A Riemannian Take on Distance Fields and Geodesic Flows in Robotics

Journal Title
XX(X):1–26
©The Author(s) 0000
Reprints and permission:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Yiming Li[1,2,*], Jiacheng Qiu[1,*] and Sylvain Calinon[1,2]

Abstract

Distance functions are crucial in robotics for representing spatial relationships between a robot and its environment. They provide an implicit, continuous, and differentiable representation that integrates seamlessly with control, optimization, and learning. While standard distance fields rely on the Euclidean metric, many robotic tasks inherently involve non-Euclidean structures. To this end, we generalize Euclidean distance fields to more general metric spaces by solving the Riemannian eikonal equation, a first-order partial differential equation whose solution defines a distance field and its associated gradient flow on the manifold, enabling the computation of geodesics and globally length-minimizing paths. We demonstrate that *geodesic distance fields*—the classical Riemannian distance function represented as a global, continuous, and queryable field—are effective for a broad class of robotic problems where Riemannian geometry naturally arises. To realize this, we present a neural Riemannian eikonal solver (NES) that solves the equation as a mesh-free implicit representation without grid discretization, scaling to high-dimensional robot manipulators. Training leverages a physics-informed neural network (PINN) objective that constrains spatial derivatives via the PDE residual and boundary/metric conditions, so the model is supervised by the governing equation and requires no labeled distances or geodesics. We propose two NES variants, conditioned on boundary data and on spatially varying Riemannian metrics, underscoring the flexibility of the neural parameterization. We validate the effectiveness of our approach through extensive examples, yielding minimal-length geodesics across diverse robot tasks involving Riemannian geometry. Additionally, we validate the method in a dynamics-aware motion-planning task for energy-efficient trajectory generation, with comparisons to baseline approaches.

Keywords

Differential Geometry, Riemannian Manifold, Distance Field, Geodesic, Eikonal Equation, Gradient Flow, Energy Conservation

1 Introduction

In robotics, measuring distances constitutes a fundamental concept for determining spatial relationships and enabling effective physical and non-physical interactions with the environment. These metrics provide a systematic means for quantifying the geometric relationships between various entities, such as points, poses, shapes or trajectories. They are widely applicable across robotic tasks, including inverse kinematics (Chiacchio et al. 1991) and motion planning (Ratliff et al. 2009). Signed distance fields (SDFs), in particular, have gained popularity for representing geometries using implicit functions, as they enable efficient distance and gradient queries which are suitable to integrate into learning (Weng et al. 2022), optimization (Li et al. 2024b) and control (Liu et al. 2022).

SDFs are conventionally employed in Euclidean spaces, representing the shortest distance from any point in the environment to the boundary of a given object or surface (Park et al. 2019). However, many robot tasks inherently operate in non-Euclidean spaces, with manifolds that can be described implicitly by a smoothly varying weighting matrix, which locally measures distances. For example, distance fields can also be applied to joint configuration space, indicating the minimum joint motion required by the robot to establish contact with a given point or object (Li et al. 2024a). In this case, *geodesic distance fields* enable the consideration of inertia, stiffness, or manipulability ellipsoids in the processing, by providing a Riemannian metric constructed with a smoothly varying, symmetric, positive definite (SPD) weighting matrix in the robot configuration space. Riemannian geometry provides a principled and systematic way to generalize algorithms from Euclidean spaces to more general manifolds (Calinon 2020). Figure 1 shows minimal-distance paths in Euclidean space and on a Riemannian manifold. In the latter, geodesics are shaped by the manifold's geometry—for example, when the Riemannian metric is defined by the robot's inertia matrix to reflect its dynamic properties.

While the geometric nature of robot problems on Riemannian manifolds is well established, many methods for computing geodesics operate on prescribed endpoint pairs and rely on local optimization (Jaquier and Asfour

[1] Idiap Research Institute, Switzerland.
[2] École Polytechnique Fédérale de Lausanne (EPFL), Switzerland
[*] Equal Contribution

Corresponding author:
Yiming Li, Idiap Research Institute, Centre du Parc, Rue Marconi 19, Martigny, CH-1920, Switzerland.
Email: yiming.li@idiap.ch

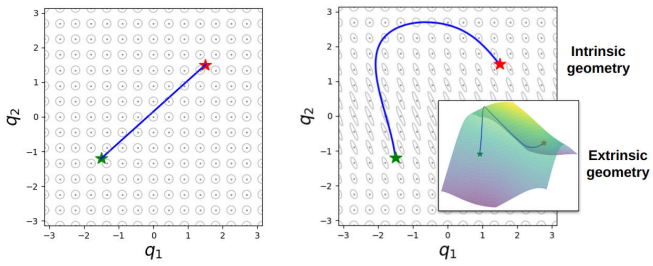


Figure 1. Minimal distance paths as geodesics in the Euclidean space (a) and in another Riemannian metric space (b). The ellipses depict the SPD weighting matrices used to locally compute distances with this metric (isocontours of inverse matrices). A Riemannian manifold can be described intrinsically by the depicted metric. For visualization, it can also be depicted with corresponding extrinsic geometry in a higher-dimensional space (see inset), but geodesic computation does not require this construction and instead only requires the metric as an intrinsic geometry representation.

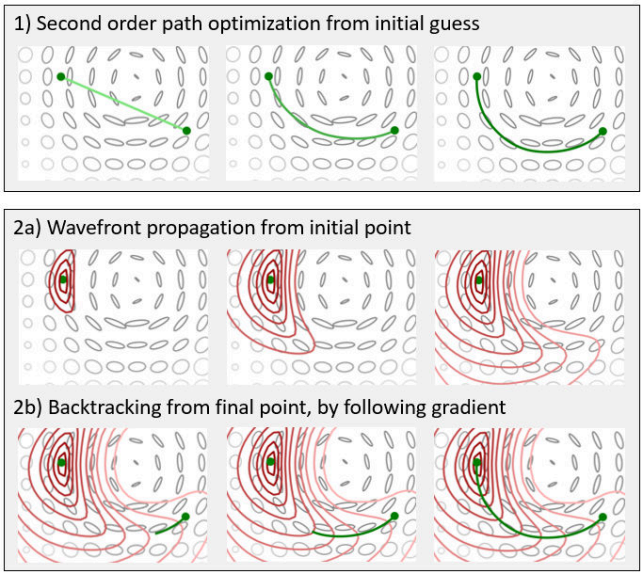


Figure 2. Two approaches for computing geodesics on inhomogeneous manifolds. The traditional method (1) formulates the problem as an iterative solution to a differential equation, often using second-order optimization (e.g., Gauss-Newton path optimization). This approach requires a good initial guess and must be solved separately for each point pair. In contrast, we propose a wavefront propagation approach that first computes a geodesic distance field from a source point by solving the Riemannian eikonal equation (2a), then retrieves geodesics by backtracking along the gradient of this field (2b). We employ physics-informed neural networks (PINNs) to solve the eikonal equation, enabling scalable solutions in high-dimensional settings. This method encodes the manifold's intrinsic geometry and yields globally optimal geodesics. It also offers modularity and efficiency: the distance field can be trained offline and used online for fast distance and geodesic queries.

2022; Klein et al. 2023; Cabrera and Hatton 2024). These approaches are computationally intensive, must be re-run for every new endpoint pair, and do not provide a global representation of the manifold. As an alternative, we represent the manifold with a *geodesic distance field*, a global and continuous value function that implicitly encodes geodesics (Crane et al. 2020) and from which minimal paths are recovered by backtracking. However, constructing such fields on Riemannian manifolds is challenging due to nonlinear metrics and complex topology; classical geodesic ray tracing and geodesic ordinary differential equation (ODE) integration/shooting are computationally expensive, sampling dependent, and lack global guarantees (Rawlinson and Sambridge 2005). Motivated by wavefront propagation methods that are well established for distance fields in Euclidean space, we compute the distance field by solving the Riemannian eikonal equation, a first-order partial differential equation (PDE) for wave travel time, which yields an efficient and globally consistent single source solution (Kimmel and Sethian 1998). This perspective is closely related to recent implicit shape models based on SDFs, which appear as viscosity solutions of the eikonal equation (Gropp et al. 2020; Marić et al. 2024). The resulting field captures the manifold's global structure, provides gradient information for fast queries, and reveals geodesic flow for straightforward trajectory backtracking. Beyond geodesic computation, this representation enables learning, optimization, and control methods based on distance fields on the manifold. An illustration is given in Figure 2.

Classical PDE solvers for wavefront propagation include the Fast Marching Method (FMM) (Sethian 1996) and fast sweeping methods (Zhao 2005), widely used in level set and Hamilton–Jacobi settings (Osher et al. 2004). FMM computes the entire single-source distance field in a single, monotone outward pass, after which geodesics to any target are obtained by inexpensive backtracking. These schemes run on discretized grids or meshes and yield deterministic solutions under standard conditions, though fixed resolution limits scalability in high-dimensional spaces. We build on this framework along two axes. First, we generalize the eikonal equation from Euclidean spaces to Riemannian manifolds, enabling geodesic distance fields and flows in robot configuration spaces with curved, non-Euclidean geometry. Second, to overcome grid scalability, we introduce a neural Riemannian eikonal solver (NES) based on physics-informed neural networks (PINN) (Raissi et al. 2019; Kelshaw and Magri 2024). NES replaces the grid with a coordinate-based neural representation, computes gradients via automatic differentiation, and yields a continuous field that can be queried at arbitrary resolution. Crucially, it can be conditioned on start–goal pairs to produce geodesic flows— which is a challenging problem for fixed-source PDE solvers. Once trained, NES outputs arbitrary state-to-goal distance queries with millisecond-level latency. Training is self-supervised by the eikonal constraint and boundary conditions, avoiding precomputed distance labels. We also introduce two NES variants that can be conditioned on boundary points or on the Riemannian metric, enabled by the flexibility and differentiability of neural network representation. In summary, our approach leverages deep learning to solve the Riemannian eikonal equation, delivering a scalable, flexible, and real-time representation of distance fields and geodesic flows that integrates naturally with learning, optimization, and control in robotics.

To demonstrate the advantages of our neural Riemannian eikonal solver for distance fields and geodesic flows, we present examples across common robotics geometries: energy-aware (kinetic/potential) metrics for dynamics-aware motion, pullback metrics for task-space minimization, and tailored metrics for stability shaping and collision avoidance. We also study dynamics-aware motion formulated as geodesic minimization under an energy-aware metric that encodes the system dynamics. NES computes geodesic flows efficiently for arbitrary source–target pairs, providing a global, geometry-aware prior toward lower-effort motions; in practice, the resulting shortest paths often align with energy-efficient trajectories and are easier to track. We demonstrate efficient planning and simple tracking on planar robots and a 7-DoF Franka arm, highlighting scalability, efficiency, and flexibility. The main contributions are summarized as follows:

- We propose solving the *eikonal equation* to obtain distance fields and geodesic flows for robot problems on Riemannian manifolds. Unlike geodesic shooting or optimal-control methods, this approach reframes geodesic computation as a *single global* implicit representation of the manifold, enabling fast backtracking and direct integration with planning and control.
- We introduce a Neural Eikonal Solver (NES) that learns the eikonal PDE in a physics-informed manner without distance labels, yielding a continuous, differentiable field with efficient gradient queries. NES scales to high-dimensional spaces beyond grid-based solvers. We also present two variants conditioned on boundary data and on spatially varying Riemannian metrics, underscoring the flexibility of the neural parameterization.
- We provide extensive examples across kinematics, dynamics, motion planning, and control. On energy-aware manifolds, NES yields shorter paths under the chosen Riemannian metric and, in our experiments, achieves lower energy costs than baselines on high-dimensional systems, with both quantitative and qualitative comparisons.

The rest of the article is structured as follows: Section 2 introduces the necessary mathematical background, and Section 3 reviews related work. Section 4 describes our NES for solving the Riemannian eikonal equation to compute geodesic distance fields and flows. Section 5 presents extensive examples using our approach across robot problems involving Riemannian geometry. Section 6 reports experimental results on dynamics-aware motion generation, demonstrating the effectiveness of our method. Finally, Section 7 concludes the paper and outlines potential applications and future directions.

2 Background

In this section, we introduce the mathematical background of Riemannian manifolds, geodesics, the eikonal equation and introduce commonly used Riemannian metrics in robotics.

2.1 Riemannian Metrics and Geodesics

A d-dimensional Riemannian manifold $\mathcal{M}$ is a topological space equipped with a smooth metric tensor $\boldsymbol{G}(\boldsymbol{x})$ defined at each point $\boldsymbol{x} \in \mathcal{M}$. The metric tensor $\boldsymbol{G}(\boldsymbol{x})$ is a symmetric positive definite matrix that defines the Riemannian metric, allowing us to calculate distances and angles on the manifold. For each point $\boldsymbol{x} \in \mathcal{M}$, there exists a tangent space $T_{\boldsymbol{x}}\mathcal{M}$, which locally linearizes the manifold.

The inner product of two velocity vectors, $\boldsymbol{u}$ and $\boldsymbol{v}$, in the tangent space $T_{\boldsymbol{x}}\mathcal{M}$ at a point $\boldsymbol{x} \in \mathcal{M}$ is given by

$$\langle \boldsymbol{u}, \boldsymbol{v} \rangle_{\boldsymbol{G}} = \boldsymbol{u}^\top \boldsymbol{G}(\boldsymbol{x}) \boldsymbol{v}. \tag{1}$$

Using this inner product, we define the Riemannian norm of a vector $\boldsymbol{u}$ as

$$\|\boldsymbol{u}\|_{\boldsymbol{G}} = \sqrt{\langle \boldsymbol{u}, \boldsymbol{u} \rangle_{\boldsymbol{G}}}. \tag{2}$$

These definitions allow us to measure vector lengths and angles within the tangent space $T_{\boldsymbol{x}}\mathcal{M}$. With this, we can define the Riemannian distance between two points, $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, on the given manifold as

$$U(\boldsymbol{x}_1, \boldsymbol{x}_2) = \int_{s_0}^{s_1} \|\dot{\gamma}(s)\|_{\boldsymbol{G}(\gamma(s))} \, \mathrm{d}s, \tag{3}$$

where $\gamma(s)$ is a smooth curve connecting $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, with $\gamma(s_0) = \boldsymbol{x}_1$ and $\gamma(s_1) = \boldsymbol{x}_2$. The minimization of this expression allows us to define geodesic distances and shortest geodesic paths between two points on the manifold.

2.2 Eikonal Equation

2.2.1 Isotropic eikonal equation

The eikonal equation is a nonlinear first-order partial differential equation (PDE) that models wavefront propagation. Let $\Omega \subset \mathbb{R}^n$ be the domain, $\boldsymbol{x}_1 \in \Omega$ a prescribed source, and $c : \Omega \to (0, \infty)$ a positive scalar function specifying the travel speed. The standard form is

$$\|\nabla U(\boldsymbol{x}_2)\| = c(\boldsymbol{x}_2) \quad \text{s.t.} \quad U(\boldsymbol{x}_1) = 0, \tag{4}$$

where $U : \Omega \to \mathbb{R}$ is the distance-to-source function. Here, $U(\boldsymbol{x}_2)$ is used as a notation instead of $U(\boldsymbol{x}_1, \boldsymbol{x}_2)$ when the source $\boldsymbol{x}_1$ is fixed, corresponding to the geodesic γ:

$$U(\boldsymbol{x}_2) = \min_{\gamma} \int_0^1 c(\gamma(s)) \, \|\dot{\gamma}(s)\| \, ds, \tag{5}$$

Here $\gamma : [0, 1] \to \Omega$ is a path parameterized by $s \in [0, 1]$ with endpoints $\gamma(0) = \mathbf{x}_1$ (source) and $\gamma(1) = \mathbf{x}_2$ (target), and $\dot{\gamma}(s)$ denotes its velocity. $c(\cdot)$ specifies the local speed that shapes the arrival time/distance. The function $U(\boldsymbol{x}_2)$ with fixed source $\boldsymbol{x}_1$ is the *single-point* solution; for global distance fields, one can encode $U(\boldsymbol{x}_1, \boldsymbol{x}_2)$ over arbitrary *source–goal* pairs.

2.2.2 Riemannian eikonal equation

The isotropic eikonal equation describes the distance field and yields shortest paths in Euclidean space. This generalizes to Riemannian manifolds (Mirebeau 2019; Peyré et al. 2010):

$$\|\nabla U(\boldsymbol{x}_2)\|_{\boldsymbol{G}^{-1}(\boldsymbol{x}_2)} = 1 \quad \text{s.t.} \quad U(\boldsymbol{x}_1) = 0, \tag{6}$$

which characterizes wavefront propagation and minimal distance on a manifold endowed with a Riemannian metric

defined by $\boldsymbol{G}$. Unless noted otherwise, we adopt the unit-weight case $c(\boldsymbol{x}_2) = 1$; any strictly positive field can be absorbed by rescaling the metric. The corresponding geodesic distance is

$$U(\boldsymbol{x}_2) \;=\; \min_{\gamma} \int_0^1 \big\|\dot\gamma(s)\big\|_{\boldsymbol{G}(\gamma(s))}\, ds, \tag{7}$$

which is the minimal Riemannian distance (3). Equation (6) measures the norm of the covector ∇U with the inverse matrix $\boldsymbol{G}^{-1}$, whereas (7) measures the speed of the vector $\dot\gamma$ with the metric tensor $\boldsymbol{G}$. These are *dual norms*, induced by $\boldsymbol{G}$ on vectors and by $\boldsymbol{G}^{-1}$ on covectors. This dual pairing underlies the Hamilton–Jacobi characterization of Riemannian distance (Mantegazza and Mennucci 2002) and is made explicit in dual-metric formulations for eikonal solvers and geodesic backtracking (Mirebeau 2019).

To backtrack geodesics, we can define the geodesic flow:

$$\boldsymbol{V}(\boldsymbol{x}_2) := \boldsymbol{G}^{-1}(\boldsymbol{x}_2)\,\nabla U(\boldsymbol{x}_2). \tag{8}$$

From (6) it follows that $\boldsymbol{V}$ is unit-speed with respect to $\boldsymbol{G}$:

$$\begin{aligned}\|\boldsymbol{V}(\boldsymbol{x}_2)\|^2_{\boldsymbol{G}(\boldsymbol{x}_2)} &= \boldsymbol{V}(\boldsymbol{x}_2)^\top\, \boldsymbol{G}(\boldsymbol{x}_2)\, \boldsymbol{V}(\boldsymbol{x}_2)\\ &= \big(\boldsymbol{G}^{-1}(\boldsymbol{x}_2)\,\nabla U(\boldsymbol{x}_2)\big)^\top\, \boldsymbol{G}(\boldsymbol{x}_2)\, \big(\boldsymbol{G}^{-1}(\boldsymbol{x}_2)\,\nabla U(\boldsymbol{x}_2)\big)\\ &= \nabla U(\boldsymbol{x}_2)^\top\, \boldsymbol{G}^{-1}(\boldsymbol{x}_2)\,\nabla U(\boldsymbol{x}_2)\\ &= 1.\end{aligned} \tag{9}$$

Given a target $\boldsymbol{x}_2$ and source $\boldsymbol{x}_1$, a geodesic is recovered by integrating the flow backward in time

$$\dot\gamma(s) = -\,\boldsymbol{V}\big(\gamma(s)\big), \qquad \gamma(0) = \boldsymbol{x}_2,\;\; \gamma(T) = \boldsymbol{x}_1. \tag{10}$$

Along this backward flow, U decreases at a unit rate. Starting from $U(\boldsymbol{x}_2)$ and terminating at the source where $U(\boldsymbol{x}_1) = 0$, the elapsed time equals the initial value, i.e., $T = U(\boldsymbol{x}_2)$.

A common numerical approach for solving eikonal equations is the Fast Marching Method (FMM). Like Dijkstra's algorithm, FMM relies on discretization and sequentially propagates information from the boundary or solved nodes on the mesh (Sethian 1996). This method extends to non-Euclidean domains, allowing computation of geodesic distances on manifolds (Kimmel and Sethian 1998). More recent work leverages machine learning to approximate solutions, producing continuous, differentiable distance fields in Euclidean spaces (Grubas et al. 2023) as well as on nonlinear manifolds (Kelshaw and Magri 2024).

2.3 Common Riemannian Metrics in Robotics

In this section, we introduce commonly used Riemannian metrics that furnish a geometric perspective on robotic problems. These include metrics that encode intrinsic kinematic and dynamic properties of the robot, as well as task-shaped designs. We also clarify the practical meaning of geodesics on the resulting manifolds and indicate how they can be computed via solving the Riemannian eikonal equation.

2.3.1 Kinetic–energy metric. Let $\boldsymbol{q}$ denote the joint configuration and $\mathbf{M}(\boldsymbol{q})$ the inertia matrix. The kinetic–energy metric is the smooth, symmetric positive-definite matrix that defines the Riemannian metric (Bullo and Lewis 2004)

$$\mathbf{G}_{\mathrm{ke}}(\boldsymbol{q}) := \mathbf{M}(\boldsymbol{q}). \tag{11}$$

The kinetic energy is defined as

$$T(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \tfrac{1}{2}\,\dot{\boldsymbol{q}}^\top \mathbf{M}(\boldsymbol{q})\,\dot{\boldsymbol{q}} = \tfrac{1}{2}\,\|\dot{\boldsymbol{q}}\|^2_{\mathbf{G}_{\mathrm{ke}}(\boldsymbol{q})}. \tag{12}$$

Under the unit–speed condition $\|\dot{\boldsymbol{q}}\|_{\mathbf{G}_{\mathrm{ke}}} = 1$, geodesics on this Riemannian manifold coincide with trajectories that minimize the kinetic–energy functional.

