

Geometric Learning: Leveraging differential geometry for learning and control

Présentée le 1^{er} juillet 2024

Faculté des sciences et techniques de l'ingénieur
Laboratoire d'algorithmes et systèmes d'apprentissage
Programme doctoral en robotique, contrôle et systèmes intelligents

pour l'obtention du grade de Docteur ès Sciences

par

Bernardo FICHERA

Acceptée sur proposition du jury

Prof. M. Kamgarpour, présidente du jury
Prof. A. Billard, directrice de thèse
Prof. A. Albu-Schäffer, rapporteur
Prof. R. Calandra, rapporteur
Prof. C. Jones, rapporteur

Acknowledgements

“Do you know anything about machine learning. . . ?”

“I worked on reinforcement learning during my master’s thesis.”, I said.

“Never mind. Let’s start from scratch.”, and so we did.

People who have worked at LASA will likely have no trouble recognizing whom I am talking about. This was my first meeting with Prof. Aude Billard. She drew a picture in which crosses and circles, arranged randomly, were separated by a curvy line. . . often, we do not realize the significance of certain moments and others’ gestures. Looking back at those hours spent together talking about learning and kernels, I can still see the “sliding doors” that made this journey possible. Thank you, Aude.

From the beginning to the end of this journey, I would like to express my gratitude to the members of my thesis defense committee: Prof. Roberto Calandra, Prof. Alin Albu-Schäffer, Prof. Colin Jones, and Prof. Maryam Kamgarpour. Thank you for taking the time to read my work and for the beautiful discussion we had during the exam. Special thanks to Roberto for the detailed feedback that helped improve the thesis.

Quite something happened in the middle of this journey. I apologize if, dear reader, you will not find your name in these acknowledgments. In fact, nobody will. . . had you been part of this journey, I wish you could, once again, experience the beauty of certain moments. Otherwise, I hope you enjoy these few lines, which may bring a fleeting smile to your lips.

To all those with whom I. . . shared a “Garpez” in the office, swam and ran on the “good” French side of the lake, swam and ran on the “bad” Swiss side of the lake, spent lovely nights in between Cut and the lab, enjoyed the sweet and warm harmony of a spring night with whiskey and cigars, watched football, loved Barça-Real 2010 5-0, ate pizza, cooked pizza, dreamed about Inglewood, savored kebab, hit the gym, won at FIFA, lost at FIFA, drank Guinness, sipped a spritz by the lake from a “stolen” glass from Abordage, watched “Big Bang Theory”, watched “How I Met Your Mother”, enjoyed the beauty of C++, loved and shared music, experienced “La Scala”, watched “Dragon Ball”, engaged in endless conversations without finding a faith, drank good wine, coded and tuned controllers, played дурак, went to Migros, went to Esselunga, ran a marathon, had barbecues, regretted the past, struggled in the present and hoped for the future, discussed research with interest and curiosity, savored блины, worked on papers and projects into the night, went to Michelin Star restaurants, sang De André, suffered, tasted a terrible truffle in a camper, rejoiced, had a coffee, played "Scala 40" and fell asleep . . . together.

Thank you.

Lausanne, April 15, 2024

B. F.

Preface

“The structure of the world seems to be intimately tied to the deep mathematical concepts, concepts which were developed out of considerations rooted only in logic and the beauty of the form.”

These words were pronounced by Chen Ning Yang at the Nobel Prize award ceremony in 1954. Yang’s work on Non-Abelian Gauge theory, was, yet, another demonstration, in the 20th century, of how mathematics could lead the way, providing insights into the underlying structures of the physical world before they were directly observed. Mathematical formalism needed for these theories was developed by mathematicians decades earlier with no reference to physics. Without any claim of being anywhere near close to those minds that revolutionized physics in the past century, this thesis represents a sincere endeavor to investigate the utility of mathematical tools forged by earlier generations of mathematicians. Our aim is to uncover their latent potential for application within the realm of robotics.

While this research has been motivated by the conviction that a more sophisticated mathematical framework could significantly enhance the field of robotics, we admit that, in certain situations, we let ourselves be captivated, solely, by “the beauty of the form”.

Lausanne, January 4, 2024

B. F.

Abstract

In this thesis, we concentrate on advancing high-level behavioral control policies for robotic systems within the framework of Dynamical Systems (DS). Throughout the course of this research, a unifying thread weaving through diverse fields emerges, and that is the fundamental role played by differential geometry. This study delves into various realms of this mathematical framework, with three distinct projects at its core. The first work revolves around graph Laplacian-based embedding space reconstruction, followed by an exploration of chart-based geometry in the second project. The third project shifts its focus towards harmonic analysis in non-Euclidean spaces. These facets of differential geometry, while seemingly distinct, converge in their practical application within the realm of robotics, specifically in the domain of Dynamical Systems (DS) based robot motion generation. The first two projects employ differential geometry tools for the purpose of learning and clustering DS on Euclidean spaces, whereas the third project ventures into the potential domain of learning DS on non-Euclidean spaces, otherwise known as manifolds. Our investigation into a more sophisticated geometry-based formalism is directed not only at enhancing the expressivity and complexity of DS policies for navigating intricate and dynamic real-world scenarios but also at favouring a more profound comprehension of the practical application of rather abstract mathematical concepts in the field of robotics and machine learning.

We dedicate the first part of thesis to studying manifold learning techniques for clustering and learning of nonlinear DS characterized by multiple equilibrium points. As determining an analytical description of the dynamics is often difficult, data-driven approaches are preferred for identifying and controlling nonlinear DS with multiple equilibrium points. We focus on an unsupervised learning scenario where neither the number nor the type of dynamics is known. We propose a Graph-based spectral clustering method that takes advantage of a velocity-augmented kernel to connect data points belonging to the same dynamics, while preserving the natural temporal evolution. We study the eigenvectors and eigenvalues of the Graph Laplacian and show that they form a set of orthogonal embedding spaces, one for each sub-dynamics. We prove that there always exists a set of 2-dimensional embedding spaces in which the sub-dynamics are linear and n -dimensional embedding spaces where they are quasi-linear. We learn a diffeomorphism from the Laplacian embedding space to the original space and show that the Laplacian embedding leads to good reconstruction accuracy and a faster convergence compared to the state-of-the-art diffeomorphism-based approaches.

The second part of this thesis explores chart-based differential geometry to empower the learning of DSs. Drawing inspiration from Einstein's concept of a four-dimensional manifold

Abstract

in the context of space-time, we introduce a novel approach to learn non-linear DSs. In our method, the non-linearity of these systems does not emerge from external forces, but rather, it arises from the intrinsic curvature of the underlying space. Every d -dimensional DS is modeled as a damped harmonic oscillator on a given manifold. By learning the manifold's $d + 1$ -dimensional Euclidean embedded representation, our approach encodes non-linearity of the DS within the curvature of the space. Asymptotic stability to an equilibrium point of the learnt DS is always preserved, independently from the curvature of the space. Having at our disposal an explicit representation of the manifold, we propose a new method of performing convex and concave obstacle avoidance via direct local deformation of the space, without re-learning the DS. Our approach demonstrates superior performance with respect to the current state-of-the-art in the common metrics typically employed in learning of DSs. The proposed geometry-based approach not only enhances the efficiency of the learning process, enabling faster convergence, but it also introduces a novel framework that could be further exploited for configuration space learning, whether in the form of DS-based policies or intrinsic robot dynamics.

If the first and second part of the thesis found application in clustering and learning DS in Euclidean space, the third part of the thesis represents the first step towards bringing these applications to, potentially unknown and high-dimensional, manifolds. In this latest work, we concentrate on extending Gaussian process regression to non-Euclidean spaces. Gaussian process regression is widely used because of its ability to provide well-calibrated uncertainty estimates and handle small or sparse datasets. However, it struggles with high-dimensional data. One possible way to scale this technique to higher dimensions is to leverage the implicit low-dimensional manifold upon which the data actually lies, as postulated by the manifold hypothesis. Prior work ordinarily requires the manifold structure to be explicitly provided though, i.e. given by a mesh or be known to be one of the well-known manifolds like sphere. In contrast, we propose a Gaussian process regression technique capable of inferring implicit structure directly from data (labeled and unlabeled) in a fully differentiable way. Our technique scales up to hundreds of thousands of data points, and improves the predictive performance and calibration of the standard Gaussian process regression in high dimensional settings as well as complex non-Euclidean spaces.

Zusammenfassung

In dieser Dissertation konzentrieren wir uns auf die Weiterentwicklung von hochrangigen Verhaltenssteuerungspolitiken für Robotersysteme im Rahmen von Dynamischen Systemen (DS). Im Verlauf dieser Forschung tritt ein verbindendes Element durch verschiedene Bereiche hervor, nämlich die fundamentale Rolle der Differentialgeometrie. Diese Studie taucht in verschiedene Bereiche dieses mathematischen Rahmens ein, mit drei unterschiedlichen Projekten im Kern. Die erste Arbeit dreht sich um die Rekonstruktion des Einbettungsraums basierend auf dem Graph-Laplace, gefolgt von einer Erkundung der kartenbasierten Geometrie im zweiten Projekt. Das dritte Projekt verlagert seinen Fokus auf die harmonische Analyse in nicht-euklidischen Räumen. Diese Aspekte der Differentialgeometrie, obwohl scheinbar unterschiedlich, konvergieren in ihrer praktischen Anwendung im Bereich der Robotik, speziell im Bereich der robotergestützten Bewegungsgeneration basierend auf Dynamischen Systemen (DS). Die ersten beiden Projekte verwenden Werkzeuge der Differentialgeometrie zum Zweck des Lernens und Gruppierens von DS in euklidischen Räumen, während das dritte Projekt in das potenzielle Gebiet des Lernens von DS in nicht-euklidischen Räumen, auch bekannt als Mannigfaltigkeiten, vordringt. Unsere Untersuchung eines ausgefeilteren geometriebasierten Formalismus zielt nicht nur darauf ab, die Ausdrucksfähigkeit und Komplexität von DS-Politiken für das Navigieren in komplexen und dynamischen realen Szenarien zu verbessern, sondern auch auf ein tieferes Verständnis der praktischen Anwendung von eher abstrakten mathematischen Konzepten im Bereich der Robotik und des maschinellen Lernens.

Der erste Teil der Dissertation widmet sich dem Studium von Mannigfaltigkeitslern-Techniken für das Clustern und Lernen von nichtlinearen DS, die durch mehrere Gleichgewichtspunkte gekennzeichnet sind. Da eine analytische Beschreibung der Dynamik oft schwierig ist, werden datengetriebene Ansätze bevorzugt, um nichtlineare DS mit mehreren Gleichgewichtspunkten zu identifizieren und zu steuern. Wir konzentrieren uns auf ein unüberwachtes Lernszenario, in dem weder die Anzahl noch die Art der Dynamiken bekannt ist. Wir schlagen eine graphbasierte spektrale Clustermethode vor, die einen geschwindigkeitserweiterten Kernel nutzt, um Datenpunkte, die zur gleichen Dynamik gehören, zu verbinden und gleichzeitig die natürliche zeitliche Entwicklung zu bewahren. Wir untersuchen die Eigenvektoren und Eigenwerte des Graph-Laplacians und zeigen, dass sie eine Reihe orthogonaler Einbettungsräume bilden, jeweils einen für jede Subdynamik. Wir beweisen, dass es immer eine Reihe von 2-dimensionalen Einbettungsräumen gibt, in denen die Subdynamiken linear sind, und n -dimensionale Einbettungsräume, in denen sie quasi-linear sind. Wir lernen eine

Zusammenfassung

Diffeomorphismus vom Laplace-Einbettungsraum zum ursprünglichen Raum und zeigen, dass die Laplace-Einbettung zu einer guten Rekonstruktionsgenauigkeit und einer schnelleren Konvergenz im Vergleich zu den neuesten diffeomorphismusbasierten Ansätzen führt.

Der zweite Teil dieser Dissertation erforscht die kartenbasierte Differentialgeometrie, um das Lernen von DSs zu stärken. Inspiriert von Einsteins Konzept einer vierdimensionalen Mannigfaltigkeit im Kontext von Raum-Zeit, führen wir einen neuartigen Ansatz ein, um nicht-lineare DSs zu lernen. In unserer Methode entsteht die Nichtlinearität dieser Systeme nicht aus externen Kräften, sondern aus der intrinsischen Krümmung des zugrunde liegenden Raums. Jedes d -dimensionale DS wird als gedämpfter harmonischer Oszillator auf einer gegebenen Mannigfaltigkeit modelliert. Indem wir die $d+1$ -dimensionale euklidische eingebettete Darstellung der Mannigfaltigkeit lernen, kodiert unser Ansatz die Nichtlinearität des DS innerhalb der Krümmung des Raums.

Die asymptotische Stabilität zu einem Gleichgewichtspunkt des gelernten DS wird immer bewahrt, unabhängig von der Krümmung des Raums. Mit einer expliziten Darstellung der Mannigfaltigkeit zur Verfügung, schlagen wir eine neue Methode vor, um konvexe und konkave Hindernisvermeidung durch direkte lokale Verformung des Raums durchzuführen, ohne das DS neu zu lernen. Unser Ansatz zeigt eine überlegene Leistung im Vergleich zum aktuellen Stand der Technik in den üblicherweise verwendeten Metriken beim Lernen von DSs. Der vorgeschlagene geometriebasierte Ansatz verbessert nicht nur die Effizienz des Lernprozesses, was eine schnellere Konvergenz ermöglicht, sondern führt auch ein neuartiges Framework ein, das weiter für das Lernen des Konfigurationsraums genutzt werden könnte, sei es in Form von DS-basierten Politiken oder intrinsischen Roboterdynamiken.

Wenn der erste und zweite Teil der Dissertation Anwendung im Clustern und Lernen von DS im euklidischen Raum gefunden haben, stellt der dritte Teil der Dissertation den ersten Schritt dar, diese Anwendungen auf potenziell unbekannte und hochdimensionale Mannigfaltigkeiten zu übertragen. In dieser neuesten Arbeit konzentrieren wir uns darauf, die Gaußsche Prozessregression auf nicht-euklidische Räume zu erweitern. Die Gaußsche Prozessregression wird aufgrund ihrer Fähigkeit, gut kalibrierte Unsicherheitsschätzungen zu liefern und kleine oder spärliche Datensätze zu bewältigen, häufig verwendet. Sie hat jedoch Schwierigkeiten mit hochdimensionalen Daten. Ein möglicher Weg, diese Technik auf höhere Dimensionen zu skalieren, ist die Nutzung der impliziten niedrigdimensionalen Mannigfaltigkeit, auf der die Daten tatsächlich liegen, wie von der Mannigfaltigkeitshypothese postuliert. Frühere Arbeiten erfordern normalerweise, dass die Struktur der Mannigfaltigkeit explizit vorgegeben wird, d.h. durch ein Mesh gegeben oder als eine der bekannten Mannigfaltigkeiten wie die Kugel bekannt. Im Gegensatz dazu schlagen wir eine Gaußsche Prozessregressionstechnik vor, die in der Lage ist, die implizite Struktur direkt aus den Daten (beschriftet und unbeschriftet) auf eine vollständig differenzierbare Weise abzuleiten. Unsere Technik skaliert auf Hunderttausende von Datenpunkten und verbessert die Vorhersageleistung und Kalibrierung der standardmäßigen Gaußschen Prozessregression in hochdimensionalen Einstellungen sowie in komplexen nicht-euklidischen Räumen.

Résumé

Dans cette thèse, nous nous concentrons sur l'avancement des politiques de contrôle comportemental de haut niveau pour les systèmes robotiques dans le cadre des Systèmes Dynamiques (DS). Au cours de cette recherche, un fil conducteur unifiant émerge à travers divers domaines, à savoir le rôle fondamental joué par la géométrie différentielle. Cette étude explore divers domaines de ce cadre mathématique, avec trois projets distincts à son cœur. Le premier travail porte sur la reconstruction de l'espace d'incrustation basé sur le Laplacien du graphe, suivi d'une exploration de la géométrie basée sur les cartes dans le second projet. Le troisième projet déplace son focus vers l'analyse harmonique dans des espaces non euclidiens. Ces facettes de la géométrie différentielle, bien que distinctes en apparence, convergent dans leur application pratique dans le domaine de la robotique, spécifiquement dans le domaine de la génération de mouvement de robot basée sur les Systèmes Dynamiques (DS). Les deux premiers projets utilisent des outils de géométrie différentielle pour l'apprentissage et le clustering de DS dans des espaces euclidiens, tandis que le troisième projet se lance dans le domaine potentiel de l'apprentissage de DS dans des espaces non euclidiens, autrement connus sous le nom de variétés. Notre enquête sur un formalisme basé sur une géométrie plus sophistiquée vise non seulement à améliorer l'expressivité et la complexité des politiques DS pour naviguer dans des scénarios réels complexes et dynamiques, mais aussi à favoriser une compréhension plus profonde de l'application pratique de concepts mathématiques plutôt abstraits dans le domaine de la robotique et de l'apprentissage automatique.

Nous consacrons la première partie de la thèse à l'étude des techniques d'apprentissage sur les variétés pour le clustering et l'apprentissage de DS non linéaires caractérisés par de multiples points d'équilibre. Comme il est souvent difficile de déterminer une description analytique de la dynamique, les approches basées sur les données sont préférées pour identifier et contrôler les DS non linéaires avec plusieurs points d'équilibre. Nous nous concentrons sur un scénario d'apprentissage non supervisé où ni le nombre ni le type de dynamiques ne sont connus. Nous proposons une méthode de clustering spectral basée sur les graphes qui tire profit d'un noyau augmenté de vitesse pour connecter les points de données appartenant à la même dynamique, tout en préservant l'évolution temporelle naturelle. Nous étudions les vecteurs propres et les valeurs propres du Laplacien du graphe et montrons qu'ils forment un ensemble d'espaces d'incrustation orthogonaux, un pour chaque sous-dynamique. Nous prouvons qu'il existe toujours un ensemble d'espaces d'incrustation bidimensionnels dans lesquels les sous-dynamiques sont linéaires et des espaces d'incrustation n -dimensionnels où elles sont quasi-linéaires. Nous apprenons un difféomorphisme de l'espace d'incrusta-

Résumé

tion Laplacien vers l'espace original et montrons que l'incrustation Laplacienne conduit à une bonne précision de reconstruction et à une convergence plus rapide par rapport aux approches basées sur le difféomorphisme de l'état de l'art.

La deuxième partie de cette thèse explore la géométrie différentielle basée sur les cartes pour renforcer l'apprentissage des DS. S'inspirant du concept d'Einstein d'une variété quadri-dimensionnelle dans le contexte de l'espace-temps, nous introduisons une approche novatrice pour apprendre les DS non linéaires. Dans notre méthode, la non-linéarité de ces systèmes n'émerge pas de forces externes, mais plutôt, elle résulte de la courbure intrinsèque de l'espace sous-jacent. Chaque DS d -dimensional est modélisé comme un oscillateur harmonique amorti sur une variété donnée. En apprenant la représentation embarquée euclidienne $d+1$ -dimensionnelle de la variété, notre approche encode la non-linéarité du DS dans la courbure de l'espace. La stabilité asymptotique

à un point d'équilibre du DS appris est toujours préservée, indépendamment de la courbure de l'espace. Ayant à notre disposition une représentation explicite de la variété, nous proposons une nouvelle méthode pour effectuer l'évitement d'obstacles convexes et concaves via une déformation locale directe de l'espace, sans réapprendre le DS. Notre approche démontre une performance supérieure par rapport à l'état de l'art actuel dans les métriques couramment utilisées dans l'apprentissage des DS. L'approche basée sur la géométrie améliore non seulement l'efficacité du processus d'apprentissage, permettant une convergence plus rapide, mais introduit également un cadre novateur qui pourrait être davantage exploité pour l'apprentissage de l'espace de configuration, que ce soit sous la forme de politiques basées sur les DS ou de dynamiques intrinsèques des robots.

Si la première et la deuxième partie de la thèse ont trouvé application dans le clustering et l'apprentissage des DS dans l'espace euclidien, la troisième partie de la thèse représente le premier pas vers l'apport de ces applications à des variétés potentiellement inconnues et de haute dimension. Dans ce dernier travail, nous nous concentrons sur l'extension de la régression par processus gaussien à des espaces non euclidiens. La régression par processus gaussien est largement utilisée en raison de sa capacité à fournir des estimations d'incertitude bien calibrées et à gérer des ensembles de données petits ou épars. Cependant, elle peine avec des données de haute dimension. Une manière possible d'adapter cette technique à des dimensions plus élevées est de tirer parti de la variété à faible dimension implicite sur laquelle les données reposent réellement, comme postulé par l'hypothèse de la variété. Les travaux antérieurs nécessitent généralement que la structure de la variété soit explicitement fournie, c'est-à-dire donnée par un maillage ou connue pour être l'une des variétés bien connues comme la sphère. En revanche, nous proposons une technique de régression par processus gaussien capable de déduire la structure implicite directement à partir des données (étiquetées et non étiquetées) de manière entièrement différentiable. Notre technique monte en charge jusqu'à des centaines de milliers de points de données, et améliore la performance prédictive et la calibration de la régression par processus gaussien standard dans des paramètres de haute dimension ainsi que dans des espaces non euclidiens complexes.

Contents

Acknowledgements	i
Preface	iii
Abstract (English/Français/Deutsch)	v
List of Figures	xv
List of Tables	xxi
Introduction	1
1 Background	17
1.1 Manifolds, Coordinate Charts and Smooth Embeddings	18
1.2 Manifold learning: a differential geometry perspective	20
1.2.1 Learning the embedding: Diffusion Maps	21
1.3 Geometrical Dynamical Systems	24
1.4 Gaussian Process Regression on Manifolds	26
2 Linearization and Identification of Multiple-Attractor Dynamical Systems through Laplacian Eigenmaps	29
2.1 Foreword	29
2.2 Introduction	29
2.2.1 Learning Stable Dynamical Systems	30
2.2.2 Learning Dynamical Systems with Multiple Attractors	31
2.2.3 Manifold Learning for Latent Embedding Spaces of Dynamical Systems	32
2.3 Background & Problem Formulation	33
2.4 Graph Embedding and Linearization of Dynamical Systems	34
2.4.1 Determining the Eigenvectors that Generate a Linear Embedding	35
2.4.2 Creating a Linear Embedding	38
2.5 Generating the Graph with Real Data	41
2.5.1 Velocity-Augmented Kernel	41
2.5.2 Selecting the Hyperparameters	43
2.6 Clustering Dynamics & Attractor Search	43
2.7 Results	46

Contents

2.8	Dynamical System Learning via NVP Transformations	52
2.9	Conclusion	57
3	Learning Dynamical Systems Encoding Non-Linearity within Space Curvature	59
3.1	Foreword	59
3.2	Introduction	59
3.3	Related work	63
3.4	Background	64
3.5	Learning the Latent Manifold Embedding	66
3.5.1	Gradient Systems & Incremental Learning	69
3.6	Online Kernel-Based Local Space Deformation	69
3.6.1	Obstacle Avoidance via Direct Space Deformation	70
3.6.2	Local Space Deformation In First Order DS	72
3.6.3	Local Space Deformation In Second Order DS	73
3.7	Synthetic Example	76
3.7.1	Locally Active Space Deformation	79
3.7.2	Directional & Exponential Dissipation	81
3.7.3	Obstacle Avoidance Online Direct Deformation	82
3.7.4	Evaluation	84
3.8	Robotics Experiments	86
3.8.1	Trajectory Tracking Evaluation	88
3.9	Conclusion	88
4	Implicit Manifold Gaussian Process Regression	91
4.1	Foreword	91
4.2	Introduction	91
4.2.1	Related Work and Contribution	92
4.3	Gaussian Processes	93
4.3.1	Matérn Gaussian Processes on Explicit Manifolds and Graphs	94
4.4	Implicit Manifolds and Gaussian Processes on Them	95
4.4.1	Background on Approximating the Eigenpairs of the Laplace–Beltrami Operator	95
4.4.2	Approximating Matérn Kernels on Manifolds	97
4.4.3	Implicit Manifold Gaussian Process	97
4.5	Efficient Training of the Implicit Manifold Gaussian Processes	99
4.5.1	Noiseless Supervised Learning	100
4.5.2	Noiseless Semi-Supervised Learning	101
4.5.3	Handling Noisy Observations	102
4.5.4	Resulting Algorithm	102
4.6	Experiments	103
4.6.1	Synthetic Examples	103
4.6.2	High Dimensional Datasets	104
4.7	Conclusion	105

