Discovering and modelling dynamics on latent manifolds with neural geodesic flows

Julian Bürge
ETH Zürich
Zürich, Switzerland

Lewis O'Donnell
Imperial College London
London, UK

Ben Moseley
Imperial College London
London, UK

Abstract

We present neural geodesic flows (NGFs), a framework for discovering and modelling dynamical systems which assumes the system evolves along the geodesics of a latent Riemannian manifold. Both the metric of the manifold and coordinate transforms between observational data and the manifold are simultaneously learned from observational data. Whilst most approaches for dynamical system modelling make rigid physical assumptions, NGFs only assume geometrical properties of the system's evolution and have the capacity to model a wide range of systems. NGFs are trained in an end-to-end fashion, backpropagating through the coordinate transforms, a numerical geodesic solver on the manifold, and the metric of the manifold, and several techniques such as residual learning and extreme value soft-clipping are required to ensure stable gradient flow. We show that NGFs can accurately model particle flow on a sphere and the two-body problem, and can be applied to generative modelling on arbitrary manifolds; with further work, NGFs could provide a powerful geometry-based framework for dynamical systems modelling.

1 Introduction

Discovering and accurately modelling dynamical systems is an essential task across the sciences, from understanding planetary motion to modelling fluid flow, as it enables the reconstruction of governing equations, prediction of long-term behaviour, and control of complex systems. The classical scientific method is to iterate between handcrafting an ordinary (ODE) or partial (PDE) differential equation to describe the system and testing its ability to predict observational data, however this relies on human intuition and often expensive numerical solvers. More recently, data-driven approaches use observational data to aid the discovery and modelling process, for example by directly estimating parameters and coefficients of governing equations via a suitable optimisation algorithm [2]. Most recently, scientific machine learning (SciML)-based approaches discover and model dynamical systems by integrating prior physical knowledge with ML models, for example by using neural networks to approximate terms in ODEs and PDEs [4, 15, 10], using physics-informed neural networks to discover and model PDEs [6], and using ML to discover and model the Hamiltonian or Lagrangian of physical systems [9, 7]. However, a key challenge with SciML approaches is that they often assume a rigid physical structure of the dynamical system, for example that the system evolves in an Euclidean observed data space, or that the governing equation is derived from a predetermined physics equation, which can lead to poor modelling accuracy and lack of generality to new systems.

In this work we propose an approach for discovering and modelling dynamical systems which instead only makes broad geometrical assumptions on the dynamics. In particular, we propose neural geodesic flows (NGFs), which assume that the system evolves along the geodesics of a latent Riemannian manifold. Many physical systems, such as general relativity and classical dynamics, can be mathematically interpreted in this way [14] and therefore NGFs have the capacity to model a wide range of systems. We investigate their ability to model different dynamical systems, specifically particle flow on a sphere and the two-body problem with restricted orbits, showing that NGFs can

1st Workshop on Differentiable Systems and Scientific Machine Learning @ EurIPS 2025

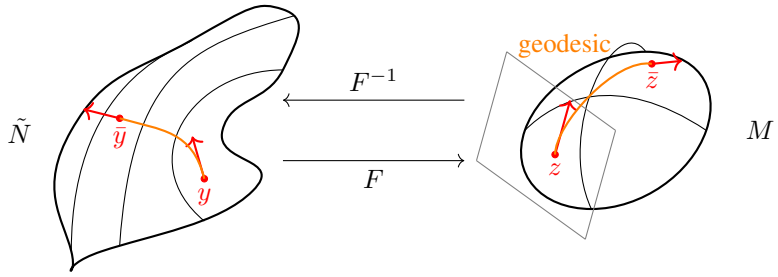

Figure 1: A schematic of a neural geodesic flow (NGF). Here, y represents observations of the state of a dynamical system, z represents associated coordinates on a latent Riemannian manifold.

accurately model their evolution. We also discuss strategies to improve their training stability and convergence, and show that NGFs can be extended to generative modelling on arbitrary manifolds.