2.3.2 Jacobi metric. The metric defined by $\mathbf{G}_{\mathrm{ke}}(\boldsymbol{q}) = \mathbf{M}(\boldsymbol{q})$ captures only the kinetic part of the dynamics and neglects the potential energy $P(\boldsymbol{q})$. For a conservative system with fixed total energy H, the Jacobi metric is defined by (Casetti et al. 2000):

$$\mathbf{G}_{\mathrm{Jac}}(\boldsymbol{q}) = 2\big(H - P(\boldsymbol{q})\big)\,\mathbf{M}(\boldsymbol{q}), \qquad H > P(\boldsymbol{q}). \tag{13}$$

The Hamiltonian satisfies $H = T + P$ and remains constant along conservative motion (Lutter and Peters 2023). The Jacobi metric is a conformal transformation of the kinetic–energy metric and is therefore smooth, symmetric, and positive-definite on the configuration space manifold. By the Maupertuis–Jacobi principle, trajectories with fixed energy H correspond to geodesics on the Riemannian manifold endowed with this metric (Albu-Schäffer and Sachtler 2022).

2.3.3 Pullback metric. The pullback operation induces a configuration–space metric from a task–space metric (Ratliff et al. 2018). Let $\varphi : \mathcal{Q} \to \mathcal{X}$ be a smooth task map with Jacobian $\mathbf{J}(\boldsymbol{q})$, and let $\mathbf{G}_{\mathcal{X}}$ be a smooth positive–definite matrix on $\mathcal{X}$. We define

$$\mathbf{G}_{\mathrm{pb}}(\boldsymbol{q}) := \mathbf{J}(\boldsymbol{q})^\top\, \mathbf{G}_{\mathcal{X}}\, \mathbf{J}(\boldsymbol{q}). \tag{14}$$

as a smooth positive semidefinite matrix. One may add a small joint–space regularization,

$$\tilde{\mathbf{G}}_{\mathrm{pb}}(\boldsymbol{q}) := \mathbf{G}_{\mathrm{pb}}(\boldsymbol{q}) + \lambda \mathbf{I} \quad \text{with } \lambda > 0, \tag{15}$$

which yields a strictly SPD matrix while preserving the task–induced geometry. Common choices for $\mathbf{G}_{\mathcal{X}}$ include *stiffness* metrics with $\mathbf{G}_{\mathcal{X}} = \mathbf{K}_x$, where $\mathbf{K}_x \succ 0$ may be diagonal for per–axis stiffness or a full SPD matrix capturing cross–axis coupling (Hogan 1985); and *manipulability* metrics, either kinematic with $\mathbf{G}_{\mathcal{X}} = \mathbf{J}(\boldsymbol{q})\mathbf{J}(\boldsymbol{q})^\top$ (Yoshikawa 1985) or more general with $\mathbf{G}_{\mathcal{X}} = \mathbf{J}(\boldsymbol{q})\,\mathbf{\Lambda}(\boldsymbol{q})^{-1}\,\mathbf{J}(\boldsymbol{q})^\top$, where $\mathbf{\Lambda}(\boldsymbol{q}) \succ 0$ is a joint–space weighting, for example $\mathbf{\Lambda}(\boldsymbol{q}) = \mathbf{M}(\boldsymbol{q})$ to account for inertia in dynamic manipulability (Lachner et al. 2020), though other choices are possible. Additionally, setting $\mathbf{G}_{\mathcal{X}} = \mathbf{I}$ yields the Euclidean metric in task space; its pullback $\mathbf{J}(\boldsymbol{q})^\top\mathbf{J}(\boldsymbol{q})$ measures how joint–space motions map to task–space displacements. Geodesics under this metric correspond to joint–space paths that minimize task–space path length.

2.3.4 Other task-specific metrics. Apart from the metrics above that reflect the robot's inherent geometry, one can also design problem-specific metrics to encode task objectives directly. These task-specific metrics reshape the manifold so that geodesics realize desired behaviors. We illustrate this idea with two examples:

Obstacle avoidance. Obstacle avoidance typically relies on a collision checker or a distance to the nearest obstacle. To obtain adaptive behavior that is fast when far from obstacles and slow near boundaries, we introduce a smoothly varying scale $\phi(\boldsymbol{x})$ and a distance-aware Riemannian metric tensor that penalizes motion along the normal direction to the obstacle (Klein et al. 2023):

$$\begin{aligned} \phi(\boldsymbol{x}) &= f\big(\alpha(\tau - d(\boldsymbol{x}))\big)\,(1-\phi_{\min}) + \phi_{\min}, \\ \boldsymbol{G}_{\text{obs}}(\boldsymbol{x}) &= \phi(\boldsymbol{x})\,\boldsymbol{r}_1(\boldsymbol{x})\boldsymbol{r}_1(\boldsymbol{x})^\top + \boldsymbol{r}_2(\boldsymbol{x})\boldsymbol{r}_2(\boldsymbol{x})^\top, \end{aligned} \tag{16}$$

where f is a smooth monotone function (e.g., the sigmoid $f(z) = 1/(1+e^{-z})$), $d(\boldsymbol{x})$ is the signed distance function, $\alpha > 0$ controls the transition sharpness, τ sets a safety margin, and $\phi_{\min} \ll 1$ enforces a minimum scale near obstacles. The vectors $\{\boldsymbol{r}_1(\boldsymbol{x}), \boldsymbol{r}_2(\boldsymbol{x})\}$ form a local orthonormal frame with $\boldsymbol{r}_1(\boldsymbol{x})$ aligned with $\nabla d(\boldsymbol{x})$. Here, we use $\boldsymbol{x}$ to denote a task–space point, however, the formulation extends to configuration space $\boldsymbol{q}$ by employing distance fields for articulated robots (Li et al. 2024b,a). With ϕ increasing as proximity grows, the metric amplifies the cost of motion toward obstacles while leaving tangential motion comparatively less penalized. From a geometric perspective, this metric warps the local geometry near obstacles, causing geodesics to bend around them while maintaining clearance. The induced speed scaling further slows motion in close proximity, enhancing safety.

Stability-aware passive motion. In motion planning it is often desirable to bias trajectories toward passively stable, low–potential–energy regions (Ortega et al. 2002). A simple and effective construction is to scale the inertia by a normalized potential, yielding a position-dependent Riemannian metric tensor

$$\begin{aligned} \mathbf{G}_{\text{sta}}(\boldsymbol{q}) &= \mathbf{M}(\boldsymbol{q})\,\big[1 + \epsilon\,\tilde{P}(\boldsymbol{q})\big], \\ \tilde{P}(\boldsymbol{q}) &= \frac{P(\boldsymbol{q}) - P_{\min}}{P_{\max} - P_{\min}} \in [0,1], \quad \epsilon > 0, \end{aligned} \tag{17}$$

where $\mathbf{M}(\boldsymbol{q})$ is the physical inertia matrix, $P(\boldsymbol{q})$ the potential energy, and $\epsilon > 0$ controls the strength of the stability bias. $P_{\max}$ and $P_{\min}$ are the maximum and minimum potential energy, respectively. Larger $P(\boldsymbol{q})$ produces a larger effective metric and hence longer paths, so geodesics on this manifold naturally avoid unstable regions while preserving the coupling encoded by $\mathbf{M}(\boldsymbol{q})$. This metric preserves natural dynamics via uniform inertia modulation, regulates speed by slowing motion in high–potential zones, and guides the system toward low–energy basins.

Embedding stability in the metric makes robustness intrinsic: the eikonal is solved on a geometry already biased toward low–energy, stable states. The resulting distance field and geodesic flow integrate naturally with learning, planning, and control approaches to incorporate additional task–specific objectives and constraints.

3 Related Work

3.1 Distance Fields in Robotics

Distance fields are fundamental representations in robotics, due to their capacity to implicitly encode spatial information while offering continuous, differentiable representations and efficient computational properties. This versatility has led to extensive exploration of signed distance fields (SDFs) for representing scenes and objects (Millane et al. 2024; Marić et al. 2024), with demonstrated applications in collision detection (Macklin et al. 2020), grasp synthesis (Liu et al. 2021), motion generation (Ratliff et al. 2009), and manipulation planning (Yang and Jin 2024). Moreover, distance fields are increasingly utilized as latent geometric features for downstream tasks such as dynamics models learning (Driess et al. 2022), grasp pose estimation (Breyer et al. 2021; Weng et al. 2022), and motion policy generation (Fishman et al. 2023). Recent advances have introduced distance fields encoded with joint angles (Liu et al. 2022; Koptev et al. 2022; Li et al. 2024b), enabling efficient distance queries between arbitrary points and the surfaces of articulated robots. Building on this foundation, our previous work (Li et al. 2024a) extended the concept of distance fields to the configuration space, wherein the representation measures the minimal joint motion required for a robot to reach specified points. The representation of articulated robots using distance fields can be interpreted as an implicit forward/inverse kinematics model, facilitating the utilization of distance and gradient information directly in joint space. By inherently capturing joint positions and velocities, this approach opens up new possibilities for advancing applications in reactive motion planning and control (Koptev et al. 2024).

3.2 Solving the Eikonal Equation for Distance Fields

Early work computed travel-time (distance) fields by integrating geodesic ODEs as initial/boundary-value problems using shooting or ray tracing (Julian and Gubbins 1977). While effective locally, ray-based methods are sensitive to initialization and step size. They provide limited coverage away from sampled rays and can suffer from efficiency and convergence issues on manifolds with complex geometry or strong anisotropy (Rawlinson and Sambridge 2005). PDE-based solvers take a different route: they solve the eikonal equation, whose viscosity solution is the shortest-path distance from prescribed sources and whose characteristics encode front propagation. Such solvers are widely used in seismic tomography (Lin et al. 2009), rendering (Ihrke et al. 2007), image segmentation (Alvino et al. 2007), and collision avoidance (Garrido et al. 2013). The Fast Marching Method (FMM) (Sethian 1996) computes eikonal solutions on discretized grids with near-linear complexity, though grid discretizations face memory and resolution limits in high-dimensional spaces. Unlike ODE ray tracing, which returns distances only along sampled geodesics, PDE formulations yield a *global* distance field whose gradient flow induces geodesics.

Recent advances in physics-informed neural networks (PINNs) enable grid-free eikonal solvers by representing the solution as a differentiable neural field and training it via the PDE residual and boundary conditions, with gradients obtained through backpropagation (Raissi et al. 2019). In this view, the network provides a continuous scalar field, while the eikonal constraint in the loss drives it to behave as a distance-to-go function even without supervised distance

labels, making its gradients directly usable for geodesic backtracking. This approach has been applied to isotropic eikonals (e.g., EikoNet) (Smith et al. 2020). In robotics, the eikonal constraint also appears when training implicit signed distance fields for shape representation (Gropp et al. 2020; Xie et al. 2022). For motion planning, NTFields solves neural eikonal equations under collision-avoidance constraints and demonstrates fast, scalable generation in high-dimensional spaces (Ni and Qureshi 2022, 2024b). Beyond Euclidean settings, the eikonal constraint has been extended to manifolds (Ni and Qureshi 2024a). The Riemannian Fast Marching method adapts fast marching to anisotropic (Riemannian) metrics (Mirebeau 2019), and the heat method computes geodesic distances via short-time heat flow (Crane et al. 2013). More recently, neural eikonal solvers have been studied directly on manifolds to compute geodesics and distance fields, further broadening applicability (Kelshaw and Magri 2024).

3.3 Motion Planning on Manifolds

Recent advances in motion planning leveraged Riemannian manifolds to tackle complex challenges. Obstacles are often treated as features that reshape the geometry of the space, allowing geodesics to naturally navigate around them and achieve collision-free motion (Ratliff et al. 2015; Laux and Zell 2021). Building on this concept, Riemannian motion policies can be used in joint space using a pullback metric (Ratliff et al. 2018). This framework was later extended to Geometric Fabrics (Van Wyk et al. 2022) by incorporating principles of classical mechanics for more adaptable motion planning. Beyond static obstacle avoidance, dynamic-aware motions have been explored through kinetic energy-based Riemannian metrics (Jaquier and Asfour 2022; Klein et al. 2023), with further extensions to the Jacobi metric that account for both kinetic and potential energy, enabling energy-conserving paths (Albu-Schäffer and Sachtler 2022). Additionally, Riemannian metrics have been applied to human motion modeling, where geodesics represent minimum-effort paths in configuration space (Neilson et al. 2015). These ideas have inspired methods to transfer human arm motions to robots, facilitating more natural and human-like behavior (Klein et al. 2022). Unlike these approaches that focus on local policies or optimizing for the shortest geodesics, our method emphasizes constructing a comprehensive distance field over the entire configuration space, allowing for more flexible and efficient motion planning.

4 Riemannian Eikonal Solver

In this section, we present the methodology for solving the Riemannian eikonal equation to compute geodesic distances and flows. For clarity, we first describe a numerical PDE solver based on Riemannian Fast Marching (RFM), and then introduce our *Neural Riemannian Eikonal Solver (NES)*, which parameterizes the PDE and learns the distance field and geodesic flows directly. We also present two NES variants that condition NES on boundary points and metrics. Because our tasks primarily involve solving the equation on the configuration-space manifold, we denote configurations by $\boldsymbol{q}$ unless we explicitly refer to task-space points, which we denote by $\boldsymbol{x}$.

4.1 Riemannian Fast Marching

Riemannian Fast Marching (RFM) (Mirebeau 2019) is an extension of classical Fast Marching Methods for solving the eikonal equation on anisotropic, inhomogeneous manifolds endowed with a Riemannian metric. As described in Section 2.2.2, we seek the single-point solution $U(\boldsymbol{q})$ such that $\|\nabla U(\boldsymbol{q})\|_{\boldsymbol{G}^{-1}(\boldsymbol{q})} = 1$, and we recover geodesics by backtracking the unit-$\boldsymbol{G}$-speed flow (8). Like FMM, RFM tackles single-source problems by discretizing the manifold into a grid and applying an upwind finite-difference scheme. The wavefront starts from a fixed source $\boldsymbol{q}_s$ and marches outward. RFM updates the travel time at each neighbor $\boldsymbol{q}_n$ of a current front point $\boldsymbol{q}_e$ via

$$U(\boldsymbol{q}_n) \;=\; \min_{\boldsymbol{q}_e \in \boldsymbol{Q}_e} \Big(U(\boldsymbol{q}_e) + \|\boldsymbol{q}_n - \boldsymbol{q}_e\|_{\boldsymbol{G}(\boldsymbol{q}_e)} \Big), \qquad (18)$$

where $\boldsymbol{Q}_e$ is the evolving wavefront, $\boldsymbol{q}_n \in \boldsymbol{Q}_n = N(\boldsymbol{q}_e)$ is a neighbor, and $\|\boldsymbol{q}_n - \boldsymbol{q}_e\|_{\boldsymbol{G}(\boldsymbol{q}_e)}$ is the local metric-induced norm. A detailed algorithm for computing geodesic distances on configuration-space manifolds is given in Algorithm 1. After obtaining $U(\boldsymbol{q})$, geodesics from an arbitrary point $\boldsymbol{q}$ to the source $\boldsymbol{q}_s$ are retrieved efficiently by backtracking using (10) and the geodesic field (8).

Algorithm 1: Riemannian Fast Marching

Input: Discretized grid $\mathcal{M}$; matrix $\boldsymbol{G}(\boldsymbol{q})$; source $\boldsymbol{q}_s$
Output: $U(\boldsymbol{q})$: geodesic distance from $\boldsymbol{q}_s$ to $\boldsymbol{q} \in \mathcal{M}$
Initialization:
set $U(\boldsymbol{q}_s) = 0$; $U(\boldsymbol{q}) = \infty$ for all $\boldsymbol{q} \neq \boldsymbol{q}_s$.
set $\boldsymbol{Q}_e = \{\boldsymbol{q}_s\}$ `wavefront`
Propagation Step:
while *there exists* $\boldsymbol{q} \in \mathcal{M}$ *with* $U(\boldsymbol{q}) = \infty$ **do**
  for *each* $\boldsymbol{q}_e \in \boldsymbol{Q}_e$ **do**
    find neighbors $\boldsymbol{Q}_n = N(\boldsymbol{q}_e)$
    for *each* $\boldsymbol{q}_n \in \boldsymbol{Q}_n$ **do**
      if $U(\boldsymbol{q}_n) = \infty$ **then**
        update $U(\boldsymbol{q}_n)$ via (18)
        $\boldsymbol{Q}_e \leftarrow \boldsymbol{Q}_e \cup \{\boldsymbol{q}_n\}$
    $\boldsymbol{Q}_e \leftarrow \boldsymbol{Q}_e \setminus \{\boldsymbol{q}_e\}$
Termination: for all $\boldsymbol{q} \in \mathcal{M}$, $U(\boldsymbol{q}) \neq \infty$.

As a finite-difference, single-pass method, RFM is computationally efficient and numerically accurate on grids. However, the dependence on an explicit discretization limits scalability for high-dimensional manipulators. We therefore propose a neural parameterization in the next section to address this limitation.

4.2 Neural Riemannian Eikonal Solver (NES)

The Neural Riemannian Eikonal Solver (NES) is inspired by recent developments in solving PDEs through deep neural networks. Different from classical grid-based approaches, it operates in a continuous space without explicit discretization. Gradients are calculated through network backpropagation by automatic differentiation, allowing for high-dimensional manifolds with continuously varying metrics.

Instead of seeking *single-point* solutions, NES allows for global geodesic distances for *source-goal* pairs, where $U(\boldsymbol{q}_s, \boldsymbol{q}_e)$ is a function of both source $\boldsymbol{q}_s$ and goal $\boldsymbol{q}_e$ points, thanks to the flexible structure provided by neural networks. Therefore, the Riemannian eikonal equation to be solved is written as

$$\|\nabla_{\boldsymbol{q}_e} U(\boldsymbol{q}_s, \boldsymbol{q}_e)\|_{\boldsymbol{G}^{-1}(\boldsymbol{q}_e)} = 1, \quad \text{s.t.} \quad U(\boldsymbol{q}_s, \boldsymbol{q}_s) = 0. \tag{19}$$

This equation lets us train a neural field $U_\theta(\boldsymbol{q}_s, \boldsymbol{q}_e)$ directly from the eikonal constraint: we sample source–goal pairs and optimize a PDE-residual loss that acts on $\nabla_{\boldsymbol{q}_e} U_\theta$. To obtain a consistent and continuous distance field, we further impose the following physical constraints:

Symmetry. The geodesic distance from a source point $\boldsymbol{q}_s$ to the destination point $\boldsymbol{q}_e$, and in the other direction are identical, following the symmetry property that $U(\boldsymbol{q}_s, \boldsymbol{q}_e) = U(\boldsymbol{q}_e, \boldsymbol{q}_s)$. It also applies to their partial derivatives: $\nabla_{\boldsymbol{q}_e} U(\boldsymbol{q}_s, \boldsymbol{q}_e) = \nabla_{\boldsymbol{q}_e} U(\boldsymbol{q}_e, \boldsymbol{q}_s)$. To impose this constraint, we define a symmetric function $u_{\boldsymbol{\theta}}^{\text{sym}}$ (Ni and Qureshi 2022)

$$u_{\boldsymbol{\theta}}^{\text{sym}}(\boldsymbol{q}_s, \boldsymbol{q}_e) = \frac{u_{\boldsymbol{\theta}}(\boldsymbol{q}_s, \boldsymbol{q}_e) + u_{\boldsymbol{\theta}}(\boldsymbol{q}_e, \boldsymbol{q}_s)}{2}, \tag{20}$$

where $u_{\boldsymbol{\theta}}$ is the output of neural network parameterized by $\boldsymbol{\theta}$. This equation ensures the symmetry of the network output with respect to permuted source-to-goal pairs.

Non-negativity and non-singularity. The geodesic distance is strictly positive between any two distinct points, and zero when the points coincide. It can be simply achieved by adding a non-negative activation function σ. However, the geodesic distance should approach zero for points close to one another, which might cause singularity issues, leading to numerical errors in distances and gradients for points close to the source point. To overcome this problem, we follow the approach in (Kelshaw and Magri 2024; Smith et al. 2020) that factorizes the distance function as

$$U_{\boldsymbol{\theta}}(\boldsymbol{q}_s, \boldsymbol{q}_e) = \|\boldsymbol{q}_e - \boldsymbol{q}_s\| \sigma\left(u_{\boldsymbol{\theta}}^{\text{sym}}(\boldsymbol{q}_s, \boldsymbol{q}_e)\right), \tag{21}$$

where $\sigma(\cdot)$ is a non-negative activation function and $\|\boldsymbol{q}_e - \boldsymbol{q}_s\|$ is the Euclidean term between two joint configurations for non-singularity. This equation guarantees the non-negativity of geodesic distance and implicitly constrains the gradient pointing to the destination.