5 Conclusion and Future Developments	107
5.1 Main Contributions	107
5.2 Limitations & Future Developments	108
A Appendix of Chapter 1	111
A.1 Proof of Lem. 1 from Sec 2.4	111
A.2 Proof of Lem. 2 from Sec 2.4	111
A.3 Proof of Prop. 2 from Sec 2.4	112
A.4 Proof of Prop. 3 from Sec 2.4	113
A.5 Proof of Prop 4 from Sec 2.4	114
B Appendix of Chapter 2	117
B.1 Proofs of Prop. 5 & Thm. 2	117
B.1.1 Proof of Prop. 5	117
B.1.2 Proof of Thm. 2	118
B.2 Kernel-Based Space Deformation	118
B.2.1 Derivation of the Metric Tensor	119
B.2.2 Derivative of the Metric Tensor	119
B.3 Ablation Study	119
C Appendix of Chapter 3	121
C.1 Theory	121
C.2 Additional Experimental Results and Details	124
C.2.1 1D Dumbbell Manifold	124
C.2.2 2D Dragon Manifold	125
C.2.3 High Dimensional Datasets	126
C.3 Hyperparameter Priors and Initialization	128
Bibliography	140
Curriculum Vitae	141

List of Figures

1	Differential Geometry in Machine Learning & Robotics; highlighted in yellow the areas touched in this thesis.	2
2	(left) 3D representation of the Earth; (center) chart map representing a portion of the earth in 2D; (right) graph structure approximating the Earth.	4
3	(left) A planar robotic arm tracking a specific streamline, highlighted in green, sampled from a 2D dynamical system. The other sampled streamlines are depicted in black, providing a visual representation of the potential 'flow' paths the robot would follow if it were positioned along these alternative trajectories. (right) Kinesthetic teaching, where the robot is physically guided through the task by the human.	5
4	One sampled streamline from a geometrical DS on a flat Euclidean space and on the sphere. The avoidance of obstacles is achieved via local deformation of the space (not visible in the picture). The color gradient represent the potential function driving the DS towards the goal (the star).	7
5	dragon	11
1.1	The differential manifold philosophy.	18
1.2	Manifold learning working flow.	20
1.3	Contraction metric reconstruction via pullback of the Euclidean embedding metric.	25
2.1	Schematic representation of a multiple-attractor DS. Each sub-dynamics \mathbf{f}_q takes within a sub-space \mathcal{B}_q and converges towards the attractor \mathbf{x}_q^*	33
2.2	Each sub-dynamics is embedded in a graph where each trajectory forms a path connected by a set of termination nodes that form a cyclic path around the attractor. $v_1^k v_{p_k}^k$ are the first and last of the k -th path graphs, respectively. p_k is the number of nodes with the k -th path graph.	34
2.3	Coordinates of three points on two eigenvectors with constant rate of change, resulting in a linear path.	38
2.4	(a) Toy graph composed by 3 path graphs connected each other with a cycle graph passing through nodes {5, 10, 15}; (b) Spectrum of the first 9 eigenvalues of the Laplacian applied to graph in Fig. 2.4a; (c) Entries of the first 2 components $\{\mathbf{u}^2, \mathbf{u}^3\}$; (d) 2D embedding space reconstructed using components $\{\mathbf{u}^2, \mathbf{u}^3\}$. . .	39

List of Figures

2.5	(a) Toy graph composed by 4 path graphs connected each other with a cycle graph passing through nodes {5, 10, 15, 20}; (b) Spectrum of the first 9 eigenvalues of the Laplacian applied to graph in Fig. 2.5a; (c) 2D embedding space reconstructed using components $\{\mathbf{u}^2, \mathbf{u}^3\}$; (d) 3D embedding space reconstructed using components $\{\mathbf{u}^2, \mathbf{u}^3, \mathbf{u}^4\}$	40
2.6	(a) Toy graph composed by 5 path graphs connected each other with a cyclic graph passing through nodes {5, 10, 15, 20, 25}; (b) Spectrum of the first 9 eigenvalues of the Laplacian applied to graph in Fig. 2.6a; (c) 2D embedding space reconstructed using components $\{\mathbf{u}^2, \mathbf{u}^3\}$; (d) 2D embedding space reconstructed using components $\{\mathbf{u}^4, \mathbf{u}^5\}$	40
2.7	Illustration of the effect of the spatial distribution of points and velocity vectors on the kernel Eq. 2.20: (a) the first term in the exponential generates low values for points that are far apart; (b) the third term in the exponential generates low values when consecutive velocity vectors are not aligned; (c) the second and third term in the exponential generate low values whenever the distance vector connecting two points is not aligned with the velocity vector of one of them. . .	42
2.8	(a) Background vector field and reference evaluation point/velocity; (b) Contour of the velocity-augmented kernel.	42
2.9	Toy data set of 2-attractor samples DS.	43
2.10	Structure of the constant right eigenvectors associated to eigenvalues equal to 0.	44
2.11	(left) $\{\mathbf{u}_3, \mathbf{u}_6\}$ embedding space; (center) Scatter matrix of the embedding space for the DS in 2.9; (right) $\{\mathbf{u}_4, \mathbf{u}_5\}$ embedding space.	44
2.12	Spectrum analysis of the multiple-attractor DS for three demonstrated trajectories and increasing number of attractors.	45
2.13	Finding the attractor in the embedding space.	46
2.14	(a) Vector field generated by Eq. 2.24. (b) Sampled trajectories from the DS in Fig.2.14a. (c) Embedding space of the sub-dynamics with local attractor in $(1, -2)$. (d) Embedding space of the sub-dynamics with local attractor in $(1, 2)$. . .	47
2.15	Clustering results of Tab. 2.1a.	48
2.16	(a) Vector field generated by Eq. 2.25. (b) Sampled trajectories from the DS in Fig.2.16a. (c) Embedding space of the sub-dynamics with local attractor in $(0, 0)$. (d) Embedding space of the sub-dynamics with local attractor in $(0, 2\pi)$	48
2.17	Clustering results of Tab. 2.2a.	49
2.18	(a) Vector field generated by Eq. 2.27. (b) Sampled trajectories from the DS in Fig.2.18a. (c) Embedding space of the sub-dynamics with local attractor in $(0, \sqrt{-\alpha/\beta})$. (d) Embedding space of the sub-dynamics with local attractor in $(0, -\sqrt{-\alpha/\beta})$	50
2.19	Clustering results of Tab. 2.3a.	50
2.20	(a) Demonstrated trajectories. (b) Embedding space of the red sub-dynamics. (d) Embedding space of the cyan sub-dynamics.	51
2.21	Clustering results of Tab. 2.4a.	51

2.22	Quasi-zero velocity heuristic for attractor location. Threshold at (a) 10%, (b) 5% and (c) 1% of the average velocity.	52
2.23	(left) Original Euclidean Space where the dataset is sampled from; (center) manifold where the DS is taking place; (right) extracted Euclidean embedding space.	52
2.24	(a) Linear DS generated when using identity diffeomorphism; (b) reconstructed dynamics through learned diffeomorphism; (c) initial quadratic potential function generating a linear DS in Fig. 2.24a; (d) deformed potential under the action of the learned diffeomorphism generating the nonlinear DS in Fig. 2.24b.	54
2.25	(a) Linear DS generated via identity diffeomorphism; (b) deformed DS under the action of the learned diffeomorphism; (c) initial quadratic potential function generating the linear DS in Fig. 2.25a; (d) deformed potential under the action of the learned diffeomorphism generating the nonlinear DS in Fig. 2.25b.	54
2.26	For Laplacian Embedding (LE) and Euclideanizing Flows (EF) approaches: (a) loss decay over 1000 epochs; (b) training time for 1000 epochs.	55
2.27	For each demonstrated DS (column wise), the first row shows the DS generated by the identity diffeomorphism (no train), the second row shows the DS generated after having learned the diffeomorphism between the original space and the embedding space, the third row shows the embedding space reconstructed using the eigenvectors extracted from the eigen-decomposition of the Laplacian matrix.	55
2.28	Deformed potential under the action of the learned diffeomorphism generating the nonlinear DS for the LASA dataset.	56
2.29	For Laplacian Embedding (LE) and Euclideanizing Flows (EF) approaches box plots comparison of DTWD and CS metrics.	57
3.1	End-effector motion of a robotic arm guided by a 2D learned DS. The surface corresponds to the 3D Euclidean space embedded representation of the learnt 2D manifold. The color gradient represents the value of the potential function that drives the linear vector field taking place on the 2D manifold. The manifold's curvature induces "apparent" non-linearity in the 2D chart Euclidean space representation of the vector field taking place on the 2D manifold. During the learning process, the curvature of the manifold adapts so that the streamlines of the 2D chart Euclidean space representation of the vector field follow closely the demonstrated trajectories (red dots), preserving the stability towards a desired equilibrium point (yellow star).	60
3.2	Mapping structure across manifolds.	66
3.3	First-order DS in flat space with localized deformation in the obstacle area: (a) Vector field with one sampled streamline avoiding the obstacle; (b) 3D embedded representation of the manifold; (c) one sampled trajectory with eigenvalue decomposition ellipses of the inverse of the metric for selected location; (d) metric determinant function with eigenvalue decomposition ellipses of the metric.	72
3.4	Kernel-based metric tensor eigenvalues and eigenvectors.	73

List of Figures

3.5	(a)-(c) Geodesic motion at time instants: 1s, 5s and 10s; (d) 3D embedded representation of case (c). In background the color gradient represents the $d+1$ embedding coordinate.	74
3.6	Geodesic motion after 1s with ellipses representing eigenvalues and eigenvectors of (a) the inverse of the metric and (b) the Christoffel symbols for selected locations.	74
3.7	Second-order DS convex obstacle avoidance: (a) velocity independent local deformation; (b) velocity dependent deformation.	75
3.8	Concave obstacle avoidance: (a) harmonic motion; (b) geodesic motion; (c)-(d) hybrid motion for semicircle and horseshoe obstacles.	75
3.9	(a) 2-joints planar robotic arm performing a straight point-to-point motion of the end-effector; (b) corresponding trajectories in configuration space starting from different initial states.	76
3.10	From top to bottom 3D embedded representation of the latent manifold, chart space representation of the DS field induced by the manifold curvature and metric tensor's determinant and ellipsoids for selected locations: (center) before learning; (left) learned first-order DS; (right) learned second-order DS.	77
3.11	Learned potential function with stiffness and dissipation matrices principal direction ellipsoids for (a) first-order and (b) second-order DS.	78
3.12	Evolution of (a) inverse metric and (b) Christoffel symbols principal directions along one sampled streamline.	79
3.13	(a) Gradient of the $d+1$ embedding component; (b) gradient of the $d+1$ embedding component with bump function; (c) one sampled trajectory with initial position far away from the demonstrated trajectories; (d) one sampled trajectory with initial position far away from the demonstrated trajectories with bump function; (e) metric determinant with eigenvalue decomposition ellipses of the metric from selected location.	79
3.14	Second-order DS (a) without, (b)-(c) with ($\lambda = 10$ and $\lambda = 20$) directional dissipation.	81
3.15	Vector field and metric determinant for (a)-(d) first-order DS with classic method; (b)-(e) first-order DS with direct deformation method; (c)-(f) second-order DS with direct deformation method.	82
3.16	Embedding space visualization of the local deformation due to the obstacle presence for (a) first-order DS and (b) second-order DS.	83
3.17	Evaluation: (top) Motion frames of 2-joints planar robotic manipulator; (bottom) sampled DSs in configuration space, corresponding to the above depicted robotic motion.	84
3.18	Learning three dimensional Dynamical Systems for Robotic Arm Control. . . .	86
3.19	Control Structure.	87

4.1	<i>Euclidean</i> (standard Matérn- $5/2$ kernel) vs <i>ours</i> (implicit manifold) Gaussian process regression for data that lies on a dumbbell-shaped curve (1-dimensional manifold) assumed unknown. The data contains a small set of labeled points and a large set of unlabeled points. Our technique recognizes that the two lines in the middle are intrinsically far away from each other, giving a much better model on and near the manifold. Far away from the manifold it reverts to the Euclidean model.	92
4.2	Kernel values $k(\cdot, \cdot)$ and samples for the Matérn- $3/2$ Gaussian processes on the sphere manifold \mathbb{S}_2 and for the approximating Matérn- $5/2$ process on a geodesic polyhedron graph $\mathbb{P}\mathbb{S}_2$	94
4.3	Different quantities connected to kernel extension. Notice that the values on sub-figures (a) and (b) are artificially restricted to the set $\text{dist}(\cdot, \mathcal{M}) < 3\alpha$ to maintain numerical stability.	99
4.4	The ground truth function on the dumbbell manifold and the predictions of the implicit manifold Gaussian process regression (IMGP) under different levels of noise.	103
C.1	Root Mean Square Error (RMSE) and Negative Log-Likelihood (NLL) for increasing number of eigenpairs L (left panels) and increasing fraction $f = n\%N$ of labeled points (right panels). The legend in (a) and (b) refers to the number of hyperparameter optimization iterations.	125
C.2	(a) Ground truth function on the complex 2D manifold and (b)-(d) predictions of the implicit manifold Gaussian process regression (IMGP) for increasing level of sampling noise β . (e)-(d) Euclidean GP prediction and uncertainty and (g) IMGp uncertainty in noiseless scenario.	126
C.3	The histogram of \mathcal{D} and the prior for the bandwidth hyperparameter α	128

List of Tables

1.1	Operators philosophy.	23
2.1	For sampled points in Fig. 2.14b: (a) Clustering labeling and attractor location error results, (b) Kernel K-Means clustering error over iteration, (c) Gaussian Mixture Model clustering error over iteration.	47
2.2	For sampled points in Fig. 2.16b: (a) Clustering labeling and attractor location error results, (b) Kernel K-Means clustering error over iteration, (c) Gaussian Mixture Model clustering error over iteration.	49
2.3	For sampled points in Fig. 2.18b: (a) Clustering labeling and attractor location error results, (b) Kernel K-Means clustering error over iteration, (c) Gaussian Mixture Model clustering error over iteration.	50
2.4	For sampled points in Fig. 2.20a: (a) Clustering labeling and attractor location error results, (b) Kernel K-Means clustering error over iteration, (c) Gaussian Mixture Model clustering error over iteration.	51
2.5	Performance evaluation at reconstructing the demonstrations for each of the 12 handdrawn examples of Figure 2.27. Performance is measured according to three metrics: root mean square error (RSMR), dynamic time warping distance (DTWD) and cosine similarity error (CS).	56
3.1	Evaluation results: for each sampled DS in Figure 3.17 we evaluate 1st and 2nd order learning DS against the Baseline with respect to the three metrics RMSE, Cosine Similarity Kernel in velocity space, DTWD.	85
3.2	Training results.	85
3.3	Experimental results.	88
4.1	Performance metrics for the dumbbell manifold with varying magnitude of noise β	104
4.2	Negative log likelihood on test samples for real datasets. For RMSE see Tables C.2 and C.3.	105
B.1	Ablation Study for the Neural Network function approximator.	120
C.1	Results for a complex 2D manifold with varying magnitude of sampling noise.	125
C.2	Results for the rotated MNIST dataset.	127

List of Tables

C.3 Results for Relative location of CT slices on axial axis ($d = 385$, $N = 48150$) from UCI Machine Learning Repository. 128

Introduction

Is mathematics discovered or invented? Galileo Galilei, often hailed as the father of the scientific method, in his work “Il Saggiatore” (1624) wrote: “The book of nature is written in the language of mathematics, with its characters being triangles, circles, and other geometrical figures. Without these elements, understanding the world is akin to aimlessly wandering through a dark labyrinth.” Galilei, a devout believer, was convinced that God articulated natural truths in the precise language of mathematics. To him, mathematics, as the language of God, was not just a tool but an eternal truth in itself.

The notion of mathematics as eternal truth persisted well after the “death of God”, proclaimed by Nietzsche at the end of the 19th century. Even as human thought increasingly moved away from the concept of any eternal truth, including God, mathematics continued to be viewed as a domain of certainty and, consequently, truth. The prevailing belief was that any imperfections in current theories could be rectified through rigorous proof and self-consistency. However, this perception was profoundly challenged in 1931 with Kurt Gödel’s Incompleteness Theorems. These theorems demonstrated that even in the most sophisticated mathematical systems, there will always be true statements that cannot be proven. This revelation was a seismic event in the academic world, similar in magnitude to the shift in geometry where Euclidean geometry, once seen as the sole truth, became just one of many geometrical systems alongside non-Euclidean geometries. Mathematics was thus stripped of its perceived incorruptibility and eternity.

While these developments seem to shift the answer to the original question from “discovered” to “invented”, an intriguing phenomenon in the realm of science has emerged. Traditionally, science was driven by induction —deriving universal laws from specific experimental observations. However, in the 20th century, this process started to move back from induction to deduction of Aristotelian memory. Theories grounded in mathematical language began to offer insights beyond just explaining natural phenomena; they started to “suggest” new experimental content. Differential geometry appears to occupy a unique position in this context. In 1950, Murray Gell-Mann postulated the existence of quarks and anti-quarks, not through empirical evidence, but based on their correspondence with the two fundamental three-dimensional representations of the Special Unitary group of dimension three, $SU(3)$. These particles had never been observed before; their existence was predicted based on mathematical symmetry patterns. Similarly, gravitational waves, first postulated

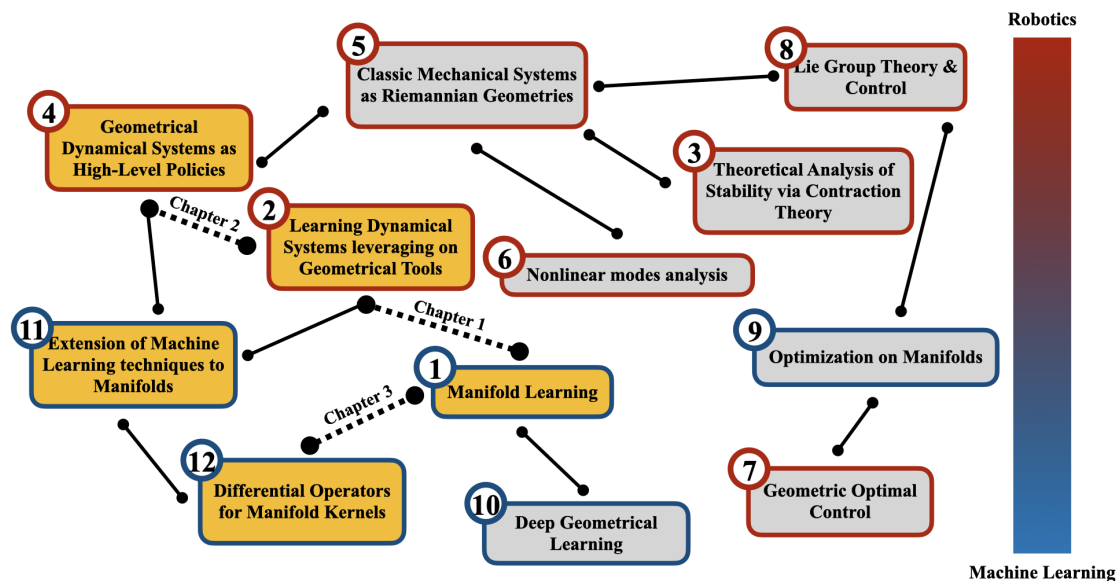


Figure 1: Differential Geometry in Machine Learning & Robotics; highlighted in yellow the areas touched in this thesis.

by Einstein's General Theory of Relativity in 1915, were only experimentally confirmed much later, in 2015, by the Laser Interferometer Gravitational-Wave Observatory (LIGO).

A Journey into Differential Geometry Applications

In the 20th century, differential geometry marked a conceptual advance in understanding the structure of the universe. Today, its remarkable representational power continues to unfold across diverse fields. This thesis explores the use of differential geometry within the realms of Machine Learning and Robotics. Our approach is twofold: firstly, to address existing limitations in these fields through the lens of differential geometry, and secondly, to look at a unified geometrical and mathematically formal framework that enhances our comprehension of these domains. This endeavor is guided by the belief that a deeper understanding of the subject will lead to new, applicable knowledge that is crucial for the next technological innovations.

In preparation for the topics covered in this thesis, it is essential to introduce the concept of a *graph*, a recurring element in our discussion. In mathematics, a graph is a structure defined by two primary components: nodes and edges. Nodes can represent a wide range of entities, from simple spatial points to complex abstract concepts. Edges express some form of connection or relationship between the nodes. Figure 1 is conceptualized around the idea of a graph. Here, each node symbolizes a specific area either in Machine Learning or Robotics that has been significantly influenced by differential geometry principles. A red frame around a node indicates a predominant application in Robotics, while a blue frame

signifies an application in Machine Learning. The core areas of this thesis are highlighted in yellow. The numbering refers to the order of appearance throughout this introduction.

A probabilistic viewpoint common in graph theory, treats graph construction as analogous to a roadmap. Nodes are akin to distinct locations or checkpoints, linked together by the edges that enable movement between them. This perspective sets the stage for our journey, as we explore the nodes illustrated in Figure 1, navigating through the domains of Machine Learning and Robotics where differential geometry has been influential. Our exploration will primarily concentrate on those nodes directly relevant to this work. Yet, we will also glimpse into other applications of differential geometry. These areas, while not the main focus of our study, provide the foundational knowledge essential for the development of the projects detailed in this thesis and might offer potential linkages for future expansions of our research. The solid lines connecting nodes represent the well-established interrelations among different disciplines, brought together through the lens of differential geometry. Meanwhile, the dashed lines signify the emerging connections we intend to develop and delve into throughout this thesis. In this introduction, we will refer to the nodes in Figure 1 using the symbol for areas within the Machine Learning field, and for those pertaining to the Robotics field. Consistent with the color scheme previously described, these nodes will be highlighted in yellow to denote areas explicitly addressed in this thesis.

Manifold Learning: representation is all that matters...

Our journey begins at the heart of the graph in Figure 1, focusing on a pivotal concept in differential geometry: *representation*. In today's world, it's commonplace to see Artificial Intelligence algorithms accurately classify images of people, objects, animals, and more. Classification here refers to the ability to generate a consistent output for a semantically similar input. For example, when an algorithm is presented with a picture of a cat, it consistently assigns a label or number that has been predetermined to represent the concept of a "cat". Each image is represented by a large vector that captures the color of each pixel. The process of training these algorithms involves feeding them millions of such vectors and gradually adjusting internal parameters until they consistently produce the desired output. While current computational power facilitates this mechanism, it raises some questions. How do humans recognize a cat in an image without needing millions of examples? Does our brain process images at the pixel level or recognize some more fundamental and simpler structure for easier and more effective learning and inference? The prevailing belief is that **the complexity and unstructured nature of raw data can hinder learning performance**. In the field of Unsupervised Learning, the subfield known as "**Manifold Learning**"—node —**emerged as an attempt to improve data representation**, aiming at extracting relevant features or fundamental structures, on top which, learning strategies could be more effective. Although its name hints at its roots in differential geometry, well-known Manifold Learning techniques like ISOMAP, Tenenbaum et al. (2000), or Graph Laplacian, Belkin and Niyogi (2002, 2003), are often simply categorized as dimensionality reduction strategies. This classification, focused

Introduction



Figure 2: (left) 3D representation of the Earth; (center) chart map representing a portion of the earth in 2D; (right) graph structure approximating the Earth.

more on the effect than the cause, obscures the true nature of these algorithms and, along with it, a whole range of potential applications.

Having already introduced the concept of a graph, we now turn to another fundamental element in differential geometry: the *manifold*. Instead of delving into its formal and abstract mathematical definition, let us understand a manifold in a more intuitive way. Essentially, it is a structure where the distance between two points is not measured by the length of a straight line connecting them. Consider the Earth as an example, Figure 2 on the left. To measure the (minimum) distance between the South and North Poles, one must calculate the length of a meridian, which is not a straight line. This is due to the “curvature” that characterizes the Earth’s surface, which is approximately spherical. A graph, as previously defined, can be seen as a discrete, approximated representation of a manifold, Figure 2 on the right. In Manifold Learning techniques, starting from our raw dataset, the underlying hypothesis is that each sample lies on an unknown manifold. These techniques construct a graph over the raw data and, rather than directly learning the manifold as the name might imply, they learn an alternative representation of it with some desirable features. Sometimes this representation has a lower dimension than the original space, making these methods useful in situations where dimensionality reduction is needed for computational efficiency. Other times, the new representation better emphasizes differences or similarities among samples, aiding in clustering tasks, for example. The effectiveness of the result is largely dependent on how the graph is constructed. In the first chapter of this thesis, we will explore this aspect in depth, using it to identify and learn *Dynamical Systems* (DS)s.

