}_j\| = 1$. To ensure the correct orientation of the coordinate frame, we assume the manifold has a known orientation on $\mathcal{N}_\epsilon(\boldsymbol{x})$, and we setup the basis so $\boldsymbol{\psi}_1 \times \boldsymbol{\psi}_2 = \boldsymbol{\psi}_3$ is aligned with the outward normal at $\bar{\boldsymbol{x}}$. We remark these procedures and PCA also can be replaced by learning alternative operators during training to obtain other representations and coordinate systems.

We further canonicalize the point data for our training methods by using transformations that impose invariances to rotations and translations, and equivariances to scalings. The invariance to rotations and translations is accomplished by using the representation $\xi^j(\mathbf{y})$ obtained from the basis $\boldsymbol{\psi}_j$. In particular, we use the change of coordinates from $\mathbf{y} = (y_1, y_2, y_3)$ to to $\boldsymbol{\xi} = (\xi^1, \xi^2, \xi^3)$ over basis $\{\boldsymbol{\psi}_j\}$. We also perform further scalings of ξ^j to normalize the point data, which we discuss below. Our normalizations allow our learning methods to find common patterns and features spanning a broad range of similar geometries. Our approach provide ways to amplify the statistical power of the point-cloud training datasets for learning operations.

An important inductive bias we incorporate into our methods is to find a local Monge-Gauge parametrization of the surface patch $\mathcal{N}_\epsilon(\boldsymbol{x})$ [60, 61]. We use parameterizations of the form

$$\boldsymbol{\sigma}(\xi^1, \xi^2) = \bar{\boldsymbol{x}} + \xi^1\boldsymbol{\psi}_1 + \xi^2\boldsymbol{\psi}_2 + h(\xi^1, \xi^2)\boldsymbol{\psi}_3. \tag{5}$$

This requires finding a height function $h(\xi^1, \xi^2)$ that matches $\xi^3(\mathbf{y})$ for each sample point $\mathbf{y} \in \mathcal{N}_\epsilon$. We can then use $\boldsymbol{\sigma}$ as part of extracting local geometric quantities, contributions to the action of differential operators, and other features. We show an example in Figure 3. We give explicit expressions for common geometric quantities and operators that can be obtained from $\boldsymbol{\sigma}$ and h in Appendix A.

To further normalize the data, we perform additional scalings to obtain results more robust to manifolds having different local characteristic scales. While scales are associated with important geometric properties, we utilize that geometric quantities satisfy many types of equivariances allowing for further canonicalization in terms of more intrinsic features of the geometry. We use the following canonical rescaling of the coordinates $(\tilde{u}, \tilde{v}, \tilde{w}) = (\xi^1, \xi^2, \xi^3)$ to $(u, v, w) = (\epsilon\tilde{u}, \epsilon\tilde{v}, \delta\tilde{w})$. The ϵ and δ give characteristic scales that serve to standardize the representation of the local geometry. We take ϵ as the radius of the neighborhood $\mathcal{N}_\epsilon$. We obtain δ by $\delta = \max(2\lambda, \delta_0)$, where λ is the standard deviation in the $\boldsymbol{\psi}_3$ direction, and δ_0 is a user-specified parameter. This yields a rescaled Monge-Gauge parameterization of the form

$$\boldsymbol{\sigma}(u, v) = \bar{\boldsymbol{x}} + u\boldsymbol{\psi}_1 + v\boldsymbol{\psi}_2 + s(u, v)\boldsymbol{\psi}_3. \tag{6}$$

This gives the rescaled height $s(u, v) = \delta \cdot h(\xi^1, \xi^2)$. Our rescaling approach provides in the geometric setting methods similar to batch-normalization [62, 63] to provide a more uniform set of scales to enhance learning.

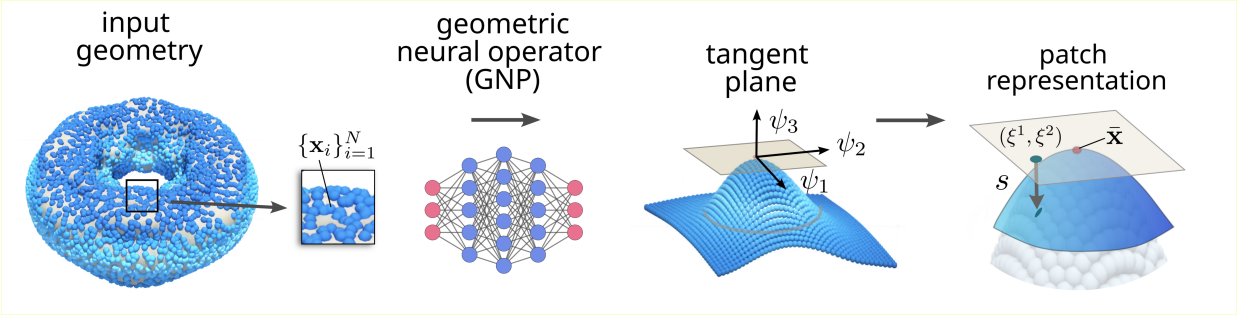

Figure 2: ***Learning Latent Geometric Representations.*** *We show how data is processed by our geometric neural operators (GNPs). We obtain latent representations at each* $\bar{\mathbf{x}}$ *by learning GNPs that map a collection of points* $\{\mathbf{x}_i\}_{i=1}^N$ *sampling the geometry to a local parameterization* (ξ^1, ξ^2) *and surface height function* $s(\xi^1, \xi^2)$*. This provides geometric information for further processing and tasks.*

An important issue that arises in practice is that we need to be cautious in extreme cases when $\mathcal{N}_\epsilon(\boldsymbol{x})$ is nearly flat. In this case, λ is nearly zero so $s(u, v)$ is prone to fit noise in the data or numerical round-off errors. To mitigate this issue, we introduce the user-specified parameter δ_0 and set it in practice to be 0.005 as a default value.

To further filter noise during processing, we limit the capacity of the height function in equation 6. We use in the final projection operations the functional form

$$s(u, v) = \sum_k s_k \phi_k(u, v). \tag{7}$$

The choice of $\Phi = \{\phi_1, \ldots, \phi_N\}$ allows for different ways to locally filter the geometry. We use a set of basis functions that are tensor products of the form $\Phi = \{l_i(u) l_j(v)\}_{i,j=0}^d$, where l_i give the basis set of functions. We use for this purpose the orthogonal Legendre Polynomials [64, 65]. By taking Taylor expansions for smooth surfaces, we see any shape can be well approximated locally with this choice by taking sufficiently small neighborhoods $\mathcal{N}_\epsilon(\boldsymbol{x})$. In practice, we use $d = 3$ for approximating $s(u, v)$ which serves to help further filter out noise and over-fitting. We use this as part of the GNP mappings from the input point-cloud data $\{\mathbf{x}_i\}_{i=1}^N$ to obtain at each $\bar{\mathbf{x}}$ the local latent representations $s(u, v)$ for the geometry. This allows for the GNP training methods to obtain latent geometric representations that can handle data containing noise and other artifacts. We show the latent representations in Figure 2.

2. Training Transferable GNP Models for Geometric Tasks

We now show how GNPs can be trained to obtain transferable models for geometric tasks. We start by training GNPs by leveraging geometric information readily available from the parameterizations of radial manifolds computed using spectral numerical methods and spherical harmonics [16]. We show an example radial manifold in Figure 3. We then show how the trained GNPs can then be used for other more general geometries and topologies.

For training, we consider radial manifolds which consist of shapes having the star property that each point on the surface can be connected by a line segment to the origin without self-intersections.

Radial manifolds can be parameterized by two coordinate charts in spherical coordinates in the form [16, 66]

$$\boldsymbol{x}(\theta,\phi) = \boldsymbol{\sigma}(\theta,\phi) = r(\theta,\phi)\boldsymbol{r}(\theta,\phi). \tag{8}$$

The $\boldsymbol{r}(\theta,\phi)$ is the unit vector from the origin to the point on the sphere with spherical coordinates (θ,ϕ). The $r(\theta,\phi)$ is a positive scalar function giving the distance from the origin. For analysis and numerical computations, we use spherical harmonics expansions of

$$r(\theta,\phi) = \sum_{l=0}^{L}\sum_{m=-l}^{l} a_l^m Y_l^m(\theta,\phi), \tag{9}$$

where Y_l^m are the spherical harmonic of index m and order l. We compute expressions numerically using the package sphericart [67] for the spherical harmonics and our approaches in [16, 66].