Loss Functions. The parameterization involves a multi-layer perceptron (MLP) neural network, with a batch of concatenated joint configuration pairs $\boldsymbol{q}_s$ and $\boldsymbol{q}_e$ as input and outputs the predicted geodesic distance $U_{\boldsymbol{\theta}}(\boldsymbol{q}_s, \boldsymbol{q}_e)$. Ground truth geodesic distances are unknown, and the neural network is supervised through the physical law defined by the Riemannian eikonal equation (19). Therefore, for each source-to-goal point pair, we minimize the loss function

$$\mathbb{L}_{\text{eik}}(\boldsymbol{q}_s, \boldsymbol{q}_e) = \left(\|\nabla_{\boldsymbol{q}_e} U_{\boldsymbol{\theta}}(\boldsymbol{q}_s, \boldsymbol{q}_e)\|_{\boldsymbol{G}^{-1}(\boldsymbol{q}_e)} - 1\right)^2 \tag{22}$$

to construct the geodesic distance field. The partial derivatives $\nabla_{\boldsymbol{q}_e} U_{\boldsymbol{\theta}}(\boldsymbol{q}_s, \boldsymbol{q}_e)$ are computed through automatic differentiation. In addition, to produce a smooth geodesic distance field, we add a regularization term based on the Laplace-Beltrami operator, which defines the divergence of the vector field:

$$\mathbb{L}_{\text{div}}(\boldsymbol{q}_s, \boldsymbol{q}_e) = \left(G^{ij}\left(\frac{\partial^2 U(\boldsymbol{q}_s, \boldsymbol{q}_e)}{\partial q_{e,i} \partial q_{e,j}} - \Gamma_{ij}^k \frac{\partial U(\boldsymbol{q}_s, \boldsymbol{q}_e)}{\partial q_{e,k}}\right)\right)^2. \tag{23}$$

The Laplace–Beltrami term discourages spurious high curvature and reduces gradient oscillations, yielding more stable geodesic backtracking. For details on the derivation of the divergence term, please refer to Appendix C.

The total loss is

$$\mathbb{L}_{\text{total}} = \frac{1}{N} \sum_{n=1}^{N} \left(\mathbb{L}_{\text{eik}} + \lambda \mathbb{L}_{\text{div}}\right), \tag{24}$$

where N is the batch size and λ weights the divergence regularization term.

4.3 Conditioned NES

The key advantage of NES over traditional numerical PDE solvers and optimal control approaches is its grid-free and end-to-end differentiability, which provides great flexibility for the neural network parameterization. Building on this property, we introduce conditioned NES (C-NES), which contains two variants, either conditioned on boundaries or conditioned on Riemannian metrics.

4.3.1 Conditioned on boundaries. In the standard NES formulation, both source and goal lie in the same space (e.g., the robot's configuration space), and the solution traces a minimal–distance curve on that manifold. Many manipulation tasks, however, involve coupling the task space and joint space. We address this by conditioning the boundary on a task–space specification while still propagating the wavefront in joint space—a hybrid process that implicitly handles inverse kinematics.

Here, we condition the boundary on a task–space source point $\boldsymbol{x}_s$. Conceptually, the boundary in configuration space is the set of configurations satisfying $f(\boldsymbol{q}) = \boldsymbol{x}_s$, where f is the forward kinematics. In practice, we do *not* enumerate these IK solutions. Instead, we use f directly in the network to anchor the boundary implicitly. The neural network is parameterized as:

$$U_{\boldsymbol{\theta}}(\boldsymbol{x}_s, \boldsymbol{q}_e) = \left\|f(\boldsymbol{q}_e) - \boldsymbol{x}_s\right\| \sigma\big(u_{\boldsymbol{\theta}}(\boldsymbol{x}_s, \boldsymbol{q}_e)\big), \tag{25}$$

where $\sigma(\cdot)$ enforces nonnegativity and the Euclidean factor $\|f(\boldsymbol{q}_e) - \boldsymbol{x}_s\|$ drives $U_{\boldsymbol{\theta}}$ to zero whenever the end–effector reaches the task–space source $\boldsymbol{x}_s$. Because $\boldsymbol{x}_s$ (task space) and $\boldsymbol{q}_e$ (configuration space) live in different spaces, symmetry does not apply; we therefore use the raw output $u_{\boldsymbol{\theta}}$ rather than a symmetrized variant. The corresponding Riemannian eikonal equation we aim to solve is

$$\begin{aligned} &\|\nabla_{\boldsymbol{q}_e} U(\boldsymbol{x}_s, \boldsymbol{q}_e)\|_{\boldsymbol{G}^{-1}(\boldsymbol{q}_e)} = 1, \\ &U(\boldsymbol{x}_s, \boldsymbol{q}) = 0 \quad \text{whenever } f(\boldsymbol{q}) = \boldsymbol{x}_s. \end{aligned} \tag{26}$$

The wavefront originates on the boundary set $\{\boldsymbol{q} : f(\boldsymbol{q}) = \boldsymbol{x}_s\}$ in configuration space and propagates outward; geodesics are recovered by backtracking the flow from the query configuration $\boldsymbol{q}_e$. Under this formulation, $U(\boldsymbol{x}_s, \boldsymbol{q}_e)$ equals the Riemannian distance (under $\boldsymbol{G}$) from $\boldsymbol{q}_e$ to the IK set. Backtracking the geodesic flow from $\boldsymbol{q}_e$ therefore terminates at an IK configuration that is *geodesically closest* to $\boldsymbol{q}_e$. Crucially, this selection is obtained *without* computing or sampling all IK solutions—the PDE solution together with (25) imposes the boundary/source implicitly and recovers the minimizing IK endpoint via geodesics.

The training is identical to standard NES (22)–(23), except the source is specified in task space. The losses are

$$\mathbb{L}_{\text{eik}}(\boldsymbol{x}_s, \boldsymbol{q}_e) = \Big(\|\nabla_{\boldsymbol{q}_e} U_{\boldsymbol{\theta}}(\boldsymbol{x}_s, \boldsymbol{q}_e)\|_{\boldsymbol{G}^{-1}(\boldsymbol{q}_e)} - 1\Big)^2,$$
$$\mathbb{L}_{\text{div}}(\boldsymbol{x}_s, \boldsymbol{q}_e) = \Big(G^{ij}(\boldsymbol{q}_e)\big[\frac{\partial^2 U(\boldsymbol{x}_s, \boldsymbol{q}_e)}{\partial q_{e,i} \partial q_{e,j}} - \Gamma^k_{ij}(\boldsymbol{q}_e) \frac{\partial U(\boldsymbol{x}_s, \boldsymbol{q}_e)}{\partial q_{e,k}}\big]\Big)^2. \tag{27}$$

4.3.2 Conditioned on metrics. Another variant of C–NES conditions the eikonal on a *parameterized* Riemannian metric. Let the metric depend on a task/physics parameter ω, written $\boldsymbol{G}(\boldsymbol{q}; \omega)$, and let the corresponding distance be $U(\boldsymbol{q}_s, \boldsymbol{q}_e \mid \omega)$. The neural parameterization remains as in (20)–(21); we simply augment the network inputs with ω, i.e., $u_{\boldsymbol{\theta}} = u_{\boldsymbol{\theta}}(\boldsymbol{q}_s, \boldsymbol{q}_e, \omega)$, while enforcing symmetry only over $(\boldsymbol{q}_s, \boldsymbol{q}_e)$. The parameter ω acts through the metric, so the eikonal becomes

$$\|\nabla_{\boldsymbol{q}_e} U(\boldsymbol{q}_s, \boldsymbol{q}_e \mid \omega)\|_{\boldsymbol{G}^{-1}(\boldsymbol{q}_e;\omega)} = 1, \qquad U(\boldsymbol{q}_s, \boldsymbol{q}_s \mid \omega) = 0. \tag{28}$$

The training objectives are identical to (22)–(23), evaluated with the augmented input ω. Conditioning NES on the Riemannian metric allows a single network to represent a *family* of distance fields and geodesic flows. At test time, we adapt to different metrics by adjusting the parameter ω, allowing us to interpolate smoothly across geometries by varying ω.

4.4 Benefits of NES

Building on the RFM approach introduced earlier for solving the Riemannian eikonal equation, we summarize the key advantages of NES over classical numerical PDE solvers for computing distance fields and geodesics:

Grid-free and end-to-end differentiable. NES operates in continuous space without explicit meshes and is trained via automatic differentiation. The resulting distance fields and geodesic flows are end-to-end differentiable, enabling gradient-based optimization, policy learning, and closed-loop control within a unified framework.

Scales to high dimensions. By avoiding discretization and leveraging neural parameterization, NES remains tractable on high-degree-of-freedom configuration manifolds with continuously varying metrics, where grid- or mesh-based methods become impractical due to resolution and memory constraints. The inference time is also fast once the model is trained.

Globality. Motivated by Fast Marching, NES enforces the eikonal PDE as a derivative constraint on U over the whole domain and approximates its viscosity solution, producing a single, globally consistent distance potential on the configuration manifold. In contrast, shooting/BVP methods require explicit geodesic solves and can become trapped in local minima.

Flexibility. NES encodes a two-point value function $U(\boldsymbol{q}_s, \boldsymbol{q}_e)$ rather than a single-source field, and it naturally supports conditioning on boundary specifications as well as on parameterized Riemannian metrics $\boldsymbol{G}(\cdot; \omega)$. These changes are handled by inputs to a single model, whereas numerical PDE solvers typically require re-discretization or re-solving the problem to accommodate them.

5 Examples

In this section we illustrate how the proposed methods produce distance fields and geodesic flows that are directly useful for robot problems. We present two groups of examples. First, we validate NES and its conditioned variant (C–NES) under an energy-related metric that reflects the robot's dynamics. Second, we show how different Riemannian metrics shape distance fields and motions for various robot applications. We provide quantitative visualizations to clarify the behavior of the learned fields and flows.

5.1 NES for Minimal Energy Geodesics

We consider a 2D planar manipulator with two links, each having a length of $l_1 = l_2 = 2$ and masses $m_1 = m_2 = 1$ concentrated at each articulation. The joint angle ranges from $-\pi$ to π for both links. Consequently, the corresponding inertial mass matrix $\boldsymbol{M}(\boldsymbol{q})$ is expressed as:

$$\boldsymbol{M}(\boldsymbol{q}) = \begin{bmatrix} (m_1+m_2)l_1^2 + m_2 l_2^2 + 2m_2 l_1 l_2 \cos(q_2) & m_2 l_2^2 + m_2 l_1 l_2 \cos(q_2) \\ m_2 l_2^2 + m_2 l_1 l_2 \cos(q_2) & m_2 l_2^2 \end{bmatrix}, \tag{29}$$

where q_1 and q_2 are the joint angles. The configuration–space manifold is endowed with the kinetic–energy Riemannian metric $\boldsymbol{G}_{\text{ke}}(\boldsymbol{q}) = \boldsymbol{M}(\boldsymbol{q})$. We visualize the manifold and metric as ellipsoids in Figure 3(a). Given a fixed source, we solve the Riemannian eikonal equation using the RFM approach in Algorithm 1, yielding the distance field (b) and geodesic flows (c). Panel (d) shows a 3D visualization of the resulting geodesics, together with a Euclidean reference path and the corresponding robot motions. Geodesics are curved paths in configuration space that minimize kinetic energy.

NES Solution . Using the approach in Section 4.2, we can also obtain solutions through the proposed neural Riemannian eikonal solver, visualized in Figure 4. Panels (a) and (b) show the learned distance field and associated geodesic flows. The differences with RFM in (c) are small, indicating that the network accurately approximates the Riemannian eikonal and yields results comparable to the RFM baseline. Panel (d) shows geodesics between arbitrary configuration pairs, including symmetric round trips, underscoring the model's ability to generalize to arbitrary source–goal pairs. In contrast, RFM is single–source and must be recomputed for each new query.

C-NES on the Task-space Boundary. Using the same kinetic–energy metric, we further show NES conditioned on a boundary induced by a task–space source, as visualized in Figure 5. The end–effector target is $\boldsymbol{x}_s = (2.0, 2.0)$, shown in Figure 5(a). This point admits two IK solutions, e.g., $(q_1, q_2) = (0, 1.57)$ and $(1.57, -1.57)$, but these are not provided to C–NES a priori. C–NES takes $(\boldsymbol{x}_s, \boldsymbol{q}_e)$ as input; the resulting distance map is shown in Figure 5(a), and the corresponding geodesic flows in Figure 5(b). Backtracking the flow from $\boldsymbol{q}_e$ discovers the IK solutions (red stars) and, upon integration, yields the geodesic that minimizes kinetic–energy distance. Figure 5(c) displays four motions that all reach the same end–effector position but start from different configurations. Due to redundancy, different geodesic flows terminate at different IKs. The orange trajectory follows the energetically optimal branch, leading to IK solutions distinguishing from the others.

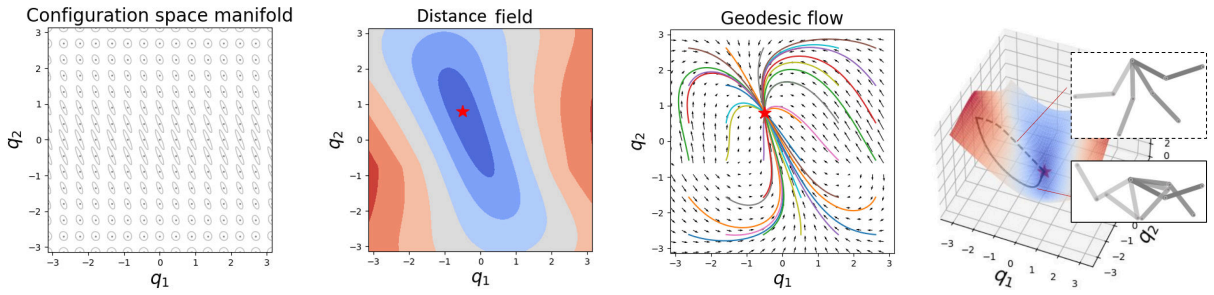


Figure 3. (a) Configuration space manifold endowed with a Riemannian metric using inertia as weighting matrix (visualized as isocontours of inverse matrices). The geodesics on this manifold correspond to minimal kinetic energy paths. By starting from a given point (red star), we can solve the eikonal equation on this manifold, accounting for the distance field (b) and gradient flow (c), which can then be used to backtrack geodesics in a very rapid manner (in milliseconds), see colored paths for examples of retrieved trajectories. Here, the source point is fixed for visualization. By using the proposed Neural Riemannian eikonal Solver (NES), these points are given as inputs, meaning that geodesics from any starting point to any final point are considered altogether. (d) Geodesic path (solid line) and Euclidean path (dashed line) on this manifold with corresponding robot motions.

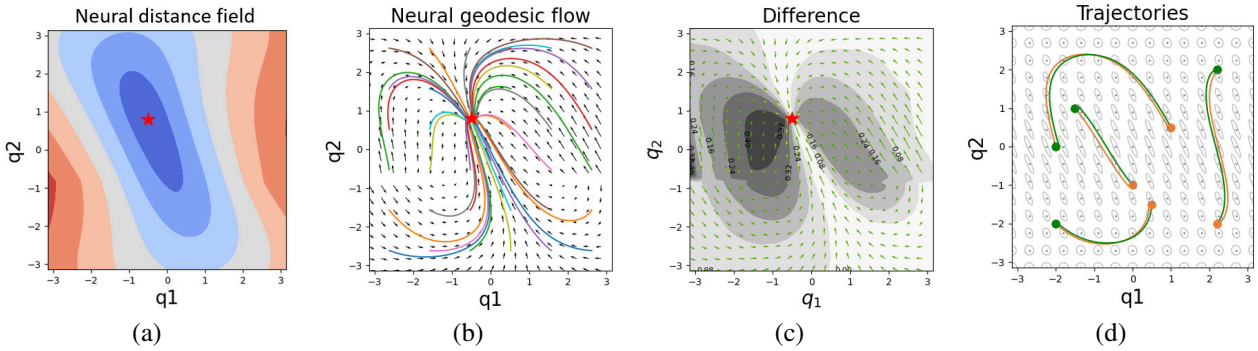


Figure 4. Solutions of Neural Riemannian Eikonal Solver (NES). (a) and (b) show the distance field and geodesic flow with the same parameters as Riemannian Fast Marching (RFM). (c) compares the difference with Figure 3 (b) (c), where geodesic flows produced by NES and RFM are shown in yellow and green, respectively. These three figures demonstrate that the neural network parameterization can solve the Riemannian eikonal equation, yielding results similar to those of RFM. (d) shows trajectories from source (green) to goal (orange) points and vice versa, highlighting the generalizability and symmetry of NES for arbitrary joint angle configuration pairs.

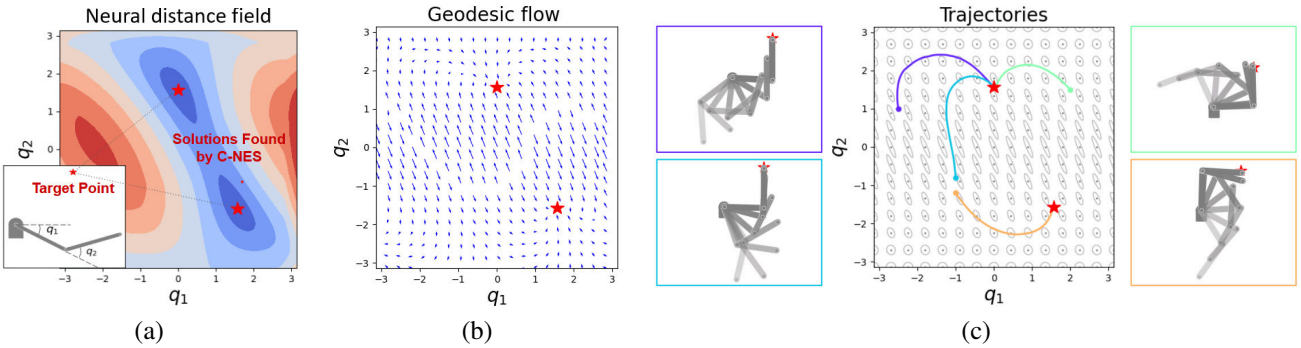


Figure 5. Given a kinetic-energy metric, (a) and (b) show the distance field and geodesic flow for the target position $(2.0, 2.0)$ in task space by using C-NES. Here, we do not specify the target joint angles (red stars). These joint angle targets are instead learned implicitly by the neural network. (c) shows four robot motions in task and configuration spaces (with different colors). We can observe that the motion solution for the task in orange color differs from the other three, which are automatically computed in accordance with the distance field and geodesic flow.

C–NES on the Riemannian Metric. A convenient instantiation of the metric-conditioning is to use the Jacobi metric derived from Hamiltonian dynamics that considers both kinetic and potential energy (Section 2.3.2). Specifically, we set the parameter in (28) to the total energy, i.e., $\omega = H$, and train our NES conditioned on this parameter. With this choice, C–NES represents distance fields and geodesic flows at fixed energy H by simply augmenting the network input with H, while keeping the training objectives unchanged. Varying H smoothly deforms the geometry through the conformal factor $2(H - P)$. As H increases, the conformal factor $2(H - P(\boldsymbol{q}))$ grows and

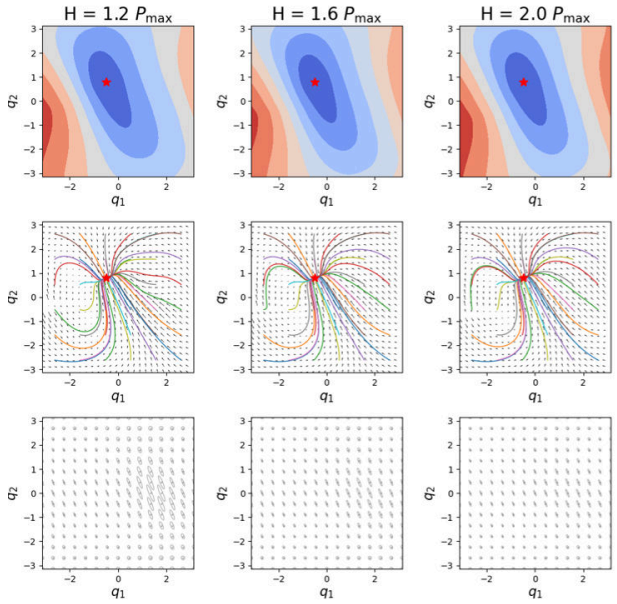


Figure 6. C–NES conditioned on the Jacobi metric parameterized by the total energy H. *Top*: distance fields. *Middle*: geodesics. *Bottom*: configuration-space manifolds.

$\mathbf{G}_{\mathrm{Jac}}(\boldsymbol{q}; H)$ more closely resembles a constant scaling of $\mathbf{M}(\boldsymbol{q})$. Accordingly, the resulting geodesics become closer to those induced by the kinetic–energy metric $\mathbf{G}_{\mathrm{ke}}$; in the idealized limit $H \to \infty$, they coincide. For smaller H, the available kinetic energy $T = H - P(\boldsymbol{q})$ is reduced, so the induced geodesics correspond to lower–energy motions. However, if H approaches $P(\boldsymbol{q})$ anywhere along a path, the conformal factor $2(H - P)$ tends to zero and the Jacobi metric becomes nearly singular, potentially causing numerical ill-conditioning. Figure 6 visualizes the distance fields, geodesics, and configuration–space manifolds for total energies $H \in \{1.2,\ 1.6,\ 2.0\}P_{\max}$, where $P_{\max}$ represents the maximum potential energy.

5.2 Comparison of Distance Field-based Approaches

We advocate solving the Riemannian eikonal equation to obtain a global, metrically consistent distance field from which geodesics are recovered by backtracking. To justify this choice, we compare NES against two baselines: (i) geodesic ray tracing (GRT) and (ii) a purely supervised learning (SL) regression of a distance field. In all methods, geodesics are extracted *a posteriori* by backtracking the resulting field.