Dynamical Systems as high level control policies

Before moving forward, it is crucial to define and contextualize the concept of Dynamical Systems (DS), which plays a central role in this thesis. A DS is a specialization of the abstract mathematical concept of ordinary differential equations to the case where the integration variable is interpreted as time. These equations allow for the description of a particle or

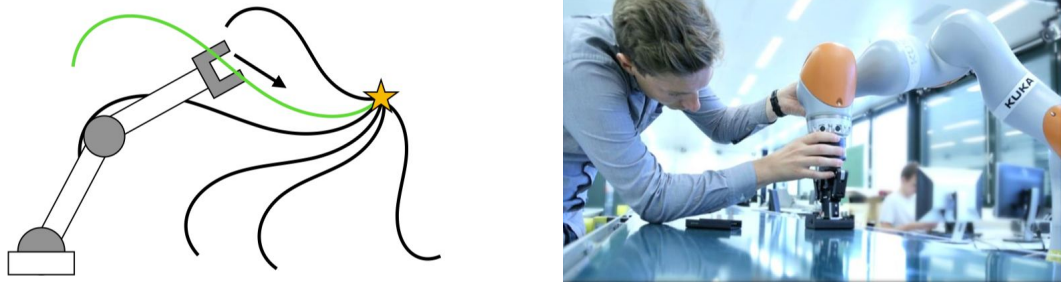



Figure 3: (left) A planar robotic arm tracking a specific streamline, highlighted in green, sampled from a 2D dynamical system. The other sampled streamlines are depicted in black, providing a visual representation of the potential 'flow' paths the robot would follow if it were positioned along these alternative trajectories. (right) Kinesthetic teaching, where the robot is physically guided through the task by the human.

ensemble of particles whose state varies over time. However, across various disciplines, this term has been broadly used beyond its original context of particles' motion, to describe abstract systems governed by differential equations involving time derivatives. At the dawn of the 21st century, **the emerging field of robotics**, drawing from both control theory and machine learning, **saw potential in using DS formalism to describe high-level behavioral policies of robotic systems**. For example, consider a robotic hand designed to move from any point within its reachable space to a specific target, such as grasping a glass of water. In this scenario, a DS can model this behavior as an ideal flow, within which the robotic hand is immersed, continuously directing it from any point in space towards the glass. At the glass location, this flow ceases. Such special locations is called equilibrium point or *attractor* of the DS. Figure 3, on the left, offers a schematic representation of a robotic arm following a streamline of the DS. Learning from Demonstration (LfD) has developed as an effective approach to embed human-like motions, Figure 3 on the right, into a DS, leveraging a limited number of observations as a foundation, Billard et al. (2008).

Learning a DS guaranteeing the existence of such unique attractor is a complex challenge, potentially impacting the encoding of the demonstrated behavior's complexity. Over the past decade, considerable research has been conducted to address this issue. Differential geometry, though not always explicitly mentioned, began to emerge in those approaches that learn DSs through diffeomorphic transformations, Perrin and Schlehuber-Caissier (2016). Without delving into mathematical intricacies, these approaches essentially seek to transform the original space in which the DS resides. This transformation aims to simplify and enhance the learning process while maintaining the existence of a unique attractor for the learned DS. The link to differential geometry in this context was first explicitly revealed by Rana et al. (2020). As highlighted earlier, the concept of representation is central to our discussion. Rana et al. explains how the various transformations of the original space can be seen as ways of accessing different representations of the same underlying manifold, with the original space itself being just one such representation. In Figure 1, we have gathered all strategies that

attempt to learn DSs in one manifold's representation space under the node —"Learning Dynamical Systems leveraging on Geometric Tools".

From Manifold Learning to Dynamical Systems

Real-world complex tasks are rarely accurately modeled by a single DS. The intricacy and diversity of movements in even the simplest real-world scenarios necessitate a model capable of integrating multiple DSs. In this context, **Multiple-Attractor Dynamical Systems have emerged as a promising approach to modeling complex tasks**. A Dynamical System composed of multiple stable attractors offers the flexibility to encode various methods for reaching and grasping objects, as well as the capability to execute complex tasks through a series of simpler subtasks. The concept of Multiple-Attractors DS, along with Manifold Learning, is the key focus of Chapter 2 in this thesis. **In realistic scenarios, proper segmentation and labeling of data might be lacking**, either because the data are generated by inexperienced users or because the dataset comprises complex physical task demonstrations. It is unreasonable to expect non-expert users to accurately identify the number of sub-dynamics and the locations of corresponding attractors. Requiring users to segment the task into sub-components can be overly demanding and hinder the effective demonstration and transfer of skills. **To address the challenge of identifying multiple-attractor DS without prior knowledge of the dynamics or attractors, we propose a fully unsupervised learning approach**. The objective of our algorithm is twofold: (a) to cluster the sub-dynamics within a dataset, and (b) to uncover the underlying structure of the data that facilitates the identification of attractors and eases the learning of a stable vector field. To accomplish our goal, **we demonstrate that by creating an appropriate graph structure, one can unveil, through Manifold Learning strategies, a new resourceful representation**. This representation effectively reveals the number of sub-dynamics and their respective attractor locations within a Multiple-Attractor DS. Moreover, this newly derived representation is inherently compatible with diffeomorphic learning DS strategies, thereby establishing the link between first two nodes, in Figure 1. This final connection was established by examining both Manifold Learning and diffeomorphic learning of DSs through the lens of differential geometry.

The application of differential geometry in reinterpreting established concepts might initially appear as an elaborate exercise in translating ideas into a different mathematical language. However, this process is far from being an academic redundancy. Reformulating problems within a more profound and sophisticated mathematical framework is a critical step towards advancement. An illustrative example, which is relevant to the next chapter of this thesis, is Contraction Theory (CT), Lohmiller and Slotine (1998). Originally, CT's formulation did not explicitly connect with the concepts of manifolds or their representations. However, a unified and formal reinterpretation of CT through the lens of differential geometry, Simpson-Porco and Bullo (2014), has enriched our understanding of the subject revealing aspects that were remained hidden. This new perspective on CT is intimately linked to the concepts of manifolds and their representations, discussed earlier. Presently, a significant area of research,

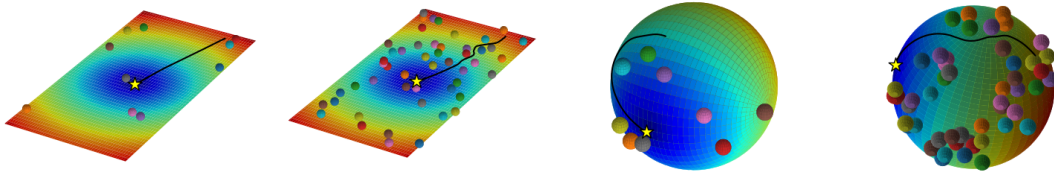


Figure 4: One sampled streamline from a geometrical DS on a flat Euclidean space and on the sphere. The avoidance of obstacles is achieved via local deformation of the space (not visible in the picture). The color gradient represent the potential function driving the DS towards the goal (the star).

which we refer to as “Theoretical Analysis of Stability on Manifolds”—node [3](#), is actively utilizing geometry to broaden various results in control theory, Forni and Sepulchre (2014). As highlighted for the first work, the second study we are going to introduce will derive results from the differential geometry interpretation of CT by Simpson-Porco and Bullo (2014).

Learning via Geometrical Dynamical Systems

While having multiple attractors clearly extends the application of Dynamical Systems (DSs) to complex, real-world scenarios, **another research trajectory focuses on enhancing the adaptability and expressivity of DSs characterized by a single attractor**. In this realm, “**Geometrical Dynamical Systems**”—node [4](#)—**proposed a significant advancement in shaping complex policies**, Figure 4. These approaches involve shaping DSs that operate on carefully defined manifolds, aiming to achieve two key goals: 1) increase the expressivity and adaptability of DS policies, and 2) enhance the modularity of these systems for addressing the increasing complexity of real-world scenarios in robotics. A notable contribution in this field is the Riemannian Motion Policy (RMP), introduced by Ratliff et al. (2018). This modular mathematical framework is designed for robotic motion generation. By strategically designing the so-called Riemannian metrics, it allows for a wide range of behaviors to be exhibited and combined. This foundational work has spurred active research in the area, leading to extensions of the original concept, Cheng et al. (2020); Bylard et al. (2021), and more recent explorations into more advanced geometrical frameworks involving pseudo-Riemannian metrics, Xie et al. (2021); Ratliff et al. (2021).

This line of research has not been explicitly focused on learning DSs from demonstrations. The work presented in [Chapter 3](#) aims to bridge this gap. **We combine insights from DS learning literature with the emerging area of geometry-based shaping of DS policies.** Inspired by Einstein’s concept of a four-dimensional manifold in space-time, we propose a novel approach to learning nonlinear DSs. **In our method, the nonlinearity of the DSs is not a result of external forces; instead, it originates from the intrinsic curvature of an underlying**

Introduction

manifold. This approach, which may initially appear as a complex mathematical construct, results in learning strategies that significantly enhance trajectory reproduction fidelity. Notably, this method also reduces computational costs both during training and at the time of querying. Our framework enhances the expressivity of geometrical policies. It illuminates the relationship between the nonlinearity of DSs and the curvature of the manifold.

The foundational work of Ratliff et al. (2018), and by extension our own, was enabled by a differential geometry-based reinterpretation of Lagrangian mechanics, an idea that began to take shape over a decade earlier. The work of Bullo and Lewis (2005), at the onset of the 21st century, proposed a unified differential geometric treatment of modeling, analysis, and design for mechanical control systems. This research domain, identified as “Classical Mechanical Systems as Riemannian Geometries”—node ⑤, has sparked several other intriguing research areas. Although not covered in this thesis, these fields could potentially intersect with our work in the future. One particularly interesting field deeply rooted in robotics is “Nonlinear Modes Analysis”—node ⑥, Albu-Schäffer and Della Santina (2020). This area extensively adopts a differential geometry perspective to enhance the understanding of robotic systems’ intrinsic dynamics. This endeavor to deepen our understanding is ultimately aimed at designing more efficient and effective controllers for specific tasks, Bjelonic et al. (2022). Additionally, it seeks to conceptualize systems that are inherently designed and optimized for particular functions, enhancing their suitability and performance for those designated tasks, Albu-Schäffer and Sachtler (2023).

Learning on Manifolds... not Manifold Learning

While the concept of a manifold has been intermittently referenced, the primary focus so far has been on its representation. In both the first and second studies of this thesis, the DS we aimed to identify or learn were ultimately addressed within a representation of the manifold, leaving the manifold itself somewhat in the background. To put it in more technical terms, our discussion has primarily concerned the “charts” that cover a particular manifold. Consider once again the Earth. We typically conceptualize it as a sphere in a 3D space. However, when we need to locate a specific city on the globe, we refer to an atlas and flip through its pages until we find the map showing the part of the Earth where that city is located, Figure 2 in the center. In mathematical terms, this specific map page is what we call a “chart” representing a portion of our manifold. A manifold can be depicted through multiple charts, each serving as a local, flat representation of the manifold. Until now, our approach has relied on the (latent) presence of the manifold to induce a specific structure in the chart, a local flat Euclidean space. But what if we want to consider problems directly on the manifold itself, like addressing issues on the entire sphere in our Earth example?

Numerous scientific disciplines deal with underlying structures that are non-Euclidean in nature. This includes areas such as computational social sciences, sensor networks in communications, functional networks in brain imaging, regulatory networks in genetics, and

meshed surfaces in computer graphics. In these applications, optimization and learning are critical processes. **Extending optimization and learning strategies to non-Euclidean spaces is a complex task that necessitates a novel reinterpretation of the most basic and fundamental mathematical operators.** When extending optimization problems to manifolds, it becomes necessary to view parameter spaces as non-Euclidean. When adapting learning problems to manifolds, the focus shifts to domains that are inherently non-Euclidean.

Building on the abstract but foundational reinterpretation of optimization through differential geometry, as explored by Absil et al. (2008), these tools have since gained widespread application across a variety of fields, most notably in machine learning and robotics. In machine learning, for example, optimization on manifolds has been applied to enhance Principal Component Analysis (PCA), a well-known unsupervised learning strategy. Structures such as the Stiefel or Grassmann manifolds have emerged as solutions to optimization problems characterized by sparsity and increased robustness against outliers. In robotics, optimization on Special Orthogonal groups (SO(d)) is ubiquitous, as these structures naturally describe the orientation of a system. This is particularly evident in simultaneous localization and mapping (SLAM), a process where a robot must map its environment and determine its location within it as it navigates Rosen et al. (2021). Furthermore, the fields of “Geometric Optimal Control”—node 7, Watterson et al. (2018); Bonalli et al. (2019), and “Lie Group Theory for Identification & Control”—node 8, Solà et al. (2021), actively employ tools from gradient-based optimization extended to non-Euclidean spaces. Today, “Optimization on Manifolds”—node 9—represents a broad and dynamic area of research, Boumal (2023), reflecting its growing importance and applicability across a spectrum of scientific and technological domains. Optimization on Manifold treatment of differential geometry from both extrinsic and group-based perspectives, has significantly enhanced our grasp of geometry-based control. This understanding was pivotal in designing the low-level controller for the robotics experiments. Moreover, it actively inspired and facilitated the development of various concepts introduced in this work.

Probabilistic Models on non-Euclidean Domains

The second key area where an extension to non-Euclidean spaces shows significant potential is learning. The third and final work of this thesis concentrates on learning on manifolds. This area, involving the learning of scalar or vector-valued functions on manifolds, has rapidly become a vibrant and productive research field in both machine learning and robotics. In the realm of deep learning, this extension was already underway with advancements in “Deep Geometrical Learning”—node 10, Bronstein et al. (2017). However, probabilistic learning strategies, which are particularly appealing for robotics and decision-making processes in general, were still in need of a robust differential geometry framework.

Probabilistic model learning represents an interesting avenue, particularly in the field of robotics where it is essential to handle hardware noise and model uncertainty in order to


Introduction

design safe and robust controllers. In this context, Gaussian Processes (GPs) are among the most adopted models for learning unknown functions within the Bayesian framework. Their data efficiency and capability for uncertainty quantification make them an appealing strategy not only for control design but also for decision-making applications in general. Another key element behind the success of GPs is their ability to encode different kinds of prior information about the function they aim to approximate. For example, by choosing different kernels, one can encode various degrees of differentiability or specific patterns, such as periodicity and symmetry.

In the field of robotics, various kernels have been developed by leveraging domain knowledge about the specific task at hand. Task-specific kernels were introduced by Antonova et al. (2017) who learned a distance metric through simulated bipedal locomotion patterns to optimize gait controllers. Similarly, Rai et al. (2018) utilized gait feature transformations to design kernels for optimizing locomotion controllers. Although these kernels can be utilized for robot control or policy search across a wide range of tasks and systems, their application is specific to the problems for which they were originally designed.

A more general approach, of interest to many fields, aims at incorporating into the kernel information about the geometry of the space. Many quantities of interest in robotics exhibit non-Euclidean geometries, necessitating the construction of specific kernels tailored to these geometries. For instance, three-dimensional rotations can be represented as elements of the Lie group $SO(3)$ or the sphere S^3 . Control gains, inertia, and manipulability ellipsoids are situated in the manifold of symmetric-positive-definite matrices \mathcal{S}_{++}^d . Meanwhile, the joint configuration of a d -degree-of-freedom robot with revolute joints can be considered as a point on a torus \mathbb{T}^d .

Constructing Kernels via Differential Operators... Manifold Learning again?

Initially, progress was made with less mathematically formal approaches, either by working locally in the so-called tangent space, Zeestraten et al. (2017), or by adapting the notion of distance to account for the curvature of the manifold, Jaquier et al. (2020). These “Extensions of Machine Learning Techniques on Manifolds”—node —drew the attention of mathematicians, leading to collaborative efforts in developing both theoretically and practically sound extensions of learning strategies to manifolds. A significant breakthrough came from insights that trace back to the mid-20th century. Building on Whittle’s pioneering work, Whittle (1963a), which established a connection between Gaussian Processes and certain stochastic partial differential equations (SPDE) driven by white noise, Borovitskiy et al. (2020) proposed an extension of the well-known Heat and Matérn kernels—central in GP regression and broadly in ridge regression algorithms—to manifolds. **A significant milestone in learning on known manifolds was achieved by embedding information about the geometry of the specific structures underlying the data directly into the kernel in a principled way**, Figure 5. This development has sparked a range of studies, which we categorize under “Differential Operators

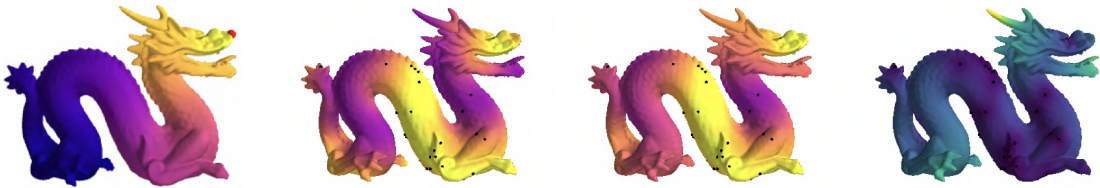


Figure 5: Learning on complex manifold. (left) Geometric kernel evaluated at red dot; the color gradient represent bell-shaped functions (similar to a multivariate Gaussian), centered in the red dot, taking place on the surface of the dragon. The kernel decays gradually following distances calculate on the surface of the manifolds. Note how the two sides of the dragon’s snout have very different values, despite close Euclidean proximity. (center) Ground truth and prediction. (right) Uncertainty. ¹

for Constructing Kernels on Manifolds”—node [12](#). Soon after, a more precise redefinition of Bayesian learning strategies for well-known manifolds relevant to robotics was introduced by Jaquier et al. (2022).

What if the structure of the manifold is unknown? Chapter 4 of this thesis addresses this question by revisiting manifold learning strategies. To bridge the gap between manifold learning and Bayesian regression on manifolds, a conceptual advancement was necessary. Following Belkin and Niyogi (2003) foundational work, a line of harmonic analysis research, influenced by Coifman and Lafon (2006); Nadler et al. (2006b), began exploring manifold learning as a means to approximate differential operators on manifolds. This approach, inspired by the work of Whittle (1963a); Lindgren et al. (2011), set a clear direction for our investigation. In this final part of the thesis, **we demonstrate that by constructing an appropriate Geometric Graph Laplacian, one can recover the differential operator needed to extend the Matérn kernel to manifolds, without prior knowledge of the manifold’s structure.** Moreover, we treat the two learning challenges—identifying the manifold structure and the function on the manifold—as interconnected problems. We introduce a Gaussian Process regression method that concurrently learns a specific function and deduces the underlying manifold structure from both labeled and unlabeled data in a fully differentiable way. Our technique is scalable to hundreds of thousands of data points and enhances both the predictive performance and calibration of standard Gaussian process regression in high-dimensional and complex non-Euclidean spaces. Although not directly related to learning Dynamical Systems, the research in this chapter lays the groundwork for future extensions of existing theories Robert-Nicoud et al. (2024) to learning vector-valued fields on unknown manifolds.

The End...?

Our exploration of Machine Learning and Robotics through the lens of differential geometry has reached its conclusion. The upcoming chapters will delve into each topic, adhering to a

¹©Borovitskiy et al. (2020).

Introduction

structured format designed to guide the reader through the diverse theoretical landscapes. Each chapter begins with an introduction providing a general overview of the current state of the art in a specific area and the challenges we aim to address. This is followed by a background section, which compiles all the necessary concepts and tools for the reader to fully grasp the topic at hand. The core content of each chapter is presented in one or more sections, thoroughly discussing the main subject matter. Finally, each chapter concludes with a discussion section. Here, we summarize the key points of our work, examine its limitations, and contemplate future developments. This section also serves as a conceptual bridge to the next chapter, ensuring a coherent and continuous narrative throughout the thesis.

Contributions and Thesis Outline

After a first background chapter, this thesis is organized in three main parts corresponding to the three different “edges” highlighted in Figure 1. Following, we present brief overviews of each chapter and highlight their corresponding contributions.

Chapter 1: Background

This chapter provides a background summary and reviews the preliminary materials needed to follow the developed approaches in the thesis.

Chapter 2: Linearization and Identification of Multiple-Attractor Dynamical Systems through Laplacian Eigenmaps

In complex real-world scenarios, single models often fail to capture the intricacies involved. A more effective method involves employing Multiple-Attractor Dynamical Systems (DSs), which integrate various DSs, each distinguished by unique non-linearities and equilibrium points (attractors). However, human demonstrations of such tasks are typically unstructured, lacking predefined information about the number and location of sub-dynamics or attractors. We introduce an unsupervised learning approach that, using a dataset of unlabeled position-velocity pairs, autonomously determines the number of sub-dynamics and the positions of attractors. This work has two primary contributions. Theoretically, we explore the structure of embedding spaces formed by eigenvectors of the Graph Laplacian when applied to specific graph structures, verifying the existence of multiple embedding spaces corresponding to each identified sub-dynamic, where demonstrated streamlines appear linear. Practically, we propose a new kernel, fusing the RBF and Cosine kernels, to reconstruct the desired graph structure. Additionally, we demonstrate the efficacy of combining differential perspective Manifold Learning with diffeomorphic transformations for learning stable Dynamical Systems.

Chapter 3: Learning Dynamical Systems Encoding Non-Linearity within Space Curvature

We concentrate on enhancing the learning of stable Dynamical Systems (DS) using geometric tools, with several key contributions. Firstly, we introduce a learning framework based on differential geometry, which encodes the non-linearity of the DS through the curvature of the space. This approach enables us to establish the stability of the DS on a manifold, independent of its complexity in a chart-based representation. This integration effectively combines Learning from Demonstration (LfD) with Geometric Policies, taking advantage of the latter's superior expressivity. We demonstrate how this framework allows for online manipulation of the space to seamlessly shift from standard behavior to task-specific actions, or to address local non-linearities in situations like obstacle avoidance, without the need for re-learning. Finally, our method extends the learning of DS to the realm of second-order systems. This not only significantly enhances the fidelity of velocity profiles and accommodates a wider array of trajectory behaviors, but also aims to create a connection between high-level DS policies and low-level intrinsic DS, both interpreted as Riemannian geometries.

Chapter 4: Implicit Manifold Gaussian Process Regression

Learning on manifolds is increasingly important due to its effectiveness in modeling a variety of phenomena. Gaussian Process Regression (GPR) is particularly notable in this context, offering the ability to infer complex functions from limited data while providing reliable uncertainty quantification, which is crucial in decision-making processes like those in robotics. Recent theoretical advancements have introduced novel kernel structures for situations where the manifold's structure is known. Our work, however, addresses scenarios where this structure is unknown. We introduce a method that synergizes Manifold Learning with GPR on Riemannian manifolds. This method learns a specific function and simultaneously infers the manifold's structure. This problem utilizes the Matérn precision matrix's unique structure on Riemannian manifolds, optimizing both the GP model's hyperparameters and the graph structure approximating the unknown manifold. We demonstrate the convergence of the approximated kernel to its continuous counterpart. Our approach, which incorporates K-nearest neighbors (KNN) for graph construction and Nystrom-formula out-of-sample eigenvectors extension, scales efficiently to hundreds of points thanks to Krylov subspace methods like Conjugate Gradient (CG) and Lanczos tridiagonalization. Our method significantly enhances inference and uncertainty quantification in high-dimensional settings, particularly where the manifold hypothesis is applicable.

Chapter 5: Conclusion and Future Developments

In this chapter we summarize the presented contributions and outline limitations and future research directions.

Introduction

Appendix

The appendix of this thesis is composed of five sections focused on proofs, expansions and extensions to the presented contributions.

Publications

The contributions in this thesis have been published, presented or are currently under review in peer-reviewed journals and conferences. The work presented in Chapter 2 has been published in the Journal of Machine Learning Research (JMLR), Fichera and Billard (2022). The work described in Chapter 2 is currently under review in the International Journal of Robotics Research (IJRR), Fichera and Billard (2024). Despite not presented in this thesis, the work in Chapter 2 was inspired and built on top of a “Blue Sky Idea” published in International Symposium of Robotic Research 2022 (ISRR), Fichera and Billard (2023). The content of Chapter 4 has been published and presented in the Thirty-seventh Annual Conference on Neural Information Processing Systems (NeurIPS 2023), Fichera et al. (2023).