We generate shapes for our training datasets of varying complexity by adjusting L in our spherical harmonics expansions for the radial functions. We obtain for $r(\theta,\phi)$ the complex coefficients $a_l^m = \alpha_l^m + i\beta_l^m$ for $m = 0,\ldots,l$, where the real α_l^m and imaginary β_l^m, by sampling the components from normal distributions with mean 0 and standard deviation $\frac{1}{l}$. To ensure real-valued functions we generate coefficients for a_l^m for $m < 0$ and set the other coefficients using conjugacy conditions. To vary the shapes of the geometry, we also select different values of L ranging over $L \in \{3,6,8,10,12,15,18,22\}$. As another convention in generating our training datasets, we also translate and scale $r(\theta,\phi)$ so that it has mean 1 and satisfies $r(\theta,\phi) \in [0.7, 1.3]$ for all (θ,ϕ).

Our training dataset consists of 320 manifold shapes having varying levels of complexity. We also hold out 40 manifold shapes for testing. For all shapes, once a parameterization is chosen, we generate a mesh and a quasi-uniform sampling consisting of $100,000$ points on the surface using the PyACVD package from PyVista [68, 69]. The generated meshes are used only for sampling purposes and visualizations. For the supervised training tasks for extracting geometric quantities, we also compute for these manifolds as a reference the unit normals, the first and second fundamental forms, and Gaussian curvatures.

We choose the neighborhoods $\mathcal{N}_\epsilon(\boldsymbol{x})$ by using randomly selected center points $\boldsymbol{x}$ and querying all points within the ϵ-neighborhood. As discussed above, we use a user specified number of nearest neighbors k which is computed in practice using a KD-tree. To ensure accurate approximation of the geometry near the boundary of $\mathcal{N}_\epsilon(\boldsymbol{x})$, we include all points within radius 1.5ϵ of $\boldsymbol{x}$. This is used as initial input to the model $\mathcal{G}_\theta$, targeting predictions for output on all points within the neighborhood $\{(u_i,v_i,w_i)\}_{i=1}^N \in \mathcal{N}_\epsilon(\boldsymbol{x})$.

When making comparisons of accuracy between the GNP predictions $\mathcal{G}_\theta$ and the ground truth geometric quantities, we use the local coordinate system provided by the GNPs. In particular, for the first two fundamental forms $\{\mathbf{I}\}_{i=1}^N, \{\mathbf{II}\}_{i=1}^N$ we convert the coordinate-dependent values based on spherical harmonics to the corresponding coordinate-dependent values in the local frame given by $\{\boldsymbol{\psi}_1,\boldsymbol{\psi}_2,\boldsymbol{\psi}_3\}$. For some of our studies, we also use in the loss function the matrix inverse of the first fundamental form $\mathbf{I}_i^{-1} = \{g^{ij}\}$, where the inverse of the metric tensor g is denoted by g^{ij}. The inverse metric plays an important in the approximation of differential operators, such as the Laplace-Beltrami operator. This provides a set of geometric quantities for evaluating the accuracy of the GNP predictions important in many geometric procedures and operators on manifolds.

When training the GNPs, we use the following loss function

$$\begin{aligned}\mathcal{L}(\mathcal{N}_\epsilon(\boldsymbol{x});\theta) = \mathcal{L}_{\text{rel}}\left(\{\hat{s}(u_i,v_i)\},\{w_i\}\right) + \lambda_1\mathcal{L}_{\boldsymbol{\eta}}\left(\{\hat{\boldsymbol{\eta}}_i\},\{\boldsymbol{\eta}_i\}\right) + \lambda_2\mathcal{L}_{\text{rel}}\left(\left\{\hat{\mathbf{I}}_i^{-1}\right\},\left\{\mathbf{I}_i^{-1}\right\}\right) \\ + \lambda_3\mathcal{L}_{\text{rel}}\left(\left\{\hat{\mathbf{II}}_i\right\},\{\mathbf{II}_i\}\right) + \lambda_4\mathcal{L}_{\text{rel}}\left(\left\{\hat{K}_i\right\},\{K_i\}\right).\end{aligned} \tag{10}$$

The relative loss terms $\mathcal{L}_{\text{rel}}$ are defined as

$$\mathcal{L}_{\text{rel}}(f,g) = \frac{\|f-g\|_2}{\|f\|_2}. \tag{11}$$

For the normals $\boldsymbol{\eta}$, we use

$$\mathcal{L}_{\boldsymbol{\eta}}\left(\{\hat{\boldsymbol{\eta}}_i\},\{\boldsymbol{\eta}_i\}\right) = \frac{1}{N}\sum_{i=1}^{N}\left(1-\hat{\boldsymbol{\eta}}_i\cdot\boldsymbol{\eta}_i\right). \tag{12}$$

The $\hat{s}(u,v) = \mathcal{G}_\theta(\mathcal{N}_\epsilon(\boldsymbol{x}))$ denotes the function constructed by the GNPs for parameters θ. The λ_k provide parameters for adjusting the relative strength of the different contributions to the loss. Typical values we use in our training are $\lambda_n = 0.5$ for $n = 1,2,3,4$. When training with a batch size greater than 1, we compute the mean of this loss over each of the neighborhoods.

In our training protocols, we consider both the case of no noise and two cases with different types of noise. This includes (i) uniform perturbations of points obscuring the geometry, and (ii) large outlier points corrupting the geometric sampling. In the first case, we consider Gaussian noise that is applied to every point with a fixed standard deviation. In the second case, we consider outliers that are generated by applying to 10% of the points a large Gaussian noise. This allows for GNPs to be trained to be robust to the uniform noise and the outlier noise. This is done by requiring that the training be accurate for all neighborhood points $\mathbf{x}$ that are not outliers. This serves to signal the GNPs to learn to ignore the misleading outlier points. In the uniform noisy cases, we require the GNPs to give accurate results for across all neighborhood points when evaluating the loss. In this way, we can introduce deliberately into the training dataset a set of artifacts that obscure the underlying geometry and for which the GNPs need to compensate to obtain robustness results. Other types of noise and artifacts also can readily be incorporated into our training protocols for the GNPs. Our approach provides ways to learn data-driven filtering and compensations for noise and other artifacts in the point-clouds.

3. Results

We learn GNPs using our training datasets based on radial manifold shapes from our spherical harmonics methods discussed in Section 1.1. We show how our pre-trained GNPs are transferable to handle new geometries, topologies, and tasks beyond the training datasets. We demonstrate how the GNP methods can be used for (i) estimating metrics and curvatures of surfaces, (ii) deforming shapes driven by mean-curvature flows, and (iii) developing numerical methods for solving geometric PDEs.

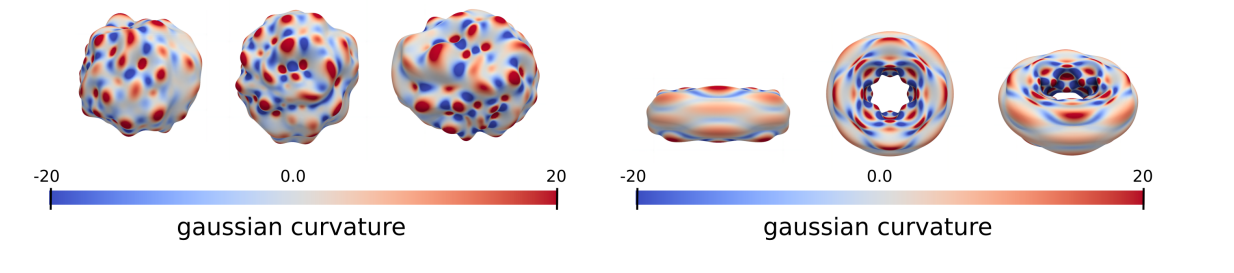

Figure 3: ***GNP Estimators for Gaussian Curvatures on Test Shapes.*** *Gaussian curvatures of radial manifold (left) and toroidal manifold (right). The views show the same manifold from a few different vantage points.*

3.1. Accuracy, Robustness, and Transferability of GNP Estimators for Geometric Quantities

We show how the pre-trained GNPs can be used beyond the radial manifold shapes in the training datasets by considering a class of toroidal manifolds. We consider toroidal shapes diffeomorphic to a torus with parameterizations of the form

$$\boldsymbol{\sigma}(u,v) = ((a(u,v)\cos(v) + b(u,v))\cos(u), (a(u,v)\cos(v) + b(u,v))\sin(u), a(u,v)\sin(v)), \quad (13)$$

where $u, v \in [0, 2\pi)$. The standard torus is obtained when a, b are constant. More generally, we obtain shapes by using functions $a(u,v),\ b(u,v)$ of the form

$$a(u,v) = a_0 + r_0 \sin(A_0 u)\cos(B_0 v), \quad b(u,v) = b_0 + r_1 \sin(A_1 u)\cos(B_1 v). \quad (14)$$

We sample shapes using random variates with $r_0 \sim \mathcal{U}(0.05a_0, 0.2a_0)$, $r_1 \sim \mathcal{U}(0, 0.08b_0)$, where $a_0 = \frac{1}{3}, b_0 = \frac{2}{3}$. We also randomly select $A_i \in \{1, 2, \ldots, 5\}$ and $B_i \in \{3, 4, \ldots, 7\}$. The $\mathcal{U}[c_1, c_2]$ denotes the uniform distribution on the interval $[c_1, c_2]$. For testing the predictions of the trained GNP models, we use that these shapes have explicit parameterizations in a single coordinate chart. This allows for comparisons between the pre-trained GNPs and the geometric quantities when computed numerically using the parameterization in equation 13. We give the expressions needed for numerical calculations of the geometric quantities in Appendix A.