Geodesic Ray Tracing (GRT). GRT is a grid-free baseline that traces geodesic rays outward from multiple sources and, recursively, from points visited along those rays (also known as wavefront construction). From each source, we sample unit directions on the Riemannian unit sphere and integrate the geodesic ODE at unit metric speed; each ray assigns an arc length to the points it traverses. The tentative distance at any location is the point-wise minimum of accumulated arc lengths over all rays; values at non-hit locations are interpolated from nearby ray samples. GRT has two main limitations compared to the eikonal formulation (Rawlinson and Sambridge 2005):

1. **Optimality and consistency.** The point-wise minimum over traced rays is a sampling-dependent lower envelope. The accumulated arc length from the source to a point x equals the true geodesic distance only up to the first conjugate point along that ray; beyond this point, the ray ceases to be minimizing and its length exceeds the minimal distance (Law 2021). Moreover, when directions are re-sampled at intermediate points, segments from different rays can be *spliced*; such stitched paths are not guaranteed to be globally optimal and, in general, do not satisfy the eikonal PDE. Therefore, this approach does not yield a single, globally consistent distance potential.

2. **Efficiency, coverage, and parallelization.** Achieving dense coverage requires fine angular/spatial sampling and repeated re-seeding from many intermediate points, which grows with the number of rays and integration steps (de Kool et al. 2006). Additionally, because new rays are computed iteratively based on previously traced rays, the method has sequential dependencies and data-dependent branching, causing warp divergence and irregular memory access, which hampers GPU efficiency (Aila and Laine 2009). In contrast, eikonal solvers (e.g., Fast Marching) compute a dense single-source distance field in a single monotone outward pass, after which geodesics are recovered by inexpensive backtracking (Kimmel and Sethian 1998).

Supervised Learning (SL). A direct alternative is to learn a pairwise distance map by generating ground-truth labels on the fly and minimizing the discrepancy between predicted and labeled distances, e.g.,

$$\mathcal{L}_{\mathrm{MSE}} = \mathbb{E}_{(q_s,q)}\left[\left(U_\theta(q_s,q) - \hat{d}(q_s,q)\right)^2\right],$$

where $\hat{d}(\cdot,\cdot)$ is obtained by a separate procedure (e.g., solving a two-point boundary-value problem or running geodesic front propagation/FMM to create ground-truth annotations). This approach yields a fast, continuously differentiable distance oracle at inference time, but has key drawbacks: (i) *computational cost*—label generation dominates training and scales with the number of source–query pairs; (ii) *coverage/bias*—supervision is limited to sampled pairs and inherits any inaccuracies or sparsity of the data; (iii) *no PDE structure*—the loss does not enforce the eikonal metric constraint; and (iv) *flow fidelity*—even with accurate distance labels, a neural regressor can produce different gradients, which can deflect paths and yield incorrect geodesic flow. In contrast, our NES eliminates distance labels and trains by enforcing the eikonal equation and boundary conditions, producing a globally consistent field whose gradients are directly suitable for reliable geodesic backtracking.

Evaluation. We evaluate GRT and SL on the same 2D Riemannian manifold with a kinetic–energy metric. Figure 7 shows distance fields (*left*) and backtracked geodesics (*right*), in comparison to NES (Figure 4a–b). For geodesic ray tracing, we initialize 64 rays from the source with random unit directions and integrate the geodesic ODE at unit metric

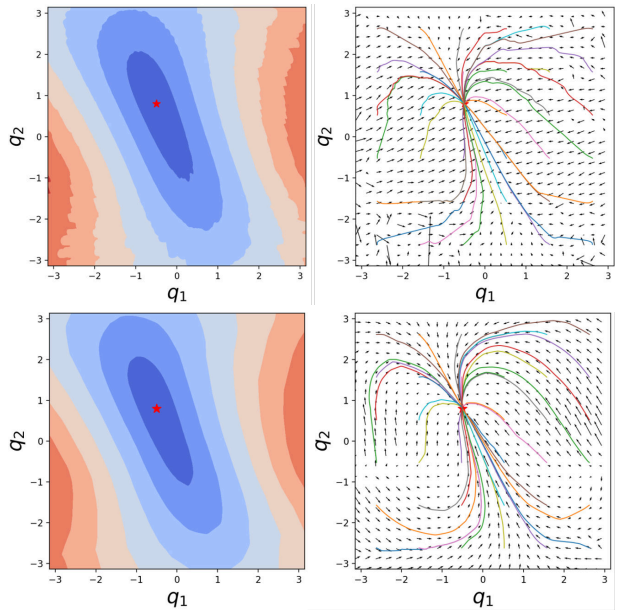


Figure 7. Distance field (*left*) and geodesics (*right*). *Top*: results obtained by the Geodesic ray tracing (GRT) baseline. *Bottom*: supervised regression trained on GRT labels.

Table 1. Comparison of distance-field-based approaches

Method	Geodesic Length	Runtime (offline/online)	Type
RFM	6.44 ± 2.93	**0.58s**/0.58s	Single-point
NES	6.42 ± 2.95	5min/**0.27s**	Source-goal
GRT	6.68 ± 3.11	30min/10.6s	Single-point
SL	6.47 ± 2.98	32min/0.27s	Single-point

speed. Every 15 integration steps, we re-seed 16 new rays at the wavefront (from the tips of active rays) and record their accumulated arc lengths. We continue until the total number of rays reaches 5,000. We then interpolate these samples and take the pointwise minimum across all rays to form the tentative distance field. Despite dense sampling, the field exhibits artifacts that induce non-smooth geodesics and inconsistent gradients (*top row*). For supervised learning, we use the GRT field as supervision and train a neural regressor to produce a continuous distance map. This reduces visible artifacts and yields smoother flows (*bottom row*), but inherits GRT's coverage/bias and requires costly label generation.

Results. Table 1 summarizes geodesic lengths backtracked from the learned distance fields and computation times—split into offline (data collection/training) and online (trajectory integration). RFM and NES recover near-optimal lengths with low online times. GRT yields longer paths and slower inference despite substantial offline cost. SL improves geodesic length and matches NES's online speed, but requires even more offline time for label generation and training; the complexity of data generation further limits it to single-point eikonal formulations. Overall, eikonal methods produce dense, metrically consistent fields from which geodesics can be reliably backtracked. NES is preferred over SL because it *enforces the eikonal PDE in the neural network during training* rather than regressing to sampled labels; this avoids costly label generation, reduces sampling bias, and imposes a principled, physics-based structure that promotes globally consistent distance fields and stable geodesic backtracking.

5.3 NES for Other Robot Applications

In addition to minimum–energy motions, we present three further applications that illustrate the generalization capability of NES across broader robotic problems. These tasks instantiate different Riemannian geometries discussed in Section 2.3, including pullback metrics, stability–aware energy shaping, and obstacle avoidance. In all cases, the eikonal is solved on the chosen geometry, and geodesics are recovered by backtracking as in the previous sections.

5.3.1 Task–Space Distance Minimization We apply our Riemannian distance field framework to generate motions that minimize *task–space* displacement. Using the pullback construction from Section 2.3 with the Euclidean task metric defined by $\mathbf{G}_{\mathcal{X}} = \mathbf{I}$, the configuration–space metric is defined by

$$\mathbf{G}_{\text{pb}}(\mathbf{q}) \;=\; \mathbf{J}(\mathbf{q})^{\top}\,\mathbf{J}(\mathbf{q}).$$

The task–space speed induced by a joint velocity $\dot{\mathbf{q}}$ is

$$\|\dot{\mathbf{x}}\| \;=\; \|\mathbf{J}\dot{\mathbf{q}}\| \;=\; \sqrt{\dot{\mathbf{q}}^{\top}\mathbf{J}^{\top}\mathbf{J}\dot{\mathbf{q}}} \;=\; \|\dot{\mathbf{q}}\|_{\mathbf{J}^{\top}\mathbf{J}}. \tag{30}$$

The minimum–length task–space path corresponds to geodesics under this Riemannian metric, which encodes how joint–space motions translate to task–space displacements. This formulation minimizes the accumulated task–space displacement, which is desirable for applications such as reaching, drawing, or tool usage, in which end–effector path length matters more than joint–space path length. The Riemannian formulation ensures that among all possible paths connecting two configurations, the one that minimizes task–space travel distance is selected. Note that the task–space paths are not necessarily straight lines, especially for the non-redundant manipulators. This contrasts with task–space interpolation or null–space control, which enforces Cartesian line segments or waypoints that may be kinematically infeasible. An illustration of this task is provided in Figure 8, which depicts the Riemannian manifold, the geodesic distance field and flow, as well as the resulting robot motions using the same 2D planar robot. This formulation can be further extended to incorporate task–specific metrics by choosing $\mathbf{G}_{\mathcal{X}} \succ 0$ to prioritize motions along specific task–space directions.

5.3.2 Stability–Aware Energy Shaping We embed passive stability directly into the planning geometry for a pendulum–on–cart system with two degrees of freedom: the cart position x and the pendulum angle q, as illustrated in Figure 9 (*top-left*).

As introduced in Section 2.3, we define a position-dependent Riemannian metric that scales inertia with normalized potential using (17). Since the state variables include both cart translation x and pole orientation q, we define

$$\begin{aligned} \mathbf{G}_{\text{sta}}(x,q) &= \mathbf{M}(x,q)\left[1+\epsilon\tilde{P}(x,q)\right], \\ \tilde{P}(x,q) &= \frac{P(x,q)-P_{\min}}{P_{\max}-P_{\min}} \in [0,1]. \end{aligned} \tag{31}$$

A detailed description is provided in Section 2.3. Figure 9 (*top–right*) shows three motions starting from different initial states, all computed under the same energy-shaped metric ($\epsilon = 2.0$) and same eikonal formulation. As the required

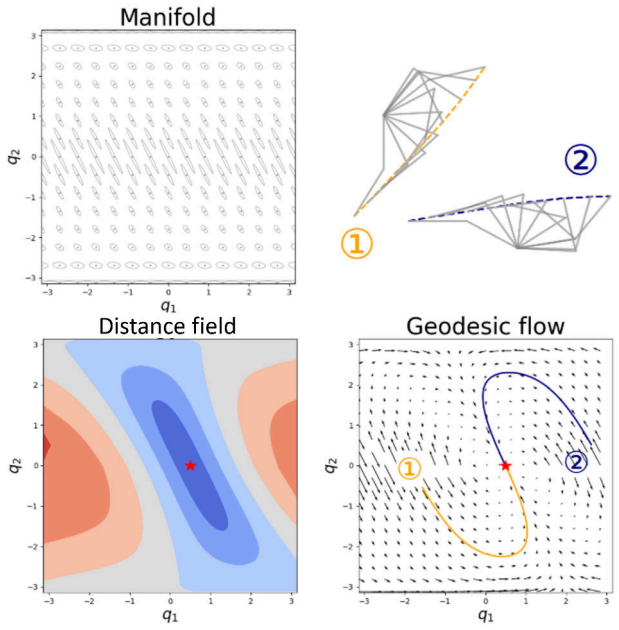


Figure 8. Illustration of task-space distance minimizing motion generation. Geodesics on the Riemannian manifold defined by the metric $\mathbf{G}(\mathbf{q}) = \mathbf{J}^\top \mathbf{J}$ correspond to motions minimizing end-effector displacement in task space, solved by the Riemannian eikonal equation.

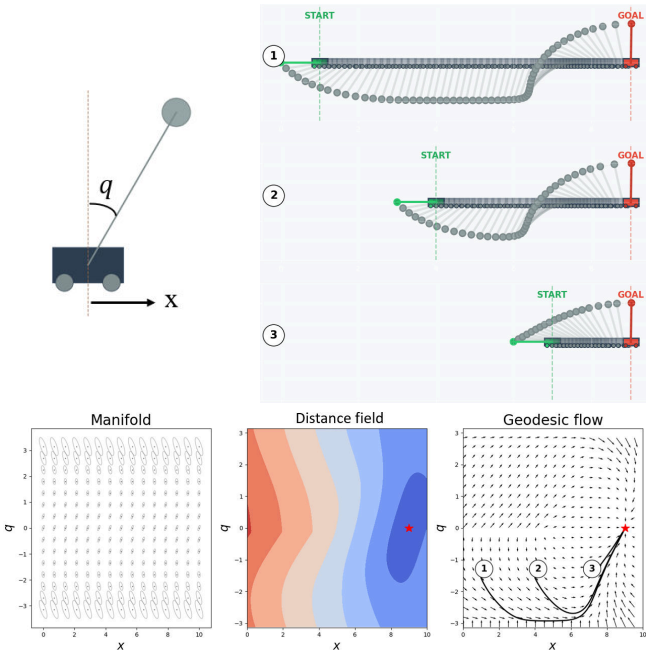


Figure 9. Stability-aware motion generation for the pendulum-on-cart system. *Top-left*: configuration space of the system. *Top-right*: representative trajectories from distinct initial conditions, computed via backtracking geodesic flow without tuning. *Bottom*: the Riemannian metric, energy field, and induced geodesic flow for a fixed goal.

cart translation increases, the geodesic flow relies more on low–potential configurations: long moves lower the pole and translate near the down configuration before lifting at the goal; moderate moves lower it partially; short moves stay near upright. This behavior requires no mode switching or heuristics, because we define the Riemannian metric inflates distances in high–energy regions, trajectories naturally dwell in low–energy basins and traverse high–energy postures only when necessary. The corresponding manifold, distance field, and geodesic flow are shown in Figure 9 (*bottom*).

5.3.3 Collision avoidance Eikonal-based distance fields and geodesic flows handle collision avoidance by making obstacles impassable in the PDE. In the isotropic form (4), this can be modeled by assigning zero-speed velocity to points inside obstacles $\mathcal{O}$. This yields global optimal paths that respect hard collision constraints and have shown promising results in motion planning (Ni and Qureshi 2022, 2024b). The same idea extends from isotropic to anisotropic settings by replacing the Euclidean norm with the Riemannian metric. Figure 10 illustrates the distance field (a) and flows/paths (b) on the same configuration–space manifold endowed with the kinetic–energy metric described in Section 5.1, now with obstacles present. Solving the eikonal equation produces a global distance field and corresponding optimal paths without additional asymptotic computational complexity.

However, because proximity to obstacles is not explicitly encoded, the resulting paths can exhibit vanishing clearance, which reduces robustness and elevates collision risk. To address this limitation, we replace the hard, non-smooth boundary (i.e., zero-speed zones) with a smoothly varying weight matrix that continuously reflects proximity to obstacles, using the Riemannian eikonal formulation described in Section 2.3. This formulation increases the cost

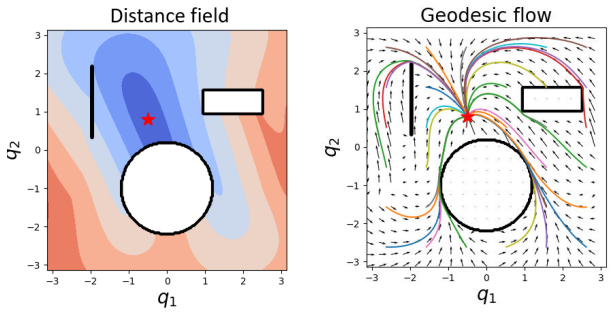
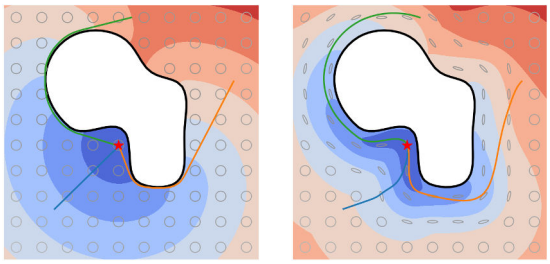


Figure 10. Solution of the Riemannian eikonal equation in case of obstacles (black contours). (a) distance field. (b) Geodesic flow.

of moving toward obstacles while preserving flexibility in tangential directions. As a result, paths naturally bend around obstacles, effectively maintaining smooth and safe clearance. Figure 11 compares paths generated by the standard isotropic eikonal formulation (a) and our Riemannian manifold-based approach (b). The latter produces trajectories that are smoother with a more natural usage of obstacle proximity information.

6 Experiments

We evaluate *NES* on motion planning problems to demonstrate its ability to generate *dynamics-aware, energy-efficient* trajectories—a long-standing challenge in robotics. From a Riemannian-geometry viewpoint, Lagrangian and Hamiltonian mechanics induce the kinetic-energy metric and the Jacobi metric respectively (Section 2.3). NES seeks geodesics under these metrics, yielding paths that respect system dynamics while reducing energy. The



(a) (b)

Figure 11. Obstacle-avoidance comparison. (a) Trajectory generated by the standard isotropic eikonal equation. (b) Trajectory generated using a Riemannian eikonal formulation.

underlying robot dynamics on configuration manifolds and the connection between geodesics and optimal control are summarized in Appendix A.

The experiments are designed to investigate the following key questions:

- **Q1:** How effective and efficient is NES in computing distance fields and geodesic paths on Riemannian manifolds?
- **Q2:** How does our method compare to commonly used approaches for computing geodesic paths?
- **Q3:** In which ways can the geodesic path on a Riemannian manifold account for reduced control inputs?
- **Q4:** How well can our neural Riemannian eikonal solver scale to high-dimensional robot manipulation problems?
- **Q5:** How effective is our C-NES approach at propagating wavefronts from joint space to task space?
- **Q6:** How easily can our approach be integrated into other motion optimization frameworks to facilitate dynamics-aware motion planning?

To address these questions, we conducted a series of experiments in both simulation and real-world settings by considering two robot manipulators: a 2-DoF planar robot described in Section 5, and a 7-axis Franka robot. The inertial mass matrix and potential energy of this robot are derived from the robot's Unified Robot Description Format (URDF) file using the Composite Rigid Body Algorithm (Walker and Orin 1982), and we modified the implementation by (Johannessen et al. 2019) to enable batch computation in PyTorch.

6.1 Baselines

Computing geodesics on a Riemannian manifold is challenging due to the metric's nonlinearity and anisotropy. In Section 5, we demonstrate the effectiveness of NES for computing distance fields and geodesic flows, and we compare it with other distance–field–based methods. Here, we further compare NES with classical approaches that directly compute geodesics.

Geodesic Shooting (GS): This method solves an initial value problem by numerically integrating the geodesic equation forward in time, starting from a given initial configuration and velocity. A key characteristic of GS is that it does not require explicit knowledge of the goal configuration in advance. Instead, it relies on choosing an initial velocity vector that heuristically points approximately toward the desired target. That differs from our NES method, which directly finds the geodesic path between start and goal pairs.

Optimal Control (OC): This approach formulates geodesic computation as a boundary value problem and solves it through iterative optimization techniques. Specifically, we optimize the objective:

$$\begin{aligned} \min_{\{\boldsymbol{u}_\zeta\}} \quad & \|\boldsymbol{q}_N - \boldsymbol{q}_s\|_Q^2 + r \sum_{\zeta=1}^{N-1} \boldsymbol{u}_\zeta^\top \boldsymbol{G}(\boldsymbol{q}_\zeta)\boldsymbol{u}_\zeta \\ \text{subject to} \quad & \boldsymbol{q}_\zeta = \boldsymbol{q}_{\zeta-1} + \boldsymbol{u}_\zeta \, d\zeta, \quad \zeta = 1, \dots, N \\ & dt = \sqrt{\boldsymbol{u}_\zeta^\top \boldsymbol{G}(\boldsymbol{q}_\zeta)\boldsymbol{u}_\zeta} \, d\zeta, \end{aligned} \tag{32}$$

where $\boldsymbol{q}_\zeta$ is the system state indexed by the phase parameter ζ, which serves as a reparameterization of time along the trajectory corresponding to the arc-length under the Riemannian metric defined by $\boldsymbol{G}$. The cost minimizes the weighted squared distance between the final state $\boldsymbol{q}_N$ and the target $\boldsymbol{q}_s$ along with a regularization term on control effort weighted by $\boldsymbol{G}(\boldsymbol{q}_\zeta)$. The dynamics constraint describes the discrete evolution of states with respect to ζ. The time reparameterization equation connects the physical time increment dt to the control magnitude measured by the energy metric, ensuring that ζ parameterizes the trajectory consistently with the system's geometry and temporal evolution. We refer to this baseline as the *constant energy path* (CEP), as the optimal control approach seeks the shortest geodesics on the configuration space manifold defined by the Jacobi metric, representing constant-energy motion.

A brief overview of these baselines and our approach is listed in Table 2. In contrast to GS and OC, which rely on numerical optimization or iterative integration and are often sensitive to initialization, our NES learns a differentiable approximation to the geodesic distance field. Once the model is trained, it can be used to query distances and geodesic flows efficiently and can be generalized to arbitrary start-to-goal joint configuration pairs. NES provides a closed-form policy by evaluating the analytic gradient of the learned field, enabling fast and generalizable geodesic inference across the configuration space. Unlike OC-based methods, which are typically computationally expensive and less flexible for real-time control, NES supports dynamic, efficient, and scalable geodesic computation.

6.2 Control Strategies

In our experiments, we exploit a key property of NES: rather than solving a new geodesic problem for each start–goal pair, it learns offline a continuous, differentiable distance field on the configuration manifold. At runtime, we query the distance field and its analytic gradient efficiently to recover geodesic directions, enabling high rate in feedback loops. Thus, in practice, NES is not only a geodesic solver but also a compact policy/value representation that integrates naturally with standard controllers to handle constraints. In other

Table 2. Comparison of Geodesic Computation Methods

Criterion	Optimal Control	Geodesic Shooting	Neural Eikonal Solver (Ours)
Output	Policy / Path	Policy / Path	Field + Policy / Path
Optimality	Local	Local	Approx. Global
Online Cost	High	Low	Low
Init. Sensitivity	High	High	Low
Flexibility	Low	High	High
Differentiability	Indirect	Indirect	Explicit

words, NES provides high-level guidance on the manifold via the geodesic flow, while a separate low-level controller enforces actuation limits, kinematic/dynamic constraints, and tracking accuracy. This separation preserves constraint awareness encoded in the metric and distance field, and affords flexibility and adaptability in dynamic environments.