Developed Libraries

In this section we list the most important libraries developed along the projects presented in this thesis. Here we are not listing paper related code but general purpose libraries that can and should be used beyond the scopes of this work.

learn-embedding: <https://github.com/nash169/learn-embedding.git>

Python library. Given an embedding automatically generates the isometric pulled-back embedding geometry necessary to define first or second order DSs on a chart-based representation of the manifold. It offers the possibility of modularly defining the embedding via different function approximators. The fully differentiable pipeline allows for embedding curvature adaptation to learn stable dynamical systems.

beautiful-bullet: <https://github.com/nash169/beautiful-bullet.git>

C++ library. It allows to quickly setup your robotic simulation and test your controllers. This library intends to leverage the core of Bullet Physics Engine and empowers it with different tools offering a performant, modular and user-friendly solution. All the mathematics is wrapped by eigen linear algebra library, current standard in C++. It uses urdfdom for efficient model parsing and assimp for creation of 3D meshed collision objects. Visualization is based on the light-weight and modular magnum graphics library. All model-based operations, such as forward/inverse kinematics/dynamics and calculation of various model matrices, is powered by pinocchio that offers state-of-the-art implementation of many Rigid Body Algorithms.

control-lib: <https://github.com/nash169/control-lib.git>

C++ library. Manifold based set of controllers. Each controller is defined over a specific manifold or group, such \mathbb{R}^n , $SO(n)$ or $SE(n)$. All the operation done via that controller au-

tomatically adapt to the geometry of the underlying space. It offers a “dynamic” version of Quadratic Programming (QP) control. The quadratic functional can be dynamically defined to operate on first-order or second-order state derivatives to target either kinematics or dynamics tasks, respectively. Both functional and constraints components can be modularly construct allowing to easily define task-specific QP control structures.

geometric-control: <https://github.com/nash169/geometric-control.git>

C++ library. It uses Visitor Pattern with Variadic Template meta-programming strategy to generate an automatic sequence of connections’ pullback, alias unconstrained inverse dynamics, across manifolds. It generates tree structures where each node is a different manifold. The edges connecting nodes corresponds to maps across manifolds and they are parent/child node specific. The leaf nodes consist of specific manifolds on which a connection (or covariant derivative) defines a “task”, alias second-order DS accomplishing a specific goal. At runtime all the connections are automatically pulled-back towards the root node generating an overall second-order DS encoding complex and various motions.

manifold-gp: <https://github.com/nash169/manifold-gp.git>

Python library. Library providing high-quality implementation of Riemannian Matérn kernel along with fully differentiable structure that allows to perform end-to-end—graph-bandwidth plus kernel hyperparameters—optimization of the graph Laplacian based GP model. The library relies on efficient implementation of K-nearest neighbors provided by faiss, to construct Random Walk and Symmetric Laplacian sparse operators as well as out-of-sample eigenvectors extension via Nystrom formula. Combined with modern matrix-free Krylov subspaces strategies, the library can scale up to hundreds of data points, operating in both supervised—only labeled data—and semi-supervised—both labeled and unlabeled data—learning.

kernel-lib: <https://github.com/nash169/kernel-lib.git>

C++ library. It offers a high-quality implementation of the most common adopted kernels in machine learning. It employs Curiously Recurring Template Pattern meta-programming strategy to achieve runtime high-performance static polymorphism. Powered by OpenMP multi-threading it can compute Gram matrices on several thousands of points in few milliseconds.

1 Background

In this thesis, differential geometry finds its application across three distinct domains: Manifold Learning, Geometrical Dynamical Systems, and Gaussian Processes on manifolds. After a preliminary section designed to establish foundational terminology and definitions, we start introducing the main topics faced in this work.

In Section 1.2, our objective is to demonstrate the multifaceted nature of Manifold Learning, unveiling new insights at each level of exploration that typically remain concealed. Initially, we focus on how Manifold Learning strives to reconstruct embeddings, aiming to retain certain properties, such as isometry. This viewpoint is instrumental for comprehending the methodologies presented in Chapter 2 of this thesis. We then extend this interpretation further, illustrating how Manifold Learning fundamentally seeks to approximate specific differential operators on manifolds. The eigenfunctions of these operators are versatile; sometimes they assist in approximating embeddings, while at other times they contribute to defining appropriate Gaussian Process (GP) covariance functions on manifolds. These applications will be explored in detail in Chapter 4.

In Section 1.3, we introduce essential terminology and concepts that are crucial for a thorough understanding of the ideas developed in Chapter 3. Specifically, we delve into the derivation of a geometric Dynamical System (DS) conceptualized as the stationary curve of a massive particle's world line in the context of action. This approach not only illuminates the intriguing interplay between Riemannian geometry and Lagrangian mechanics but also paves the way for the future development of more sophisticated and intricate Dynamical Systems.

Finally, in Section 1.4, we establish a link between the aspect of Manifold Learning, which focuses on approximating differential operators, and Gaussian Process Regression on manifolds. In the initial interpretation of Manifold Learning, the eigenfunctions of such differential operators were seen as components of embeddings. However, building upon the relationship between stochastic partial differential equations and Gaussian Processes, in Chapter 4, these eigenfunctions are re-envisioned as a basis, or harmonics. These harmonics are key in capturing the intrinsic geometry of the manifold, enabling us to accurately reconstruct the

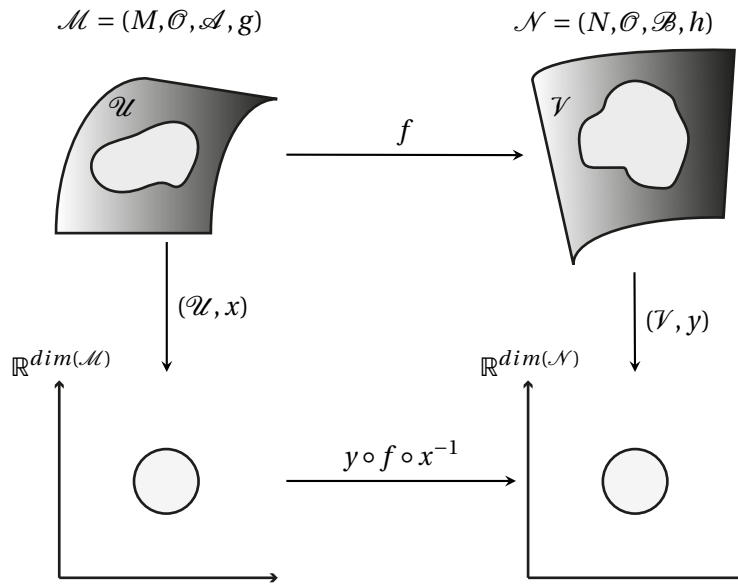


Figure 1.1: The differential manifold philosophy.

appropriate covariance function for the manifold's specific geometry.

1.1 Manifolds, Coordinate Charts and Smooth Embeddings

We first consider the geometric problem of manifold and metric representation, and define a smooth manifold in terms of coordinate charts.

Manifold A topological space $(\mathcal{M}, \mathcal{O})$ is called a d -dimensional manifold if:

$$\forall p \in \mathcal{M} : \exists \mathcal{U} \in \mathcal{O} : \exists x : \mathcal{U} \rightarrow x(\mathcal{U}) \subseteq \mathbb{R}^d \quad (1.1)$$

and:

1. x is continuous
2. x is invertible
3. x^{-1} is continuous

\mathcal{O} is a topology and \mathcal{U} is an open connected set that defines a portion of the manifold. x is called the *chart map*. It maps every point $p \in \mathcal{M}$ to the point $x(p) = \{x^1(p), \dots, x^d(p)\} = \mathbf{x}$ into the \mathbb{R}^d Euclidean space which the manifold is locally homeomorphic to. $\{x^1(p), \dots, x^d(p)\}$ take the name of *coordinate maps*. We assume to deal with *differentiable Riemannian C^∞ -manifold*.

1.1 Manifolds, Coordinate Charts and Smooth Embeddings

From geometrical point of view, the operation that leads to the calculation of the metric starting from a differential coordinate transformation is formalized defining an embedding.

Embedding Let \mathcal{M} and \mathcal{N} be smooth manifolds and $f : \mathcal{M} \rightarrow \mathcal{N}$ be an injective continuous map. Then f is called an immersion if its derivative (push-forward map $f_* = df$) is everywhere injective. An embedding, or a smooth embedding, is defined to be an injective immersion which is an embedding in the topological sense that f yields a homeomorphism onto its image.

The derivative of the embedding, also called push-forward map or Jacobian, is a tool that, informally, allows to send vectors that lie on \mathcal{M} to \mathcal{N} .

Pushforward map Let $f : \mathcal{M} \rightarrow \mathcal{N}$ be a continuous map between manifolds \mathcal{M} and \mathcal{N} . The push-forward map f_* is the map

$$\begin{aligned} f_* : T\mathcal{M} &\rightarrow T\mathcal{N} \\ X &\mapsto f_*(X) \end{aligned} \tag{1.2}$$

where

$$f_*(X)\phi := X(f \circ \phi) \quad \forall \phi \in C^\infty(\mathcal{N}), \forall X \in \Gamma(T\mathcal{M}). \tag{1.3}$$

$T\mathcal{M}$ and $T\mathcal{N}$ are the tangent bundle of \mathcal{M} and \mathcal{N} , respectively.

The C^∞ – module $\Gamma(T\mathcal{M})$ is defined as the set collecting all the *sections* from the manifold \mathcal{M} to the tangent bundle $T\mathcal{M}$ (informally vector fields on \mathcal{M}).

The "reverse" is called pullback map and it sends co-vectors from \mathcal{N} to \mathcal{M} .

Pullback map Let $f : \mathcal{M} \rightarrow \mathcal{N}$ be a continuous map between manifolds \mathcal{M} and \mathcal{N} . The pullback map f^* is the map

$$\begin{aligned} f^* : T^*\mathcal{N} &\rightarrow T^*\mathcal{M} \\ \omega &\mapsto f^*(\omega) \end{aligned} \tag{1.4}$$

where

$$f^*(\omega)(X) := \omega(f_*(X)) \quad \forall X \in \Gamma(T\mathcal{M}), \forall \omega \in T^*\mathcal{N}. \tag{1.5}$$

$T^*\mathcal{N}$ is the cotangent bundle of \mathcal{N} .

In Riemannian geometry the specific embedding that preserves the metric is called *isometric embedding*.

Isometric embedding Let (\mathcal{M}, g) and (\mathcal{N}, h) be Riemannian manifolds. An isometric embedding is a smooth embedding $f : \mathcal{M} \rightarrow \mathcal{N}$ which preserves the metric in the sense that g is equal to the pullback of h by f , i.e. $g = f^*h$. Explicitly, for any two tangent vectors $v, w \in T_p(\mathcal{M})$ we have

$$g(v, w) = h(f_*(v), f_*(w)) = (f^*h)(v, w) \tag{1.6}$$

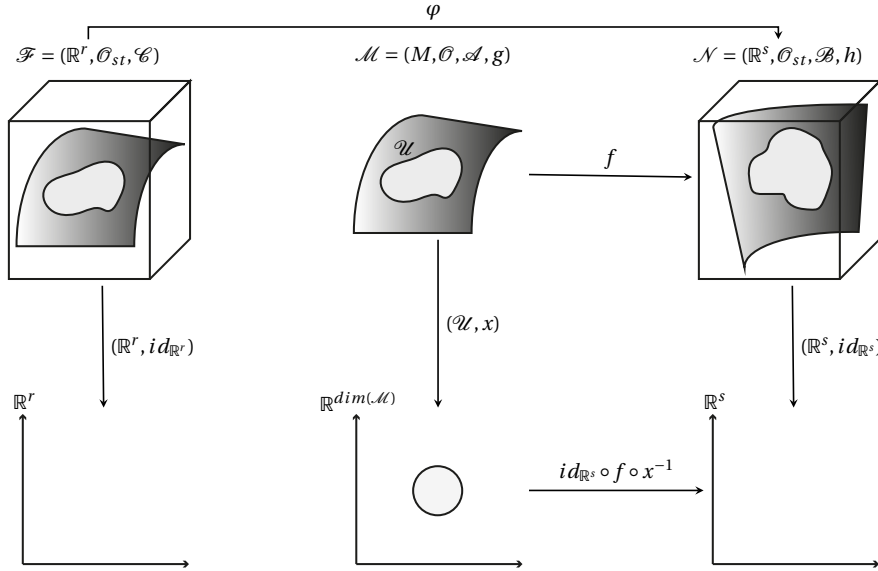


Figure 1.2: Manifold learning working flow.

The map across the two local coordinate charts is called the *local representative* of the map f .

Local representative of a map Let (\mathcal{U}, x) and (\mathcal{V}, y) be charts of manifolds \mathcal{M} and \mathcal{N} , respectively. Given the map $f: \mathcal{U} \rightarrow \mathcal{V}$, the local representative of f with respect to the two charts is the map $f_{xy}: x(\mathcal{U}) \rightarrow y(\mathcal{V})$ given by

$$f_{xy}(\mathbf{x}) = y \circ f \circ x^{-1}(\mathbf{x}), \quad (1.7)$$

where $\mathbf{x} = x(p)$, $p \in \mathcal{M}$.

Figure 1.1 shows, on the top-left, the manifold \mathcal{M} . On the top-right, it is represented the embedding manifold \mathcal{N} . Below these two manifolds there are the related maps chart x and y . f is the embedding while $y \circ f \circ x^{-1}$ is the local representative of f .

1.2 Manifold learning: a differential geometry perspective

Manifold learning algorithms start from the assumption that data lie near or along a smooth sub-manifold \mathcal{M} of dimension d embedded in an Euclidean space $\mathcal{M} \subset \mathbb{R}^r$ with $d \ll r$. From a geometric perspective, one would ideally like to recover directly the d -dimensional coordinate maps $x(p) = \{x^1(p), \dots, x^d(p)\} = \mathbf{x}$ with $p \in \mathcal{M}$, Figure 1.2. This is not possible as manifolds in general cannot be represented by a global coordinate chart (the sphere is an example). Thus, what manifold learning algorithms aim to achieve is to map data into $s \geq d$ dimensions. In order to avoid heavy notation all the sampled data-points in Euclidean space are expressed in bold vector notation with lower index referring to the label. Let $id_{\mathbb{R}^r}$

1.2 Manifold learning: a differential geometry perspective

be the identity chart map of a r -dimensional Euclidean space endowed with the standard Euclidean metric. The i -th sampled point on the (Euclidean) manifold p_i is expressed in local coordinates as $id_{\mathbb{R}^r}(p_i) \equiv \xi_i$. Given a data set $\mathcal{X} = \{\xi_1, \dots, \xi_m\} \subset \mathbb{R}^r$, representing the sampling in \mathbb{R}^r of points $p \in \mathcal{M}$, the goal of these algorithms is to map the data points into vectors $\{f(\xi_1), \dots, f(\xi_m)\} \subset \mathbb{R}^s$ where $s \ll r$ and $d \leq s$. f is the embedding from \mathcal{M} to $\mathcal{N} = \mathbb{R}^s$. The *strong Whitney embedding theorem* states that any d -dimensional smooth manifold can be embedded into \mathbb{R}^{2d} . Given a manifold of dimension d , small compared to the observed data dimension r , manifold learning algorithms try to recover the embedding $f : \mathcal{M} \rightarrow \mathbb{R}^s$, where $s \leq 2d$, preserving the geometry of the manifold. As we have a discretized representation of the manifold, given by the data set \mathcal{X} , manifold learning does not recover a continuous embedding but a discretized version \tilde{f} represented by a set of vectors $\tilde{f} = [\tilde{f}^1, \dots, \tilde{f}^m]$. \tilde{f} will provide us with the coordinate in \mathbb{R}^s for each point in the dataset¹.

1.2.1 Learning the embedding: Diffusion Maps

Diffusion Maps, proposed by Coifman and Lafon (2006), provides a unified probabilistic framework relating diffusion process and corresponding differential operators to Manifold Learning and more generically Spectral Methods. Assuming that the data is randomly sampled from an underlying general probability distribution $e^{-U(\xi)}$, as the number of samples goes to infinity, the eigenvectors of each diffusion map converge to the eigenfunctions of a corresponding differential operator defined on the support of the probability distribution.²

Suppose we have m data points ξ_i , independently and identically distributed, belonging to a set \mathcal{X} , generated by random variables distributed according to a density q over a sub-manifold $\mathcal{M} \in \mathbb{R}^n$ of dimension d . In addition to this structure a positive definite and symmetric kernel $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is given. The kernel offers a measure of affinity or similarity between points and defines a prior on the local geometry of \mathcal{X} . Let $k_\epsilon(\xi_i, \xi_j) = h\left(\frac{\|\xi_i - \xi_j\|^2}{\epsilon}\right)$ a rotation invariant kernel. The degree of the graph constructed over the data points is defined as $d_\epsilon(\xi_i) = \sum_{j=1}^m k_\epsilon(\xi_i, \xi_j)$.

In diffusion maps, the graph Laplacian normalization is not applied on the graph with isotropic weights but on a re-normalized graph. We define the anisotropic kernel as $k_\epsilon^{(\alpha)}(\xi_i, \xi_j) = \frac{k_\epsilon(\xi_i, \xi_j)}{q_\epsilon^\alpha(\xi_i)q_\epsilon^\alpha(\xi_j)}$, where $\alpha \in \mathbb{R}$ is a parameter that can be tuned to regulate the influence of the density in the infinitesimal transitions of the diffusion. The degree of the graph for the new kernel is $d_\epsilon^{(\alpha)}(\xi_i) = \sum_{j=1}^m k_\epsilon^{(\alpha)}(\xi_i, \xi_j)$. Applying the graph Laplacian normalization we get the transition matrix of the Markov chain

$$\mathbf{M}_{\epsilon, \alpha}(\xi_i, \xi_j) = \frac{k_\epsilon^{(\alpha)}(\xi_i, \xi_j)}{d_\epsilon^{(\alpha)}(\xi_i)}. \quad (1.8)$$

In the limit for $m \rightarrow \infty$ the discrete sums converge to integrals over the density q . For $\phi :$

¹For out-of-sample extension of the manifold learning problem see Bengio et al. (2004).

²For a deeper analysis about approximation of differential operator on manifolds see Nadler et al. (2006b,a). For connection between heat kernel and Laplace-Beltrami operator on manifolds see Belkin and Niyogi (2002, 2003).

Chapter 1. Background

$\mathcal{M} \rightarrow \mathbb{R}$ we have $\lim_{m \rightarrow \infty} \frac{1}{m} \sum_{j=1}^m k_\epsilon(\xi, \xi_j) \phi(\xi_j) = \int_{\mathcal{M}} k_\epsilon(\xi, \xi') \phi(\xi') q(\xi') d\xi'$, where ξ and ξ' are two continuous variables. For a finite value of ϵ , the Markov chain in discrete time and space converges to a Markov process in discrete time but continuous space. The forward and backward continuous transport operator can be defined

$$\begin{aligned} T_{\epsilon, \alpha}^f \phi(x) &= \int_{\mathcal{M}} M_{\epsilon, \alpha}(\xi | \xi') \phi(\xi') q(\xi') d\xi' \\ T_{\epsilon, \alpha}^b \phi(x) &= \int_{\mathcal{M}} M_{\epsilon, \alpha}(\xi' | \xi) \phi(\xi') q(\xi') d\xi', \end{aligned} \quad (1.9)$$

where $M_{\epsilon, \alpha}(\xi | \xi')$ is the continuous in space relative of the discrete Markov chain in Eq. 1.8. $T_{\epsilon, \alpha}^f$ and $T_{\epsilon, \alpha}^b$ are thus the continuous analogues of the left and right multiplication by the finite matrix $\mathbf{M}_{\epsilon, \alpha}$.

The operator $M_{\epsilon, \alpha}$ admits a spectral decomposition where λ_k , ϕ_k and ψ_k denote the eigenvalues, the left and right eigenfunctions, respectively. For a fixed s , decided by evaluating the spectrum decay of the operator, the mapping between the original and embedding space, known as *diffusion maps*, is

$$\Psi(\xi) = [\lambda_1 \psi_1(\xi), \lambda_2 \psi_2(\xi), \dots, \lambda_s \psi_s(\xi)]. \quad (1.10)$$

The first eigenfunction ψ_0 is ignored since as it well known for stochastic matrix it corresponds to the constant eigenvector for the connected graph.

Another important concept is the *diffusion distance*:

$$D(\xi, \xi') = \|p(\xi, \cdot) - p(\xi', \cdot)\|_{L^2(\mathcal{D}, dq/\pi)}^2 = \int_{\mathcal{D}} (p(\xi, \mathbf{u}) - p(\xi', \mathbf{u}))^2 \frac{dq(\mathbf{u})}{\pi(\mathbf{u})}, \quad (1.11)$$

where π is the stationary distribution. $D(\xi, \xi')$ is related to the transition probability between points over the graph; it is small if there is a large probability of transition from ξ to ξ' , and large, otherwise. As shown by Coifman Coifman et al. (2005) the diffusion distance is equal to the Euclidean distance in the embedding generated by the diffusion maps $\|\Psi(\xi) - \Psi(\xi')\| = D(\xi, \xi')$.

In the limit, for $\epsilon \rightarrow 0$, the random walk converges to a diffusion process in continuous time and space, whose probability density evolves according to

$$\frac{\partial p(x, t)}{\partial t} = \lim_{\epsilon \rightarrow 0} \frac{I - T_{\epsilon, \alpha}^f}{\epsilon} p(x, t). \quad (1.12)$$

The objects of study are the infinitesimal generators (or propagators)

$$\mathcal{H}_\alpha^f = \lim_{\epsilon \rightarrow 0} \frac{I - T_{\epsilon, \alpha}^f}{\epsilon} \quad \mathcal{H}_\alpha^b = \lim_{\epsilon \rightarrow 0} \frac{I - T_{\epsilon, \alpha}^b}{\epsilon} \quad (1.13)$$

For $q = e^{-U}$ and considering the anisotropic diffusion kernel in Eq. 1.8 the propagators can be

1.2 Manifold learning: a differential geometry perspective

Case	Operator	Stochastic Process
$\epsilon > 0, n < \infty$	Markov $n \times n$ matrix M	Random walk discrete in space and time
$\epsilon > 0, n \rightarrow \infty$	Transport Operators T_f, T_b	Random walk continuous in space discrete time
$\epsilon \rightarrow 0, n \rightarrow \infty$	Infinitesimal Generators H_f, H_b	Diffusion process continuous in space and time

Table 1.1: Operators philosophy.

expressed in term of the parameter α :

$$\begin{aligned}\mathcal{H}_\alpha^f p &= \Delta p - 2(1 - \alpha)\nabla p \cdot \nabla U \\ \mathcal{H}_\alpha^b p &= \Delta p - 2\alpha\nabla p \cdot \nabla U + (2\alpha - 1)p((\nabla U)^2 - \Delta U).\end{aligned}\tag{1.14}$$

There three interesting cases:

- for $\alpha = 0$, the classical normalized Laplacian, introduced by Belkin and Niyogi (2003), with the infinitesimal backward generator being

$$\mathcal{H}_0^b p = \Delta p - 2\nabla U \cdot \nabla p,\tag{1.15}$$

that shows how for uniform distribution the propagator coincides with the Laplace-Beltrami operator;

- for $\alpha = 1/2$ the construction yields the forward Fokker-Plank equation

$$\frac{\partial p}{\partial t} = \nabla \cdot (\nabla p + p\nabla U),\tag{1.16}$$

where $p(x, t)$ represents the density of points at position x and time t of a dynamical system satisfying the Langevin equation

$$\dot{x} = -\nabla U(x) + \sqrt{2}\dot{\omega},\tag{1.17}$$

with ω being a n -dimensional Brownian motion;

- for $\alpha = 1$, the backward operator returns the Laplace-Beltrami operator independently from the density distribution generating the samples

$$\mathcal{H}_1^b = \Delta.\tag{1.18}$$

The random walk generated through $\mathbf{M}_{\epsilon,1}$ converges to Brownian motion on \mathcal{M} . Excluding the eigenvalues connected to the stationary distributions, the initial region of the spectrum of the Markov matrix can be used to estimate the dimension of the manifold; the corresponding eigenvectors (eigenfunctions in the continuous case) are intrinsic coordinate on \mathcal{M} . In this case, diffusion maps provide a map from \mathcal{M} to the embedding space $\mathbb{E} = \text{span}\{\psi_1, \dots, \psi_d\}$ and it is possible to recover the Riemannian geometry of the data set, regardless of the distribution of the data points.

Chapter 1. Background

In conclusion, the left and right eigenvectors of the finite matrix M can be viewed as discrete approximations to those of the operators T^f and T^b , which in turn can be viewed as approximations to those of H^f and H^b .

1.3 Geometrical Dynamical Systems

Let $(\mathcal{M}, \Theta, \mathcal{A}, g)$ be a topological smooth manifold endowed with the Hausdorff topology Θ , the C^∞ atlas \mathcal{A} and the Riemannian metric g . Consider a generic *action* of matter coupled with the chosen manifold metric g

$$S_{total}[g, \Phi] = S_{metric}[g] + S_{matter}[\Phi, g], \quad (1.19)$$

where $\Phi = \sigma, A, \phi, \dots$ massive, field (covector, scalar, ...) or other matter types. For massive particle, we consider just the interaction term on the right-hand side with given metric (no variation with respect to the metric). The action of a massive particle world line is

$$S[\sigma; g] = \int d\lambda \left(m \sqrt{g(v_\sigma, v_\sigma)} \right), \quad (1.20)$$

where $\sigma : I \rightarrow \mathcal{M}$ is curve on the manifold \mathcal{M} and v_σ is the vector field generated by the tangent velocities of curve σ . Considering the presence of (given) covector fields acting on the space we have

$$S[\sigma; A, g] = \int d\lambda \left(m \sqrt{g(v_\sigma, v_\sigma)} + D(v_\sigma) \right). \quad (1.21)$$

The dynamics equations can be derived from the Euler-Lagrange equation of such an action or equivalently considering the infinitesimal variation of the action itself leading to

$$m \nabla_{v_\sigma} v_\sigma = \mathcal{F}(\sigma, v_\sigma, \lambda)^\# = -d\phi^\# - D(\cdot, v_\sigma)^\#, \quad (1.22)$$

where $\lambda \in I$, \mathcal{F} is the forces covectors and the sharp operator $(\cdot)^\#$ transforms covectors to vectors. The chart components formulation is

$$\begin{aligned} g_{ak} (m \nabla_{v_\sigma} v_\sigma)^k &= \partial_a(\phi \circ x^{-1}) - D_{ma} \dot{\sigma}^m \\ (m \nabla_{v_\sigma} v_\sigma)^k &= g^{ak} \partial_a \phi - D_m^k \dot{\sigma}^m. \end{aligned} \quad (1.23)$$

Here some attention is required in the pulling up of the indices, In addition the "dissipative" is not properly derived. Nevertheless the equation should be correct. The covariant derivative $\nabla_{v_\sigma} v_\sigma$ is the so-called geometric acceleration.