As a further study of the pre-trained GNP methods, we also make comparisons with alternative numerical methods based on Generalized Moving Least Squares (GMLS) [31, 70]. We showed in prior work that GMLS can used for both geometric estimation and surface function approximation on scattered data by solving a collection of local least squares problems [31]. In these GMLS methods, at each point $\boldsymbol{x}_i$ in the point cloud we consider a neighborhood $\mathcal{N}_\epsilon(\boldsymbol{x}_i) = \{\boldsymbol{x}_j\}_{j=1}^n$ and solve the least-squares problem

$$q^*(u,v) = \arg\min_{q \in \Phi} \sum_{j=1}^{n} \left(s(u_j, v_j) - q(u_j, v_j)\right)^2 w\left(\|(u_j, v_j)\|_2\right). \quad (15)$$

Similar to the GNP methods, the GMLS estimator for geometric quantities at $\boldsymbol{x_i}$ are computed by using for $q(u_j, v_j)$ the local coordinates $(u_j, v_j, s(u_j, v_j))$ as in equation 6 and from $q^*(u,v)$ using

model	k	normal	metric inverse	shape	gaussian	mean
radial manifolds						
gnp, no-noise	30	2.04e-06	6.57e-04	1.89e-02	4.07e-02	1.93e-02
gmls, no-noise	30	3.11e-08	5.95e-05	4.10e-03	1.92e-02	7.15e-03
gnp, no-noise	50	4.55e-06	7.99e-04	1.70e-02	4.56e-02	2.24e-02
gmls, no-noise	50	1.89e-07	1.16e-04	7.06e-03	3.17e-02	1.21e-02
gnp, outliers, σ =5e-03,	50	3.70e-05	1.97e-03	2.75e-02	6.26e-02	3.14e-02
gmls, outliers, σ =5e-03,	50	3.39e-04	7.34e-03	3.72e-01	1.01e+00	4.11e-01
gnp, noise, σ =1e-03,	70	6.10e-05	5.62e-03	1.32e-01	1.75e-01	1.03e-01
gmls, noise, σ =1e-03,	70	6.11e-04	1.79e-02	1.17e+00	2.19e+00	7.00e-01
toroidal manifolds						
gnp, no-noise	30	6.19e-07	3.79e-05	2.30e-02	4.03e-02	2.50e-02
gmls, no-noise	30	3.97e-08	1.27e-06	5.90e-03	2.10e-02	8.34e-03
gnp, no-noise	50	1.43e-06	6.07e-05	2.48e-02	4.38e-02	3.34e-02
gmls, no-noise	50	3.97e-10	3.61e-03	1.03e-02	2.55e-02	1.49e-02
gnp, outliers, σ =5e-03	50	4.16e-05	2.11e-04	3.71e-02	6.31e-02	4.45e-02
gmls, outliers, σ =5e-03	50	2.43e-04	3.98e-04	3.58e-01	1.41e+00	3.94e-01
gnp, noise, σ =1e-03	70	7.80e-05	6.25e-04	1.54e-01	2.16e-01	1.19e-01
gmls, noise, σ =1e-03	70	8.74e-05	1.25e-03	1.25e+00	2.24e+00	6.93e-01

Table 1: ***Accuracy and Robustness of GNPs to Noise and Outliers.*** *We show results for the radial and toroidal manifolds for predictions of our trained GNPs compared with GMLS estimators. These are compared when varying the number of nearest neighbors k for determining* $\mathcal{N}_\epsilon(\boldsymbol{x})$ *and in the presence of noise perturbations or outliers. The Gaussian noise in these cases has standard deviation denoted by* σ*. We show example point-cloud data with noise in Figure 4.*

the expressions in Appendix A. The approximation space Φ is taken to be polynomials up to degree $n = 3$. For the weight function w, we use $w(r) = (1-r)_+^4$ where $(z)_+ = \max(z, 0)$. For more details on GMLS, see [31].

While GNPs and GMLS use some similar geometric representations, there are a few key differences between the approaches. The GNP methods utilize a data-driven neural operator approach to avoid needing to solve least-squares problems at each evaluation site $\mathbf{x}_i$. The GNP methods also allow for more general loss functions which can through training target local and more global geometric features that impact the latent geometric representations beyond only targeting local least-squares reconstructions of shape or other locally known functionals at the time of estimation. In addition, the GNP methods can be trained to learn latent information for filtering out noise and outliers. Data-driven filtering of empirical artifacts and other idiosyncratic behaviors of particular types of measurements or datasets can be readily incorporated into our GNP training through empirical examples, data augmentations, and other protocols. As we discuss in our examples, this provides unified strategies for utilizing GNPs to further impact the latent geometric representations to capture relevant information for different types of geometric tasks involving estimators and operators for point-clouds and manifolds.

To validate the methods for important geometric quantities associated with metric and curvature, we perform studies comparing the pre-trained GNPs, GMLS, and analytic results. We consider

both radial and toroidal manifolds to show performance across different topologies. These test cases also demonstrate how the methods perform on in-distribution shapes and out-of-distribution shapes relative to the training dataset based on radial manifolds. We consider 40 manifolds for each case of manifold. We characterize the methods using the same relative loss terms as used in training when evaluating the performance of the methods on the test data. The loss terms are computed over neighborhoods of the manifolds for each point $\mathbf{x}_i$ and averaged across the 40 manifolds. In our comparisons, we compute the accuracy of the (i) shape reconstruction, (ii) normals, (iii) metric inverse, (iv) gaussian curvature, and (v) mean curvature. We emphasize that the mean curvature H was not included in the training loss, which provides an indication of the accuracy of the composite Weingarten map $\boldsymbol{W} = \mathbf{I}^{-1}\mathbf{II}$, since $H = \text{trace}(\boldsymbol{W})$. We report our results for the methods with and without noise in Table 1.

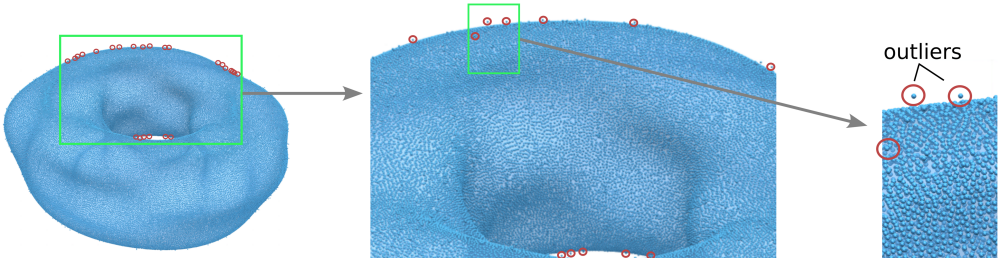

Figure 4: ***Point-Cloud Data with Artifacts.*** *The underlying geometry can be obscured by noise when working with point-cloud representations. We focus on the case of non-uniform samplings and outlier artifacts. For an example shape sampled with approximately* 10% *outliers (left), we highlight a subset of the outliers in the data by circles (middle). The shape also exhibits non-uniform sampling as can be seen in the most magnified view (right). These and other artifacts can be introduced into our datasets for training to enhance the robustness of the GNP methods.*

We find for the no-noise and noisy cases that the GNPs are capable of learning accurate estimators for the geometric quantities. In the no-noise cases with $k = 30$ and $k = 50$, we find while both methods give accurate results for both the radial and toroidal manifolds, the GMLS estimators give more accurate results. This is somewhat expected given that GMLS solves a local least-squares objective at each surface location $\mathbf{x}_i$ specialized for targeting local shape reconstruction. Both the GNPs and GMLS methods provide accurate estimators with errors less than 5%.