2 Methods

2.1 Model definition

We first assume that a dynamical system with some observable state $y \in \tilde{N} \subseteq \mathbb{R}^n$ (for example, the positions and velocities of two planets) evolves on a submanifold $\tilde{N} \subseteq \mathbb{R}^n$ described by a flow $\Xi^t : y \in \tilde{N} \mapsto \bar{y} \in \tilde{N}$. Then the core assumption of NGFs is that there exists a diffeomorphism F (smooth bijective map with smooth inverse) between $\tilde{N}$ and the tangent bundle $N = TM$ of a Riemannian manifold (M, g) such that the original flow Ξ^t on $\tilde{N}$ corresponds to the geodesic flow $\exp_g(\cdot, t)$ on N. An overview of this framework is shown in Figure 1.

To predict the evolution of the system given an initial observation of the system y, we first map y to a coordinate $z \in U \subseteq \mathbb{R}^{2m}$ on a chart of N, with $m = \dim(M)$, using a neural network encoder $z = \psi_\theta(y)$ to represent the mapping F. Next, we evolve the system over time by moving it along geodesics on the latent manifold, i.e. computing the exponential map $\bar{z} = \exp_{g_\theta}(z, t)$, where we use a neural network to represent the metric, g_θ, of the manifold M. Practically this is done by numerically solving the geodesic flow ODE on the chart. Finally, we map the chart coordinates back to the observed data space using a neural network decoder $\bar{y} = \phi_\theta(\bar{z}) \approx \psi_\theta^{-1}(\bar{z})$, representing F^{-1}. The entire forward pass of the NGF is given by

$$y \mapsto \bar{y} = \left(\phi_\theta \circ \exp_{g_\theta}(\cdot, t) \circ \psi_\theta\right)(y) \,. \tag{1}$$

2.2 Training

We use the following loss function to simultaneously train ψ_θ, ϕ_θ and g_θ via gradient descent,

$$\begin{aligned} L = \frac{1}{|I|} \sum_{i \in I} \sum_{j=0}^{k} \Bigg(& \underbrace{\|(\phi_\theta \circ \psi_\theta)(y_j^{(i)}) - y_j^{(i)}\|^2}_{\text{reconstruction loss}} \\ & + \underbrace{\|(\phi_\theta \circ \exp_{g_\theta}(\cdot, t_j) \circ \psi_\theta)(y_0^{(i)}) - y_j^{(i)}\|^2}_{\text{data space prediction loss}} \\ & + \underbrace{\|(\exp_{g_\theta}(\cdot, t_j) \circ \psi_\theta)(y_0^{(i)}) - \psi_\theta(y_j^{(i)})\|^2}_{\text{latent space prediction loss}} \Bigg) \,, \end{aligned} \tag{2}$$

given a training dataset, $D = \left\{y_0^{(i)}, .., y_k^{(i)}\right\}_{i \in I}$, defined on k time points, $\{t_0, ...t_k\}$, of many example state trajectories with different initial conditions. This loss function compares the difference between predicted and ground truth trajectories in both the observed and latent space, as well as the invertibility of the coordinate transform.

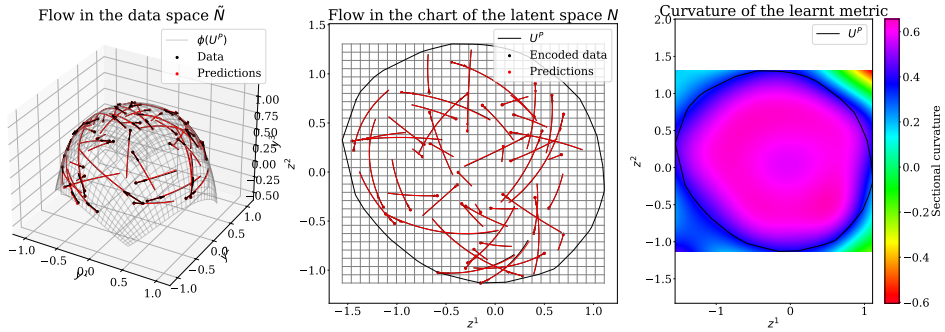

Figure 2: Modelling particle flow on the sphere; (a) ground truth test trajectories (black), NGF predictions (red), and learned manifold (grey) in observed data space, (b) ground truth and predicted trajectories in manifold chart coordinates, (c) sectional curvature of the learned manifold.