The metric can be derived as the pullback of the Euclidean metric with respect to the isometric embedding $f : \mathcal{M} \rightarrow \mathbb{R}^s$, where \mathbb{R}^s is and Euclidean space of dimension s . This represents a particular case where the co-domain of the embedding is an Euclidean space of given dimension. Fig. 1.3 shows on the top left the embedding Euclidean space \mathbb{R}^s , endowed with the standard topology \mathcal{O}_{st} and the Euclidean metric g^E , whose components with respect

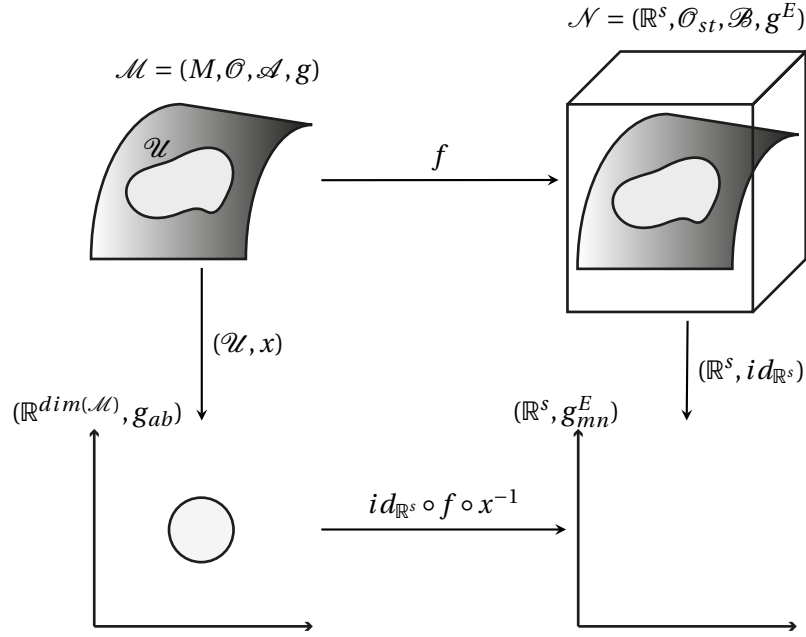


Figure 1.3: Contraction metric reconstruction via pullback of the Euclidean embedding metric.

to the identity chart map $id_{\mathbb{R}^s}$ are represented by the identity matrix.

The local representative map between the manifolds is defined as

$$(id_{\mathbb{R}^s} \circ f \circ x^{-1}) = (f \circ x^{-1}) : \mathbb{R}^{\dim(\mathcal{M})} \rightarrow \mathbb{R}^s \quad (1.24)$$

and it gives us across the two respective charts. With a little abuse of notation let's define such transformation as $f(x(p)) = (f \circ x^{-1})x(p)$, where $p \in \mathcal{M}$. The differential relation can be expressed as

$$\delta f^i = \partial_j (f \circ x^{-1})^i x(p) \delta x^j = \left(\frac{\partial f^i}{\partial x^j} \right)_p \delta x^j \quad (1.25)$$

In "vector" this relation is equivalent to $\delta z = J \delta x$, where δx and $\delta z = \delta f(x)$ are the infinitesimal variation in base and target space, respectively, and J is usually called the Jacobian. Then we can derive the metric of \mathcal{M} pulling back the Euclidean metric of \mathcal{N}

$$\begin{aligned} g_{ab}(p) &= (f^* g^E)_{ab}(p) \\ &= (\partial_a (f \circ x^{-1})^m x(p)) (\partial_b (f \circ x^{-1})^n x(p)) g_{mn}^E(\tilde{p}) \\ &= \left(\frac{\partial f^m}{\partial x^a} \right)_p \left(\frac{\partial f^n}{\partial x^b} \right)_p g_{mn}^E(\tilde{p}), \end{aligned} \quad (1.26)$$

where $\tilde{p} \in \mathcal{N}$.

1.4 Gaussian Process Regression on Manifolds

Up to this point, we have made a clear distinction between chart coordinates, denoted as \mathbf{x} , and embedded coordinates, represented by $\boldsymbol{\xi}$. Going forward, however, we will simplify our notation by using \mathbf{x} to refer to both, as the distinction between the two will no longer be significant in our context. Additionally, to align more closely with the established conventions in the field of Gaussian Process Regression, we will use the symbol f to represent the generic function that we aim to learn. It is important to note that when f is accompanied by a subscript, such as in f_n , it specifically refers to an eigenfunction of a particular differential operator.

The last work presented in this thesis we focus on extending to manifold setting Gaussian Process Regression. A Gaussian process is a distribution over functions of the form $f : \mathcal{X} \rightarrow \mathbb{R}$. It is defined implicitly by the requirement that $p(f(\mathbf{x}_1), \dots, f(\mathbf{x}_N))$ be jointly Gaussian, for any set of points $\mathbf{x}_i \in \mathcal{X}$. The parameters of this Gaussian can be computed using a mean function $\mu(\cdot)$ and covariance function (or kernel) $k(\cdot, \cdot)$. We write $f \sim \text{GP}(\mu(\cdot), k(\cdot, \cdot))$. In order to extend the GP formalism to non-Euclidean domains it is fundamental to design new kernels that can encode the geometry of the underlying space.

The problem of kernel regularization beyond non-Euclidean spaces started by focusing on graphs and discrete spaces with the work of Smola and Kondor (2003). Although its non-continuous nature, the work clearly hints at the idea that as long as a kernel can be defined in the input space all the kernel-based algorithms with their theoretical guarantees can be exported to non-flat geometries, namely manifolds. In addition, since every manifold can be isometrically embedded in Euclidean space, as shown by Nash (1956), it is always possible to define some cordal distance between points combined with a radial basis function kernel. Although this type of kernel suffers from the curse of dimensionality and it discards information contained in the geometry induced by the structure of the manifold.

Another solution was found in replacing the Euclidean norm with the geodesic distance as in Jayasumana et al. (2013, 2015). Unfortunately this approach does not always lead to well-defined kernel. Gneiting (1998) showed that, in general, one cannot expect this approach yielding symmetric positive semi-definite kernel or, in the context of GP, valid covariance functions. In addition, Feragen et al. (2015a) showed that for positive semi-definite geodesic squared exponential kernels with positive length scale hyper-parameter the underlying manifold is isometric to a Euclidean space. This implies an ill defined kernel for any compact Riemannian manifold.

Other interesting approaches leverage on the fact that manifolds are locally homeomorphic to the Euclidean space. Most of the technique used in Gaussian-based probabilistic modeling can be applied in the Euclidean tangent space of the manifold at a certain point by taking advantage of the so-called *exponential* and *logarithmic* maps. As shown in Zeestraten et al. (2017) and Calinon (2020) squared exponential kernel can be modified via such maps and concepts like Gaussian product, conditioning and mixture regression redefined consid-

ering the geometry of the space. Although this approximation demands for two important requirements: the manifold has to be homogenous for point-independent definition of the exponential and logarithmic maps and most importantly the geometry of the manifold has to be known to define these maps.

In order to take advantage of the geometry of the manifold, Belkin et al. (2006) suggested to construct the kernel via the Laplace-Beltrami operator used as a semi-norm of the form

$$\langle f, -\Delta_{\mathcal{M}} \rangle = \langle \nabla f, \nabla f \rangle. \quad (1.27)$$

Similar works by Sindhwani et al. (2007) and Zhu et al. (2003) take advantage of the Laplace-Beltrami operator to define a covariance function for GP. Limitation and improvements of such approach are discussed in Nadler et al. (2009) and Zhou and Belkin (2011). These approaches suggest two important ideas: first, differential operators can be used to construct kernels as formalized by Steinke and Schölkopf (2008); second, data-driven approach to construct a discrete version of the Laplace-Beltrami operator can be successfully applied to the problem as in Belkin and Niyogi (2004) and Belkin et al. (2006).

The second point will be the main topic of Chapter 4. The first point, coupled with the mathematical theory that connects GPs with stochastic partial differential equations (SPDE), provided a robust foundation for extending kernels to non-Euclidean spaces in a mathematically sound and principled manner. Given a differential linear operator \mathcal{L} , the SPDE on manifold \mathcal{M}

$$\mathcal{L}(f) = \mathcal{W} \quad (1.28)$$

with appropriate boundary conditions on $\partial\mathcal{M}$ and where \mathcal{W} represents a Gaussian white noise source, has solution

$$f \sim \text{GP}(0, \mathcal{L}^{-1}(\mathcal{L}^t)^{-1}), \quad (1.29)$$

where \mathcal{L}^t is the adjoint operator of \mathcal{L} and \mathcal{L}^{-1} is the inverse of the differential operator. The pioneering work of Whittle (1963a) has shown that the Matern GP on \mathbb{R}^d satisfies the SPDE

$$\left(\frac{2\nu}{\epsilon^2} - \Delta_{\mathbb{R}^d} \right)^{\frac{\nu}{2} + \frac{d}{4}} f = \mathcal{W}, \quad (1.30)$$

whose the connected covariance function is the well-known Matern kernel

$$k_{\nu}(\mathbf{x}, \mathbf{x}') = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} \frac{\|\mathbf{x} - \mathbf{x}'\|}{\epsilon} \right)^{\nu} K_{\nu} \left(\sqrt{2\nu} \frac{\|\mathbf{x} - \mathbf{x}'\|}{\epsilon} \right). \quad (1.31)$$

Following the same argument it is possible to show that

$$e^{-\frac{\epsilon^2}{4} \Delta_{\mathbb{R}^d}} f = \mathcal{W} \quad (1.32)$$

Chapter 1. Background

satisfies a GP characterized by the limiting squared exponential kernel

$$k_{\infty}(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\epsilon^2}\right). \quad (1.33)$$

This viewpoint on GPs has been reintroduced in statistical literature by Lindgren et al. (2011). The main advantage of this approach is that it generalizes to Riemannian manifold in a straightforward way by substituting $\Delta_{\mathbb{R}^d}$ with the Laplace-Beltrami operator $\Delta_{\mathcal{M}}$ on manifold.

The recent work of Borovitskiy et al. (2020) makes a decisive step forward in the development of the topic. Starting from the *Sturm–Liouville decomposition* of the Laplace-Beltrami operator

$$-\Delta_g f = \sum_{n=0}^{\infty} \lambda_n \langle f, f_n \rangle f_n \quad (1.34)$$

a kernel construction based on infinite expansion of eigenfunctions of the Laplace-Beltrami operator has been proposed. The Matern kernel takes the form of

$$k_v(\mathbf{x}, \mathbf{x}') = \frac{\sigma^2}{C_v} \sum_{n=0}^{\infty} \left(\frac{2v}{\epsilon^2} + \lambda_n\right)^{-v-\frac{d}{2}} f_n(\mathbf{x}) f_n(\mathbf{x}'), \quad (1.35)$$

while the exponential squared kernel becomes

$$k_{\infty}(\mathbf{x}, \mathbf{x}') = \frac{\sigma^2}{C_{\infty}} \sum_{n=0}^{\infty} e^{-\frac{\epsilon^2}{2} \lambda_n} f_n(\mathbf{x}) f_n(\mathbf{x}'). \quad (1.36)$$

This approach allows working in non-conjugate settings, such as classification, or using recently-proposed techniques for scalable GPs via sparse inducing point methods, see Hensman et al. (2018). In order to calculate the necessary eigenfunctions required for the construction of the kernel the approach relies on PDE-theoretic discretization techniques, such as Galerkin finite element methods.

2 Linearization and Identification of Multiple-Attractor Dynamical Systems through Laplacian Eigenmaps

2.1 Foreword

The work presented in this chapter has been published in *Fichera, B. and Billard, A. (2022). Linearization and Identification of Multiple-Attractor Dynamical Systems through Laplacian Eigenmaps. Journal of Machine Learning Research.*

2.2 Introduction

Relying on the mathematical framework of differential equations, *Dynamical Systems* (DS) describe how a system evolves temporally and spatially. Their application span various fields, such as physics, biology, engineering, and economics. In recent years, DS have been successfully applied to model and control robots (e.g. Heinzmann and Zelinsky (2003); Corteville et al. (2007)).

Finding an analytical description of the dynamics is, however, often difficult. This led researchers to turn to data-driven identification of the dynamics using machine learning algorithms. Data is usually provided by an expert, within the generic framework of learning from demonstration (Billard et al. (2016)). Data can also be gathered from trial and error in a reinforcement learning framework (Wabersich and Zeilinger (2021)).

From a machine learning perspective, approximating a DS can be framed as a regression problem. One estimates a non-linear function $\mathbf{f}: \mathbb{R}^d \rightarrow \mathbb{R}^d$, mapping the d-dimensional input state $\mathbf{x}(t) \in \mathbb{R}^d$ to its time-derivative $\dot{\mathbf{x}}(t) \in \mathbb{R}^d$, such that:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)). \quad (2.1)$$

Training data consists of a set of trajectories, as examples of path integrals of the DS. These trajectories cover a limited portion of the state space. To ensure that the learned DS stably generalizes over regions not covered by the data represents one of the most challenging and active research area. One option to tackle this problem is to embed constraints explicitly in

Chapter 2. Linearization and Identification of Multiple-Attractor Dynamical Systems through Laplacian Eigenmaps

the algorithm so that the learned dynamics offers similar guarantees as those generated from control theory. One desirable property is stability at an equilibrium point, or *attractor*. If \mathbf{x}^* is the attractor, we wish to guarantee that

$$\lim_{t \rightarrow \infty} \mathbf{x}(t) = \mathbf{x}^*, \quad \mathbf{f}(\mathbf{x}^*) = 0. \quad (2.2)$$

2.2.1 Learning Stable Dynamical Systems

In the Stable Estimator of Dynamical Systems (SEDS) proposed by Mohammad Khansari-Zadeh and Billard (2014), the density estimation with Gaussian Mixture Models is reformulated to enforce constraints on the parameters of the Gaussian Mixtures, by imposing conditions derived via Lyapunov's second method for stability. The DS is then estimated through Gaussian Mixture Regression (GMR). This approach was, however, limiting. Constraints from a quadratic Lyapunov function were conservative and led to a poor approximation of the flow when the dynamics was highly nonlinear.

Accuracy and stability turned out to be conflicting objectives whilst constraints were derived from a quadratic Lyapunov function, and this prevented modeling dynamics that are non-monotonic, namely temporarily moving away from the attractor. To address this issue, other works used more complex expression for the Lyapunov function (e.g. Mirrazavi Salehian (2018); Figueroa and Billard (2018)). An alternative to using Lyapunov stability is Contraction Theory (CT) introduced by Lohmiller and Slotine (1998). Stability under CT is less restrictive, as it follows a differential perspective and enforces solely that the flow contracts locally. Further, it does not require prior knowledge of the location of the attractor. Ravichandar et al. (2017) used CT to reformulate the SEDS constraints. Similarly Sindhvani et al. (2018) constraints a Support Vector regression problem with CT constraints. Although CT represents a promising approach, current works adopting it are limited to local stability guarantees making the approximated vector field unsuitable for regions with sparse or no data.

A third approach to tackle the problem of increasing accuracy while preserving stability is based on the idea of using latent representation to ease the stabilization of the DS. That is achieved via diffeomorphic mapping. One of the first attempts was offered in Neumann and Steil (2015), which uses a diffeomorphic map to send a complex Lyapunov function, learned from the sampled trajectories, to a space where it appears quadratic. In this space, SEDS is applied, and the original vector field is recovered via the inverse diffeomorphic map. The two-step learning approach requires fitting both a Lyapunov function (or a diffeomorphism) and a DS from training data, increasing the number of tunable parameters and the overall learning time for non-convex optimization problems. Perrin and Schlehuber-Caissier (2016) follow a similar approach but apply the diffeomorphism directly to the DS using one single sampled trajectory. The diffeomorphism learning process is purely geometrical, namely only original and target points' position are considered. Reconstruction of the proper velocity profile is achieved via re-scaling the learned DS. The recent work proposed by Rana et al.

(2020) follows a similar approach but introduces a formulation of the optimization framework that includes dynamics information (e.g., velocity) within the process, providing for a one-step learning algorithm.

All the methods reviewed above focus on single-attractor DS, and, except for works using stability constraints based on CT-derived conditions, they require prior knowledge of the location of the attractor. Assuming single dynamics and single attractor DS considerably constrains the applicability of these methods to learn uni-modal dynamics. Embedding multiple dynamics in a single control law based on DS increases the complexity of the dynamics that we can model. Next, we review recent efforts that have been dedicated to learning DS with multiple attractors.

2.2.2 Learning Dynamical Systems with Multiple Attractors

Learning DS with multiple-attractor can be achieved by explicitly partitioning the space to separate each dynamics and their respective attractors, as shown by Shukla and Billard (2012). This work requires knowing how many sub-dynamics exist in the system and the attractors' locations. Such knowledge may not always be easy to obtain.

Hence, if, for learning single-attractor DS, the main requirement is to know the location of the attractor, when learning multiple-attractor DS, correct labelling and classification of data points to distinguish the attractors and their associated dynamics is necessary. In realistic scenarios, proper segmentation and labeling might not be available either because data are generated by naive users or because the dataset is constructed by sampling demonstrations of complicated physical tasks (e.g., cleaning up a messy office). One cannot expect lay users to properly identify the number of sub-dynamics with the relative attractors location. Asking users to divide the task into sub-segment will be very cumbersome and prevent proper skill display and transfer; finally, even for expert users, it might often be difficult to classify the dynamics, especially when these require data that are not easy to interpret, such as force and haptic information.

To overcome this limitation, one option is to offer automatic segmentation and identification of the dynamics. Such an approach was followed by Medina and Billard (2017), using Hidden Markov Models to extract automatically the attractors' positions of multiple-attractor DS. The algorithm assumes that such multiple-attractor DS can be thought of as a long sequence of multiple sub-dynamics to be performed one after the other. Such approach implicitly assumes time labeling of the points and can handle only multiple-attractor DS considered as a long sequence of dynamics. Manschitz et al. (2018) presented another interesting application of multiple-attractor DS. The DS-based control law is modeled as a weighted combination of linear systems, each of which is stable with respect to an attractor. The location of the attractors along with the parameters of each linear sub-dynamics is automatically derived via optimization problem. However the minimum number of attractors necessary to solve a specific task is a user-defined hyper-parameter that has to be known a priori.

Chapter 2. Linearization and Identification of Multiple-Attractor Dynamical Systems through Laplacian Eigenmaps

To avoid providing prior information on both the number of dynamics and attractors, we see a need for a fully *unsupervised learning* approach to the identification of multiple-attractor DS. In this light, we offer an algorithm whose goal is to: (a) cluster the sub-dynamics within a certain data set, (b) find underlying interesting structure of the data that would allow to locate the attractors and ease the task of learning stable vector field. To do so, we seek to exploit structure discovery techniques to identify the number of sub-dynamics automatically.

2.2.3 Manifold Learning for Latent Embedding Spaces of Dynamical Systems

Manifold learning techniques, such as *Laplacian Eigenmaps* (Belkin and Niyogi (2003)) and *Isometric Mapping* (Tenenbaum et al. (2000)), are particularly relevant to our problem. These techniques can generate Euclidean spaces recovering the intrinsic geometry of a certain manifold (e.g., Tenenbaum et al. (2000)). As in Kernel PCA (Principal Component Analysis) (Schölkopf and Smola (2002)), these methods are based on eigenvalue decomposition of a matrix. The matrix embeds information about the feature of the data, as encapsulated by the kernel. Depending on the kernel used, different features can be extracted by the manifold.

We find applications of manifold learning techniques for DS in biology for analyzing emergent dynamics behaviors in data measured from bioelectric signals (Erem et al. (2016)), or for person identification from ECG (Sulam et al. (2017)); in control theory for data-driven time series analysis (Shnitzer et al. (2020)); in finance for describing the characteristics of financial system (Huang et al. (2018)); in chemistry for stochastic model of cellular chemotaxis (Dsilva et al. (2018)).

In the works reviewed above, the primary goal is to use manifold learning to reduce the dimensionality of the data and simplify the estimation of the DS. The scope of our work is different. We do not aim at reducing the dimensionality but rather at finding an embedding that can simplify the control of DS by linearizing a non-linear dynamics that would otherwise be difficult to stabilize.

The success of manifold learning depends heavily on the choice of kernel. We define a velocity-augmented kernel to extract the temporal evolution of data points. We generate the desired graph structure with this kernel and compute the associated Laplacian. We study the embedding spaces generated by the eigendecomposition of such a graph-based Laplacian matrix. We show that an analysis of the eigenvalues enables us to determine the number of underlying dynamics and to identify a set of eigenvectors that forms an embedding space in which the DS is linearized.

We validate our technique for sub-dynamics clustering and stable equilibria localization of multiple-attractor DS using theoretical and noisy instances of DS. We compare our algorithm to Kernel K-Means, Spectral Clustering, and Gaussian Mixtures. We showcase that even when these algorithms are provided with the correct number of sub-dynamics, they fail to cluster them correctly.

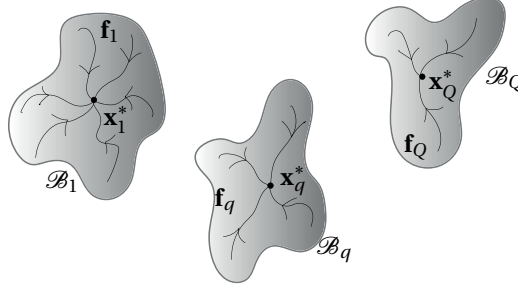


Figure 2.1: Schematic representation of a multiple-attractor DS. Each sub-dynamics \mathbf{f}_q takes within a sub-space \mathcal{B}_q and converges towards the attractor \mathbf{x}_q^* .

2.3 Background & Problem Formulation

Let $\mathbf{x} \in \mathbb{R}^d$ be the state of a DS. Its temporal evolution is governed by the non-linear function $\mathbf{f}(\mathbf{x}(t))$:

$$\mathbf{f}: \mathbb{R}^d \rightarrow \mathbb{R}^d, \quad \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)). \quad (2.3)$$

Assume that \mathbf{f} is composed of Q sub-dynamics, each asymptotically stable at one equilibrium point, the attractor.

Let the q -th sub-dynamics be

$$\dot{\mathbf{x}}(t) = \mathbf{f}_q(\mathbf{x}(t)) \quad \forall q \in [1, Q] \subset \mathbb{N}. \quad (2.4)$$

Each *sub-dynamics*, $\mathbf{f}_q(\mathbf{x})$, is Lyapunov stable and there exists $\delta > 0$ such that if $\|\mathbf{x}(0) - \mathbf{x}_q^*\| < \delta$, then

$$\lim_{t \rightarrow \infty} \|\mathbf{x}(t) - \mathbf{x}_q^*\| = 0, \quad (2.5)$$

where \mathbf{x}_q^* is the attractor of the sub-dynamics $\mathbf{f}_q(\mathbf{x})$, Fig. 2.1.

We know neither the number of sub-dynamics Q , the shape of the dynamics \mathbf{f}_q , nor the number and location of the attractors. All we have at our disposal is a finite set of observations, samples of trajectories from the DS. These are constituted by *unlabeled* position-velocity pairs $\mathcal{V} = \{v_i = (\mathbf{x}_i, \dot{\mathbf{x}}_i), i = 1, \dots, M\}$ sampled from $\mathbf{f}(\mathbf{x})$ at a constant frequency, f_{sampling} . Among these trajectories, a subset belongs to some of the sub-dynamics, but we know neither to which dynamics they belong nor how many trajectories belong to the same dynamics. All sampled trajectories end at the attractor of the associated sub-dynamics.

This paper presents a method by which we can a) determine the number Q of sub-dynamics present in the dataset, b) identify to which sub-dynamics each data point belongs and the attractor of the corresponding sub-dynamics and c) project each dynamics into a separate subspace in which the dynamics is linear. The method is based on a representation of the data through a graph and exploits properties from the eigendecomposition of the graph-based Laplacian matrix, as we present next.

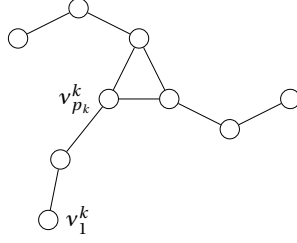


Figure 2.2: Each sub-dynamics is embedded in a graph where each trajectory forms a path connected by a set of termination nodes that form a cyclic path around the attractor. v_1^k $v_{p_k}^k$ are the first and last of the k -th path graphs, respectively. p_k is the number of nodes with the k -th path graph.