In the case of uniform noise and outliers, we find the GNPs perform significantly better than the GMLS methods. In the outlier case, we use $k = 50$ which provides additional samples for estimating the geometry. We find that GNPs are still able to reconstruct the radial shapes with an accuracy of 2.75% while GMLS only to an accuracy of 37%. Similarly, GNPs are able to obtain accurate estimates despite outliers for gaussian curvature with error 6.25% and for mean curvature with error 3.14%. The GMLS estimates have gaussian curvatures with error 101% and mean curvatures with error 41.1%, see Table 1. The GNPs also outperform GMLS for the toroidal manifolds with outliers. The GNPs have errors in reconstructing the shape 3.71%, gaussian curvature 6.31%, mean curvature 4.45%. The GMLS methods have errors in reconstructing the shape 35.8%, gaussian curvature 140%, mean curvature 39.4%. In the case of uniform Gaussian noise, we also find that GNPs are more robust than the GMLS methods for both the radial and toroidal manifolds. The GNPs have errors respectively for the radial and toroidal manifolds for the shape reconstruction

13.2%, 15.4%, gaussian curvature 17.5%, 21.6% and mean curvature 10.3%, 11.9%. The GMLS methods have errors for shape reconstruction 117%, 125% and inaccurate estimates for the gaussian and mean curvatures.

The results show the utility in GNPs of being able to use data-driven methods to train the operators for mapping point-cloud samples $\{\mathbf{x}_i\}$ to latent geometric features. The GNP methods also can accommodate through training other types of noise and artifacts arising in practice for point-clouds without the need to design auxiliary regularization terms. In GNPs can be trained using empirical artifacts by incorporating them directly into the training datasets, performing data augmentations, or using other protocols. In this way, the GNPs can be used to learn robust estimators for geometric quantities of point-cloud representations obtained in practice. The results also show the transferability of the GNP estimators which were trained on a dataset of radial manifolds and were found to also provide accurate and robust results for the out-of-distribution toroidal shapes. This allows for GNPs to be pretrained and transferred for use in other data processing pipelines and computational methods.

3.2. GNPs for Shape Deformations based on Mean-Curvature Flows

As a further demonstration of how our pre-trained GNP methods can be transferred for use in other tasks, we perform computations of shape deformations based on Mean-Curvature Flows (MCF). We show how the GNPs can be used as part of the numerical methods for these flows. We consider deformations of an initial smooth manifold $\mathcal{M}$ that has an immersion given by the map $\varphi_0 : \mathcal{M} \to \mathbb{R}^3$. The mean curvature flow (MCF) of $\mathcal{M}$ is a family of smooth immersions $\varphi_t : \mathcal{M} \to \mathbb{R}^3$ for $t \in [0, T)$, where $\varphi(p, t) = \varphi_t(p)$ is the solution of the PDE

$$\begin{cases} \frac{\partial}{\partial t}\varphi(p,t) = H(p,t)\boldsymbol{\eta}(p,t), & p \in \mathcal{M}, t \in [0,T), \\ \varphi(p,0) = \varphi_0(p), & p \in \mathcal{M}. \end{cases} \tag{16}$$

The mean curvature at $p \in \mathcal{M}$ is denoted by $H(p, t)$ and the outward unit normal by $\boldsymbol{\eta}(p, t)$. This flow deforms the surface by moving points p in the direction of the normal vector $\boldsymbol{\eta}$ at a rate proportional to the mean curvature H.

We numerically approximate the PDE in equation 16 by discretizing time into steps t_k with $t_k = k\Delta t + t_0$ with time-step Δt and treat the manifold geometry through a finite point-cloud sampling $\{\mathbf{x}_i\}_{i=1}^N$. At each time, we deform the surface by moving each sample point $\boldsymbol{x}_i^n$ in the normal direction in proportion to the local mean curvature using

$$\boldsymbol{x}_i^{n+1} = \boldsymbol{x}_i^n + \Delta t H(\boldsymbol{x}_i^n)\boldsymbol{\eta}(\boldsymbol{x}_i^n). \tag{17}$$

This requires estimation from the point-cloud of the mean curvature $H(\boldsymbol{x}_i^n)$ and normals $\boldsymbol{\eta}(\boldsymbol{x}_i^n)$. For this purpose, we use our pre-trained GNPs to obtain estimates of H and $\boldsymbol{\eta}$ for each point and time-step.

To help ensure stability of the numerical methods, we apply a smoothing step on the values of H to obtain

$$\tilde{H}(\boldsymbol{x}_i^n) = C_i^n \sum_{\boldsymbol{x}_j^n \in B(\boldsymbol{x}_i^n; 3r_0)} w(\left\|\boldsymbol{x}_j^n - \boldsymbol{x}_i^n\right\|_2) H(\boldsymbol{x}_j^n), \qquad w(r) = \exp\left(-r^2/(2r_0^2)\right). \tag{18}$$

The C_i^n are chosen to normalize the sum so that the weights $C_i^n w\left(\left\|\boldsymbol{x}_j^n - \boldsymbol{x}_i^n\right\|_2\right)$ sums to 1. In practice, we set $r_0 = 0.00667$. As time evolves, areas of negative mean curvature will shrink and point samplings will become more dense. Similarly, regions also can become more rarefied. To help maintain uniformity of the points and avoid round-off errors, we sub-sampled in areas of high density to eliminate points when they get too close together by only keeping one representative of the cluster. After each time step, we sub-sample as needed to ensure points are always at least distance r_1 from it's nearest neighbor, and we set $r_1 = 0.005$. In our examples, we did not need to do anything to mitigate rarefaction of the points, which could in principle be handled by interpolation and resampling using our local GNP surface reconstructions. Our GNP estimators have some build-in robustness to density variations since they are already handled to some degree by our criteria for selecting neighborhood patch sizes based on the k nearest neighbors criteria. In this way we are able to use the GNPs as part of the numerical methods for deforming the surface by mean curvature flow.

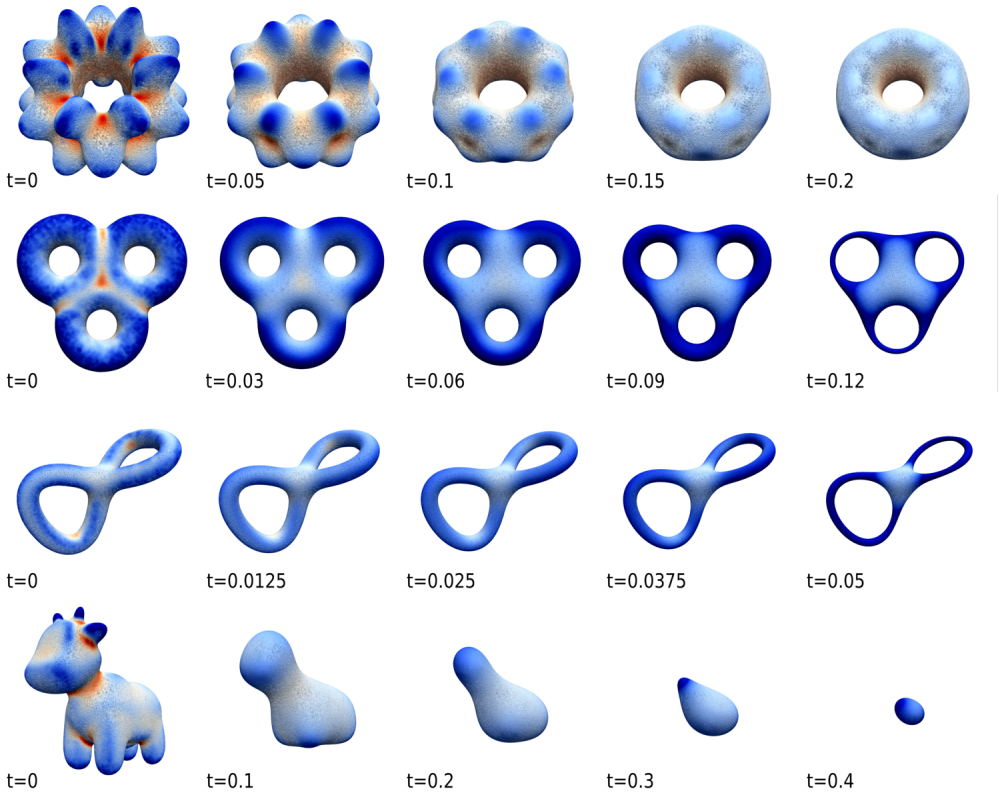

Figure 5: ***Mean-Curvature Driven Flow.*** *We show shape deformations evolved under mean-curvature flow (MCF) in equation 16 using our numerical methods based on pretrained GNPs.*

To test the performance of the numerical methods for MCF based on the pretrained GNPs, we considered four different test shapes having non-trivial geometries and topologies. These consisted of shapes we refer to as the (i) bumpy torus, (ii) fidget spinner (genus three shape), (iii) genus two shape, and (iv) cow figurine. The cow and genus shapes serve as common benchmark shapes used in the graphics community [71, 72]. For MCF, it is known that the genus one and larger shapes

can develop singularities. For each of our shapes, we show results of the mean-curvature flows up to the time of approach of a singularity or when N_* points get closer together than a threshold δ_*. Another challenge for validating MCF is the lack of analytic solutions for the shape deformations for non-trivial geometries. We present qualitative results for our GNP methods for MCF, and then present quantitative results for numerical solvers for PDEs on manifolds in the next section.