Differentiability and gradient stability Assuming that Ξ^t and F are smooth, we expect the entire forward pass (Equation (1)) to be smooth. By choosing smooth neural network architectures for ψ_θ, ϕ_θ (which model F) and g_θ (which models g and completely determines $\exp_g$ on N) we can ensure smoothness and therefore differentiability of the forward pass, which allows us to train ψ_θ, ϕ_θ and g_θ via end-to-end gradient descent in JAX [1]. Whilst differentiable, in practice we find a number of strategies significantly improve gradient stability and training convergence. First, any Riemannian metric must be symmetric positive definite, and therefore we hard-constrain our metric to be $g_\theta = I + L_\theta^T L_\theta$ where I is the identity matrix and L_θ is a learnt upper triangular matrix with a soft plus applied to the diagonal. Note we learn a residual correction, i.e. the model is initialised with a flat (Euclidean) metric, which we find further stabilises convergence. Secondly, we use $\tanh$ activation functions in ψ_θ and ϕ_θ to ensure smoothness. Finally, we employ extreme value soft-clipping during time stepping in the numerical geodesic solver to guard against exploding values.

2.3 Related work

Floryan & Graham [8] and others [17, 16] propose models to discover dynamics on a manifold in a similar fashion to NGFs. The central difference is that a standard neural ODE [4] is used in their latent space, while ours is strictly a geodesic ODE, which endows NGFs with inherently more geometric structure. NGFs automatically obtain the underlying Lagrangian and Hamiltonian of the learnt dynamics, similar to Lagrangian (LNNs [7]) and Hamiltonian neural networks (HNNs [9]), but in addition to these approaches NGFs also allow coordinate transforms to be learned.

3 Results

3.1 Modelling particle flow on the sphere

As a proof of concept, we first task NGFs with learning the dynamics of particles flowing on the upper half sphere, $\mathbb{S}^2_+$. We construct a training set of 16384 trajectories on $\mathbb{S}^2_+$ and test on 1024 trajectories. The trajectories are constructed by setting $\tilde{N} = T\mathbb{S}^2_+ \subseteq \mathbb{R}^6$, drawing uniform random initial points $\{y_i\}_{i\in I}$ on $T\mathbb{S}^2_+$, and evolving them for unit time along spherical geodesics using the standard Runge Kutta 4 (RK4) scheme with 49 timesteps, $\bar{y}_i = \exp_{g_{\mathbb{S}^2_+}}(y_i, t=1)$. This yields trajectories with 50 points, i.e., the dataset: $\left\{y_i = y_0^{(i)}, \ldots, y_{49}^{(i)} = \bar{y}_i\right\}_{i\in I}$ on $\left\{t_0 = 0, \ldots, t_{49} = 1\right\}$. For this problem, since $\tilde{N}$ is itself a tangent bundle, we can restrict ψ_θ and ϕ_θ so that only the point mapping is learned whilst tangents are mapped using its derivative, i.e., use the Jacobian split described in [3]. We use MLPs with 2 hidden layers and 32 neurons to define ψ_θ and ϕ_θ, whilst for the metric, g_θ, we directly learn the values of the upper triangular matrix L_θ (as described in Section 2.2). We use the same RK4 solver to compute the exponential map $\exp_{g_\theta}(\cdot, t)$, and the Adam optimiser for training.

We observe stable convergence during training, and after training we find that our model accurately predicts our test set trajectories, achieving a mean squared trajectory error (MSE) of $8.5 \cdot 10^{-5}$ in the

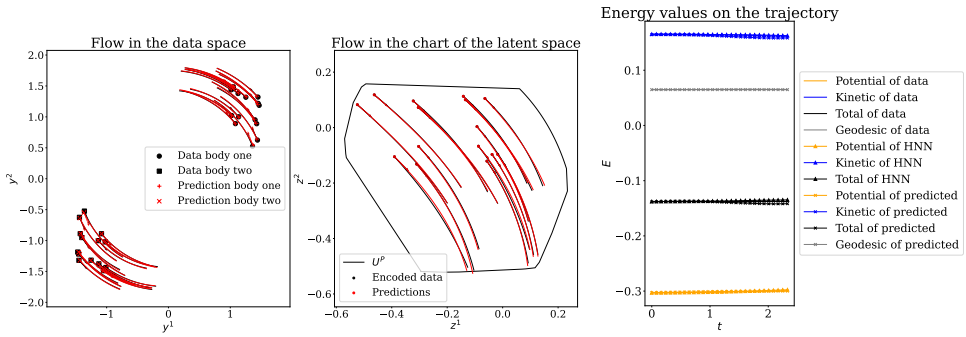

Figure 3: Modelling the two-body problem; (a) ground truth test trajectories (black) and NGF predictions (red) in observed data space, (b) ground truth and predicted trajectories in manifold chart coordinates, (c) energy of ground truth system, NGF, and reference HNN.