2.4 Graph Embedding and Linearization of Dynamical Systems

Consider a symmetric weighted graph $G(\mathcal{V}, \mathcal{E})$. The nodes $v_i \in \mathcal{V}$ represent the data points $v_i = \{\mathbf{x}_i, \dot{\mathbf{x}}_i\}$, $i = \{1, \dots, M\}$ sampled from our DS, where $M = \sum_{q=1}^Q p_q$ with p_q being the number of data points sampled from each of the sub-dynamics and Q being the number of sub-dynamics. \mathcal{E} is the set of edges $e_{ij} \in \mathcal{E}$ connecting nodes v_i and v_j . For a generic DS containing Q sub-dynamics, the graph G is the result of the union of Q sub-graphs \mathcal{F} such that $G = \mathcal{F}_1 \cup \mathcal{F}_2 \cup \dots \cup \mathcal{F}_Q$ and $\mathcal{F}_1 \cap \mathcal{F}_2 \cap \dots \cap \mathcal{F}_Q = \emptyset$. Each sub-graph \mathcal{F} is given by the composition of K path graphs, equal to the number of sampled trajectories. We order the nodes (vertices) of \mathcal{F} monotonically as $\{v_1^k, v_2^k, \dots, v_{p_k}^k\}$, $k = 1, \dots, K$, such that each $v_{p_k}^k$ is the last node of the k -th path graph and $p_k > 1$. The edges are $e_{ij}^k = \{v_i^k, v_{i+1}^k\}$. The K nodes $\{v_{p_1}^1, \dots, v_{p_K}^K\}$ form a cyclic graph (or simple circuit). Each node belonging to the cyclic path has degree 3. All other nodes along each path graph have degree 1 or 2. The nodes are numbered as

$$\{v_1^1, \dots, v_{p_1}^1, v_1^2, \dots, v_{p_2}^2, \dots, v_1^K, \dots, v_{p_K}^K\} \sim \{1, 2, \dots, M\}$$

With reference to Fig. 2.2, $\{v_1^k, \dots, v_{p_k}^k\}$ represents the nodes of the k -th path graph where p_k is the number of samples within the k -th path graph.

The Graph Laplacian matrix, $L(G) = D(G) - A(G)$, where $A(G)$ is the adjacency matrix

$$A_{ij} = \begin{cases} 1, & \text{if } e_{ij} \in \mathcal{E} \\ 0, & \text{otherwise} \end{cases} \quad (2.6)$$

and $D(G)$ is a diagonal matrix composed of the sum of the rows of $A(G)$

$$D_{ii} = \sum_j A_{ij}. \quad (2.7)$$

$L(G)$ is positive semi-definite and, hence, admits an eigendecomposition. The eigenvalues of the $L(G)$ are non-negative and we have at least one eigenvalue equal to zero. The multiplicity of the eigenvalue zero allows to determine the number of sub-dynamics embedded in the

graph.

Proposition 1. *The eigendecomposition of the Laplacian generates a set of M positive eigenvalues $\Lambda = \{0 = \lambda_0^1 = \lambda_0^2 = \dots = \lambda_0^Q < \lambda_1 \leq \dots \leq \lambda_{M-Q}\}$. The multiplicity of the eigenvalue 0 is equal to the number of sub-dynamics Q .*

Proof. The multiplicity of the eigenvalue zero of the Laplacian matrix is equal to the number of connected components, hence, by construction of G is equal to the number of sub-dynamics Q . \square

As the graph G is composed of a set of disconnected components, the eigendecomposition of $L(G)$ can be performed by blocks and we can associate a subset of eigenvectors to each of the Q blocks. We show in the next subsection that each set of non-trivial— $\lambda \neq 0$ —eigenvectors of $L(G)$ generates Q separate sub-spaces in which each sub-dynamics is linearized.

Observe first that, in each of the graph connected component, the nodes are connected in such a way that each trajectory forms a path in the graph. With K trajectories for each sub-dynamics, we have K paths. Each trajectory is connected to another trajectory through the last node of each path graph.

2.4.1 Determining the Eigenvectors that Generate a Linear Embedding

We are now ready to present the main results of this paper, namely that a well-chosen set of eigenvectors of the graph Laplacian generates a linear embedding of the sub-dynamics. All the following results relate to the non-trivial eigenvectors, for which $\lambda \neq 0$.

We start by showing that, for each path in the graph, the corresponding entries in the eigenvectors follow a recursive law.

Lemma 1. *Consider K different path graphs that form a graph G with one connected component. Let \mathbf{u} be an eigenvector of the Graph Laplacian $L(G)$. The elements of \mathbf{u} entail K sets of scalars $\{u_1^k, \dots, u_{p_k}^k\}$, corresponding to the nodes within the k -th path graph. Each set follows the recursive relation:*

$$\begin{aligned} u_1^k &\in \mathbb{R}, \\ u_2^k &= (1 - \lambda)u_1^k, \\ u_n^k &= (2 - \lambda)u_{n-1}^k - u_{n-2}^k, \quad \text{for } n = 3, \dots, p_k. \end{aligned} \tag{2.8}$$

Proof. See Appendix A.1. \square

Lemma 2. *In each of the K sets of elements in \mathbf{u} denoted as $\{u_1^k, \dots, u_{p_k}^k\}$, the recursive relation in Eq. 2.8 corresponds to a combination of Chebyshev polynomials T and V , of first and second kind, given by:*

$$u_n^k = u_1^k \left[T_n(\lambda) - \frac{\lambda}{2} V_{n-1}(\lambda) \right] \quad \text{for } n \geq 1. \tag{2.9}$$

Chapter 2. Linearization and Identification of Multiple-Attractor Dynamical Systems through Laplacian Eigenmaps

Let $\theta := \cos^{-1}(1 - \frac{\lambda}{2})$. Eq. 2.9 can be expressed as the following combination of trigonometric functions:

$$u_n^k(\lambda, \theta(\lambda)) = u_1^k \left[\cos((n-1)\theta) - \frac{\lambda \sin((n-1)\theta)}{2 \sin(\theta)} \right]. \quad (2.10)$$

Proof. See Appendix A.2. □

Next, we show that, when we select a specific set of eigenvectors, the coordinates of the path graphs in these eigenvectors either grow or decrease monotonically. This a key property to prove the linearity of the embedding.

Proposition 2. *If each of the path graphs have at least 3 nodes, $p_k \geq 3 \forall k$, the coordinates of a path graph $\{u_1^k, \dots, u_{p_k}^k\}$ in the eigenvectors \mathbf{u} with corresponding eigenvalues $0 < \lambda \leq 2 \left[1 - \cos\left(\frac{\pi}{p_k - \frac{1}{2}}\right) \right]$, either increase or decrease monotonically within each path graph, s.t.:*

$$u_n^k \geq (\leq) u_{n+1}^k \quad \text{for } n = 1, \dots, p_k, \quad \text{and } k \in \{1, \dots, K\} \quad (2.11)$$

Proof. See Appendix A.3. □

Next, we prove that, when all the path graphs have the same length, i.e $p_k = N, \forall k$, there always exist, at least, one pair of eigenvectors with properties of Prop. 2, by showing that there exists an eigenvalue $\lambda \leq 2 \left[1 - \cos\left(\frac{\pi}{p_k - \frac{1}{2}}\right) \right]$ with algebraic multiplicity 2.

The proof is done in two steps: first, we show that the spectrum of $L(G)$ presents repeated eigenvalues; second, we show that there is at least one repeated eigenvalue upper bounded by a value inferior to the monotonicity constraint introduced in Prop. 2.

Following Gupta et al. (2022), observe that the graph Laplacian, analyzed in this work, has the following structure:

$$L(G) = 2I - J(G), \quad (2.12)$$

where J has a block circulant matrix structure of the following type

$$J(G) = \text{circ}(B_0, B_1, \underbrace{\mathbf{0}^{(N \times N)}}_{K-3 \text{ times}}, B_1), \quad (2.13)$$

with K the number of path graphs and N the number of nodes in each path graph. The

2.4 Graph Embedding and Linearization of Dynamical Systems

matrices $B_0 \in \mathbb{R}^{N \times N}$ is a tri-diagonal matrix of the form

$$B_0 = \begin{bmatrix} 1 & 1 & & & & \\ 1 & 0 & \ddots & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & 0 & 1 & \\ & & & 1 & -1 & \end{bmatrix} \quad (2.14)$$

and $B_1 \in \mathbb{R}^{N \times N}$ is defined as

$$B_1 = \begin{bmatrix} \mathbf{0}^{(N-1 \times N-1)} & 0 \\ 0 & 1 \end{bmatrix}. \quad (2.15)$$

We repeat results from Gupta et al. (2022) in Proposition 3 and 4:

Proposition 3. *If the number of paths K is even, $\frac{K}{2} - 1$ eigenvalues of $L(G)$ have multiplicity 2 and for K odd, $\frac{K-1}{2}$ eigenvalues of $L(G)$ have multiplicity 2.*

Proof. See Appendix A.4. □

Proposition 4. *The second smallest eigenvalue of the Graph Laplacian $L(G)$ with algebraic multiplicity 2 is denoted by $\lambda_{\min}(L)$ and is bounded above and below as follows:*

$$1 - 2 \cos\left(\frac{\pi}{N}\right) < \lambda_{\min}(L) \leq 2 \left(1 - \cos\left(\frac{\pi}{N - \frac{1}{2}}\right)\right) \quad (2.16)$$

where N is the number of nodes in each of the K paths.

Proof. See Appendix A.5. □

We have shown that there exist at least a pair of eigenvectors with same eigenvalues in which the corresponding entries of our path graph grow or decrease monotonically.

Next, we show that the DS is linear in the embedding formed by these two eigenvectors.

Corollary 1. *For each eigenvector u with associated eigenvalue $0 < \lambda < 1$, the rate of change $r(u_n) = u_n - u_{n-1}$ along the entries $\{u_1, \dots, u_{p_k}\}$, for each path $k = 1, \dots, K$ decreases monotonically according to:*

$$\dot{r}(u) = -\lambda u. \quad (2.17)$$

Proof. $r(u_{n+1}) = u_n - u_{n-1} - \lambda u_n = r(u_n) - \lambda u_n$. □

Theorem 1. *If the graph Laplacian $L(G)$ admits a set of eigenvectors \mathbf{u} , with same eigenvalue, the K paths of the graph expressed in the domain spanned by \mathbf{u} form K lines.*

Chapter 2. Linearization and Identification of Multiple-Attractor Dynamical Systems through Laplacian Eigenmaps

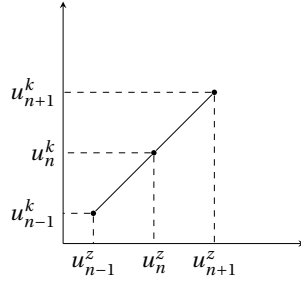


Figure 2.3: Coordinates of three points on two eigenvectors with constant rate of change, resulting in a linear path.

Proof. Consider a group of three consecutive nodes $\{v_{n-1}, v_n, v_{n+1}\}$ of one of the paths of the graph. The coordinate of these points when projected in the space spanned by two eigenvectors u^z and u^k are $\{u_{n-1}^{z,k}, u_n^{z,k}, u_{n+1}^{z,k}\}$. We show that the three points are on the same line, as illustrated in Fig.2.3. We prove that these coordinates grow at the same rate for all the eigenvectors that have equal eigenvalues $\lambda < 1$. Hence the spacing across each group of coordinates on each axis is the same, as illustrated in Fig.2.3.

In the following, we drop the upper index referring to the specific eigenvector. For any eigenvector with eigenvalue $0 < \lambda < 1$, using Eq. A.2, we have

$$u_{n+1} = \gamma u_n - u_{n-1}, \quad (2.18)$$

with $\gamma = 2 - \lambda, \gamma > 1$. Furthermore, for the first two points on the path graph, we have, from Eq. A.1: $u_2 = \delta u_1$, with $\delta = 1 - \lambda, \delta > 0$. Combining these two equations, the third point can be expressed as a function of the first coordinate u_1 only: $u_3 = \gamma(\delta u_1) - u_1 = (\gamma\delta - 1)u_1$. Similarly the fourth point can be expressed solely as a function of the first coordinate u_1 . We have: $u_4 = \gamma u_3 - u_2 = \gamma(\gamma\delta - 1)u_1 - \delta u_1 = (\gamma(\gamma\delta - 1) - \delta)u_1$. The same reasoning holds for all points and hence by induction, we have:

$$u_n = P_n(\lambda)u_1. \quad (2.19)$$

$P_n(\lambda)$ is a polynomial of order $n - 2$ that depends only on the eigenvalue λ . If the eigenvectors considered have same eigenvalue λ , the coordinates along each axis grow with the same series of polynomial $P_n(\lambda)$. Hence, all points are located on a line. For a pair of eigenvectors z, k , the line has slope $\frac{u_1^k}{u_1^z}$. \square

Note that, while Thm. 1 holds for all the eigenvectors in the spectrum of $L(G)$, the original ordering of the nodes along each path graph is preserved only when $\lambda < 2 \left[1 - \cos \left(\frac{\pi}{p^{k-\frac{1}{2}}} \right) \right]$. For the eigenvectors with larger λ , the coordinates of the nodes change sign within the path graph. Hence, while the path may still appear linear in the embedding, the order of the nodes will not be preserved anymore.

2.4.2 Creating a Linear Embedding

We summarize how the above theoretical results allow us to find embedding spaces so that we can separate each of the Q sub-dynamics and that they are linear in each embedding.

First, let us recall that, since the matrix $L(G)$ can be decomposed by block with each

2.4 Graph Embedding and Linearization of Dynamical Systems

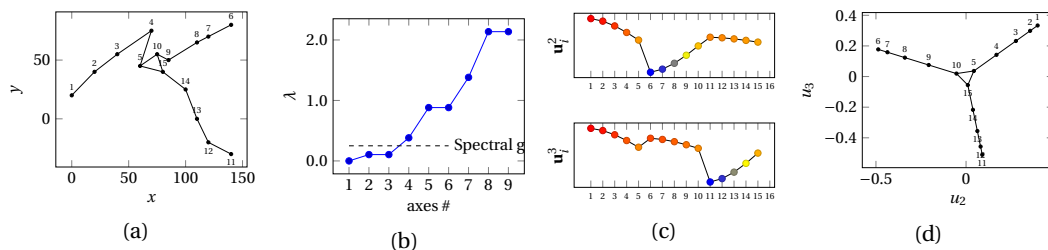


Figure 2.4: (a) Toy graph composed by 3 path graphs connected each other with a cycle graph passing through nodes $\{5, 10, 15\}$; (b) Spectrum of the first 9 eigenvalues of the Laplacian applied to graph in Fig. 2.4a; (c) Entries of the first 2 components $\{\mathbf{u}^2, \mathbf{u}^3\}$; (d) 2D embedding space reconstructed using components $\{\mathbf{u}^2, \mathbf{u}^3\}$.

block representing data points that belong to each sub-dynamics \mathbf{f}_q , we can generate a set of Q separate embedding spaces by using only the subset of eigenvectors associated to each sub-block. By orthogonality of the eigenvectors, all other sub-dynamics \mathbf{f}_k , $k \neq q$ will project to the origin in the embedding space of the q -th dynamics. We now need to determine the existence of such embedding spaces.

Given a K -paths graph, from Prop. 3, for $N \geq 3$ and $K \geq 3$, there is at least one pair of eigenvector with eigenvalue $0 < \lambda \leq 2 \left[1 - \cos \left(\frac{\pi}{N-1/2} \right) \right]$. It follows that if the Q dynamics of the graph have at least $N \geq 3$ nodes and $K \geq 3$, we have $2Q$ eigenvectors with properties of Prop. 2 and hence Q embedding spaces.

The properties stated in the previous propositions are illustrated in Fig. 2.4a. We see a three-path graph connected by a cycle path across nodes $\{5, 10, 15\}$. The entries of the second and third eigenvectors (we exclude the first eigenvector, as it corresponds to the eigenvalue zero) are displayed in Fig. 2.4c. To ease the interpretation, we enumerate the data points in increasing order as we move from the start to the end of each path. We plot the entries of the eigenvectors against the point label. Observe that, within each path graph, the corresponding entries on the eigenvectors are either strictly increasing or decreasing. Further, observe that the spectrum of the Laplacian, see Fig. 2.4b, entails a pair of eigenvalues with algebraic multiplicity equal to 2. The embedding space reconstructed using the eigenvectors $\{\mathbf{u}^2, \mathbf{u}^3\}$ corresponding to these two eigenvalues is shown in Fig. 2.4d. Observe that the structure now entails 3 paths all of which are linear. The original non-linear structure is hence linear in the embedding space.

In practice, we observe that several pairs of eigenvectors provide these embedding spaces. We also observe that the third and above eigenvalues are numerically close to the pair of smallest eigenvalues, and hence allow quasi-linear embedding spaces larger than 2 dimensions and up to $K - 1$. This follows by observing that the polynomial $P_n(\lambda)$ in Eq. 2.19 is continuous and continuously differentiable up to $N - 1$.

Chapter 2. Linearization and Identification of Multiple-Attractor Dynamical Systems through Laplacian Eigenmaps

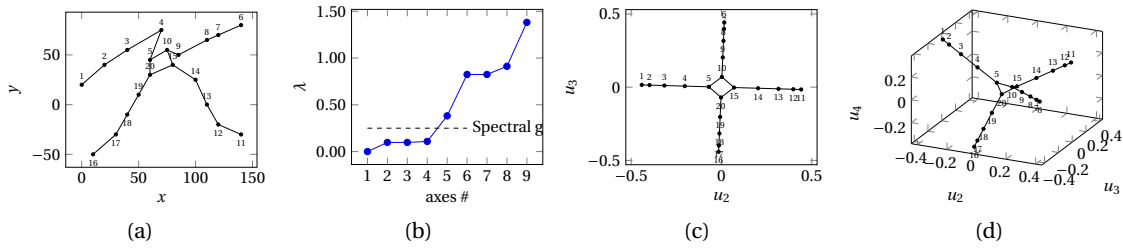


Figure 2.5: (a) Toy graph composed by 4 path graphs connected each other with a cycle graph passing through nodes $\{5, 10, 15, 20\}$; (b) Spectrum of the first 9 eigenvalues of the Laplacian applied to graph in Fig. 2.5a; (c) 2D embedding space reconstructed using components $\{\mathbf{u}^2, \mathbf{u}^3\}$; (d) 3D embedding space reconstructed using components $\{\mathbf{u}^2, \mathbf{u}^3, \mathbf{u}^4\}$.

We illustrate this property in two examples using 4-path and 5-path graphs, respectively, see Fig. 2.5 and 2.6. As in our previous example with a 3-path graph, the path graph are connected through one cycle graph. For the 4-path graph, we see that the spectrum of the Laplacian entails two eigenvalues of equal magnitude. The third eigenvalue has magnitude comparable but larger than the previous two, Fig. 2.5b. The graph can be linearized using the two eigenvectors with equal eigenvalues, see Fig. 2.5c. Quasi-linearization in the 3D embedding space can be achieved by using as coordinates the first three components, see Fig. 2.5d.

Using the 5-path graphs, we obtain two pairs of eigenvalues with equal magnitude, visible in the spectrum, Fig. 2.6b. We can hence reconstruct two distinct 2D embedding spaces where the DS is perfectly linear. While the embedding space constructed using eigenvectors $\{\mathbf{u}^2, \mathbf{u}^3\}$ preserves the radial ordering of the path graphs, Fig. 2.6c, the embedding constructed with components $\{\mathbf{u}^4, \mathbf{u}^5\}$ does not, as shown Fig. 2.6d.

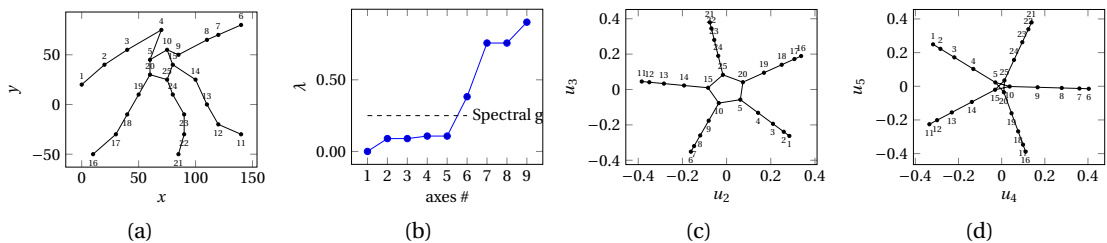


Figure 2.6: (a) Toy graph composed by 5 path graphs connected each other with a cyclic graph passing through nodes $\{5, 10, 15, 20, 25\}$; (b) Spectrum of the first 9 eigenvalues of the Laplacian applied to graph in Fig. 2.6a; (c) 2D embedding space reconstructed using components $\{\mathbf{u}^2, \mathbf{u}^3\}$; (d) 2D embedding space reconstructed using components $\{\mathbf{u}^4, \mathbf{u}^5\}$.

2.5 Generating the Graph with Real Data

To exploit the results stated in the previous section, we must first embed the data in a graph. To achieve this, we propose a kernel that takes advantage of positions and velocities.

2.5.1 Velocity-Augmented Kernel

The kernel provides a distance measure across nodes v_i, v_j :

$$k(v_i, v_j) = \exp \left(-\frac{1}{2\sigma^2} \left(\underbrace{\|\mathbf{x}_i - \mathbf{x}_j\|^2}_{\text{locality}} + \overbrace{\left(|g(\mathbf{x}_i, \mathbf{x}_j, \dot{\mathbf{x}}_i)| \right)^2 + \left(|g(\mathbf{x}_j, \mathbf{x}_i, \dot{\mathbf{x}}_j)| \right)^2}^{\text{directionality}} \right) \right), \quad (2.20)$$

where $\dot{\mathbf{x}}_i = \mathbf{f}(\mathbf{x}_i)$ is the DS map that relates positions to velocities.

We introduce g a function to measure the angular distance across the velocity vectors for a pair of data points \mathbf{x}_i and \mathbf{x}_j . We relate the distance vector $\mathbf{x}_j - \mathbf{x}_i$ and the velocity vector $\dot{\mathbf{x}}_i$ through a cosine kernel

$$g(\mathbf{x}_i, \mathbf{x}_j, \dot{\mathbf{x}}_i) = \underbrace{\delta_{\sigma_f}(\|\dot{\mathbf{x}}_i\| \|\dot{\mathbf{x}}_j\|)}_{\text{Filter}} + \underbrace{\left\langle \frac{\mathbf{x}_j - \mathbf{x}_i}{\|\mathbf{x}_j - \mathbf{x}_i\|}, \frac{\dot{\mathbf{x}}_i}{\|\dot{\mathbf{x}}_i\|} \right\rangle}_{\text{Cosine similarity kernel}}, \quad (2.21)$$

where δ_{σ_f} is a normal Gaussian distribution $\delta_{\sigma_f}(x) = \exp\left(-\frac{x^2}{2\sigma_f^2}\right)$ centered in $\|\dot{\mathbf{x}} = 0$, with σ_f set approximately to 0. The presence of the filtering part in Eq. 2.21 is necessary to withhold the "directionality" penalization among zero velocity points potentially close to an attractor. The filter δ_{σ_f} takes care of this situation outputting 1 whenever the velocity is zero. The parameter σ_f can be regulated to take into account known noise, e.g., when one knows that the velocity at an attractor point is not zero for numerical reasons.

The linear function $\gamma: \mathbb{R} \rightarrow \mathbb{R}$ in Eq. 2.20 is defined as

$$\gamma(g) = \frac{3}{2}\theta_r(1-g)\sigma. \quad (2.22)$$

The co-domain the function g , in Eq. 2.21, is in between -1 and 1 . The function γ , in Eq. 2.22, maps the part of the co-domain of g in between $\cos(\theta_r)$ and 1 to the range between 3σ and 0 . Consequently, the range of values of g in between -1 and $\cos(\theta_r)$ will be mapped to values greater than 3σ . When the angle between two velocity vectors is θ_r the linear map yields a value equal to 3σ causing a penalization of the weight about 99%. When the output of the cosine is proximal to 1 , due to the almost complete co-linearity, the linear function will yield 0 causing no penalization over the weight produced by the standard RBF (Radial Basis Functions) kernel. The reference angle, θ_r , is a parameter that can be set depending on the sampling frequency. High frequencies allow for a more strict ($\theta_r \approx 0$) selection of such angle.

Chapter 2. Linearization and Identification of Multiple-Attractor Dynamical Systems through Laplacian Eigenmaps

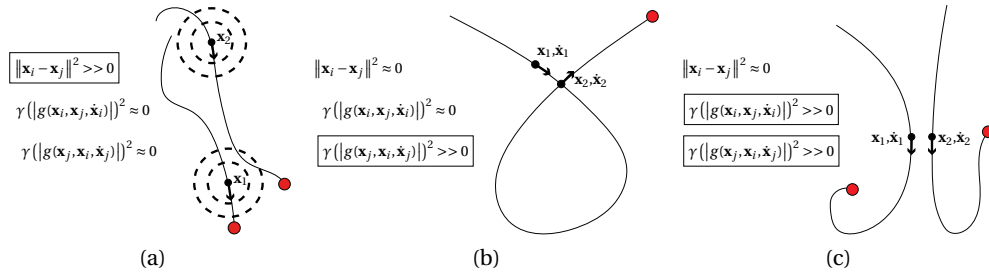


Figure 2.7: Illustration of the effect of the spatial distribution of points and velocity vectors on the kernel Eq. 2.20: (a) the first term in the exponential generates low values for points that are far apart; (b) the third term in the exponential generates low values when consecutive velocity vectors are not aligned; (c) the second and third term in the exponential generate low values whenever the distance vector connecting two points is not aligned with the velocity vector of one of them.