We present our qualitative results for the MCF deformations for each of the shapes using our GNP-based numerical methods, see Figure 5. In these results we used our pretrained GNPs with no-noise and $k = 50$ nearest neighbors for estimating $H, \boldsymbol{\eta}$ at each $\boldsymbol{x}_i^n$. We use a time-step of $\Delta t = 10^{-4}$ and evolved each shape for at least 500 iterations. We find for each of the shapes the results are consistent with how we would expect mean-curvature flows to behave and do not exhibit erratic estimates or deformations. The GNP-based numerical methods consistently provide reliable time-steps as indicated by the uniform and symmetric evolution of the bumpy torus shape toward the expected target torus. Similarly, the genus two and three shapes show consistent evolution toward the singular shapes expected from the mean curvature flows. The cow figurine consists of geometric features across several spatial scales and is found to evolve toward the correct spherical target shape expected under mean-curvature flow. The results show qualitatively how the GNPs can serve as reliable components within other computational and numerical methods requiring geometric information. To demonstrate more quantitative results, we consider next the development of numerical solvers for geometric PDEs on manifolds.

3.3. Using Transferable Pretrained GNPs for Developing Numerical Solvers for Geometric PDEs

We demonstrate the use of our pre-trained GNPs to develop numerical methods for solving PDEs on manifolds. We consider the Laplace-Beltrami (LB) equation

$$\begin{cases} \Delta_{\text{LB}}\ u(\mathbf{x}) & = -f(\mathbf{x}), \\ \int_{\mathcal{M}} u(\boldsymbol{x})\, d\boldsymbol{x} & = 0, \end{cases} \tag{19}$$

where

$$\Delta_{LB} = \frac{1}{\sqrt{|g|}} \partial_i \left(g^{ij} \sqrt{|g|} \partial_j \right). \tag{20}$$

The Δ_{LB} generalizes the Laplacian to scalar functions on surfaces $\mathcal{M}$. The ∂_i denotes the derivative in the i^{th} coordinate direction. The $|g|$ denotes the determinant of the metric tensor and g^{ij} denotes the terms of the inverse metric. The integral condition serves to provide a unique solution to the PDE for closed surfaces. A challenge that arises in development of effective numerical methods is the ability to estimate from the manifold the geometric quantities g, g^{ij}, and derivatives. For this purpose, we develop collocation numerical methods [73–75] for approximating solutions u of the PDE that leverage geometric estimators obtained from the pretrained GNPs.

In collocation numerical methods, approximate solutions $\tilde{u}$ to PDEs are obtained by requiring the target differential relations $\mathcal{L}\tilde{u}(x_i) = -f(x_i)$ hold approximately at a collection of evaluation points x_i. The function $\tilde{u}$ is represented by degrees of freedom α_k yielding $\tilde{u} = \mathcal{U}[\boldsymbol{\alpha}]$, where $\boldsymbol{\alpha}$ is the vector with $[\boldsymbol{\alpha}]_k = \alpha_k$. Almost any form of interpolation or extension $\mathcal{U}[\boldsymbol{\alpha}]$ for using the data to obtain a function $\tilde{u}$ can be utilized as long as this provides increasingly accurate estimates of the action of the differential operator $\mathcal{L}u$ as the density of collocation points x_i increases [73–75]. Central

to collocation methods is the ability to capture the local differential relations as the number of α_k and density of points x_i increases [74]. For this purpose, we use for $\mathcal{U}$ an approximate interpolation operation with $\alpha_k = u_k \approx u(\tilde{x}_k)$, so $\tilde{u} = \mathcal{U}[\mathbf{u}]$ with $[\mathbf{u}]_k = u_k$. The collection of points $\{\tilde{x}_k\}$ need not be the same as the collection $\{x_i\}$.

For the Laplace-Beltrami PDE, we let $\mathcal{L} = \Delta_{LB}$ and consider the differential relations

$$\Delta_{\mathrm{LB}}\tilde{u}(\mathbf{x}_i) = -f(\mathbf{x}_i) + r_i, \quad \text{with} \quad r_i[\mathbf{u}] = \Delta_{\mathrm{LB}}\tilde{u}(\mathbf{x}_i) - (-f(\mathbf{x}_i))\,. \tag{21}$$

The residuals are denoted by r_i which we will aim to make as small as possible. We use this to approximate the solution of the PDE in equation 19 by $\tilde{u}^*(x) = \mathcal{U}[\mathbf{u}^*](x)$ where $[\mathbf{u}^*]_k = u_k^*$ is the solution of the following least-squares problem

$$\mathbf{u}^* = \arg\min_{\mathbf{u}} \sum_{i=1}^{N} \left(r_i[\mathbf{u}]\right)^2. \tag{22}$$

For the operator producing $\tilde{u} = \mathcal{U}[\mathbf{u}]$, we use GMLS to fit locally polynomials by solving at each $\mathbf{x}_i$ the least-squares problem

$$p_i^*(\cdot) = \arg\min_{p\in\mathbb{P}} \sum_{k=1}^{n} \left(p(\boldsymbol{y}_k) - u_k\right)^2 w\left(\|\boldsymbol{y}_k - \boldsymbol{x}_i\|_2\right), \tag{23}$$

where w is given by $w(r) = (1 - \frac{r}{\epsilon})_+^4$ and $\mathbb{P}$ is a space of Legendre Polynomials [64, 65] of degree $d = 3$. This gives at each $\mathbf{x}_i$ a polynomial $p_i(\cdot)$ and we can evaluate the function as $\tilde{u}(\mathbf{x}_i) = p_i(\mathbf{x}_i)$.

We also use the local polynomial representation to evaluate the action of differential operators $\mathcal{L}$. We compute $\mathcal{L}v = \partial_j v$ for a function by letting $v_k = v(\mathbf{x}_k)$ and approximating $\partial_j v(\mathbf{x}_i)$ by $\partial_j p_i(\mathbf{x}_i)$, where p_i solves equation 23 for $u_k = v_k$. We can compose derivative operations by repeating these approximations successively using the output of the previous operations for the next sampled input function v.

For the Laplace-Beltrami $\mathcal{L} = \Delta_{LB}$ operator, further information is required beyond the derivatives ∂_j since there are also contributions from the geometry, see equation 20. For the geometric contributions to the differential operator, we use our pretrained GNPs to obtain g, g^{ij}, and other geometric terms. We evaluate the action of the differential operator $\Delta_{LB}\tilde{u}(\mathbf{x}_i)$ by composing the evaluations of ∂_j with these geometric terms from the pretrained GNPs. This provides for any choice of u_k an approximation to the action of Δ_{LB} for computing the residuals in the collocation method in equation 21 and for solving the minimization problem in equation 22.

To develop practical numerical methods for our collocation approach, we use that the operations for evaluation of Δ_{LB} are linear in $\mathbf{u}$. This allows us to collect terms together to express the problem as seeking a solution to the linear system

$$A\mathbf{u} = \mathbf{f}, \tag{24}$$

where $[\mathbf{u}]_k = u_k \approx u(x_k)$, $[\mathbf{f}] = f_k \approx f(x_k)$, and $[A\mathbf{u}]_k \approx \Delta_{LB}\tilde{u}(\mathbf{x}_k)$. The stiffness matrix A is obtained by composing the normal equations for the GMLS least-squares problem in equation 23 and the geometric contributions to Δ_{LB} from the pretrained GNPs in equation 20. Since the linear system in equation 24 will typically be over-determined in our collocation methods, we solve for the