observed data space, where the average trajectory length is approximately 0.5 units. See Figure 2 for a visual comparison between the predicted and ground truth trajectories. Furthermore, we plot the sectional curvature of our learned latent manifold using our learned metric, g_θ, in Figure 2, and find our model learned a geometry with positive constant sectional curvature, which, by the Killing-Hopf theorem [11], means that it successfully learned a geometry isometric to the sphere. We conclude that NGFs can successfully model this problem, accurately learning the dynamics of the system and its underlying manifold.

3.2 Modelling the two-body problem

Since NGFs are aimed at dynamics discovery, next we attempt to discover the dynamics of two bodies orbiting each other under the influence of Newtonian gravity. We use the code of HNNs [9] to generate our training and test sets of trajectories, where each trajectory is of two bodies of unit mass on near circular orbits with varying radii, as shown in Figure 3. Because the NGF code is currently only implemented for a single global manifold chart, we restrict orbits to the first quadrant for body A and third quadrant for body B, as a multi-chart atlas is required to cover the full orbit [8]. We generate 4000 training trajectories and 900 test trajectories, where each trajectory has 30 timesteps. Similar to Section 3.1, we use MLPs with 2 hidden layers and 32 neurons to define ψ_θ and ϕ_θ, in this case without a Jacobian split, a RK4 solver to compute $\exp_{g_\theta}(\cdot, t)$, and the Adam optimiser.

We observe stable training, and after training find that the NGF accurately predicts the test set trajectories, achieving a trajectory MSE of $3.6 \cdot 10^{-5}$ in the data space, where the average trajectory length is approximately 1.5 units. We plot 15 example test predictions in Figure 3. In addition, we find that the NGF model conserves total energy well; the total energy MSE of the test set is $1.4 \cdot 10^{-5}$, almost on par with the reference HNN used in [9] (designed specifically for energy conservation) which achieves $4 \cdot 10^{-6}$ on our test set. We note that by construction NGFs conserve geodesic energy, which is also plotted in Figure 3. In summary, NGFs effectively capture the dynamics of this system. However, extending the framework to a multi-chart atlas will be an important next step for modelling complete orbital trajectories and evaluating the model's long-term stability.

3.3 Generative flow matching on arbitrary manifolds

An interesting future extension of NGFs is to utilise their geodesic solver and latent manifold learning ability to extend existing Riemannian flow matching models [5] and carry out generative modelling on arbitrary manifolds. We present preliminary work towards this goal in Appendix A.

4 Conclusions and further work

NGFs offer a geometry-based framework for discovering and modelling dynamical systems, and we showed they are able to accurately model particle flow on a sphere and the two body problem with restricted orbits. Future work will investigate their performance on more complex systems such as the chaotic N-body problem and fluid flow, the use of multiple charts to represent more complex manifolds, and their application to generative modelling on learned manifolds.

References

[1] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[2] Steven L. Brunton, Joshua L. Proctor, J. Nathan Kutz, and William Bialek. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences of the United States of America*, 113(15):3932–3937, apr 2016.

[3] Julian Bürge. Neural geodesic flows. Master thesis, ETH Zürich, March 2025. Published in the ETH research collection at `https://doi.org/10.3929/ethz-b-000733724`.

[4] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural Ordinary Differential Equations. *NIPS*, 109(NeurIPS):31–60, jun 2018.

[5] Ricky T.Q. Chen and Yaron Lipman. Flow Matching On General Geometries. In *12th International Conference on Learning Representations, ICLR 2024*. International Conference on Learning Representations, ICLR, feb 2024.

[6] Zhao Chen, Yang Liu, and Hao Sun. Physics-informed learning of governing equations from scarce data. *Nature Communications*, 12(1):1–13, oct 2021.