To understand the kernel better, we illustrate the effect of each term in three scenarios depicted in Fig 2.7. The directed kernel encompasses two components: "locality" and "directionality". The locality term gives a measure of the spatial distance between pairs of points based on the Euclidean distance, similar to the standard RBF kernel. The two directionality terms measure the co-linearity of the velocity vectors. Two points far apart with co-linear velocity vectors or two points close with distinct velocity vector will have a small value on the kernel and hence a loose connectivity in the similarity matrix used to generate the graph. Sole points that are both close and with co-directed velocity will reach the maximum kernel value, 1.

Points \mathbf{x}_1 and \mathbf{x}_2 in Fig. 2.7a will have negligible connection weights due to the large distance between them even though the "directionality" terms would yield values closed to 1 (about 0 after mapping γ) since $\mathbf{x}_2 - \mathbf{x}_1 \parallel \dot{\mathbf{x}}_1$ and $\mathbf{x}_1 - \mathbf{x}_2 \parallel \dot{\mathbf{x}}_2$. The second example in Fig. 2.7b considers the possibility of having intersecting trajectories perhaps due to second order dynamics or imprecise user demonstration. Since the figure might be misleading we

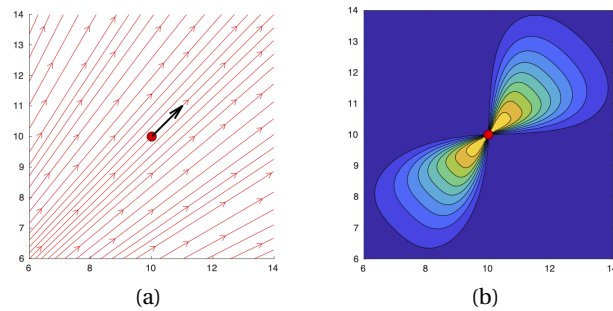


Figure 2.8: (a) Background vector field and reference evaluation point/velocity; (b) Contour of the velocity-augmented kernel.

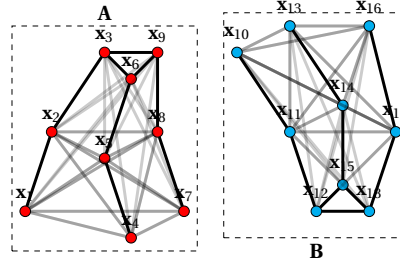


Figure 2.9: Toy data set of 2-attractor samples DS.

clarify that point \mathbf{x}_1 belongs to the beginning of the trajectory while point \mathbf{x}_2 lies at the end, close to the attractor. In this situation we would like to "separate" the two points since they are not close from a dynamical perspective. Due to the proximity of the points, the "Locality" part contributes to generate a strong connection between the two points. The second component of the "directionality" part will take care of drastically decreasing the weight connection. Indeed, if the first term yields a high value, $\mathbf{x}_2 - \mathbf{x}_1 \parallel \dot{\mathbf{x}}_1$, the second term will be decisive in cutting of the connection between the points since $\mathbf{x}_1 - \mathbf{x}_2 \approx \dot{\mathbf{x}}_1 \perp \dot{\mathbf{x}}_2$. As last scenario in Fig. 2.7c, we consider the case of two proximal trajectories belonging to different sub-dynamics. As in the previous example the "Locality" cannot account for this situation; the "directionality" part will take care of decreasing the weight connection given that $\mathbf{x}_2 - \mathbf{x}_1 \perp \dot{\mathbf{x}}_1$ and $\mathbf{x}_1 - \mathbf{x}_2 \perp \dot{\mathbf{x}}_2$.

Fig. 2.8a shows streamlines of an expanding vector field $\dot{\mathbf{x}} = \mathbf{x}$ and a reference point with its velocity vector. In Fig. 2.8b it is possible to see how the proposed kernel generates, with respect to the considered vector field, high value regions symmetrically placed with respect to the plane orthogonal to reference velocity in the proximity of the evaluation point.

2.5.2 Selecting the Hyperparameters

The choice of the hyperparameter σ is important as it modulates the granularity of the distance measure in Cartesian space. To inform the choice of σ , we can use the sampling frequency $f_{sampling}$ of the acquisition system, when recording trajectories. The higher the frequency, the closer the two consecutive data points. The maximum distance between two consecutive point is $d_{max} = \frac{\text{argmax}(\mathcal{D})}{f_{sampling}}$ with $\mathcal{D} := \{\dot{\mathbf{x}}_i, i = 1, \dots, m\}$. If sampling is perfect (no frame loss), we can set $\sigma = d_{max}$. Otherwise, a margin of error may be warranted. The adjacency matrix is computed as follows: $A_{ij} = 1$ if $k(v_i, v_j) \geq \epsilon$ otherwise $A_{ij} = 0$. The tolerance ϵ must be chosen in relation to σ . For instance, if $\sigma = d_{max}$, the kernel will be equal to at maximum 0.6 for the two most distant pair of points.

2.6 Clustering Dynamics & Attractor Search

We consider as toy example a 2-attractor DS where each sub-dynamics is constituted by 3 sampled trajectories. Fig. 2.9 shows the connection strength generated by the velocity-

Chapter 2. Linearization and Identification of Multiple-Attractor Dynamical Systems through Laplacian Eigenmaps

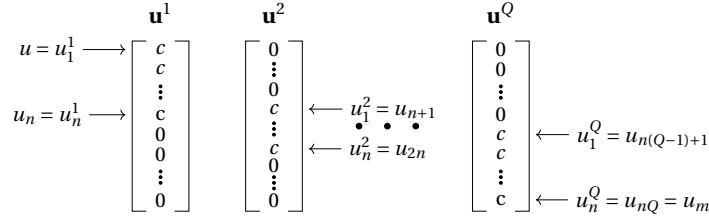


Figure 2.10: Structure of the constant right eigenvectors associated to eigenvalues equal to 0.

augmented kernel in Eq. 2.20. Connection between nodes belonging to different trajectories will have very low weight. Only points at the end of each trajectory, close to an attractor, will have high weight even if they belong to different trajectories.

The labeling of points, for clustering dynamics, is done by taking advantage of the first right eigenvectors, $\{\mathbf{u}^1, \dots, \mathbf{u}^Q\}$, connected to the eigenvalues equal to zero. Assuming n data points for each of the Q demonstrated dynamics the k -th eigenvectors extracted will have non-zero entries between $\mathbf{u}_{n(k-1)+1}$ and \mathbf{u}_{nk} , see Fig. 2.10. For each eigenvector the coefficients corresponding to the points belonging to a particular DS (sub-graph) are equal to a constant value c while all the others are 0.

In summary, the number of 0 eigenvalues is used to determine the number of attractors present in the DS. The corresponding eigenvectors are then used to cluster the data points, belonging to each sub-dynamics. Plotting the first four eigenvectors, Fig. 2.11, it is possible to observe that the spectral decomposition of the Laplacian produces a set of eigenvectors that are "expanding" one dynamics while "compressing", in zero, the other ones for each projected space. In this example \mathbf{u}^3 and \mathbf{u}^6 focused on sub-dynamics **A** while \mathbf{u}^4 and \mathbf{u}^5 on sub-dynamics **B**. The selection of the "interesting" components can be easily achieved by checking the change of the slope in the spectrum of the Laplacian matrix. Fig. 2.12 gives the spectrum analysis for varying the number of attractors, keeping the number of demonstrated trajectories to three. The number of relevant components grows proportionally to the number of sub-dynamics present in the dataset, $n_{DS} = \sum_{q=1}^K K_q - 1$ with Q being the total number of sub-dynamics and K_q the number of sampled trajectories for the q -th sub-dynamics. As shown in Alg. 1, taking advantage of the stationary right eigenvectors used to label points,

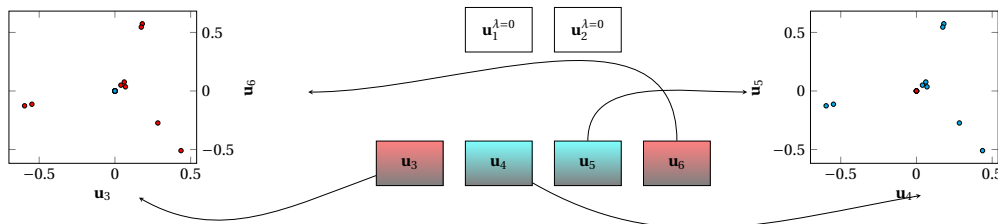


Figure 2.11: (left) $\{\mathbf{u}_3, \mathbf{u}_6\}$ embedding space; (center) Scatter matrix of the embedding space for the DS in 2.9; (right) $\{\mathbf{u}_4, \mathbf{u}_5\}$ embedding space.

2.6 Clustering Dynamics & Attractor Search

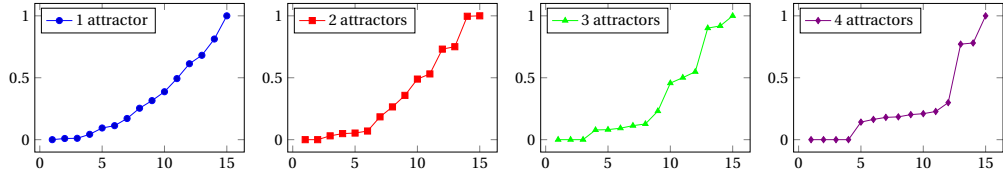


Figure 2.12: Spectrum analysis of the multiple-attractor DS for three demonstrated trajectories and increasing number of attractors.

each eigenvector is assigned to either one dynamics or the other. For the toy case in exam we have two eigenvectors associated to zero eigenvalues. The eigenvectors associated the least two eigenvalues, with algebraic multiplicity equal to two, are used for the reconstruction of the embedding spaces as shown in Fig. 2.11.

The search of the attractor positions is carried out by exploiting the particular structure of the embedding spaces. In each embedding space the related dynamics is "linearized", while the other ones, as said before, are "compressed" in zero. In this space the position of the attractor, \mathbf{u}^* , is easily found at the intersection of the lines whose slopes can be calculated using diffusion distances.

With reference to Fig. 2.13b, the algorithm randomly selects a point in the embedding space, \mathbf{u}_1 representing the vector containing the embedding coordinates of the chosen point. For a clear visualization of the problem, the process is illustrated in 2-dimensional space but, as shown before, the embedding space can have a higher dimension depending on the number of demonstrated trajectories or the input space dimension. \mathbf{u}_2 is selected searching among the nearest neighbors of \mathbf{u}_1 . The line direction $\mathbf{m}_r = \mathbf{u}_2 - \mathbf{u}_1$ is stored. The process is repeated until all the line directions, $\mathbf{m}_s = \mathbf{u}_4 - \mathbf{u}_3$, have been found. Pairwise intersections of the found lines can be simply calculated by solving the overdetermined linear system $A\mathbf{x} = \mathbf{b}$, where $A = [\mathbf{m}_s, \mathbf{m}_r]$, $\mathbf{x} = [\alpha, \beta]^T$ and $\mathbf{b} = \mathbf{u}_3 - \mathbf{u}_1$ for the two parametric lines, $r = \mathbf{u}_1 + \mathbf{m}_r\alpha$ and $s = \mathbf{u}_3 + \mathbf{m}_s\beta$. The mean of the intersection points calculated is used to established the position of the attractor, \mathbf{u}^* , in the embedding space.

Algorithm 1: Clustering Dynamics & Embedding Reconstruction.

Input: $\Lambda = \{\lambda_1, \dots, \lambda_m\}$, $U = \{\mathbf{u}_1, \dots, \mathbf{u}_m\}$

Output: \mathbf{x}_q^* , Q

$n_{\text{sub-DS}} = (\lambda_k == 0)$ // Compute the number of attractors

$U_{\text{clusters}} = \{\mathbf{u}_{\lambda=0}^1, \dots, \mathbf{u}_{\lambda=0}^{n_{\text{sub-DS}}}\}$ // Right "stationary" eigenvectors

while $\text{stop} = \text{false}$ **do**

$\text{stop} = \text{false}$

 /* From the first non-stationary eigenvector till the last one */

for $i = n_{\text{attractors}} + 1$ **to** m **do**

 /* Iterate over the number of attractors */

for $k = 1$ **to** $n_{\text{attractors}}$ **do**

 /* Assign eigenvector to sub-dynamics/attractor k */

if $\mathbf{1}^T (u_{\lambda=0}^k \circ |u_i|) \neq 0$ **then**

$U^k \leftarrow u_i$

 /* Stop loop at the spectral gap */

if $|\lambda_{i+1} - \lambda_i| > \text{tol}$ **then**

$\text{stop} = \text{true}$

Chapter 2. Linearization and Identification of Multiple-Attractor Dynamical Systems through Laplacian Eigenmaps

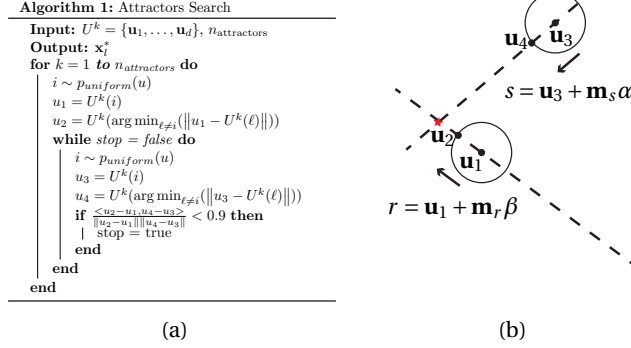


Figure 2.13: Finding the attractor in the embedding space.

2.7 Results

We validate our approach first at correctly identifying the underlying dynamics of well-known DS. We choose three nonlinear DS known to embed two separate sub-dynamics which are asymptotically stable at two separate attractors. Second, to test the sensitivity of the approach to real and noisy data, we validate the approach to decompose DS generated from handwritten data. For each of the systems, we generate a set of three example trajectories so as to obtain an embedding of the same dimension as the original system, namely $D = 2$.

To quantify the clustering results, we compare our approach, to three clustering techniques: Kernel K-means, Spectral Clustering and Gaussian Mixture Model (GMM). GMM adopts full covariance matrix. Kernel K-means and GMM require the number of clusters. For those, we provide them with the correct number of dynamics. For Kernel K-means we adopt, as similarity metric, a Radial Basis Functions (RBF) kernel with the hyper-parameter σ set as previously explained. Spectral Clustering adopts k -nearest neighborhoods technique for building the adjacency matrix and the identification of the sub-dynamics clusters is achieved through spectral decomposition of the symmetric normalized Laplacian. In order to provide the same information to all the algorithms, the feature state for the compared approaches has been augmented with the velocities.

As none of the clustering techniques we compare can extract the attractors, we compute the reconstruction error for the attractors only for our approach., the *Orthogonal Expansion*. The error is calculated as the squared error between the actual and the extracted location, normalized by the standard deviation of the position data set

$$e(\mathbf{x}^*) = ((\mathbf{x}_{real}^* - \mathbf{x}_{estimated}^*)^T \Sigma^{-1} (\mathbf{x}_{real}^* - \mathbf{x}_{estimated}^*))^{\frac{1}{2}}, \quad (2.23)$$

where Σ is the diagonal matrix containing the standard deviation of the dataset.

Note that K-means and GMM are iterative approaches sensitive to initialization. For each of them we provide average performance over 10 trials as the number of iterations increases.

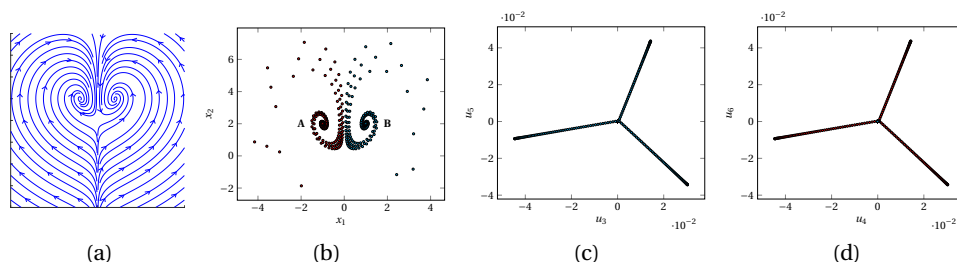


Figure 2.14: (a) Vector field generated by Eq. 2.24. (b) Sampled trajectories from the DS in Fig.2.14a. (c) Embedding space of the sub-dynamics with local attractor in $(1, -2)$. (d) Embedding space of the sub-dynamics with local attractor in $(1, 2)$.

Example. As a first example we consider the following 2-dimensional DS

$$\begin{aligned}\dot{x}_1 &= 2x_1 - x_1 x_2 \\ \dot{x}_2 &= 2x_1^2 - x_2,\end{aligned}\tag{2.24}$$

which present two stable equilibrium points in $(-1, 2)$ and $(1, 2)$. The vector field generated from Eq. 2.24 and the sampled trajectories are shown in Fig. 2.14, case (a) and (b) respectively. Such multiple-attractor DS represents a challenging case for our algorithm due to the proximity of the trajectories belonging to different sub-dynamics, symmetrically displaced with respect to the vertical axis passing through zero, Fig. 2.14b. In order to have a successful reconstruction of the graph we sampled from the DS for 10s at a frequency of 100Hz reaching a total of 6006 points sampled. The correct reconstruction of the graph can be assessed by looking at Fig. 2.14c and 2.14d, where the two sub-dynamics are correctly linearized. In particular using components \mathbf{u}_3 and \mathbf{u}_5 the embedding space linearizes sub-dynamics **B** while sub-dynamics **A** is compressed in zero; vice-versa for the embedding space constructed adopting \mathbf{u}_4 and \mathbf{u}_6 components. Results of clustering are reported in Tab. 2.1a and Fig.2.15. Kernel K-means does not yield good performance and, as shown in Fig. 2.1b, it remains fairly sensitive to centroids initialization. Spectral Clustering shows good performance being able to cluster correctly the

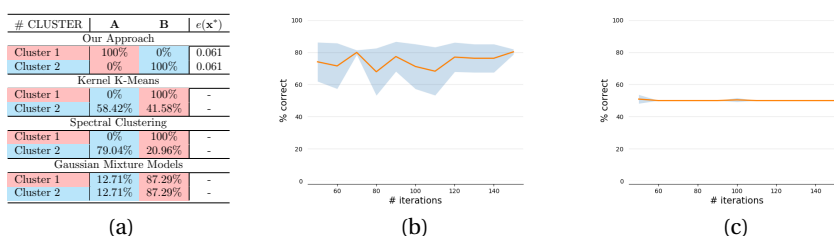


Table 2.1: For sampled points in Fig. 2.14b: (a) Clustering labeling and attractor location error results, (b) Kernel K-Means clustering error over iteration, (c) Gaussian Mixture Model clustering error over iteration.

Chapter 2. Linearization and Identification of Multiple-Attractor Dynamical Systems through Laplacian Eigenmaps

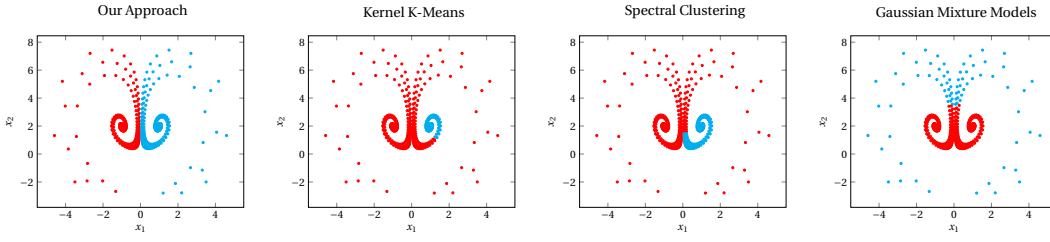


Figure 2.15: Clustering results of Tab. 2.1a.

two high-density region areas. Although it fails in clustering correctly regions with sparse data. GMM present stable solution with respect to parameters initialization, see Fig. 2.1c, but poor performance failing in placing the mixtures in a consistent way with respect to the two sub-dynamics. In particular the two high-density regions around the attractors are described by a single component yielding incorrect clustering.

Example. We consider in this example the pendulum equation with friction. The second order differential equation describing the dynamics of the pendulum is $\ddot{\theta} = -\frac{l}{g} \sin(\theta) - kl\dot{\theta}$, where l is the length of the pendulum, g the gravity constant and k the friction coefficient. The phase space of such system yields a multiple-attractor vector fields with periodic stable equilibria at $k2\pi$, with $k \in \mathbb{Z}$. Let $x_1 = \theta$ and $x_2 = \dot{\theta}$. The pendulum dynamics can be rewritten as a system of first order differential equations

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -\frac{l}{g} \sin(x_1) - klx_2 \end{aligned} \quad (2.25)$$

We consider a domain $x_1 \in [0, 2\pi]$ where such DS presents 2 attractors at the boundary of the domain and one unstable point at π . In order to asses the ability of our kernel to generate the desire graph structure to build the Laplacian matrix we appositely sampled proximal trajectories pointing to two different attractors (central region in Fig. 2.16b). Each trajectories has been sampled for 60s at a frequency of 10Hz for a total 3592 points. As it is possible to see

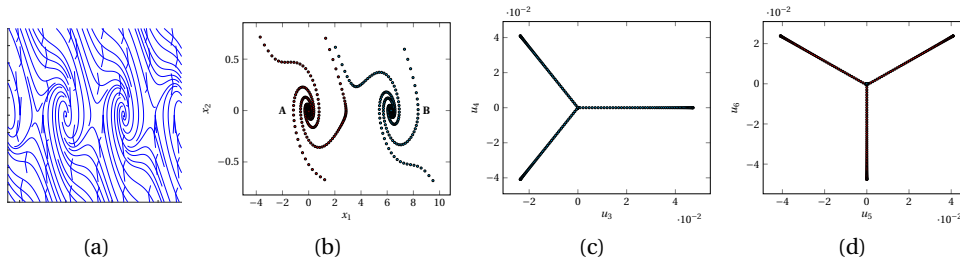


Figure 2.16: (a) Vector field generated by Eq. 2.25. (b) Sampled trajectories from the DS in Fig.2.16a. (c) Embedding space of the sub-dynamics with local attractor in (0,0). (d) Embedding space of the sub-dynamics with local attractor in (0, 2π).

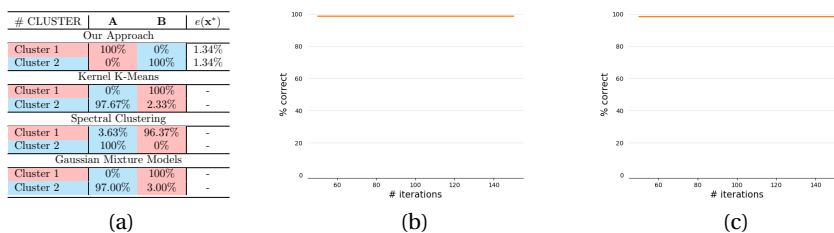


Table 2.2: For sampled points in Fig. 2.16b: (a) Clustering labeling and attractor location error results, (b) Kernel K-Means clustering error over iteration, (c) Gaussian Mixture Model clustering error over iteration.

from the embedding spaces extracted, Fig. 2.16c and 2.16d, the kernel is able to approximate the theoretical graph proposed leading to linearization of the sub-dynamics. In Tab. 2.2a and Fig. 2.17 the results of clustering are shown. All the algorithms yields good performance in this case correctly assigning points belonging to high-density regions around attractors. Nevertheless, with exception of our approach, they all fail to achieve perfect clustering due to mis-clustering in regions with sparse data.

Example. As a third example we consider the Duffing equation (or oscillator), a non-linear second-order differential equation used for modeling damped oscillator

$$\ddot{x} + \delta \dot{x} + \alpha x + \beta x^3 = 0. \quad (2.26)$$

Differently from a simple harmonic oscillator such equation models more complex potential by including a non-linear spring with restoring force $\alpha x + \beta x^3$. Such DS exhibits chaotic behavior. We consider positive damping case, $\delta = 0.3$, with $\alpha = -1.2 < 0$ and $\beta = 0.3 > 0$ which present two stable at $+\sqrt{-\alpha/\beta}$ and $-\sqrt{-\alpha/\beta}$. The phase space of the duffing equation can be achieved by the following two dimension DS

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= 0.3(4x_1 - x_1^3 - x_2). \end{aligned} \quad (2.27)$$

As for example 1, due the presence of highly compact and non-linear regions, high frequency

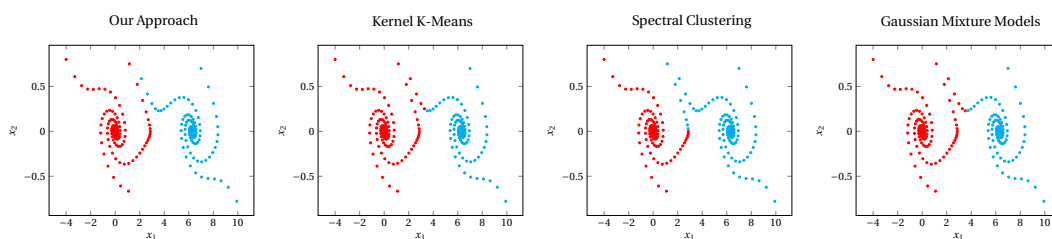


Figure 2.17: Clustering results of Tab. 2.2a.