We observed that directly following the learned geodesic flow $-\boldsymbol{V}(\boldsymbol{q})$ can lead to numerical instability, especially when $\boldsymbol{q}$ is close to the target and $U(\boldsymbol{q})$ approaches zero. To ensure stable and robust convergence near the target in our tests, we define a configuration-dependent desired joint velocity $\dot{\boldsymbol{q}}_{\text{des}}(\boldsymbol{q})$ by blending the learned geodesic flow direction $-\boldsymbol{V}(\boldsymbol{q})$ with a simple linear vector $(\boldsymbol{q}_s - \boldsymbol{q})$ pointing directly toward $\boldsymbol{q}_s$. The blending weight $\lambda(\boldsymbol{q})$ is dynamically adjusted based on $U(\boldsymbol{q})$, allowing a smooth transition between global and local guidance:

$$\begin{aligned} \lambda(\boldsymbol{q}) &= \frac{\beta\, U(\boldsymbol{q})}{1+\beta\, U(\boldsymbol{q})}, \\ \dot{\boldsymbol{q}}_{\text{des}}(\boldsymbol{q}) &= -\lambda(\boldsymbol{q})\, \boldsymbol{V}(\boldsymbol{q}) + \big(1-\lambda(\boldsymbol{q})\big)\,(\boldsymbol{q}_s - \boldsymbol{q}). \end{aligned} \tag{33}$$

Here $\beta > 0$ modulates the sensitivity to $U(\boldsymbol{q})$. When the robot is far from the target and $U(\boldsymbol{q})$ is large, $\lambda(\boldsymbol{q}) \approx 1$ and the motion follows $-\boldsymbol{V}(\boldsymbol{q})$; as $U(\boldsymbol{q})$ becomes small, $\lambda(\boldsymbol{q}) \to 0$ and the attraction term ensures stable convergence.

Below, we illustrate representative integrations used in our experiments, including kinematic and kinodynamic QP tracking and task-priority control via null-space projection. Other approaches, such as model predictive control (MPC) (Koptev et al. 2024) can also be exploited.

6.2.1 Quadratic Programming (QP) The blended velocity $\dot{\boldsymbol{q}}_{\text{des}}(\boldsymbol{q})$ specifies an instantaneous target motion. We compute a feasible command that respects limits and dynamics by solving a QP. We give kinematic (velocity) and kinodynamic (torque/acceleration) forms.

Kinematic QP Formulation At the joint velocity level, the control command is the joint velocity $\dot{\boldsymbol{q}} \in \mathbb{R}^n$. This QP is formulated to compute the feasible joint velocity $\dot{\boldsymbol{q}}$ that closely tracks the blended desired velocity $\dot{\boldsymbol{q}}_{\text{des}}(\boldsymbol{q})$, while satisfying joint velocity and position limits. The objective is to minimize the weighted squared error between the actual and desired velocities, with the cost function defined by a symmetric positive semi-definite matrix H:

$$\begin{aligned} \min_{\dot{\boldsymbol{q}} \in \mathbb{R}^n} \quad & (\dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}_{\text{des}}(\boldsymbol{q}))^\top \boldsymbol{Q} (\dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}_{\text{des}}(\boldsymbol{q})), \\ \text{s.t.} \quad & \dot{\boldsymbol{q}}_{\min} \le \dot{\boldsymbol{q}} \le \dot{\boldsymbol{q}}_{\max}, \\ & \boldsymbol{q}_{\min} \le \boldsymbol{q} + \Delta t \dot{\boldsymbol{q}} \le \boldsymbol{q}_{\max}. \end{aligned} \tag{34}$$

The weighting matrix $\boldsymbol{Q}$ allows prioritizing tracking accuracy across different joints. Joint position and velocity limits are applied to constrain the motion within the robot's physical limits.

Kinodynamic QP Formulation At the joint torque level, where the control input is $\boldsymbol{\tau} \in \mathbb{R}^n$, it is necessary to incorporate the robot's dynamics, typically modeled as $\boldsymbol{\tau} = \mathbf{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q})$. To utilize the desired velocity $\dot{\boldsymbol{q}}_{\text{des}}(\boldsymbol{q})$, we first compute a nominal desired joint acceleration $\ddot{\boldsymbol{q}}_{\text{des}}$ aimed at tracking the desired velocity over time:

$$\ddot{\boldsymbol{q}}_{\text{des}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \frac{\dot{\boldsymbol{q}}_{\text{des}}(\boldsymbol{q}) - \dot{\boldsymbol{q}}}{\Delta t}, \tag{35}$$

where Δt is the control time step. A nominal desired torque $\boldsymbol{\tau}_{\text{des}}$ required to achieve the target acceleration is computed using inverse dynamics:

$$\boldsymbol{\tau}_{\text{des}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}_{\text{des}}) = \mathbf{M}(\boldsymbol{q})\ddot{\boldsymbol{q}}_{\text{des}} + \mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}). \tag{36}$$

We formulate a kinodynamic QP with both joint torque $\boldsymbol{\tau}$ and joint acceleration $\ddot{\boldsymbol{q}}$ as decision variables. This formulation allows us to explicitly include the robot's dynamics as an equality constraint while minimizing the deviation between the applied torque $\boldsymbol{\tau}$ and the nominal desired torque $\boldsymbol{\tau}_{\text{des}}$. The resulting QP is defined as:

$$\begin{aligned} \min_{\substack{\boldsymbol{\tau} \in \mathbb{R}^n \\ \ddot{\boldsymbol{q}} \in \mathbb{R}^n}} \quad & (\boldsymbol{\tau} - \boldsymbol{\tau}_{\text{des}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}_{\text{des}}))^\top \boldsymbol{Q} (\boldsymbol{\tau} - \boldsymbol{\tau}_{\text{des}}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}_{\text{des}})), \\ \text{s.t.} \quad & \boldsymbol{\tau} - \mathbf{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} = \mathbf{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}), \\ & \boldsymbol{\tau}_{\min} \le \boldsymbol{\tau} \le \boldsymbol{\tau}_{\max}, \\ & \ddot{\boldsymbol{q}}_{\min} \le \ddot{\boldsymbol{q}} \le \ddot{\boldsymbol{q}}_{\max}, \\ & \dot{\boldsymbol{q}}_{\min} \le \dot{\boldsymbol{q}} + \Delta t \ddot{\boldsymbol{q}} \le \dot{\boldsymbol{q}}_{\max}, \\ & \boldsymbol{q}_{\min} \le \boldsymbol{q} + \Delta t \dot{\boldsymbol{q}} + \tfrac{1}{2}\Delta t^2 \ddot{\boldsymbol{q}} \le \boldsymbol{q}_{\max}. \end{aligned} \tag{37}$$

In this formulation, $\boldsymbol{Q}$ is a symmetric positive semi-definite weighting matrix for the torque error objective. The constraints enforce the robot's dynamics, as well as joint torque, acceleration, velocity, and position limits. Future joint velocities and positions are approximated using first- and second-order Euler integration, respectively. The kinodynamic QP finds the optimal feasible torque and acceleration pair that satisfies the robot's dynamics and physical limits while minimizing the weighted deviation from the nominal torque required to achieve the desired acceleration derived from the blended velocity $\dot{\boldsymbol{q}}_{\text{des}}(\boldsymbol{q})$, offering a simple yet effective framework for integrating the learned geodesic flows into real-time control strategies. Furthermore, this framework is extensible and can incorporate additional constraints, such as collision avoidance (Li et al. 2024b; Koptev et al. 2022).

6.2.2 Task Prioritization In practice, while QP-based controllers offer flexibility for optimizing multiple objectives simultaneously, they can yield suboptimal results when objectives conflict or require strict prioritization. In weighted optimization schemes, improper tuning of weights can lead

to violations of critical tasks. Task prioritization addresses this limitation by enforcing a strict hierarchy: higher-priority tasks are satisfied strictly, while lower-priority behaviors are projected into the null space of higher-priority constraints. This method is effective for simultaneous objective achievement, such as maintaining end-effector orientation while avoiding obstacles (Khatib 1987; Ratliff et al. 2018).

Here we integrate the geodesic flow policies learned by NES into a task prioritization framework to enhance robotic performance while strictly satisfying the primary task. Typically, this policy is integrated as a lower-priority task, contributing to motion generation without violating higher-priority constraints. Although projecting into the null space may compromise some global optimality of the learned policy, the resulting motion remains near-optimal by leveraging the robot's residual redundancy.

For a constraint manifold defined by $f(\boldsymbol{q})$, the joint velocity $\dot{\boldsymbol{q}}$ must satisfy:

$$\boldsymbol{J}_f(\boldsymbol{q})\dot{\boldsymbol{q}} = \boldsymbol{0}, \tag{38}$$

where $\boldsymbol{J}_f(\boldsymbol{q}) \triangleq \frac{\partial f}{\partial \boldsymbol{q}}$ is the constraint Jacobian. This ensures all motion remains tangent to the constraint surface. The desired geodesic flow $\dot{\boldsymbol{q}}_{\text{des}}$ is projected into the constraint null space:

$$\dot{\boldsymbol{q}}_{\text{TP}} = \boldsymbol{N}_f \dot{\boldsymbol{q}}_{\text{des}}, \tag{39}$$

where $\boldsymbol{N}_f \triangleq \mathbf{I} - \boldsymbol{J}_f^{\dagger}\boldsymbol{J}_f$ is the null-space projector, $\boldsymbol{J}_f^{\dagger}$ is the Moore–Penrose pseudoinverse of $\boldsymbol{J}_f$, and $\dot{\boldsymbol{q}}_{\text{TP}}$ is the task-prioritized velocity. By integrating our geodesic flow policy through null-space projection, we achieve simultaneous constraint satisfaction and trajectory optimization. The framework ensures strict adherence to constraints $\boldsymbol{J}_f(\boldsymbol{q})\dot{\boldsymbol{q}} = \boldsymbol{0}$ while preserving the optimized trajectory in the remaining degrees of freedom. As a result, the robot's motion retains the optimality properties of the original policy wherever feasible, while ensuring that all specified constraints are strictly satisfied.

6.3 Training Details and Evaluation Metrics

We use a multilayer perceptron (MLP) as the backbone of our NES to train the neural Riemannian eikonal solver for the 7-DoF Franka robot. The MLP takes concatenated source-to-goal pairs as input and outputs a scalar Riemannian distance U, and its gradient is obtained analytically via automatic differentiation in PyTorch. To better capture the local geometric structure of the configuration space manifold, we utilize a geometry-aware sampling strategy—specifically, the Riemannian Manifold Metropolis-Adjusted Langevin Algorithm (RM-MALA)—to generate input point pairs that lie on the manifold. A detailed explanation of this sampling method is provided in Appendix D. During training, we randomly sample 50,000 joint configurations $\boldsymbol{q}_e$ at each epoch as wavefront boundary points, drawn from the distribution produced by the RM-MALA algorithm. The model is trained for up to 10^5 epochs using the Adam optimizer with a learning rate of 0.001, terminating early upon convergence. Training takes approximately 5 minutes for the 2-DoF planar robot and around four hours for the 7-DoF Franka robot, using a single NVIDIA RTX 3090 GPU.

We evaluate our approach using the following metrics:

Geodesic Length: The geodesic length is computed via (3), which is a basic metric to measure the quality of the constructed distance field.

Total Torque: We measure control effort as

$$\|\boldsymbol{\tau}\|_{\boldsymbol{R}} \triangleq \Big(\int_0^T \boldsymbol{\tau}^\top \boldsymbol{R}\boldsymbol{\tau} \, \mathrm{d}t \Big)^{1/2}. \tag{40}$$

We report two variants: (i) *norm torque* with $\boldsymbol{R} = \mathbf{I}$ (standard ℓ^2 norm); and (ii) *active torque* with $\boldsymbol{R} = \dot{\boldsymbol{q}}\dot{\boldsymbol{q}}^\top$, whose integrand is $(\boldsymbol{\tau}^\top \dot{\boldsymbol{q}})^2$, indicating torque that does mechanical work). Ideally, for geodesics under the Jacobi metric, the motion is energy-conserving, so no active work is required, and this term vanishes; see Appendix A.3 for a detailed discussion.

Computation Time: We measure the total computation time required to generate control inputs and produce the full trajectory.

6.4 Results

6.4.1 Dynamics-aware motions generation on configuration space manifold We first evaluate dynamics-aware, energy-efficient motion generation on the configuration-space manifold. Specifically, we compare NES against baselines in computing minimum-length paths on the energy-conserving manifold induced by the Jacobi metric. As discussed in Section 5.1, we set the total energy to $H = 1.2\,P_{\max}$ to remain in a low-energy regime while avoiding singularities. Table 3 reports results for both 2D and 7D robotic systems. We randomly sample 100 start-goal pairs for geodesic computation and compare NES with Geodesic Shooting (GS), Optimal Control (OC), and RFM. For NES, we evaluate two tracking controllers described in Section 6.2: a velocity-based QP controller (Vel) and a torque-based QP controller (Tau) for following the planned trajectories. From these experiments, we extract the following key insights:

Energy-efficient motions via geodesic planning (Q1, Q2): In the 2D planar simulation, NES achieves comparable geodesic lengths and lower torque consumption than RFM, and significantly outperforms both GS and OC in terms of active torque and total control effort. In the 7D case, NES generates the shortest geodesics and lowest active torques among all evaluated methods. These results validate our formulation: instead of solving dynamics-aware motion planning via trajectory optimization, we compute geodesics on a configuration-space manifold shaped by the robot's energy structure. This leverages the geometry of robot dynamics to find globally optimal solutions.

Relation to minimum control effort (Q3): Our method consistently produces trajectories with lower active torque consumption by planning paths along geodesics on the energy manifold. This further results in reduced ℓ^2 torque, a standard metric in optimal control for measuring the control effort. These results align with our theoretical analysis in Appendix A.3, showing that geodesic paths on the energy-conserving manifold often correlate with lower control effort. Notably, this performance is achieved without explicitly optimizing torque-based objectives, demonstrating that our geometric formulation inherently promotes energy-efficient behaviors.

Decoupling planning and control for scalability and robustness (Q2, Q4, Q6): A key advantage of our framework is the clear separation between motion planning and control. Traditional optimal control tightly couples these components, solving high-dimensional nonlinear trajectory optimization problems at runtime while satisfying system constraints at every timestep. This approach is not only computationally expensive but also highly sensitive to initialization, often resulting in suboptimal solutions due to local minima. In contrast, our NES precomputes a global distance field and corresponding geodesic paths over the configuration space manifold. These geodesics encode dynamic feasibility and energy optimality, serving as reference trajectories. The control task is then reduced to a lightweight tracking problem solved by standard QP-based velocity or torque controllers. This decoupling enables real-time execution and robustness to disturbances. As evidenced in Table 3, it substantially improves computational efficiency, making our method scalable to high-DoF systems and adaptable in reactive scenarios.

Flexibility and adaptability (Q1): Unlike RFM, which relies on grid-based discretization, NES learns a continuous and differentiable representation of the distance field. RFM must solve the eikonal equation from scratch using iterative wavefront propagation for each new start point. This process severely limits its scalability and computational efficiency. In contrast, NES supports parallel computation over batch inputs, enabling rapid generalization across a wide range of initial conditions. It also avoids interpolation errors and scales effectively to high-dimensional configuration spaces. As shown in Table 3, NES performs reliably on high-dimensional platforms where RFM fails.

A comprehensive comparison of our approach and baselines is shown in Figure 12 (**Q2**). The ground truth trajectory is derived from our RFM approach. In (a), we observe that both NES and optimal control successfully approximate the energy-optimal path if the start and goal points are close to each other. However, if the start and goal points are far apart, as shown in (b), the optimal control method becomes trapped in a local minimum, while our NES approach still converges to the near-optimal solution. In both cases, geodesic shooting finds local minimum solutions. Figure 12(c) further highlights NES's robustness to disturbances. Thanks to its efficient computation and generalization ability, NES adapts to varying start-to-goal conditions and maintains reliable performance under the disturbance.

6.4.2 Correlation with Minimum-Torque Control To further verify the connection between geodesics on the energy-conserving manifold and the minimum control-effort problem (**Q2, Q3**), we consider an additional optimal control problem that minimizes torque:

$$\begin{aligned} \min_{\{\boldsymbol{u}_t\}} \quad & \|\boldsymbol{q}_T - \boldsymbol{q}_g\|^2_{\boldsymbol{Q}} + \frac{r}{2}\sum_{t=1}^{T-1} \boldsymbol{u}_t^\top \boldsymbol{R}\, \boldsymbol{u}_t, \\ \text{s.t.} \quad & \boldsymbol{q}_{t+1} = FD(\boldsymbol{q}_t, \boldsymbol{u}_t), \end{aligned} \tag{41}$$

where $FD(\boldsymbol{q}_t, \boldsymbol{u}_t)$ denotes the system dynamics derived from (47), and $\boldsymbol{Q} \succeq 0$, $\boldsymbol{R} \succ 0$ are weighting matrices. We refer to this formulation as the *minimum-torque* (MT) problem. Unlike previous baselines, this formulation does not strictly enforce the Riemannian manifold constraints; instead, it serves as a proof of concept to illustrate the relationship between geodesic motions on the energy-conserving manifold and the minimization of control effort.

We compare NES with a joint-torque controller as the low-level actuator against this optimal-control formulation that penalizes control effort. Qualitative comparisons are presented in Figure 13. Our NES approach consistently generates trajectories that closely follow geodesics on the energy-conserving manifold, resulting in paths that are qualitatively similar to those obtained from the minimum-torque solution computed via the optimal-control formulation (as illustrated in Figure 13-*left*). This observation supports using geodesics on the energy-conserving manifold as a geometric prior for optimal-control objectives that penalize control effort. Although friction, modeling/tracking errors, and boundary-condition constraints introduce non-conservative energy expenditure, the geodesic path still provides a principled heuristic that serves as a high-quality initial guess close to the optimal solution. In Figure 13-*right*, NES outperforms the minimum-torque solution obtained from optimal control. A critical limitation of the optimal-control formulation is its reliance on manual tuning of the weighting matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$, which introduces subjectivity and can yield suboptimal solutions if not tuned carefully. In contrast, NES derives a control policy inherently aligned with the system's dynamics, enabling adaptive and efficient energy use without heuristic parameter tuning.

6.4.3 Conditioned on Task-space Boundaries We further evaluate the performance of our proposed C-NES method in mapping joint space boundaries to task space (**Q5**). For the planar robot, we consider only the end-effector position. For the Franka robot, NES is extended to handle position and orientation separately, with:

$$\begin{aligned} U^{\text{pos}}_{\boldsymbol{\theta}}(\boldsymbol{x}_s, \boldsymbol{q}_e) &= \|f^{\text{pos}}(\boldsymbol{q}_e) - \boldsymbol{x}^{\text{pos}}_s\|\ \sigma\big(u^{\text{pos}}_{\boldsymbol{\theta}}(\boldsymbol{x}_s, \boldsymbol{q}_e)\big), \\ U^{\text{ori}}_{\boldsymbol{\theta}}(\boldsymbol{x}_s, \boldsymbol{q}_e) &= \arccos\left(f^{\text{ori}}(\boldsymbol{q}_e)^\top \boldsymbol{x}^{\text{ori}}_s\right)\ \sigma\big(u^{\text{ori}}_{\boldsymbol{\theta}}(\boldsymbol{x}_s, \boldsymbol{q}_e)\big), \end{aligned} \tag{42}$$

where $\|f^{\text{pos}}(\boldsymbol{q}_e) - \boldsymbol{x}^{\text{pos}}_s\|$ denotes the Euclidean distance in $\mathbb{R}^3$, and $\arccos\left(f^{\text{ori}}(\boldsymbol{q}_e)^\top \boldsymbol{x}^{\text{ori}}_s\right)$ corresponds to the geodesic distance on the $\mathcal{S}^3$ manifold. Here, $f^{\text{ori}}(\boldsymbol{q}_e)$ and $\boldsymbol{x}^{\text{ori}}_s$ are the unit quaternions representing the orientation. The total loss is defined as the sum of the losses and the gradient flow is computed as a linear combination of gradients from $U^{\text{pos}}_{\boldsymbol{\theta}}$ and $U^{\text{ori}}_{\boldsymbol{\theta}}$, ensuring that both position and orientation contribute to the result.