[7] Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian Neural Networks. *ICLR 2020 Workshop DeepDiffEq*, mar 2020.

[8] Daniel Floryan and Michael D. Graham. Data-driven discovery of intrinsic dynamics. *Nature Machine Intelligence*, 4(12):1113–1120, dec 2022.

[9] Sam Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. In *Advances in Neural Information Processing Systems*, volume 32. Neural information processing systems foundation, jun 2019.

[10] Patrick Kidger. *On Neural Differential Equations*. PhD thesis, University of Oxford, 2022.

[11] John M Lee. *Introduction to Riemannian manifolds*, volume 2. Springer, 2018.

[12] Yaron Lipman, Ricky T.Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow Matching For Generative Modeling. In *11th International Conference on Learning Representations, ICLR 2023*. International Conference on Learning Representations, ICLR, oct 2023.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[14] Ong Chong Pin. Curvature and mechanics. *Advances in Mathematics*, 15(3):269–311, 1975.

[15] Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, and Ali Ramadhan. Universal differential equations for scientific machine learning. *arXiv*, jan 2020.

[16] Aleksei Sholokhov, Yuying Liu, Hassan Mansour, and Saleh Nabi. Physics-informed neural ODE (PINODE): embedding physics into models using collocation points. *Scientific Reports*, 13(1):1–13, dec 2023.

[17] Kevin Zeng, Carlos E.Pérez De Jesús, Andrew J Fox, and Michael D Graham. Autoencoders for discovering manifold dimension and coordinates in data from complex dynamical systems. *Machine Learning: Science and Technology*, 5(2):025053, may 2024.

Funding

We are thankful to Imperial College London, the EPSRC Centre for Doctoral Training in Collaborative Computational Modelling at the Interface, and University College London for providing travel funding to allow us to present this work at EurIPS.

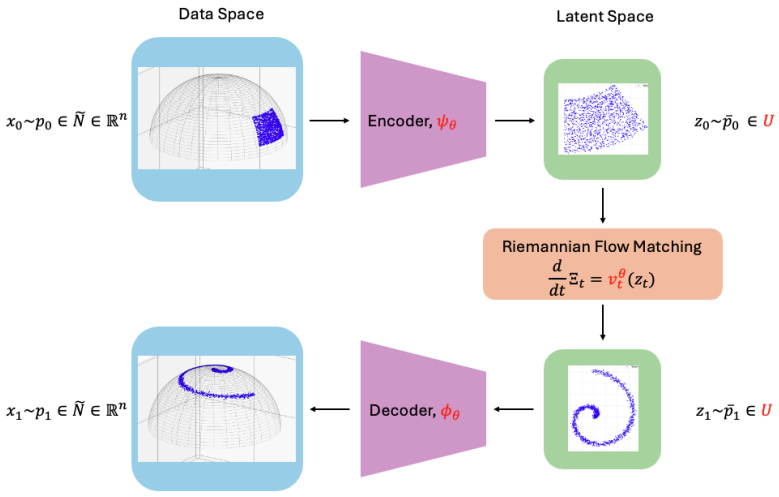

Figure 4: Schematic of a generative NGF. Learnable components are in red.

A Generative NGFs

In this section we present preliminary results on extending NGFs to carry out generative modelling on arbitrary manifolds. We define the task of generative modelling on an arbitrary manifold as follows: let $x = (x^1, x^2, \ldots, x^d) \in \tilde{N} \in \mathbb{R}^d$ be data points on a submanifold $\tilde{N}$ of $\mathbb{R}^d$. A generative model is then a model, ξ, that takes samples, $x_0 \sim p_0$ from a base distribution defined on the manifold and transforms them to samples, $x_1 \sim p_1$, so that they resemble samples, $\hat{x}_1 \sim q$ taken from some target distribution q defined on the manifold, i.e. that $\xi(x_0) = x_1 \sim p_1 \approx q$.