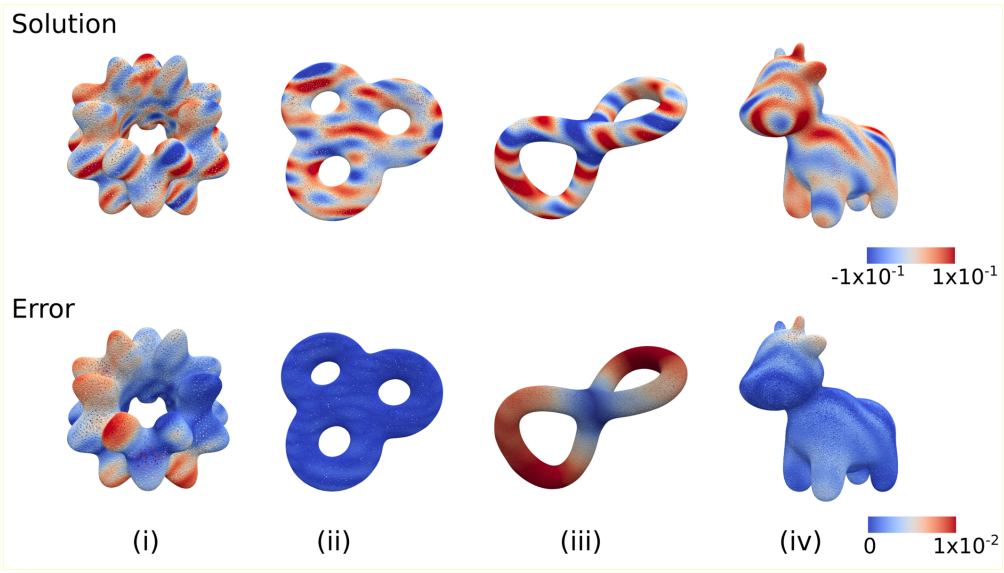

Figure 6: ***Geometric PDE Solvers based on GNPs.*** *We show solutions of the Laplace-Beltrami PDE for numerical methods based on our pre-trained GNPs.*

$\mathbf{u}$ that minimizes the residual $\|\mathbf{r}\|^2 = \|A\mathbf{u} - \mathbf{f}\|^2$. This is obtained by solving for $\mathbf{u}$ in the normal equations

$$A^T A\mathbf{u} = A^T\mathbf{f}. \tag{25}$$

To numerically approximate the solution of these equations, we develop iterative methods based on Scipy's LGMRES solver [76]. To improve the convergence rate of our iterative methods, we design algebraic multigrid preconditioners using the package PyAMG [77].

For point-cloud methods, PDE solvers for many applications need to be able to deal with the presence of noise perturbations, outliers, or other artifacts. We show how our GNP approaches can be used to develop filters for PDE solvers to deal with noise. We found that the collocation numerical methods perform quite poorly in the presence of outliers. This is a consequence of points moving not only in the normal direction, but also in the tangential directions of the manifold resulting in solution distortions. This causes significant errors in approximating the function values on the manifold shape especially at the outlier points.

We introduce the following GNP filter method for identifying and processing outlier points to improve the accuracy of the collocation methods. We perform an initial local surface reconstruction using our GNP methods on all points, since they have been trained to ignore outliers. If we find for the local GNP predicted shape reconstructions that a point $\mathbf{x}_i$ with coordinate (u_i, v_i, w_i) has a deviation value $|w_i - s(u_i, v_i)| > \alpha$ above the threshold α, we remove this point from the neighborhood during further processing. We use $\alpha = 0.1$ in practice. We then use our methods to approximate u and the remaining geometric quantities using the remaining points after the GNP filtering.

For our numerical methods, we use in our studies point-clouds with $100,000$ sample points for each shape. When imposing the differential relations in equation 21, we evaluate the operator $\Delta_{LB}\tilde{u}(\mathbf{x}_i)$ and the right-hand-side (RHS) $f(\mathbf{x}_i)$ at all $n = 100,000$ sample points $\{\mathbf{x}_i\}_{i=1}^n$. To ensure

uniqueness and robustness of the residual least-squares problem, we use a subset $N = 0.9n = 90,000$ of the sample points for determining $\tilde{u}$ with the degrees of freedom u_k associated with each of the points in $\{\mathbf{x}_k\}_{i=1}^N$. We use RHS functions $f(x)$ for testing the solvers by generating Fourier Series for $x \in \mathbb{R}^3$ with $M = 10$ modes in each direction with coefficients drawn from a normal distribution and spatial period $[-1, 1]$. We then restrict these functions to the manifold shape. We also normalize $f(x)$ so it has values in the range $[-20, 20]$. In each case, we test the solver in equation 19 with 10 samples of these RHS terms $f(x)$.

We study how our numerical methods based on the pretrained GNPs perform for Laplace-Beltrami PDEs on geometries with and without noise. We consider the four shapes discussed in Section 3.2. We refer to these shapes as the (i) bumpy torus, (ii) genus three shape, (iii) genus two shape, and (iv) cow figurine, see Figure 5. We compare our results for the GNP numerical methods with those that use as an alternative the GMLS estimators for the geometric contributions, as discussed in Section 3.1. When performing the studies with noise, we employ our GNP filtering methods discussed above for the collocation methods in both the GMLS and GNP cases. If we did not perform filtering of outliers for the collocation PDE solver, GMLS has large errors and does not produce viable outputs. We show the results of our studies in Table 2.

model	**k**	**bumpy torus**	**genus-3**	**genus-2**	**cow**
gnp, no-noise	30	7.97e-02	9.55e-03	8.23e-02	7.11e-02
gmls, no-noise	30	8.03e-02	9.88e-03	7.33e-02	7.27e-02
gnp, no-noise	50	8.49e-02	1.26e-02	3.86-02	9.93-02
gmls, no-noise	50	1.01e-01	1.39e-02	2.94-02	6.63-02
gnp, outliers, σ =5e-03	50	9.30e-02	1.72e-02	6.94e-02	1.08e-01
gmls, outliers, σ =5e-03	50	9.12e-02	1.74e-02	4.98e-02	9.20e-02
gnp, noise, σ =1e-03	70	9.95e-02	2.02e-02	5.52e-02	1.02e-01
gmls, noise, σ =1e-03	70	1.59e-01	6.09e-02	1.22e-01	9.99e-01

Table 2: ***Geometric PDE Solvers based on GNPs.*** *We show results for the accuracy and robustness of our PDE solvers using numerical methods based on pretrained GNPs. We also make comparisons with alternative GMLS methods. In both cases, we used our GNP methods for filtering outliers, otherwise the GMLS collocation methods do not produce viable results. We show cases when varying the neighborhood size k and cases with and without noise.*

We find in the no-noise case that the collocation methods based on pre-trained GNPs and GMLS methods behave comparably for $k = 30$ and $k = 50$. While both methods are accurate with a precision of at least 10% or better, we find for $k = 50$ the results are slightly less accurate. This arises since as patch sizes become larger they can smooth locally the geometric features of the shapes. In the outlier case, while the GMLS and GNP methods performed similarly, we emphasize the GNPs were used as part of the pre-processing filter for the later GMLS steps to help mitigate noise and outliers. As mentioned, if this was not done GMLS has errors that are too large to produce viable results in the collocation methods. We see our GNP filtering yields enhanced precision of the geometric and PDE estimates allowing for similar accuracies for both solvers.

In the case of uniform Gaussian noise, we find the GNPs performed consistently better than the GMLS methods. We see an especially significant difference for the genus-2 shape. The pre-trained GNP methods yield a solver with 5.5% accuracy compared to 12.2% for the purely GMLS-based

methods. We see similar improvements for the genus-3 shape, where the GNP methods yield a solver with accuracy of 2.0% compared to 6.0% for the purely GMLS-based methods. A notable aspect of each of these cases are regions having large curvatures requiring robust estimators for the geometric and PDE contributions. This shows a key advantage of the learned GNP estimators. Since the GNPs were trained with artifacts during training, they are able to learn more robust estimators that can compensate for noise in the point-cloud data. The results of these studies show how the pre-trained GNPs can be used in the development of numerical methods for approximating solutions of geometric PDEs.

Conclusions

We have shown how transferable GNP models can be learned for processing point-cloud representations for use in geometric tasks. The GNPs can be used to learn estimators for key geometric quantities and other features. The GNP methods also allow for training that incorporates data-driven filtering for handling noise, outliers, and other artifacts in non-pristine point-clouds. We demonstrated how the pretrained GNPs can be used in tasks that include (i) robustly estimating geometric quantities related to the metric and curvatures, (ii) tracking shape changes driven by mean-curvature flows, and (iii) developing numerical methods for approximating the solutions of geometric PDEs. The GNP models also can be incorporated readily into other data processing pipelines and computational methods. We release an open source package with training codes and with weights for our pre-trained GNPs. The introduced approaches provide methods for obtaining general transferable GNP models for performing geometric tasks.

Open Source Package

We release an open source package for our methods at
https://github.com/atzberg/geo_neural_op.

Acknowledgments

Authors research supported by grant NSF Grant DMS-2306101. Authors also would like to acknowledge computational resources and administrative support at the UCSB Center for Scientific Computing (CSC) with grants NSF-CNS-1725797, MRSEC: NSF-DMR-2308708, Pod-GPUs: OAC-1925717, and support from the California NanoSystems Institute (CNSI) at UCSB. P.J.A. also would like to acknowledge a hardware grant from Nvidia.

References

[1] Rishi Bommasani et al. "On the opportunities and risks of foundation models". In: *arXiv preprint arXiv:2108.07258* (2021).