A qualitative visualization of the planar robot is shown in Figure 5. In this example, the target position of the end-effector is set to $\boldsymbol{x}_s = (2.0, 2.0)$, as shown in (a). This inverse kinematics problem has two possible solutions for the joint angles: $q_1 = 0, q_2 = 1.57$ and $q_1 = 1.57, q_2 = -1.57$. Rather than explicitly solving the inverse kinematics problem, our approach generates the geodesic flow that iteratively guides the solution towards the minimal geodesic length, as shown in (a) and (b). In (c), we visualize four different robot trajectories that all reach the same end-effector position but originate from different joint configurations. Due to the system's redundancy, multiple

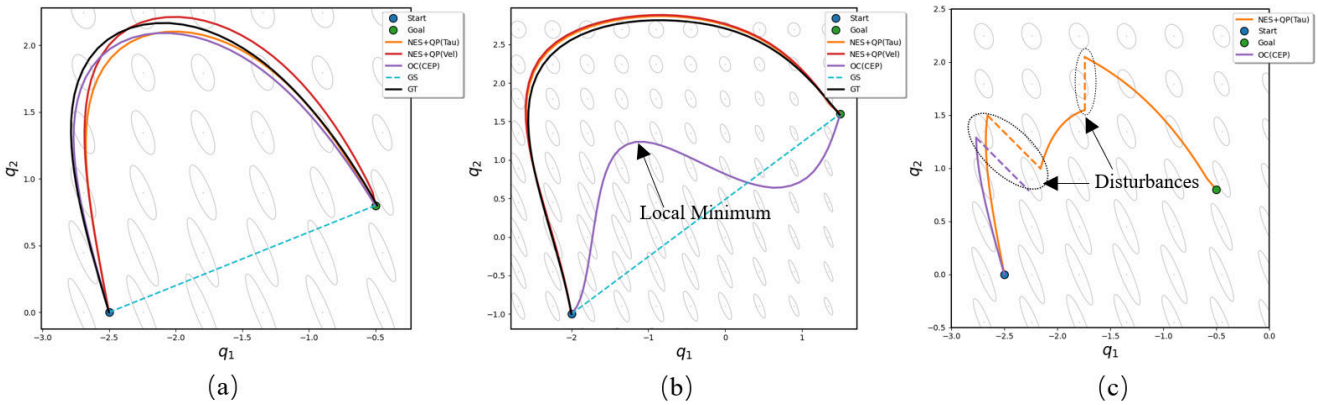


Figure 12. Comparison between our NES with a QP controller and baseline methods. NES matches OC in short-range tasks (a), outperforms long-range cases (b) by avoiding local minima, and is robust to (c) disturbances.

Table 3. Comparison of Baselines on Energy-Efficient Motion Generation

Method	Planar Robot (2D)				Franka Robot (7D)			
	Geodesic Length	L2 Torque	Active Torque	Time (Policy/Path)	Geodesic Length	L2 Torque	Active Torque	Time (Policy/Path)
GS	0.99 ± 0.37	88.7 ± 26.5	55.4 ± 19.5	**0.001/0.18**	8.73 ± 2.16	246.3 ± 82.8	58.9 ± 24.6	**0.001/0.18**
OC	0.91 ± 0.34	74.7 ± 20.6	48.2 ± 15.9	4.43/4.43	8.29 ± 2.34	201.6 ± 68.3	52.0 ± 21.2	6.14/6.14
RFM	$\mathbf{0.80 \pm 0.22}$	55.8 ± 14.6	38.9 ± 9.7	0.58/0.58	-	-	-	-
NES+QP (Vel)	0.81 ± 0.22	$\mathbf{51.6 \pm 11.1}$	$\mathbf{34.2 \pm 7.5}$	0.013/0.27	$\mathbf{7.69 \pm 2.07}$	128.9 ± 32.0	30.6 ± 11.6	0.016/0.31
NES+QP (Tau)	0.81 ± 0.22	53.9 ± 13.0	35.6 ± 8.1	0.021/0.40	7.74 ± 2.10	$\mathbf{118.0 \pm 29.3}$	$\mathbf{30.5 \pm 11.4}$	0.025/0.51

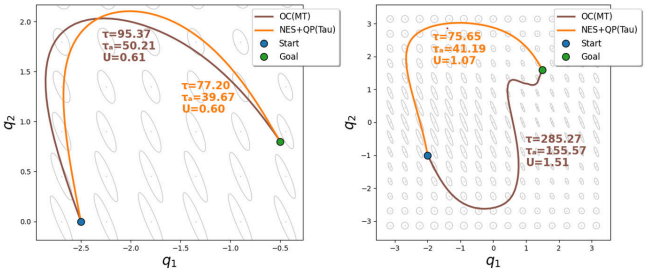


Figure 13. Comparison between our NES with a QP controller and optimal control approach with minimum torque inputs.

joint configurations can map to the same task space position, resulting in different geodesic flows starting from various joint angles. Consequently, the orange trajectory represents an optimal energy path that terminates at a different joint configuration, distinguishing it from the other three.

Quantitative results are presented in Table 4. We compare our C-NES method against two baseline methods. The first is a geodesic shooting approach guided by Gauss–Newton (GN), a second-order optimization method commonly used in inverse kinematics. GN optimizes joint configurations for the inverse kinematics problem. The second baseline is an optimal control (OC) formulation, where the original terminal cost term $\|\boldsymbol{q}_N - \boldsymbol{q}_s\|^2_{\boldsymbol{Q}}$ in (32) is replaced with a forward kinematics cost $\|f(\boldsymbol{q}_N) - \boldsymbol{x}_s\|^2_{\boldsymbol{Q}}$. The RFM method fails under this setup due to severe grid distortions introduced by the nonlinear forward kinematics function, which make solving the eikonal equation intractable. These results highlight the effectiveness of C-NES in computing geodesic paths that map joint configurations to end-effector poses, resulting in energy-efficient motions from joint space to task space. Importantly, our formulation extends the classical eikonal equation to incorporate nonlinear forward kinematics, re-framing the inverse kinematics problem as a hybrid optimization problem that jointly balances task-space accuracy and energy optimality. The network learns a global distance field that encodes energy-efficient solutions in task space, leveraging the eikonal equation constraint $\|\nabla U\|_{\boldsymbol{G}} = 1$ to capture globally shortest paths across the configuration manifold.

Table 4. Baseline Comparison for Geodesic Lengths in Dynamics-aware Motion Generation in Task Space

Method	GN+GS	OC	C-NES
Planar Robot (2D)	0.54 ± 0.20	0.50 ± 0.18	0.43 ± 0.12
Franka Robot (7D)	6.92 ± 1.71	6.27 ± 1.60	5.45 ± 1.47

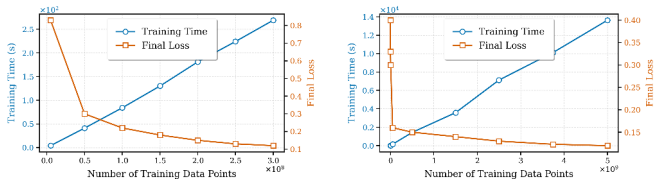
Table 5. Inference time (s) of NES across Different Batch Sizes.

Batch Size	1	10	10^2	10^3	10^4	10^5
CPU	**0.010**	**0.012**	**0.014**	0.035	0.249	3.131
GPU	0.027	0.028	0.028	**0.028**	**0.031**	**0.121**

6.4.4 Data and Computational Efficiency We further validate the data and computational efficiency of our neural Eikonal solver for geometry-aware motion generation (**Q1, Q4**). We analyze both training efficiency and inference time.

Figure 14 presents the training performance of our neural solver, evaluated by the number of data points required and total training time. For the planar robot, the network converges within 5 minutes using up to 3.0×10^8 data points. In contrast, training the Franka robot model takes approximately 4 hours and requires 5.0×10^9 data points. In both robot experiments, training time scales linearly with the size of the training dataset.

Table 5 summarizes the inference time per forward pass across varying batch sizes. The reported times include neural network evaluation, inertial mass matrix, computation of the inertial mass matrix and potential energy, and automatic differentiation to estimate the geodesic flow. Overall, inference times remain low and stable across both CPU and GPU platforms, even for batch sizes up to 10^3,



(a) Planar Robot (b) Franka Robot

Figure 14. Evaluation of efficiency for the 2D planar robot and Franka robot.

indicating strong suitability for real-time applications. The GPU's parallelism further enhances efficiency at larger batch sizes. The majority of computational bottlenecks lies in inverting the matrices to retrieve geodesic flows. Notably, for a batch size of 1, CPU inference completes in under 0.01 seconds, supporting high-frequency updates of source-to-goal pairs that are critical for reactive and real-time planning and control.

6.4.5 Task-prioritized Energy-efficient Motion Policy Previous experiments have demonstrated NES's capability to generate energy-efficient motions that can be effectively tracked using a QP controller. In this section, we further evaluate the effectiveness of NES within task-prioritized control frameworks, focusing on its ability to preserve natural energy conservation behavior while satisfying additional constraints (**Q6**). Specifically, we consider two representative scenarios: constrained motion and obstacle avoidance. The constrained motion scenario involves task-space requirements, such as maintaining a desired end-effector position and orientation. In contrast, obstacle avoidance requires generating collision-free trajectories to ensure safe operation in cluttered environments. The experimental setups for both scenarios are described in detail below:

6.4.6 Constrained motion In this setup, the robot's end-effector is restricted to moving within a specified plane while maintaining an orthogonal orientation. The constraint requires the end-effector to remain in a plane parallel to the horizontal plane at $x = 0.3$, with the vertical axis of the end-effector aligned consistently throughout the motion. To achieve energy-efficient movement, our NES policy is designed to operate within the null-space of the primary task. The Jacobian associated with this constraint defines a subspace of the robot's full Jacobian, allowing NES to optimize energy usage without violating the primary task requirements.

6.4.7 Obstacle avoidance This scenario focuses on achieving energy-efficient motion while preventing collisions with surrounding obstacles. Building on prior work (Li et al. 2024b), we utilize a distance function and its gradient to monitor proximity to obstacles. When no obstacles are within the predefined safety margin, the robot follows the NES-generated energy-efficient policy. However, if an obstacle enters a predefined safety zone, the collision avoidance strategy constrains NES within the null-space, ensuring safe, collision-free trajectories without compromising the primary task.

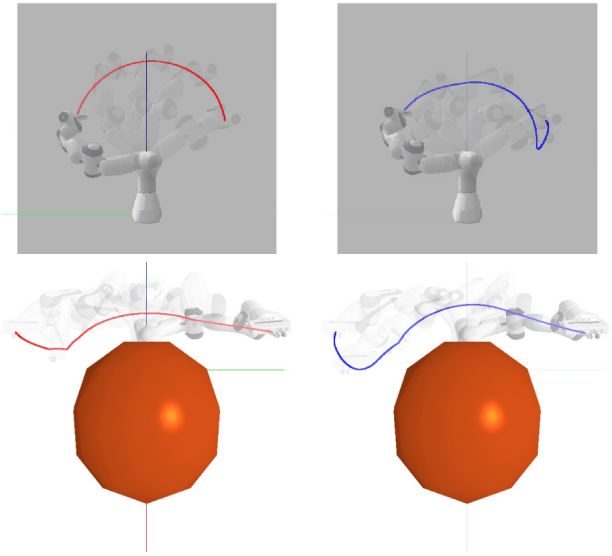


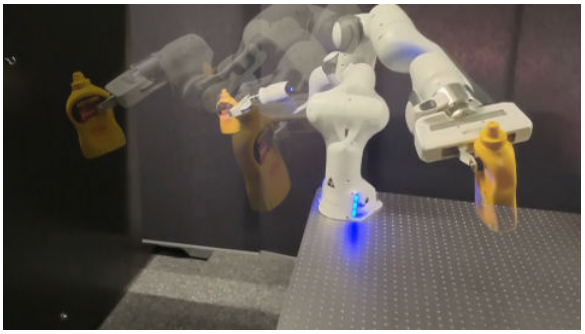
Figure 15. Trajectories generated by task-prioritized energy-efficient motion policy and baseline. Top: Constrained motion in which the end-effector is perpendicular to a plane. Bottom: Obstacle avoidance. Red and blue curves depict the GS and NES paths operating in the null-space of each principal task.

Table 6. Geodesic lengths for task-prioritized energy-aware motion policies under the Jacobi metric.

Method	No Constraint	Obstacle	Constrained Motion
GS + TP	7.94±1.67	8.52±1.83	8.75±1.83
NES + TP	**7.67±1.70**	**7.86±1.85**	**8.20±1.86**

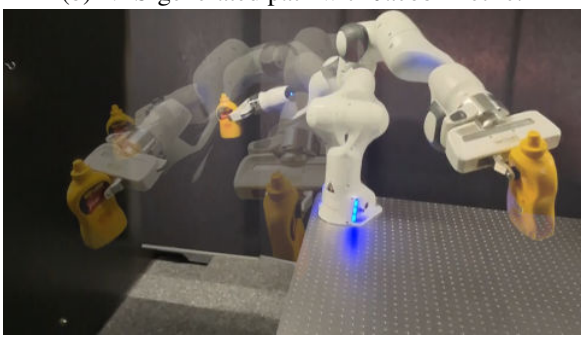
We begin by randomly sampling 100 joint configuration pairs that satisfy the above two constraints. For each pair, we apply the NES algorithm to compute energy-aware policies. We use geodesic shooting as the baseline because it is also a reactive approach. Both NES and geodesic shooting policies are projected into the null-space of the primary tasks. Table 6 presents the experimental results on average geodesic lengths for each method. As expected, introducing constraints will increase geodesic lengths on the configuration space manifold as the space of feasible solutions is restricted. Despite this, the NES-generated motion policies continue to outperform geodesic shooting in terms of energy efficiency. Figure 15 compares trajectories of task-prioritized motions with and without our energy-aware motion policy, demonstrating the effectiveness and flexibility of NES in combination with other motion policies for real-time, adaptive, and energy-efficient robot control.

6.4.8 Robot Experiments Finally, we conduct robotic experiments to demonstrate the efficacy of our approach; refer to the accompanying video (**Q4**). In the experiments, we employ the QP controller with joint velocity inputs to track the geodesic flow generated by the NES framework. Two scenarios are evaluated for energy-conserving paths: with and without gravity compensation. In the first scenario, gravity compensation is applied externally via the controller, so the Riemannian metric reflects only *kinetic energy*. In the second, both kinetic and potential energy are incorporated into the geodesic computation, resulting in full energy conservation (Jacobi metric).



(a) NES-generated path with kinetic energy metric.

(b) NES-generated path with Jacobi metric.



(c) Euclidean path.

Figure 16. Snapshots of robot motions for C-space path planning. Initial and final frames are displayed in solid color. Intermediary frames are transparent.

Figure 16 presents key frames of the robot trajectories generated by NES under both metrics. For comparison, we include the results from the geodesic shooting (GS) approach, where the initial velocity is directed toward the goal point and scaled to satisfy the manifold constraints. On both manifolds, GS produces a straight-line Euclidean path with differences lying in joint velocities to satisfy the underlying Riemannian metric constraint. To visualize differences more clearly, Figure 17 plots the joint positions for all three trajectories. Although all trajectories share the same start and goal configurations, the NES-generated paths exhibit more natural behavior with curved trajectories in configuration space, better reflecting energy-efficient motion consistent with the geometry of the underlying manifold. Similarly, robot motions and joint positions for task-space motion generation are visualized in Figures 18 and 19. Although the target joint configuration is unknown, the gradient flow produced by our C-NES allows the robot to reach the solution iteratively along the path of minimal geodesic distance. Due to the redundancy of the robot, the resulting trajectories differ and converge to distinct final joint configurations, all satisfying the same task-space goal.

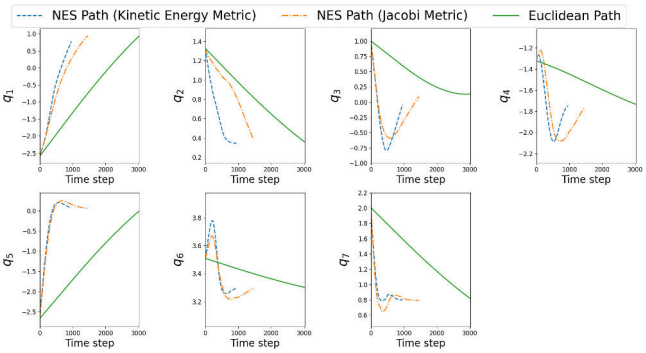
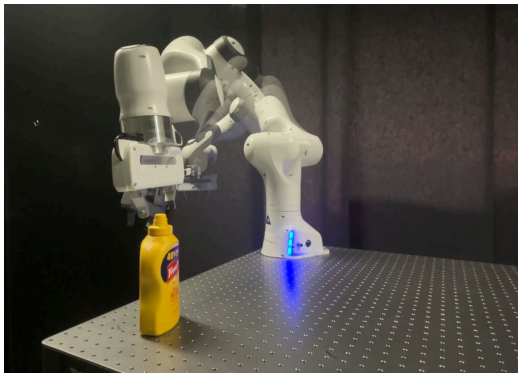


Figure 17. Joint positions for Euclidean and NES-generated paths.

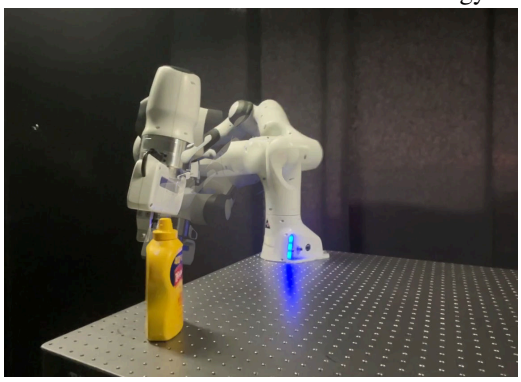
We further visualize the variations in energy and torque during the robot's motion to assess the energy efficiency of our approach. Figure 20 (a) shows the energy and torque variations on the Riemannian manifold with Jacobi metric (*solid lines*), compared to the Euclidean path (*dashed lines*). On the manifold equipped with the Jacobi metric, the total energy—comprising both kinetic and potential components—is expected to remain constant. Rather than directly moving toward the goal, the NES-generated path initially reduces the robot's potential energy, converting it into kinetic energy and resulting in a curved path. Therefore, we observe lower energy and reduced torque consumption. Similarly, Figure 20 (b) illustrates kinetic energy-efficient trajectories for the task-space motion generation problem using C-NES. In this case, the kinetic energy remains constant at each time step. It exhibits more effective utilization of kinetic energy compared to the geodesic shooting path with Gauss-Newton optimization. While it may require higher torque at the beginning to maintain kinetic energy, it compensates by reducing energy and torque demands later in the motion, ultimately producing a more efficient trajectory to reach the goal.

To further evaluate performance at scale, we simulate 100 randomly sampled start-to-goal configuration pairs and record energy and torque cost during trajectory execution. Results are summarized in Table 7. At each time step, joint positions and velocities are used to compute the energy, while the total torque is obtained by summing the actuator torques applied throughout the trajectory execution. Across both Riemannian metrics, NES consistently achieves lower overall energy cost and reduced torque inputs compared to GS. Its variant, C-NES, also outperforms GS with Gauss-Newton optimization in task space. These results highlight the strength of the NES framework in producing dynamics-aware, energy-efficient motions that are well-suited for real-world control and planning tasks.

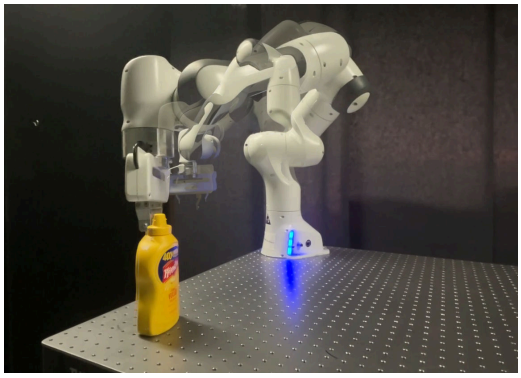
It is important to note that, in practical mechanical systems, actuators cannot perfectly share or recycle power. Negative mechanical work is often dissipated locally, and opposed closed-force loops can generate internal forces with near-zero net mechanical work while still incurring nontrivial electrical losses. Consequently, geodesic optimality (minimal length under the chosen Riemannian metric) does not, in general, coincide with minimal execution effort. We therefore use geodesics as a principled heuristic: a global, geometry-aware reference that can guide the



(a) The solution of C-NES with the kinetic energy metric.



(b) The solution of C-NES with the Jacobi metric.



(c) The solution of the Gauss-Newton method.

Figure 18. Snapshots of robot motions for inverse kinematics. Initial and final frames are displayed in solid color. Intermediary frames are transparent.

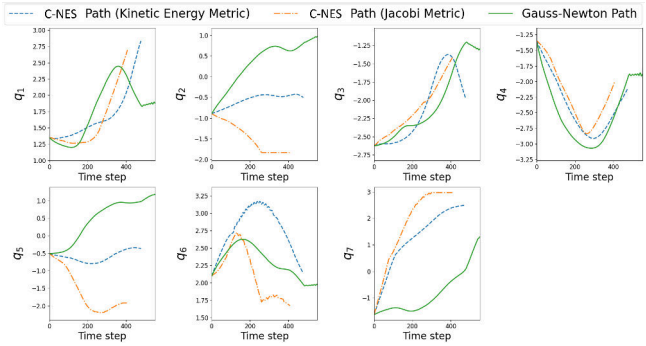
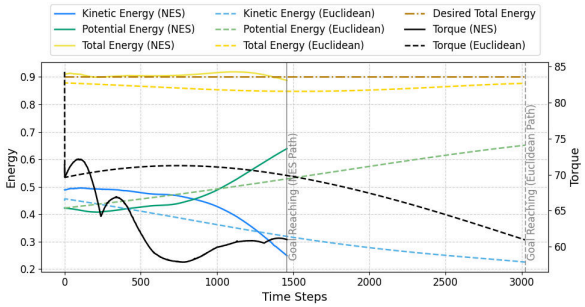
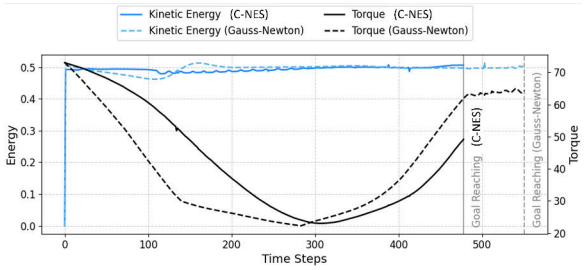


Figure 19. Joint positions for inverse kinematics solved by Gauss-Newton and C-NES with kinetic energy metric and Jacobi metric.

robot toward lower-energy motions and inform trajectory generation. Although not universally aligned with all



(a) NES and Euclidean paths with Jacobi metric.