A computationally efficient option is flow matching [12], where a neural network is used to parametrise a time-dependent velocity field, v_t^θ (with learnable parameters θ), that is then used to construct ξ as a solution to a flow ODE,

$$\frac{d}{dt}\xi_t(x) = v_t^\theta(\xi_t(x)), \qquad \xi_0(x) = x_0 \ , \tag{3}$$

for times $t \in [0, 1]$, where v_t^θ is trained using the conditional loss function,

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t, q(\hat{x}_1), p_t(x|\hat{x}_1)} ||v_t^\theta(x) - u_t(x|\hat{x}_1)||^2 \ . \tag{4}$$

Here $u_t(x)$ is the velocity field which generates $p_t(x)$, and typically linear Gaussian conditional probability paths $p_t(x|\hat{x}_1)$ are assumed to make computation of $u_t(x|\hat{x}_1)$ tractable. However, in this standard formulation, flow matching does not ensure that samples $\xi_t(x_0)$ from the flow remain on the manifold $\tilde{N}$.

Our generative model is instead defined as follows; first, we assume we have access to a pretrained NGF equipped with the mappings ψ_θ,ϕ_θ and metric g_θ related to $\tilde{N}$. Then we use the NGF encoder ψ_θ to encode base samples x_0 to chart coordinates z_0. Next, we learn a flow velocity $v_t^\theta(z)$ defined in the latent (chart) space, which moves z_0 to z_1. Finally, we use ϕ_θ to map z_1 back to a target sample x_1 in the data space. In this way, samples from the flow always remain constrained to the manifold defined by the NGF. An overview of this model is shown in Figure 4. We then use a similar conditional loss function to learn $v_t^\theta(z)$,

$$\mathcal{L}_{\text{LCRFM}}(\theta) = \mathbb{E}_{t, \tilde{q}(\hat{z}_1), \tilde{p}_t(z|\hat{z}_1)} ||v_t^\theta(z) - u_t(z|\hat{z}_1)||^2_{g_\theta} \ , \tag{5}$$

where target velocities $u_t(z|\hat{z}_1)$ are computed by first combining the NGF's geodesic solver with an iterative shooting method to search for an initial velocity u_0 which moves a randomly sampled z_0 along a geodesic to a random true sample $\hat{z}_1$, and then computing $(z_t, u_t) = \exp_{g_\theta}((z_0, u_0), t)$. Our approach closely follows Riemannian flow matching (RFM) proposed in [5], except that we compute Equation (5) directly in the latent space rather than the data space, use a pretrained metric, and a geodesic solver to generate training velocities.

In initial experiments, we compare generative NGFs to Euclidean flow matching [12] and Riemannian flow matching [5] in three test cases. The first, simplest problem, compares the models on two-dimensional Euclidean space. For the base distribution, we use a uniform distribution constrained to a unit square; the target distribution is the "make_moons" dataset from Scikit-learn [13]. We then move to two more complex geometries, namely the positive half sphere and the ring torus. The base distribution for both experiments was again a uniform patch lying on the respective geometries. On the sphere, we used a spiral distribution as our target (see Figure 4), and on the torus, the target was the same uniform patch translated to the opposite side. Each model was trained with 1024 samples from the same base and target distributions on each geometry for 5000 epochs. The neural networks for the trained velocity fields are simple MLPs with width 128, depth 5, ELU activations, and we use the Adam optimiser.

Our results demonstrate that the NGF generative model produces higher fidelity samples than Euclidean flow matching in all tests; we record an order of magnitude energy distance improvement when the data lies on a non-Euclidean manifold. Notably, on the ring torus, our model produces a distribution with an energy distance of $4.99 \cdot 10^{-3}$ between itself and the target distribution, compared to $1.26 \cdot 10^{-2}$ from Euclidean flow matching. Importantly, and as expected, our NGF model produces distributions with zero samples lying off the data manifold. The results for these experiments are displayed in Table 1. Note that ED represents the energy distance and Frac represents the fraction of points in the produced distribution that lie off the data manifold.

	2D Euclidean		**Sphere**		**Torus**	
Model	ED	Frac	ED	Frac	ED	Frac
FM	4.87×10^{-3}	0.00	4.71×10^{-2}	2.47×10^{-1}	1.26×10^{-2}	9.45×10^{-1}
RFM	–	–	$\mathbf{2.36 \times 10^{-3}}$	0.00	–	–
NGF	$\mathbf{2.76 \times 10^{-3}}$	0.00	3.13×10^{-3}	0.00	$\mathbf{4.99 \times 10^{-3}}$	0.00

Table 1: Table showing the results from the generative flow matching experiments.