[2] Thomas M Breuel. "Reflections on Foundation Models". In: *Stanford HAI* (2021). URL: https://crfm.stanford.edu/2021/10/18/reflections.html.

[3] Hugo Touvron et al. "Llama: Open and efficient foundation language models". In: *arXiv preprint arXiv:2302.13971* (2023).

[4] Mikhail V Koroteev. "BERT: a review of applications in natural language processing and understanding". In: *arXiv preprint arXiv:2103.11943* (2021).

[5] OpenAI. *ChatGPT*. [Large language model]. 2023. URL: https://chat.openai.com/chat.

[6] OpenAI. *DALL-E-3*. [Image Generation Model]. 2024. URL: https://openai.com/index/dall-e-3/.

[7] Dustin Podell et al. "Sdxl: Improving latent diffusion models for high-resolution image synthesis". In: *arXiv preprint arXiv:2307.01952* (2023).

[8] Midjourney. *Midjourney*. [Image Generation Model]. 2024. URL: https://www.midjourney.com/home.

[9] Joseph Redmon et al. "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[10] Nicolas Carion et al. "End-to-end object detection with transformers". In: *European conference on computer vision*. Springer. 2020, pp. 213–229.

[11] Matthias Minderer et al. "Simple open-vocabulary object detection". In: *European conference on computer vision*. Springer. 2022, pp. 728–755.

[12] Paul J Atzberger. "Importance of the mathematical foundations of machine learning methods for scientific and engineering applications". In: *arXiv preprint arXiv:1808.02213* (2018).

[13] Nathan Baker et al. *Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence*. Tech. rep. USDOE Office of Science (SC), Washington, DC (United States), 2019.

[14] P J Atzberger B Quackenbush. "Geometric neural operators (gnps) for data-driven deep learning in non-euclidean settings." In: *Machine Learning: Science and Technology* 5.4 (2024). ISSN: 2632-2153. DOI: https://doi.org/10.1088/2632-2153/ad8980. URL: https://iopscience.iop.org/article/10.1088/2632-2153/ad8980.

[15] Panos Stinis, Constantinos Daskalakis, and Paul J Atzberger. "SDYN-GANs: Adversarial learning methods for multistep generative models for general order stochastic dynamics". In: *Journal of Computational Physics* 519 (2024), p. 113442.

[16] Ben J Gross and Paul J Atzberger. "Hydrodynamic flows on curved surfaces: Spectral numerical methods for radial manifold shapes". In: *Journal of Computational Physics* 371 (2018), pp. 663–689. DOI: doi.org/10.1016/j.jcp.2020.109340. URL: https://doi.org/10.1016/j.jcp.2018.06.013.

[17] Gabriela González Castro et al. "A survey of partial differential equations in geometric design". In: *The Visual Computer* 24 (2008), pp. 213–225.

[18] David A Rower, Misha Padidar, and Paul J Atzberger. "Surface fluctuating hydrodynamics methods for the drift-diffusion dynamics of particles and microstructures within curved fluid interfaces". In: *Journal of Computational Physics* 455 (2022), p. 110994.

[19] Daniel Millán, Adrian Rosolen, and Marino Arroyo. "Nonlinear manifold learning for meshfree finite deformation thin-shell analysis". In: *International Journal for Numerical Methods in Engineering* 93.7 (2013), pp. 685–713.

[20] M. M. Bronstein et al. "Geometric Deep Learning: Going beyond Euclidean data". In: *IEEE Signal Processing Magazine* 34.4 (2017), pp. 18–42. ISSN: 1053-5888.

[21] Larry Wasserman. "Topological data analysis". In: *Annual Review of Statistics and Its Application* 5 (2018), pp. 501–532.

[22] Marina Meilă and Hanyu Zhang. "Manifold learning: what, how, and why". In: *Annual Review of Statistics and Its Application* 11 (2023).

[23] Trenton Kirchdoerfer and Michael Ortiz. "Data-driven computational mechanics". In: *Computer Methods in Applied Mechanics and Engineering* 304 (2016), pp. 81–101.

[24] Timo Hackel et al. "Semantic3d. net: A new large-scale point cloud classification benchmark". In: *arXiv preprint arXiv:1704.03847* (2017).

[25] Ryan Lopez and Paul J Atzberger. "GD-VAEs: Geometric Dynamic Variational Autoencoders for Learning Nonlinear Dynamics and Dimension Reductions". In: *arXiv preprint arXiv:2206.05183* (2022). URL: http://arxiv.org/abs/2206.05183.

[26] Manzil Zaheer et al. "Deep sets". In: *Advances in neural information processing systems* 30 (2017).

[27] Charles R Qi et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 652–660.

[28] Angel X. Chang et al. *ShapeNet: An Information-Rich 3D Model Repository*. Tech. rep. arXiv:1512.03012 [cs.GR]. Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[29] Patrick Mullen et al. "Signing the unsigned: Robust surface reconstruction from raw pointsets". In: *Computer Graphics Forum*. Vol. 29. 5. Wiley Online Library. 2010, pp. 1733–1741.

[30] Yulan Guo et al. "Deep learning for 3d point clouds: A survey". In: *IEEE transactions on pattern analysis and machine intelligence* 43.12 (2020), pp. 4338–4364.

[31] Ben J Gross et al. "Meshfree methods on manifolds for hydrodynamic flows on curved surfaces: A Generalized Moving Least-Squares (GMLS) approach". In: *Journal of Computational Physics* 409 (2020), p. 109340. DOI: doi.org/10.1016/j.jcp.2020.109340. URL: https://doi.org/10.1016/j.jcp.2020.109340.

[32] Louis Serrano et al. "Operator learning with neural fields: Tackling pdes on general geometries". In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 70581–70611.

[33] Christopher J Budd and Arieh Iserles. "Geometric integration: numerical solution of differential equations on manifolds". In: *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 357.1754 (1999), pp. 945–956.

[34] Marino Arroyo and Michael Ortiz. "Local maximum-entropy approximation schemes". In: *Meshfree methods for partial differential equations III*. Springer, 2007, pp. 1–16.

[35] Jian Liang and Hongkai Zhao. "Solving partial differential equations on point clouds". In: *SIAM Journal on Scientific Computing* 35.3 (2013), A1461–A1486.

[36] Gerhard Dziuk and Charles M Elliott. "Finite element methods for surface PDEs". In: *Acta Numerica* 22 (2013), pp. 289–396.

[37] Xiaolong Wang, David Fouhey, and Abhinav Gupta. "Designing deep networks for surface normal estimation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 539–547.

[38] Ying Li et al. "Deep learning for lidar point clouds in autonomous driving: A review". In: *IEEE Transactions on Neural Networks and Learning Systems* 32.8 (2020), pp. 3412–3432.

[39] Simegnew Yihunie Alaba and John E Ball. "A survey on deep-learning-based lidar 3d object detection for autonomous driving". In: *Sensors* 22.24 (2022), p. 9577.

[40] Yue Wang et al. "Dynamic graph cnn for learning on point clouds". In: *ACM Transactions on Graphics (tog)* 38.5 (2019), pp. 1–12.

[41] Lassi Ruoppa et al. "Unsupervised deep learning for semantic segmentation of multispectral LiDAR forest point clouds". In: *arXiv preprint arXiv:2502.06227* (2025).

[42] Bo Pang et al. "Neural Laplacian Operator for 3D Point Clouds". In: *ACM Transactions on Graphics (TOG)* 43.6 (2024), pp. 1–14.

[43] Bo Pang et al. "Learning the geodesic embedding with graph neural networks". In: *ACM Transactions on Graphics (TOG)* 42.6 (2023), pp. 1–12.

[44] Chenyu Zeng et al. "Point Cloud Neural Operator for Parametric PDEs on Complex and Variable Geometries". In: *arXiv preprint arXiv:2501.14475* (2025).

[45] Senwei Liang et al. "Solving pdes on unknown manifolds with machine learning". In: *Applied and Computational Harmonic Analysis* 71 (2024), p. 101652.

[46] Zongyi Li et al. "Geometry-informed neural operator for large-scale 3d pdes". In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 35836–35854.

[47] Gengxiang Chen et al. "Learning neural operators on riemannian manifolds". In: *National Science Open* 3.6 (2024), p. 20240001.

[48] Weiheng Zhong and Hadi Meidani. "Physics-Informed Geometry-Aware Neural Operator". In: *Computer Methods in Applied Mechanics and Engineering* 434 (2025), p. 117540.

[49] Zhiwei Zhao et al. "Diffeomorphism neural operator for various domains and parameters of partial differential equations". In: *Communications Physics* 8.1 (2025), p. 15.