(b) C-NES and Gauss-Newton paths with kinetic energy metric.

Figure 20. Variation of energy and torque among different paths and Riemannian metrics.

Table 7. Total Energy and torque cost during the robot's motion.

		Kinetic Energy Metric		Jacobi Metric	
		Energy	Torque	Energy	Torque
C-Space	GS	1.53 ± 0.62	13.4 ± 6.89	2.82 ± 0.85	12.8 ± 5.17
	NES	0.62 ± 0.18	5.14 ± 1.92	1.42 ± 0.46	6.68 ± 2.54
Task-space	GS + GN	0.20 ± 0.09	1.56 ± 0.62	0.51 ± 0.23	2.12 ± 1.08
	C-NES	0.15 ± 0.07	1.26 ± 0.56	0.37 ± 0.11	1.75 ± 0.88

actuator cost models, this approach is grounded in a well-established geometric framework and yields geometrically optimal paths for analyzing and planning robot motion when both kinetic and potential energy are modeled. In addition, the total energy H can be provided as a conditioning input to NES, allowing the geodesic flow to adapt dynamically to changes in energy, as described in Section 5.1.

7 Conclusion

In this article, we presented an approach to compute Riemannian distance functions by formulating the problem as wavefront propagation and solving the corresponding *Riemannian eikonal equation*. We proposed a neural Riemannian eikonal solver (NES) that learns a mesh-free, continuous, and differentiable implicit field, enabling efficient queries of distances and geodesic flows in high-dimensional configuration spaces, with globally consistent distance and flow fields. In addition, we introduced NES variants that can be conditioned on boundary data and/or on Riemannian metrics. Our method integrates with planning, control, and optimization as a principled geometric prior. We demonstrated the effectiveness of the approach in constructing distance fields and geodesic flows across kinematics, dynamics, motion planning, and control problems—including high-dimensional manipulators—with quantitative and qualitative comparisons to established baselines. Finally, we conducted extensive experiments on

energy-aware manifolds induced by kinetic and potential energy; the resulting geodesics provide informative guidance toward lower-energy and lower-torque motions, highlighting the practicality of our approach.

A limitation of our approach lies in training the neural Riemannian eikonal solver. Once trained, the model can be reused across multiple queries, enabling efficient and real-time applications. However, the lack of prior data and the strongly anisotropic nature of the Riemannian metric pose significant challenges for the training. While our formulation theoretically allows for global optimal solutions, its practical performance can sometimes unexpectedly fall short. Future work will focus on improving the robustness and accuracy of the training process to enhance overall reliability. Additionally, we plan to extend our framework to broader metric spaces beyond Riemannian manifolds. The eikonal equation itself can be generalized to other types of partial differential equations relevant to robotics, including those arising in dynamic systems modeling (Lutter et al. 2018), value function computation in dynamic programming, and reinforcement learning. These extensions could unlock new applications for geometry-aware learning in complex robotic environments.

8 Acknowledgments

This work was supported by the China Scholarship Council (No. 202204910113), and by the State Secretariat for Education, Research and Innovation in Switzerland for participation in the European Commission's Horizon Europe Program through the INTELLIMAN project (https://intelliman-project.eu/, HORIZON-CL4-Digital-Emerging Grant 101070136) and the SESTOSENSO project (http://sestosenso.eu/, HORIZON-CL4-Digital-Emerging Grant 101070310).

References

Aila T and Laine S (2009) Understanding the efficiency of ray traversal on GPUs. In: *Proc. High Performance Graphics*. pp. 145–149.

Albu-Schäffer A and Sachtler A (2022) What can algebraic topology and differential geometry teach us about intrinsic dynamics and global behavior of robots? In: *The International Symposium of Robotics Research*. Springer, pp. 468–484.

Alvino C, Unal G, Slabaugh G, Peny B and Fang T (2007) Efficient segmentation based on eikonal and diffusion equations. *International Journal of Computer Mathematics* 84(9): 1309–1324.

Breyer M, Chung JJ, Ott L, Siegwart R and Nieto J (2021) Volumetric grasping network: Real-time 6 dof grasp detection in clutter. In: *Proc. Conference on Robot Learning (CoRL)*. PMLR, pp. 1602–1611.

Bullo F and Lewis AD (2004) *Geometric Control of Mechanical Systems: Modeling, Analysis, and Design for Simple Mechanical Control Systems*, volume 49. Springer Science & Business Media.

Cabrera A and Hatton RL (2024) Optimal control of robotic systems and biased riemannian splines. *ESAIM: Control, Optimisation and Calculus of Variations* 30: 36.

Calinon S (2020) Gaussians on riemannian manifolds: Applications for robot learning and adaptive control. *IEEE Robotics & Automation Magazine* 27(2): 33–45.

Casetti L, Pettini M and Cohen E (2000) Geometric approach to hamiltonian dynamics and statistical mechanics. *Physics Reports* 337(3): 237–341.

Chiacchio P, Chiaverini S, Sciavicco L and Siciliano B (1991) Closed-loop inverse kinematics schemes for constrained redundant manipulators with task space augmentation and task priority strategy. *The International Journal of Robotics Research* 10(4): 410–425.

Crane K, Livesu M, Puppo E and Qin Y (2020) A survey of algorithms for geodesic paths and distances. *CoRR* abs/2007.10430. DOI:10.48550/arXiv.2007.10430. URL https://arxiv.org/abs/2007.10430.

Crane K, Weischedel C and Wardetzky M (2013) Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Transactions on Graphics (TOG)* 32(5): 1–11.

de Kool M, van der Hilst RDJ and Nolet G (2006) A practical grid-based method for tracking multiple refraction arrivals. *Geophysical Journal International* 167(1): 253–270. DOI: 10.1111/j.1365-246X.2006.03105.x.

Driess D, Ha JS, Toussaint M and Tedrake R (2022) Learning models as functionals of signed-distance fields for manipulation planning. In: *Proc. Conference on Robot Learning (CoRL)*. PMLR, pp. 245–255.

Fishman A, Murali A, Eppner C, Peele B, Boots B and Fox D (2023) Motion policy networks. In: *Conference on Robot Learning*. PMLR, pp. 967–977.

Garrido S, Malfaz M and Blanco D (2013) Application of the fast marching method for outdoor motion planning in robotics. *Robotics and Autonomous Systems* 61(2): 106–114.

Girolami M and Calderhead B (2011) Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society Series B: Statistical Methodology* 73(2): 123–214.

Gropp A, Yariv L, Haim N, Atzmon M and Lipman Y (2020) Implicit geometric regularization for learning shapes. In: *Proceedings of Machine Learning and Systems 2020*. pp. 3569–3579.

Grubas S, Duchkov A and Loginov G (2023) Neural eikonal solver: Improving accuracy of physics-informed neural networks for solving eikonal equation in case of caustics. *Journal of Computational Physics* 474: 111789.

Hogan N (1985) Impedance control: An approach to manipulation. part i–theory. *Journal of Dynamic Systems, Measurement, and Control* 107(1): 1–7.

Ihrke I, Ziegler G, Tevs A, Theobalt C, Magnor M and Seidel HP (2007) Eikonal rendering: Efficient light transport in refractive objects. *ACM Transactions on Graphics (TOG)* 26(3): 59–es.

Jaquier N and Asfour T (2022) Riemannian geometry as a unifying theory for robot motion learning and control. In: *The International Symposium of Robotics Research*. Springer, pp. 395–403.

Johannessen LMG, Arbo MH and Gravdahl JT (2019) Robot dynamics with urdf & casadi. In: *2019 7th International Conference on Control, Mechatronics and Automation (ICCMA)*. IEEE.

Julian B and Gubbins D (1977) Three-dimensional seismic ray tracing. *Journal of Geophysics* 43(1): 95–113.

Kelshaw D and Magri L (2024) Computing distances and means on manifolds with a metric-constrained eikonal approach. *arXiv preprint arXiv:2404.08754* .

Khatib O (1987) A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation* 3(1): 43–53.

Kimmel R and Sethian JA (1998) Computing geodesic paths on manifolds. *Proceedings of the National Academy of Sciences* 95(15): 8431–8435.

Klein H, Jaquier N, Meixner A and Asfour T (2022) A Riemannian take on human motion analysis and retargeting. In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 5210–5217.

Klein H, Jaquier N, Meixner A and Asfour T (2023) On the design of region-avoiding metrics for collision-safe motion generation on riemannian manifolds. In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 2346–2353.

Koptev M, Figueroa N and Billard A (2022) Neural joint space implicit signed distance functions for reactive robot manipulator control. *IEEE Robotics and Automation Letters* 8(2): 480–487.

Koptev M, Figueroa N and Billard A (2024) Reactive collision-free motion generation in joint space via dynamical systems and sampling-based mpc. *The International Journal of Robotics Research* : 02783649241246557.

Lachner J, Schettino V, Allmendinger F, Fiore MD, Ficuciello F, Siciliano B and Stramigioli S (2020) The influence of coordinates in robotic manipulability analysis. *Mechanism and machine theory* 146: 103722.

Laux M and Zell A (2021) Robot arm motion planning based on geodesics. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 7585–7591.

Law M (2021) Jacobi fields, conjugate points and some applications. https://mike-law.github.io/files/jacobi_fields.pdf.

Li Y, Chi X, Razmjoo A and Calinon S (2024a) Configuration space distance fields for manipulation planning. In: *Proc. Robotics: Science and Systems (RSS)*.

Li Y, Zhang Y, Razmjoo A and Calinon S (2024b) Representing robot geometry as distance fields: Applications to whole-body manipulation. In: *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*. pp. 15351–15357.

Lin FC, Ritzwoller MH and Snieder R (2009) Eikonal tomography: surface wave tomography by phase front tracking across a regional broad-band seismic array. *Geophysical Journal International* 177(3): 1091–1110.

Liu P, Zhang K, Tateo D, Jauhri S, Peters J and Chalvatzaki G (2022) Regularized deep signed distance fields for reactive motion generation. In: *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*. IEEE, pp. 6673–6680.

Liu T, Liu Z, Jiao Z, Zhu Y and Zhu SC (2021) Synthesizing diverse and physically stable grasps with arbitrary hand structures using differentiable force closure estimator. *IEEE Robotics and Automation Letters* 7(1): 470–477.

Lutter M and Peters J (2023) Combining physics and deep learning to learn continuous-time dynamics models. *The International Journal of Robotics Research* 42(3): 83–107.

Lutter M, Ritter C and Peters J (2018) Deep lagrangian networks: Using physics as model prior for deep learning. In: *International Conference on Learning Representations*.

Macklin M, Erleben K, Müller M, Chentanez N, Jeschke S and Corse Z (2020) Local optimization for robust signed distance field collision. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 3(1): 1–17.

Mantegazza C and Mennucci AC (2002) Hamilton-jacobi equations and distance functions on riemannian manifolds. *arXiv preprint math/0201296* .

Marić A, Li Y and Calinon S (2024) Online learning of continuous signed distance fields using piecewise polynomials. *IEEE Robotics and Automation Letters (RA-L)* 9(6): 6020–6026. DOI:10.1109/LRA.2024.3397085.

Millane A, Oleynikova H, Wirbel E, Steiner R, Ramasamy V, Tingdahl D and Siegwart R (2024) nvblox: Gpu-accelerated incremental signed distance field mapping.

Mirebeau JM (2019) Riemannian fast-marching on Cartesian grids, using Voronoi's first reduction of quadratic forms. *SIAM Journal on Numerical Analysis* 57(6): 2608–2655. URL https://hal.science/hal-01507334v4/document.

Neilson PD, Neilson MD and Bye RT (2015) A Riemannian geometry theory of human movement: The geodesic synergy hypothesis. *Human movement science* 44: 42–72.

Ni R and Qureshi AH (2022) Ntfields: Neural time fields for physics-informed robot motion planning. In: *The Eleventh International Conference on Learning Representations*.

Ni R and Qureshi AH (2024a) Physics-informed neural motion planning on constraint manifolds. *arXiv preprint arXiv:2403.05765* .

Ni R and Qureshi AH (2024b) Progressive learning for physics-informed neural motion planning .

Ortega R, Van Der Schaft A, Maschke B and Escobar G (2002) Interconnection and damping assignment passivity-based control of port-controlled hamiltonian systems. *Automatica* 38(4): 585–596.

Osher S, Fedkiw R and Piechor K (2004) Level set methods and dynamic implicit surfaces. *Appl. Mech. Rev.* 57(3): B15–B15.

Park JJ, Florence P, Straub J, Newcombe R and Lovegrove S (2019) Deepsdf: Learning continuous signed distance functions for shape representation. In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. pp. 165–174.

Peyré G, Péchaud M, Keriven R and Cohen LD (2010) Geodesic methods in computer vision and graphics. *Foundations and Trends® in Computer Graphics and Vision* 5(3–4): 197–397.

Raissi M, Perdikaris P and Karniadakis GE (2019) Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics* 378: 686–707.

Ratliff N, Toussaint M and Schaal S (2015) Understanding the geometry of workspace obstacles in motion optimization. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 4202–4209.

Ratliff N, Zucker M, Bagnell JA and Srinivasa S (2009) Chomp: Gradient optimization techniques for efficient motion planning. In: *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*. IEEE, pp. 489–494.

Ratliff ND, Issac J, Kappler D, Birchfield S and Fox D (2018) Riemannian motion policies. *arXiv preprint arXiv:1801.02854* .

Rawlinson N and Sambridge M (2005) The fast marching method: an effective tool for tomographic imaging and tracking multiple phases in complex layered media. *Exploration Geophysics* 36(4): 341–350.

Sethian JA (1996) A fast marching level set method for monotonically advancing fronts. *proceedings of the National Academy of Sciences* 93(4): 1591–1595.

Siciliano B, Khatib O and Kröger T (2008) *Springer handbook of robotics*, volume 200. Springer.

Smith JD, Azizzadenesheli K and Ross ZE (2020) Eikonet: Solving the eikonal equation with deep neural networks. *IEEE Transactions on Geoscience and Remote Sensing* 59(12): 10685–10696.

Spong MW, Hutchinson S and Vidyasagar M (2006) *Robot modeling and control*, volume 3. Wiley New York.

Van Wyk K, Xie M, Li A, Rana MA, Babich B, Peele B, Wan Q, Akinola I, Sundaralingam B, Fox D, Boots B and Ratliff ND (2022) Geometric fabrics: Generalizing classical mechanics to capture the physics of behavior. *IEEE Robotics and Automation Letters* 7(2): 3202–3209.

Walker MW and Orin DE (1982) Efficient dynamic computer simulation of robotic mechanisms .

Weng T, Held D, Meier F and Mukadam M (2022) Neural grasp distance fields for robot manipulation. *arXiv preprint arXiv:2211.02647* .

Xie Y, Takikawa T, Saito S, Litany O, Yan S, Khan N, Tombari F, Tompkin J, Sitzmann V and Sridhar S (2022) Neural fields in visual computing and beyond. In: *Computer Graphics Forum*, volume 41. Wiley Online Library, pp. 641–676.

Yang W and Jin W (2024) Contactsdf: Signed distance functions as multi-contact models for dexterous manipulation. *arXiv preprint arXiv:2408.09612* .

Yoshikawa T (1985) Manipulability of robotic mechanisms. *The International Journal of Robotics Research* 4(2): 3–9.

Zhao H (2005) A fast sweeping method for eikonal equations. *Mathematics of computation* 74(250): 603–627.

Appendix

A Robot Dynamics on Configuration Space Manifolds

A.1 Lagrangian Mechanics

The Lagrangian is a scalar function defined on the tangent bundle of the configuration manifold G, given by:

$$L\left(\boldsymbol{q}, \dot{\boldsymbol{q}}\right) = T\left(\boldsymbol{q}, \dot{\boldsymbol{q}}\right) - P\left(\boldsymbol{q}\right), \tag{43}$$

where $\boldsymbol{q}$ is the configuration, $\dot{\boldsymbol{q}}$ is the velocity, T is the kinetic energy, and P is the potential energy. The kinetic energy is typically expressed as:

$$T\left(\boldsymbol{q}, \dot{\boldsymbol{q}}\right) = \frac{1}{2}\dot{\boldsymbol{q}}^{\top}\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{q}}, \tag{44}$$

with $\boldsymbol{M}(\boldsymbol{q})$ representing the configuration-dependent inertia matrix. The action functional S, defined over a trajectory $\boldsymbol{q}(t)$ between times t_0 and t_1, integrates the Lagrangian along the path:

$$S\left(\boldsymbol{q}\right) = \int_{t_0}^{t_1} L\left(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t)\right)\, \mathrm{d}t. \tag{45}$$

When potential energy is ignored, this functional reduces to the kinetic-energy functional, with $\boldsymbol{M}(\boldsymbol{q})$ acting as the kinetic-energy metric. According to the principle of stationary action, the physical trajectory is one that makes the action stationary, i.e., $\delta S = 0$. This leads to the Euler–Lagrange equations:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{\boldsymbol{q}}}\right) - \frac{\partial L}{\partial \boldsymbol{q}} = \boldsymbol{0}. \tag{46}$$

Substituting the Lagrangian into the Euler–Lagrange equations yields the system's second-order dynamics:

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{0}, \tag{47}$$

where $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$ groups the Coriolis and centrifugal terms arising from $\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{q}}$ and $\frac{\partial \boldsymbol{M}}{\partial \boldsymbol{q}}$. $\boldsymbol{g}(\boldsymbol{q}) = \frac{\partial P}{\partial \boldsymbol{q}}$ is the generalized gravitational force.

A.1.1 Derivation of Equation of Motion We derive the equation of motion from Equation (46). The partial derivative of the kinetic energy T with respect to $\dot{q}_i$ is:

$$\frac{\partial T}{\partial \dot{q}_i} = \sum_j M_{ij}\dot{q}_j. \tag{48}$$

Taking the total time derivative yields:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_i}\right) = \sum_j M_{ij}\ddot{q}_j + \sum_{j,k}\frac{\partial M_{ij}}{\partial q_k}\dot{q}_k\dot{q}_j. \tag{49}$$

Next, we compute the partial derivative of T with respect to q_i:

$$\frac{\partial T}{\partial q_i} = \frac{1}{2}\sum_{j,k}\frac{\partial M_{jk}}{\partial q_i}\dot{q}_j\dot{q}_k. \tag{50}$$

The kinetic part of the Euler–Lagrange equation is then given by:

$$\begin{aligned}
&\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}_i}\right) - \frac{\partial T}{\partial q_i} \\
&= \sum_j M_{ij}\ddot{q}_j + \sum_{j,k}\left(\frac{\partial M_{ij}}{\partial q_k} - \frac{1}{2}\frac{\partial M_{jk}}{\partial q_i}\right)\dot{q}_j\dot{q}_k \\
&= \sum_j M_{ij}\ddot{q}_j + \sum_{j,k}\left(\frac{1}{2}\frac{\partial M_{ij}}{\partial q_k} + \frac{1}{2}\frac{\partial M_{ik}}{\partial q_j} - \frac{1}{2}\frac{\partial M_{jk}}{\partial q_i}\right)\dot{q}_j\dot{q}_k.
\end{aligned} \tag{51}$$

Since the potential energy P depends only on $\boldsymbol{q}$, we have:

$$\frac{\partial P}{\partial \dot{q}_i} = 0, \quad \frac{\partial P}{\partial q_i} = g_i. \tag{52}$$

Combining the kinetic and potential terms, the full equation becomes:

$$\sum_j M_{ij}\ddot{q}_j + \sum_{j,k}\Gamma_{i,jk}\dot{q}_j\dot{q}_k + g_i = 0, \tag{53}$$

where the Christoffel symbol $\Gamma_{i,jk}$ is defined as:

$$\Gamma_{i,jk} = \frac{1}{2}\left(\frac{\partial M_{ij}}{\partial q_k} + \frac{\partial M_{ik}}{\partial q_j} - \frac{\partial M_{jk}}{\partial q_i}\right). \tag{54}$$

The term $\sum_{j,k}\Gamma_{i,jk}\dot{q}_j\dot{q}_k$ corresponds to the Coriolis and centrifugal forces, i.e., the i-th component of $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$. In vector form, this yields the standard form of the robot dynamics (Siciliano et al. 2008):

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{0}. \tag{55}$$

A.2 Hamiltonian Mechanics

The Hamiltonian function $H(\boldsymbol{q},\boldsymbol{p})$ represents the total energy of a mechanical system (Lutter and Peters 2023). It is defined as the sum of kinetic and potential energy.

$$H(\boldsymbol{q},\boldsymbol{p}) = T(\boldsymbol{q},\dot{\boldsymbol{q}}) + P(\boldsymbol{q}) = \frac{1}{2}\boldsymbol{p}^\top \boldsymbol{M}^{-1}(\boldsymbol{q})\boldsymbol{p} + P(\boldsymbol{q}), \tag{56}$$

where $\boldsymbol{p} \in \mathbb{R}^n$ is the generalized momentum, given by

$$\boldsymbol{p} = \frac{\partial L}{\partial \dot{\boldsymbol{q}}} = \boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{q}}. \tag{57}$$

The Hamiltonian and Lagrangian are related via a Legendre transform:

$$H(\boldsymbol{q},\boldsymbol{p}) = \boldsymbol{p}^\top \dot{\boldsymbol{q}} - L(\boldsymbol{q},\dot{\boldsymbol{q}}), \tag{58}$$

where $\dot{\boldsymbol{q}} = \boldsymbol{M}^{-1}(\boldsymbol{q})\boldsymbol{p}$. Substituting the Euler–Lagrange equations yields Hamilton's equations:

$$\dot{\boldsymbol{q}} = \frac{\partial H}{\partial \boldsymbol{p}}, \qquad \dot{\boldsymbol{p}} = -\frac{\partial H}{\partial \boldsymbol{q}}. \tag{59}$$

These equations describe the evolution of the system in phase space $(\boldsymbol{q},\boldsymbol{p})$. In the absence of non-conservative forces, the Hamiltonian H is conserved, reflecting conservation of total mechanical energy.