[50] Nicola Rares Franco, Andrea Manzoni, and Paolo Zunino. "Mesh-informed neural networks for operator learning in finite element spaces". In: *Journal of Scientific Computing* 97.2 (2023), p. 35.

[51] Zongyi Li et al. "Fourier neural operator with learned deformations for pdes on general geometries". In: *Journal of Machine Learning Research* 24.388 (2023), pp. 1–26.

[52] Minglang Yin et al. "A scalable framework for learning the geometry-dependent solution operators of partial differential equations". In: *Nature Computational Science* 4.12 (2024), pp. 928–940.

[53] Nikola Kovachki et al. "Neural operator: learning maps between function spaces with applications to PDEs". In: *J. Mach. Learn. Res.* 24 (2023), Paper No. [89], 97. ISSN: 1532-4435,1533-7928. DOI: 10.1080/15502287.2022.2066031. URL: https://doi.org/10.1080/15502287.2022.2066031.

[54] Tianping Chen and Hong Chen. "Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems". In: *IEEE Transactions on Neural Networks* 6.4 (1995), pp. 911–917. DOI: 10.1109/72.392253.

[55] Justin Gilmer et al. "Message passing neural networks". In: *Machine learning meets quantum physics*. Springer, 2020, pp. 199–214.

[56] Marco Gori, Gabriele Monfardini, and Franco Scarselli. "A new model for learning in graph domains". In: *Proceedings. 2005 IEEE international joint conference on neural networks, 2005.* Vol. 2. IEEE. 2005, pp. 729–734.

[57] Trevor Hastie. *The elements of statistical learning: data mining, inference, and prediction*. 2009.

[58] Harold Hotelling. "Analysis of a complex of statistical variables into principal components." In: *Journal of educational psychology* 24.6 (1933), p. 417.

[59] Karl Pearson. "LIII. On lines and planes of closest fit to systems of points in space". In: *The London, Edinburgh, and Dublin philosophical magazine and journal of science* 2.11 (1901), pp. 559–572.

[60] Gaspard Monge. *Application de l'analyse a la geometrie (etc.) 4*. Bernard, 1809.

[61] Andrew N Pressley. *Elementary differential geometry*. Springer Science & Business Media, 2010.

[62] Ian Goodfellow et al. *Deep learning*. Vol. 1. 2. MIT press Cambridge, 2016.

[63] Nils Bjorck et al. "Understanding batch normalization". In: *Advances in neural information processing systems* 31 (2018).

[64] John P Boyd. *Chebyshev and Fourier spectral methods*. Courier Corporation, 2001.

[65] L.N. Trefethen. *Spectral methods in MATLAB*. Society for Industrial Mathematics, 2000. ISBN: 0898714656.

[66] Jon Karl Sigurdsson and Paul J Atzberger. "Hydrodynamic coupling of particle inclusions embedded in curved lipid bilayer membranes". In: *Soft matter* 12.32 (2016), pp. 6685–6707.

[67] Filippo Bigi et al. "Fast evaluation of spherical harmonics with sphericart". In: *J. Chem. Phys.* 159 (2023), p. 064802.

[68] C. Bane Sullivan and Alexander Kaszynski. "PyVista: 3D plotting and mesh analysis through a streamlined interface for the Visualization Toolkit (VTK)". In: *Journal of Open Source Software* 4.37 (2019), p. 1450. DOI: 10.21105/joss.01450. URL: https://doi.org/10.21105/joss.01450.

[69] Sebastien Valette, Jean-Marc Chassery, and Remy Prost. "Generic Remeshing of 3D Triangular Meshes with Metric-Dependent Discrete Voronoi Diagrams". In: *IEEE Trans. Vis. Comput. Graph.* 14 (Mar. 2008), pp. 369–381. DOI: 10.1109/TVCG.2007.70430.

[70] Davoud Mirzaei, Robert Schaback, and Mehdi Dehghan. "On generalized moving least squares and diffuse derivatives". In: *IMA Journal of Numerical Analysis* 32.3 (2012), pp. 983–1000.

[71] Xifeng Gao et al. "Robust structure simplification for hex re-meshing". In: *ACM Trans. Graph.* 36.6 (Nov. 2017). ISSN: 0730-0301. DOI: 10.1145/3130800.3130848. URL: https://doi.org/10.1145/3130800.3130848.

[72] Keenan Crane, Ulrich Pinkall, and Peter Schröder. "Robust fairing via conformal curvature flow". In: *ACM Transactions on Graphics (TOG)* 32.4 (2013), pp. 1–10.

[73] Zhuojia Fu et al. "Localized collocation schemes and their applications". In: *Acta Mechanica Sinica* 38.7 (2022), p. 422167.

[74] Douglas N Arnold and Wolfgang L Wendland. "On the asymptotic convergence of collocation methods". In: *Mathematics of Computation* 41.164 (1983), pp. 349–381.

[75] Helmut Moritz. "Least-squares collocation". In: *Reviews of geophysics* 16.3 (1978), pp. 421–430.

[76] Ralf Gommers et al. "scipy/scipy: SciPy 1.15. 0". In: *Zenodo* (2024).

[77] Nathan Bell et al. "PyAMG: Algebraic Multigrid Solvers in Python". In: *Journal of Open Source Software* 8.87 (2023), p. 5495. DOI: 10.21105/joss.05495. URL: https://doi.org/10.21105/joss.05495.

[78] Ralph Abraham, Jerrold E Marsden, and Tudor Ratiu. *Manifolds, tensor analysis, and applications*. Vol. 75. Springer Science & Business Media, 2012.

A. Fundamental Forms, Curvature, and Operators on Surfaces

We give expressions for how a few key geometric quantities can be computed from local parameterizations $\boldsymbol{\sigma}(u, v)$ of the surface. In the coordinates (u, v), the first fundamental form $\mathbf{I}$ is defined as

$$\mathbf{I} = \begin{bmatrix} \boldsymbol{\sigma}_u \cdot \boldsymbol{\sigma}_u & \boldsymbol{\sigma}_u \cdot \boldsymbol{\sigma}_v \\ \boldsymbol{\sigma}_v \cdot \boldsymbol{\sigma}_u & \boldsymbol{\sigma}_v \cdot \boldsymbol{\sigma}_v \end{bmatrix} = \begin{bmatrix} E & F \\ F & G \end{bmatrix}. \tag{26}$$

We denote the derivatives by $\boldsymbol{\sigma}_u = \partial_u \boldsymbol{\sigma}/\partial u$, $\boldsymbol{\sigma}_v = \partial_v \boldsymbol{\sigma}/\partial v$, and similarly for higher-order terms. This first fundamental form $\mathbf{I}$ is also equivalent to the metric tensor $\mathbf{g} = \mathbf{I}$. The $\mathbf{I}$ can be used for computations involving distances, arc lengths, and angles on the surface. The second fundamental form $\mathbf{II}$ is defined as

$$\mathbf{II} = \begin{bmatrix} \boldsymbol{\sigma}_{uu} \cdot \boldsymbol{n} & \boldsymbol{\sigma}_{uv} \cdot \boldsymbol{n} \\ \boldsymbol{\sigma}_{uv} \cdot \boldsymbol{n} & \boldsymbol{\sigma}_{vv} \cdot \boldsymbol{n} \end{bmatrix} = \begin{bmatrix} L & M \\ M & N \end{bmatrix}. \tag{27}$$

The $\boldsymbol{n}$ denotes the outward unit normal to the surface given by

$$\boldsymbol{n} = \frac{\boldsymbol{\sigma}_u \times \boldsymbol{\sigma}_v}{\|\boldsymbol{\sigma}_u \times \boldsymbol{\sigma}_v\|}. \tag{28}$$

These can be combined to obtain the Weingarten map $\boldsymbol{W} = \mathbf{I}^{-1}\mathbf{II}$. The $\boldsymbol{W}$ can be used to compute the Gaussian curvature K and the mean curvature H of the surface. These are given by

$$K = \det(\boldsymbol{W}) = \frac{LN - M^2}{EG - F^2}, \tag{29}$$

$$H = \frac{1}{2}\mathrm{tr}(\boldsymbol{W}) = \frac{1}{2}\left(\frac{LG - 2MF + NE}{EG - F^2}\right). \tag{30}$$

The Laplace-Beltrami operator Δ_{LB} extends the Laplacian to scalar functions on the surface. The can be expressed as

$$\Delta_{LB} = \frac{1}{\sqrt{|g|}}\partial_i\left(g^{ij}\sqrt{|g|}\partial_j\right). \tag{31}$$

The $|g|$ denotes the determinant of the metric tensor and g^{ij} denotes the terms of the inverse metric. In these expressions we use the Einstein tensor notations for implicit summation [78]. Further discussions of these geometric quantities and computations also can be found in [16, 61, 78].