A.2.1 Geometric Formulation and Maupertuis' Principle

When $H(\boldsymbol{q},\boldsymbol{p})$ is constant along a trajectory, the action integral can be reformulated in a time-independent form. This leads to *Maupertuis' Principle* (Albu-Schäffer and Sachtler 2022):

$$S_M(\boldsymbol{q}) = \int_{\boldsymbol{q}_1}^{\boldsymbol{q}_2} \boldsymbol{p}^\top d\boldsymbol{q}, \tag{60}$$

which states that the true path connecting $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ corresponds to a minimum or a saddle point of the action functional. This reformulation provides a geometric viewpoint on robot dynamics, enabling the study of energy-conserving paths using tools from Riemannian geometry and algebraic topology. In this setting, constant-energy trajectories correspond to geodesics under the *Jacobi metric* (Casetti et al. 2000)

$$\boldsymbol{G}_{\text{Jac}}(\boldsymbol{q}) = 2\big(H - P(\boldsymbol{q})\big)\boldsymbol{M}(\boldsymbol{q}), \tag{61}$$

which is a conformal scaling of the kinetic-energy metric.

A.2.2 Derivation of Jacobi metric

To derive the Jacobi metric, we can reparameterize the trajectory $\boldsymbol{q}(t)$ using the arc-length s, with

$$\boldsymbol{q}' = \frac{d\boldsymbol{q}}{ds}, \quad \dot{\boldsymbol{q}} = \frac{d\boldsymbol{q}}{dt} = \frac{d\boldsymbol{q}}{ds}\frac{ds}{dt} = \boldsymbol{q}'\frac{ds}{dt}. \tag{62}$$

Then, the kinetic energy becomes

$$\begin{aligned} T &= \frac{1}{2}\dot{\boldsymbol{q}}^\top \boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{q}} = \frac{1}{2}\left(\boldsymbol{q}'\frac{ds}{dt}\right)^\top \boldsymbol{M}(\boldsymbol{q})\left(\boldsymbol{q}'\frac{ds}{dt}\right) \\ &= \frac{1}{2}\left(\frac{ds}{dt}\right)^2 \boldsymbol{q}'^\top \boldsymbol{M}(\boldsymbol{q})\boldsymbol{q}'. \end{aligned} \tag{63}$$

By energy conservation,

$$H = T + P(\boldsymbol{q}) \quad \Rightarrow \quad T = H - P(\boldsymbol{q}). \tag{64}$$

Equating both expressions for T, we get:

$$\frac{1}{2}\left(\frac{ds}{dt}\right)^2 \boldsymbol{q}'^\top \boldsymbol{M}(\boldsymbol{q})\boldsymbol{q}' = H - P(\boldsymbol{q}). \tag{65}$$

Solving for dt gives:

$$dt = \sqrt{\frac{\boldsymbol{q}'^\top \boldsymbol{M}(\boldsymbol{q})\boldsymbol{q}'}{2(H - P(\boldsymbol{q}))}}\, ds. \tag{66}$$

The action integral becomes:

$$\begin{aligned} S_M(\boldsymbol{q}) &= \int \boldsymbol{p}^\top d\boldsymbol{q} = \int \boldsymbol{p}^\top \boldsymbol{q}'\, ds = \int \dot{\boldsymbol{q}}^\top \boldsymbol{M}(\boldsymbol{q})\boldsymbol{q}'\, ds \\ &= \int_{s_1}^{s_2} \left(\boldsymbol{q}'\frac{ds}{dt}\right)^\top \boldsymbol{M}(\boldsymbol{q})\boldsymbol{q}'\, ds \\ &= \int_{s_1}^{s_2} \left(\frac{ds}{dt}\right) \boldsymbol{q}'^\top \boldsymbol{M}(\boldsymbol{q})\boldsymbol{q}'\, ds. \end{aligned} \tag{67}$$

Substituting the expression for dt, we have:

$$\begin{aligned} S_M(\boldsymbol{q}) &= \int_{s_1}^{s_2} \sqrt{\boldsymbol{q}'^\top \left(2(H - P(\boldsymbol{q}))\boldsymbol{M}(\boldsymbol{q})\right)\boldsymbol{q}'}\, ds \\ &= \int_{s_1}^{s_2} \|\boldsymbol{q}'\|_{\boldsymbol{G}_{\text{Jac}}(\boldsymbol{q})}\, ds. \end{aligned} \tag{68}$$

Thus, energy-conserving paths correspond to geodesics under the Jacobi metric $\boldsymbol{G}_{\text{Jac}}(\boldsymbol{q})$, offering a geometric interpretation of conservative dynamics (Casetti et al. 2000).

A.3 Geodesics as Optimal Control Solutions

We now explore the geometric connection between energy conservation and optimal control strategies that minimize control input. Specifically, we show how minimizing control input leads to energy-preserving trajectories consistent with geodesic motion under the Jacobi metric. The time derivative of the Hamiltonian expression in (56) is

$$\begin{aligned} \dot{H} &= \frac{d}{dt}\left(\frac{1}{2}\dot{\boldsymbol{q}}^\top \boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{q}}\right) + \frac{d}{dt}P(\boldsymbol{q}) \\ &= \dot{\boldsymbol{q}}^\top \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \frac{1}{2}\dot{\boldsymbol{q}}^\top \dot{\boldsymbol{M}}(\boldsymbol{q})\dot{\boldsymbol{q}} + \dot{\boldsymbol{q}}^\top \nabla_{\boldsymbol{q}} P(\boldsymbol{q}). \end{aligned} \tag{69}$$

Substituting the dynamics from (47) with external torque $\boldsymbol{\tau}$:

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau}, \tag{70}$$

we obtain:

$$\begin{aligned} \dot{H} &= \dot{\boldsymbol{q}}^\top \left[\boldsymbol{\tau} - \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - \boldsymbol{g}(\boldsymbol{q})\right] + \frac{1}{2}\dot{\boldsymbol{q}}^\top \dot{\boldsymbol{M}}(\boldsymbol{q})\dot{\boldsymbol{q}} + \dot{\boldsymbol{q}}^\top \boldsymbol{g}(\boldsymbol{q}) \\ &= \dot{\boldsymbol{q}}^\top \boldsymbol{\tau} - \dot{\boldsymbol{q}}^\top \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \frac{1}{2}\dot{\boldsymbol{q}}^\top \dot{\boldsymbol{M}}(\boldsymbol{q})\dot{\boldsymbol{q}}. \end{aligned} \tag{71}$$

Using the skew-symmetry property for rigid-body systems (Spong et al. 2006)

$$\dot{\boldsymbol{q}}^\top \left(\dot{\boldsymbol{M}}(\boldsymbol{q}) - 2\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\right)\dot{\boldsymbol{q}} = 0, \tag{72}$$

we conclude that

$$\dot{H} = \dot{\boldsymbol{q}}^\top \boldsymbol{\tau}. \tag{73}$$

This equation represents the *instantaneous power input* to the system. When $\dot{H} = 0$, the torque $\boldsymbol{\tau}$ is orthogonal to the

velocity $\dot{\boldsymbol{q}}$, meaning it does not change total energy—only a redirection of motion, consistent with geodesic motion under the Jacobi metric (61). For a proof of the skew-symmetry property, see Appendix B.

From the perspective of classical optimal control, minimizing control effort is typically formulated as

$$\min_{\boldsymbol{\tau}} \int_0^T \boldsymbol{\tau}^\top \boldsymbol{R} \boldsymbol{\tau} \, \mathrm{d}t, \tag{74}$$

where $\boldsymbol{R} \succeq 0$ is a symmetric weighting matrix. Most often, this matrix penalizes control effort uniformly in all directions, i.e., the identity matrix. However, this formulation does not distinguish between torque that alters system energy and torque that merely redirects motion. To reflect the underlying physics more faithfully, we can consider minimizing the square of the instantaneous power input

$$\min_{\boldsymbol{\tau}} \int_0^T \dot{H}^2 \, \mathrm{d}t = \min_{\boldsymbol{\tau}} \int_0^T \left(\dot{\boldsymbol{q}}^\top \boldsymbol{\tau} \right)^2 \, \mathrm{d}t. \tag{75}$$

This criterion penalizes only the component of torque that performs mechanical work (i.e., the portion aligned with the system velocity). This aligns with the standard optimal control problem (74) with $\boldsymbol{R} = \dot{\boldsymbol{q}}\dot{\boldsymbol{q}}^\top$. In contrast to conventional ℓ^2 minimization that treats all directions equally, this criterion is more natural and physically grounded. As a result, geodesics under the Jacobi metric naturally emerge as solutions to this optimal control problem, highlighting the deep connection between energy-efficient control and geometric mechanics.

B Derivation of Skew-Symmetric Matrix

Assume the (i,j)-th element of the matrix $\dot{\boldsymbol{M}}(\boldsymbol{q}) - 2\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is denoted as n_{ij}. From Equations (49) and (51), we obtain:

$$\begin{aligned} n_{ij} &= \frac{\partial M_{ij}}{\partial q_k}\dot{q}_k - \left(\frac{\partial M_{ij}}{\partial q_k} + \frac{\partial M_{ik}}{\partial q_j} - \frac{\partial M_{jk}}{\partial q_i} \right) \dot{q}_k \\ &= \left(\frac{\partial M_{jk}}{\partial q_i} - \frac{\partial M_{ik}}{\partial q_j} \right) \dot{q}_k. \end{aligned} \tag{76}$$

By interchanging the indices i and j, we obtain:

$$n_{ji} = \left(\frac{\partial M_{ik}}{\partial q_j} - \frac{\partial M_{jk}}{\partial q_i} \right) \dot{q}_k = -n_{ij}. \tag{77}$$

Therefore, the matrix $\dot{\boldsymbol{M}}(\boldsymbol{q}) - 2\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is skew-symmetric.

C Laplace-Beltrami Operator

The Laplace-Beltrami operator is a second-order differential operator that generalizes the classical Laplace operator from Euclidean space to Riemannian manifolds. This extension is essential for analyzing functions on curved spaces, as it accounts for the geometry of the manifold. The Laplace-Beltrami operator allows us to compute the divergence of the gradient in more general spaces and is given by the expression

$$\Delta f = \frac{1}{\sqrt{|\boldsymbol{G}|}} \frac{\partial}{\partial_i} \left(\sqrt{|\boldsymbol{G}|} G^{ij} \frac{\partial}{\partial_j} f \right), \tag{78}$$

where G^{ij} are the components of the inverse of the metric tensor and $|\boldsymbol{G}|$ is the determinant of $\boldsymbol{G}$.

An alternative, more compact form of the Laplace-Beltrami operator can be derived from (78):

$$\begin{aligned} \Delta f &= \frac{1}{\sqrt{|\boldsymbol{G}|}} \frac{\partial}{\partial_i} \left(\sqrt{|\boldsymbol{G}|} G^{ij} \frac{\partial}{\partial_j} f \right) \\ &= (\frac{\partial}{\partial_i} G^{ij}) \frac{\partial}{\partial_j} f + G^{ij} \frac{1}{\sqrt{|\boldsymbol{G}|}} \left(\frac{\partial}{\partial_i} \sqrt{|\boldsymbol{G}|} \frac{\partial}{\partial_j} f \right) + G^{ij} \frac{\partial^2}{\partial_i \partial_j} f \\ &= (\frac{\partial}{\partial_i} G^{ij}) \frac{\partial}{\partial_j} f + G^{ij} \frac{1}{\sqrt{|\boldsymbol{G}|}} \left(\frac{1}{2} (\frac{\partial}{\partial_i} G_{kl}) G_{kl} \sqrt{|\boldsymbol{G}|} \frac{\partial}{\partial_j} f \right) + G^{ij} \frac{\partial^2}{\partial_i \partial_j} f \\ &= (\frac{\partial}{\partial_i} G^{ij}) \frac{\partial}{\partial_j} f + \frac{1}{2} G^{ij} (\frac{\partial}{\partial_i} G_{kl}) G^{kl} \frac{\partial}{\partial_j} f + G^{ij} \frac{\partial^2}{\partial_i \partial_j} f \\ &= G^{ij} \frac{\partial^2}{\partial_i \partial_j} f - G^{ij} \boldsymbol{\Gamma}^k_{ij} \frac{\partial}{\partial_k} f \\ &= G^{ij} \left(\frac{\partial^2}{\partial_i \partial_j} f - \boldsymbol{\Gamma}^k_{ij} \frac{\partial}{\partial_k} f \right). \end{aligned} \tag{79}$$

D Geometry-Aware Sampling

We present the Riemannian Manifold Metropolis-Adjusted Langevin Algorithm (RM-MALA) for geometry-aware sampling on the Riemannian manifold. A detailed algorithm is shown in Algorithm 2.

D.1 Target Probability Density Function (PDF)

In each tangent space, an infinitesimal space is induced by the Riemannian metric $d\mathcal{M}(\boldsymbol{q}) = \sqrt{|\boldsymbol{G}(\boldsymbol{q})|} d\boldsymbol{q}$, bridging the target probability density function (PDF) $\rho(\boldsymbol{q})$ with respect the Lebesgue measurement $d\boldsymbol{q}$, to the PDF $p(\boldsymbol{q})$ with respect to $d\mathcal{M}(\boldsymbol{q})$ by

$$\rho(\boldsymbol{q}) = p(\boldsymbol{q}) \sqrt{|\boldsymbol{G}(\boldsymbol{q})|}. \tag{80}$$

D.2 Sampling on the Riemannian manifold

The objective is to sample variables on the Riemannian manifold from the PDF $\rho(\boldsymbol{q})$, while taking into account the local geometric structure. Given the Riemannian metric tensor, we adopt the Metropolis-adjusted Langevin Monte Carlo algorithm on the Riemannian Manifold (Girolami and Calderhead 2011). The algorithm describes the Langevin diffusion process on the Riemannian manifold in a stochastic differential equation (SDE)

$$d\boldsymbol{q}(t) = \frac{1}{2} \tilde{\nabla}_{\boldsymbol{q}} \mathcal{L}(\boldsymbol{q}(t)) + d\tilde{\boldsymbol{b}}(t), \tag{81}$$

with $\tilde{\nabla}_{\boldsymbol{q}} \mathcal{L}(\boldsymbol{q}(t))$ representing the natural gradient equipped by the Riemannian metric tensor $\tilde{\nabla}_{\boldsymbol{q}} \mathcal{L}(\boldsymbol{q}) = \boldsymbol{G}^{-1}(\boldsymbol{q}) \, \nabla_{\boldsymbol{q}} \mathcal{L}(\boldsymbol{q})$, where

$$\mathcal{L}(\boldsymbol{q}) = \log \rho(\boldsymbol{q}). \tag{82}$$

The equation of the Brownian motion $d\tilde{\boldsymbol{b}}(t)$ is given by

$$\begin{aligned} d\tilde{\boldsymbol{b}}_i(t) = &\frac{1}{\sqrt{|\boldsymbol{G}(\boldsymbol{q}(t))|}} \sum_{j=1}^{D} \frac{\partial}{\partial \boldsymbol{q}_j} \left((\boldsymbol{G}^{-1}(\boldsymbol{q}(t)))_{ij} \sqrt{|\boldsymbol{G}(\boldsymbol{q}(t))|} \right) dt \\ &+ \left(\sqrt{\boldsymbol{G}^{-1}(\boldsymbol{q}(t))} d\boldsymbol{b}(t) \right)_i, \end{aligned} \tag{83}$$

where $\boldsymbol{b}(t)$ is the normal Brownian motion. Assuming $p(\boldsymbol{q})$ is a constant, the natural gradient is expressed using (80):

$$\tilde{\nabla}_{\boldsymbol{q}} \mathcal{L}(\boldsymbol{q}) = \boldsymbol{G}^{-1}(\boldsymbol{q}) \nabla_{\boldsymbol{q}} \sqrt{|\boldsymbol{G}(\boldsymbol{q})|}. \tag{84}$$

Algorithm 2: RM-MALA

Input: $\boldsymbol{G}(\boldsymbol{q})$: Riemannian metric function
$\mathcal{L}(\boldsymbol{q})$: Proposed likelihood function
n_{burn}: Burn-in steps
n_{sample}: Sample steps
ϵ: Step size
Output: Q: A set of sampled points $\boldsymbol{q}$
Initialization:
set Random points $\boldsymbol{q} \in [-\pi, \pi)$
Sampling Step:
for *i from 1 to* $n_{burn} + n_{sample}$ **do**
  Sample new point $\boldsymbol{q}_{\text{new}}$ through (85)
  if $\boldsymbol{q}_{new} \notin [-\pi, \pi)$ **then**
    $\boldsymbol{q}_{\text{new}} = \arctan2\left(\sin\left(\boldsymbol{q}_{\text{new}}\right), \cos\left(\boldsymbol{q}_{\text{new}}\right)\right)$
  Calculate the proposed log-likelihood $\log \mathcal{L}(\boldsymbol{q})$ and $\log \mathcal{L}(\boldsymbol{q}_{\text{new}})$ through (82)
  Calculate the transition likelihood $p\left(\boldsymbol{q}_{\text{new}}|\boldsymbol{q}\right)$ and $p\left(\boldsymbol{q}|\boldsymbol{q}_{\text{new}}\right)$ through (87)
  Calculate the acceptance ratio α through (88)
  Draw a random number $t \in [0, 1)$
  if $e^{\alpha} \geq t$ **then**
    $\boldsymbol{q} = \boldsymbol{q}_{\text{new}}$
    if $i \geq n_{burn}$ **then**
      Add $\boldsymbol{q}_{\text{new}}$ into set Q
  else
    $\boldsymbol{q}_{\text{new}}$ is not accepted

In (81), $d\tilde{\boldsymbol{b}}(t)$ defines the Brownian motion on the Riemannian manifold.

After applying the first-order Euler integration with the fixed step size ϵ to the SDE (81), we have

$$\boldsymbol{q}(t+1) = \boldsymbol{\mu}(\boldsymbol{q}(t), \epsilon) + \left(\epsilon\sqrt{\boldsymbol{G}^{-1}(\boldsymbol{q}(t))}\boldsymbol{z}(t)\right), \tag{85}$$

where $\boldsymbol{\mu}(\boldsymbol{q}(t), \epsilon)$ is the mean of the Gaussian distribution associated with the sampled variable

$$\begin{aligned}\boldsymbol{\mu}(\boldsymbol{q}(t), \epsilon) = \boldsymbol{q}_i(t) &+ \frac{\epsilon^2}{2}\left(\boldsymbol{G}^{-1}(\boldsymbol{q}(t))\nabla_{\boldsymbol{q}}\mathcal{L}(\boldsymbol{q}(t))\right)_i \\ &- \epsilon^2\sum_{j=1}^{D}\left(\left(\boldsymbol{G}^{-1}(\boldsymbol{q}(t))\right)\frac{\partial \boldsymbol{G}(\boldsymbol{q}(t))}{\partial \boldsymbol{q}_j}\boldsymbol{G}^{-1}(\boldsymbol{q}(t))\right)_{ij} \\ &+ \frac{\epsilon^2}{2}\sum_{j=1}^{D}\left(\boldsymbol{G}^{-1}(\boldsymbol{q}(t))\right)_{ij}\operatorname{Tr}\left(\left(\boldsymbol{G}^{-1}(\boldsymbol{q}(t))\right)\frac{\partial G(\boldsymbol{q}(t))}{\partial \boldsymbol{q}_j}\right)\end{aligned} \tag{86}$$

and $\boldsymbol{z}$ is a random variable sampled from the standard normal distribution $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \boldsymbol{I})$.

Finally, the probability of the sampled variable follows the Gaussian distribution

$$p\left(\boldsymbol{q}(t+1)|\boldsymbol{q}(t)\right) = \mathcal{N}\left(\boldsymbol{q}(t+1)|\boldsymbol{\mu}(\boldsymbol{q}(t), \epsilon),\ \epsilon^2\boldsymbol{G}^{-1}\left(\boldsymbol{q}(t)\right)\right). \tag{87}$$

The acceptance of the sampled variable is finally calculated with

$$\begin{aligned}\alpha = &\mathcal{L}(\boldsymbol{q}(t+1)) + \log p(\boldsymbol{q}(t)|\boldsymbol{q}(t+1)) - \mathcal{L}(\boldsymbol{q}(t)) \\ &- \log p(\boldsymbol{q}(t+1)|\boldsymbol{q}(t)).\end{aligned} \tag{88}$$