Chapter 2. Linearization and Identification of Multiple-Attractor Dynamical Systems through Laplacian Eigenmaps

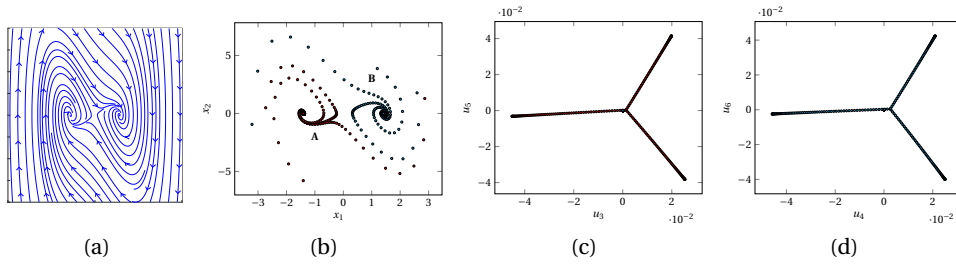


Figure 2.18: (a) Vector field generated by Eq. 2.27. (b) Sampled trajectories from the DS in Fig.2.18a. (c) Embedding space of the sub-dynamics with local attractor in $(0, \sqrt{-\alpha/\beta})$. (d) Embedding space of the sub-dynamics with local attractor in $(0, -\sqrt{-\alpha/\beta})$.

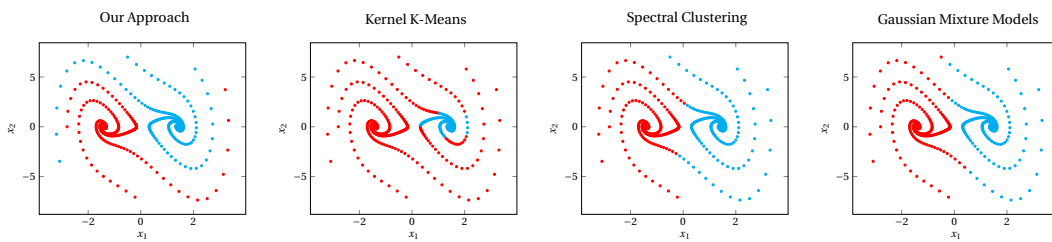


Figure 2.19: Clustering results of Tab. 2.3a.

sampling is required. Each trajectory is sample at $200Hz$ for $5s$ yielding a dataset of 6006 samples. For this case eigenvectors \mathbf{u}_3 and \mathbf{u}_5 yield an embedding space where sub-dynamics **A** is linearized while the space generate by \mathbf{u}_4 and \mathbf{u}_6 achieve linearization of sub-dynamics **B**. The results of clustering in Tab. 2.3a and Fig. 2.19 show a similar behavior for Spectral Clustering and GMM. Both algorithms yields quantitative good results. Although, due to the chaotic behavior of the DS they fail in clustering correctly trajectories close to the attractor belonging to another sub-dynamics. Kernel K-means is able to cluster correctly points lying close to the attractors but it is incapable of clustering the majority of the trajectories.

Example. As last examples we show applicability to hand-drawn multiple-attractor DS where our algorithm can be used to cluster the sub-dynamics and locate the various attractors for then providing such information to stable learning algorithms present in the literature.

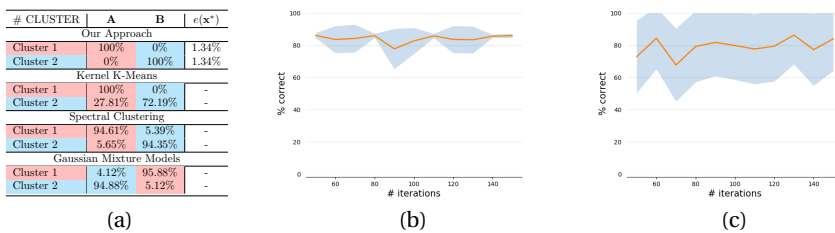


Table 2.3: For sampled points in Fig. 2.18b: (a) Clustering labeling and attractor location error results, (b) Kernel K-Means clustering error over iteration, (c) Gaussian Mixture Model clustering error over iteration.

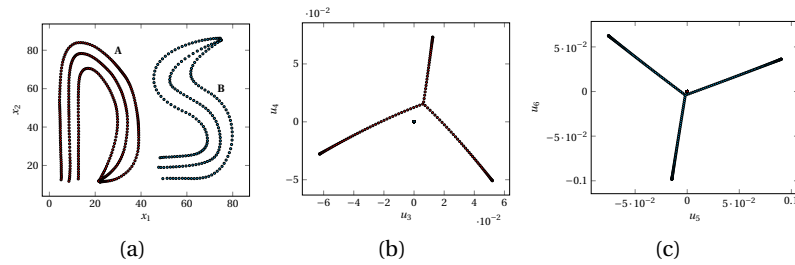


Figure 2.20: (a) Demonstrated trajectories. (b) Embedding space of the red sub-dynamics. (c) Embedding space of the cyan sub-dynamics.

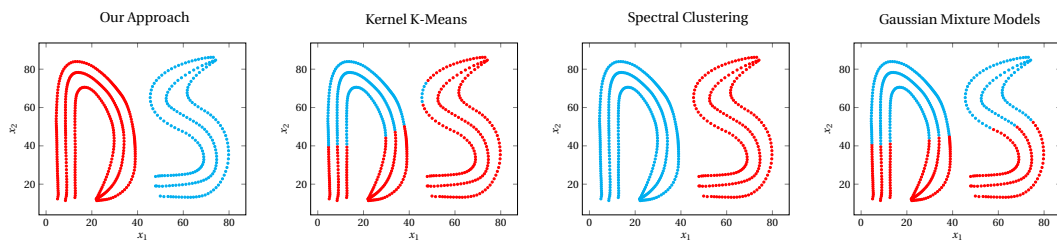


Figure 2.21: Clustering results of Tab. 2.4a.

Fig. 2.20a shows the demonstrated trajectories for a 2-attractor DS in 2D. For drawing such trajectories we took advantage of Wacom tablet and the software provided by ML_toolbox¹. Spectral Clustering is able to achieve perfect clustering while GMM misplaces the Gaussian components yielding poor results. Kernel K-means yields particular poor results in this case when a small kernel width such the one adopted for our algorithm is used. To enhance the performance of Kernel K-means, we set the kernel width to be equal to the standard deviation of the dataset. Nevertheless Kernel K-means is not able to consistently cluster the two sub-dynamics as shown in Fig. 2.4b. For such case the real attractor location is assumed to be the mean average of the position of the last point of the demonstrated trajectories. For this specific case we show results for the location of the attractor based on quasi-zero velocity heuristic.

¹https://github.com/epfl-lasa/ML_toolbox

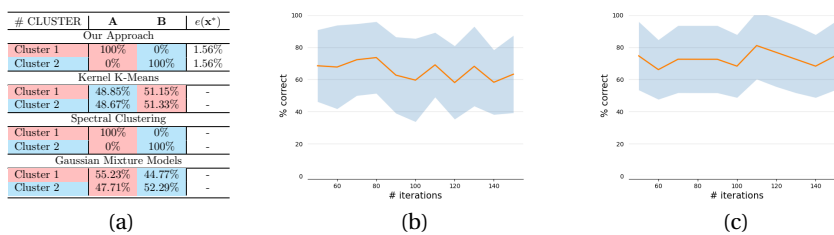


Table 2.4: For sampled points in Fig. 2.20a: (a) Clustering labeling and attractor location error results, (b) Kernel K-Means clustering error over iteration, (c) Gaussian Mixture Model clustering error over iteration.

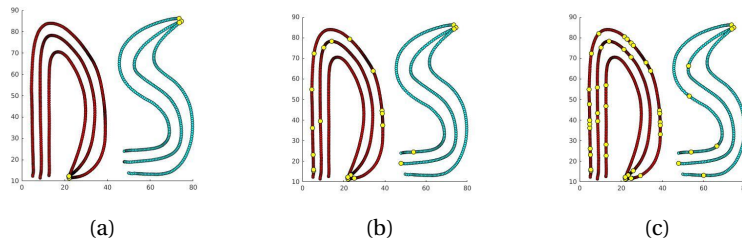


Figure 2.22: Quasi-zero velocity heuristic for attractor location. Threshold at (a) 10%, (b) 5% and (c) 1% of the average velocity.

Fig.2.22 shows the identification of potential locations for the attractors based on different velocity thresholds. Notice that zero-velocity crossing ends up with identifying multiple and incorrect attractor locations. This happens often at the beginning of a trajectory or in region of high curvature where the velocity can be proximal to zero.

2.8 Dynamical System Learning via NVP Transformations

From a differential geometry perspective Manifold Learning has a simple and straightforward interpretation. Consider the *differentiable* manifold \mathcal{M} in Fig. 2.23. The maps \mathbf{x} and \mathbf{u} , generally called *chart maps*, represent two different (local) representation of the manifold in the Euclidean spaces \mathbb{R}^d and \mathbb{R}^s .

Going back to the problem of learning stable DS, we view trajectories as motions over a differentiable manifold \mathcal{M} . Starting from an Euclidean representation \mathbb{R}^d of such motions (the original dataset), our approach is capable of reconstructing an alternative Euclidean representation \mathbb{R}^s of the manifold in which the DS looks linear. If the number of selected eigenvectors to reconstruct the embedding space is equal to the dimension of the original space, namely $s = d$, such ensemble represents the discrete counterpart of the continuous diffeomorphic map that links the two different representation of the manifold $\mathbf{u} \circ \mathbf{x}^{-1} : \mathbb{R}^d \rightarrow \mathbb{R}^s$. A differentiable manifold ensures that any two charts (representations of the same manifold) are differentiable-compatible, namely $\mathbf{u} \circ \mathbf{x}^{-1}$ and its inverse $\mathbf{x} \circ \mathbf{u}^{-1}$ are continuous and differ-

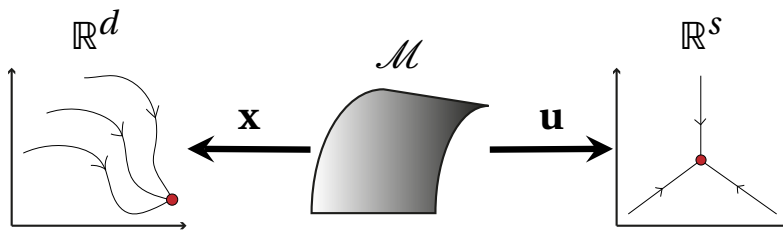


Figure 2.23: (left) Original Euclidean Space where the dataset is sampled from; (center) manifold where the DS is taking place; (right) extracted Euclidean embedding space.

2.8 Dynamical System Learning via NVP Transformations

entiable. This properties allows to relate velocity components in the two different Euclidean spaces by the means of the so called Jacobian. For easier notation let the diffeomorphism be $\psi = \mathbf{u} \circ \mathbf{x}^{-1}$, and the Jacobian $\mathbf{J}_\psi = \partial_{\mathbf{x}}\psi$ then

$$\dot{\mathbf{u}} = \mathbf{J}_\psi \dot{\mathbf{x}}. \quad (2.28)$$

It is easy to show that if Lyapunov stability is guaranteed in one Euclidean representation of \mathcal{M} the diffeomorphism *induces* the same property in the other one. Consider the existence of a Lyapunov function $V : \mathbb{R}^d \rightarrow \mathbb{R}$ radially unbounded and positive in all the space with $\dot{V}(\mathbf{x}) < 0$ except in \mathbf{x}^* , equilibrium point, where $V(\mathbf{x}) = \dot{V}(\mathbf{x}) = 0$. The diffeomorphism ψ generates a Lyapunov function $\tilde{V} : \mathbb{R}^s \rightarrow \mathbb{R}$ and a related attractor $\mathbf{u}^* = \psi(\mathbf{x}^*)$

$$\dot{\tilde{V}}(\mathbf{u}) = \frac{\partial V}{\partial \mathbf{x}} \frac{\partial \psi^{-1}}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial t} = \frac{\partial V}{\partial \mathbf{x}} \mathbf{J}_\psi^{-1} \mathbf{J}_\psi \dot{\mathbf{x}} = \frac{\partial V}{\partial \mathbf{x}} \dot{\mathbf{x}} = \dot{V}(\mathbf{x}) < 0. \quad (2.29)$$

Given the bijectivity of the diffeomorphism if \tilde{V} is a Lyapunov function defining a stable equilibrium point at \mathbf{u}^* , \mathbf{x}^* is a globally asymptotically stable equilibrium point. We apply this principle to reconstruct the dynamics in the original space.

We start by constructing a linear DS in embedding space that follows the linearized trajectories of our DS: $\dot{\mathbf{u}} = \mathbf{u}^* - \mathbf{u}$. This system is globally asymptotically stable at \mathbf{u}^* . Stability can be proved easily by considering the quadratic Lyapunov function $\tilde{V}(\mathbf{u}) = \frac{1}{2}(\mathbf{u}^* - \mathbf{u})^T(\mathbf{u}^* - \mathbf{u})$. Observe that \tilde{V} is also the potential of the vector field (namely $\dot{\mathbf{u}} = \nabla \tilde{V}$). Following from Eq. 2.28, the original dynamics can be recovered through

$$\dot{\mathbf{x}} = \mathbf{J}_\psi^{-1}(\mathbf{u}^* - \mathbf{u}) = \mathbf{J}_\psi^{-1}(\psi(\mathbf{x}^*) - \psi(\mathbf{x})). \quad (2.30)$$

The corresponding (deformed) Lyapunov function in original space can be recovered as $V = \tilde{V} \circ \psi = \frac{1}{2}(\psi(\mathbf{x}^*) - \psi(\mathbf{x}))^T(\psi(\mathbf{x}^*) - \psi(\mathbf{x}))$.

To reconstruct the diffeomorphic map using the embedding space as a ground truth, we adopt Non-Volume Preserving (NVP) transformations introduced by Dinh et al. (2017). The diffeomorphism is obtained through a sequence of k *coupling layers* each of which is given by:

$$\begin{cases} \mathbf{u}_{1:n} & = \mathbf{x}_{1:n} \\ \mathbf{u}_{n+1:d} & = \mathbf{x}_{n+1:d} \odot \exp(s(\mathbf{x}_{1:n})) + t(\mathbf{x}_{1:n}) \end{cases} \quad (2.31)$$

with $n < d$, d the dimension of the original space. $s(\cdot)$ and $t(\cdot)$ are scaling and translating functions, respectively. Each of these functions is approximated through Random Fourier feature approximation (Rahimi and Recht (2007a)) of a vector-valued isotropic Radial Basis Functions kernel as shown by Rana et al. (2020). As loss function, we use the standard Mean Squared Error, $\text{MSE} = \sum_{i=1}^{N_D} \|\mathbf{u}_i - \psi(\mathbf{x}_i)\|^2$, adopting the embedding space coordinates as ground truth.

We applied the diffeomorphism to learn a mapping from our original space to the embedding spaces for the hand-drawn multiple-attractor unknown DS, presented in Fig. 2.20a.

Chapter 2. Linearization and Identification of Multiple-Attractor Dynamical Systems through Laplacian Eigenmaps

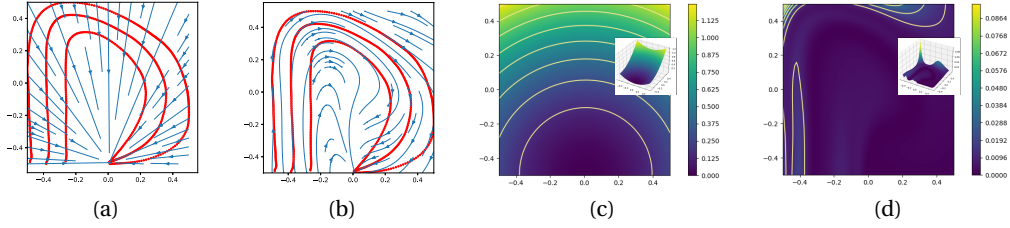


Figure 2.24: (a) Linear DS generated when using identity diffeomorphism; (b) reconstructed dynamics through learned diffeomorphism; (c) initial quadratic potential function generating a linear DS in Fig. 2.24a; (d) deformed potential under the action of the learned diffeomorphism generating the nonlinear DS in Fig. 2.24b.

Fig. 2.24 focuses on the red sub-dynamics. Fig. 2.24a shows the original linear dynamics produced by Eq. 2.30 when the diffeomorphism ψ is the identity. After having learned the diffeomorphism, the deformed nonlinear DS is shown in Fig. 2.24b. Fig. 2.24c and 2.24d show the potential function $V = \tilde{V} \circ \psi$ with identity and learned diffeomorphism, respectively. Results for the second extracted sub-dynamics are shown in Fig. 2.25.

Our approach to learning DS through diffeomorphic mapping is highly inspired by the works of Perrin and Schlehuber-Caissier (2016) and Rana et al. (2020). Different from the approach of Perrin and Schlehuber-Caissier (2016), our learned diffeomorphism generates a map from the original space to the embedding space, rather than the opposite. This leads to a different formulation of the deformed non-linear DS. The advantage of our approach is that it does not require to construct explicitly the inverse diffeomorphism. This comes at the cost of having to invert the Jacobian when reconstructing the DS. Compared to the approach proposed in Rana et al. (2020), we rely solely on the geometrical deformation of the space, induced by the embedding space information, without taking into account dynamics information. As consequence our loss function considers only position information (original vs. embedding space) discarding the velocities. This allows to achieve faster convergence, due to the simpler learning problem, and sensible lower learning time due to the advantage of

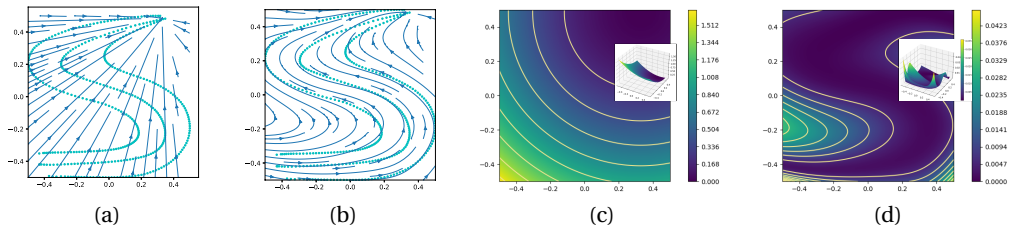


Figure 2.25: (a) Linear DS generated via identity diffeomorphism; (b) deformed DS under the action of the learned diffeomorphism; (c) initial quadratic potential function generating the linear DS in Fig. 2.25a; (d) deformed potential under the action of the learned diffeomorphism generating the nonlinear DS in Fig. 2.25b.

2.8 Dynamical System Learning via NVP Transformations

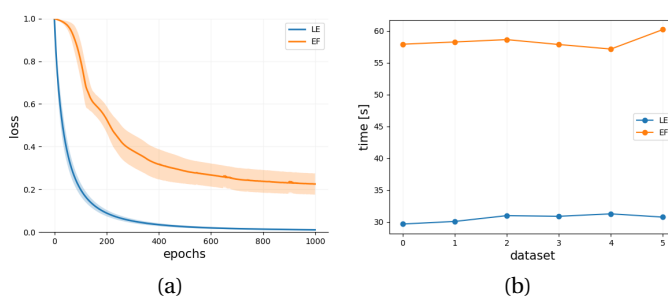


Figure 2.26: For Laplacian Embedding (LE) and Euclideanizing Flows (EF) approaches: (a) loss decay over 1000 epochs; (b) training time for 1000 epochs.

not having to calculate and invert the Jacobian in the process. Fig.2.26 shows a comparison between our approach, termed as Laplacian Embedding (LE), and Euclideanizing Flows (EF) presented in Rana et al. (2020). All the tests have been performed on a machine endowed with an Intel i9-10900K CPU and a NVIDIA GeForce RTX 2080Ti GPU². On the left it is possible to see how the easier formulation proposed in LE leads to a fast, exponential decaying of the loss while EF generally requires more epochs to properly converge. On the right the training time over 1000 epochs is shown. As LE does not require to invert the Jacobian, it requires approximately half training time compared to EF.

We evaluate the ability of the proposed non-volume preserving transformation to reconstruct the diffeomorphism between the original and the embedding space on the LASA dataset³, a dataset composed of hand-drawn letters, that has been often used to compare

²Repository available at: <https://github.com/nash169/learn-diffeomorphism>

³<https://bitbucket.org/khansari/lasahandwritingdataset>

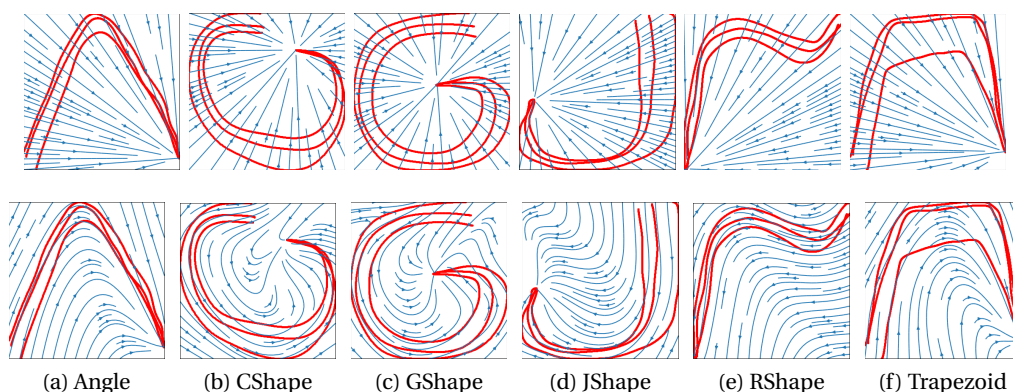


Figure 2.27: For each demonstrated DS (column wise), the first row shows the DS generated by the identity diffeomorphism (no train), the second row shows the DS generated after having learned the diffeomorphism between the original space and the embedding space, the third row shows the embedding space reconstructed using the eigenvectors extracted from the eigen-decomposition of the Laplacian matrix.

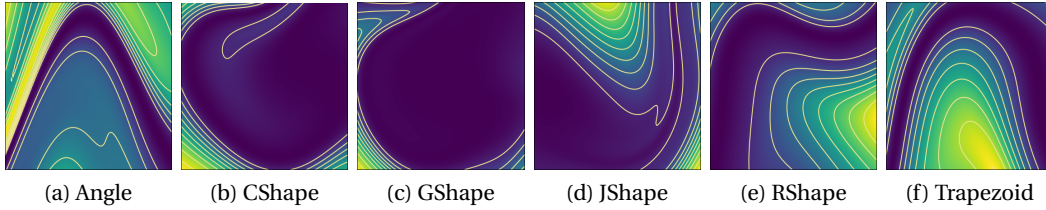


Figure 2.28: Deformed potential under the action of the learned diffeomorphism generating the nonlinear DS for the LASA dataset.

performance of learned non-linear DS. Each demonstration is composed by 7 trajectories (1000 samples of position-velocity pairs). Since, in two dimensions, the reconstruction of the embedding does not require more than three trajectories, for each demonstration, we discard 4 trajectories that we use at testing time to evaluate performance.

For the reconstruction of the embedding to be successful, it is necessary that the demonstrated trajectories are instances of a first order DS. While noise is tolerated, the trajectories should not intersect, as this would violate a fundamental property of first order DS and core assumption of the DS graph representation. Therefore, whenever it was possible, from each demonstration, we selected trajectories in line with this requirement. In those cases where it was not possible, in order to retain a good statistical analysis on the diffeomorphism learning part, we reconstructed the graph structure manually.

Fig. 2.28 shows the deformed quadratic potential function under the action of the learned diffeomorphism for a subset of demonstrations, while Fig. 2.28 shows the relative learned DS. To evaluate the performance of the algorithm we employ three metrics: (1) prediction cosine similarity error $\dot{e} = \frac{1}{N_D} \sum_{i=1}^{N_D} \left| 1 - \frac{f(\mathbf{x}_i^{ref})^T \dot{\mathbf{x}}_i^{ref}}{\|f(\mathbf{x}_i^{ref})\| \|\dot{\mathbf{x}}_i^{ref}\|} \right|$, (2) *average* Dynamic Time Warping Distance (DTWD) Salvador and Chan (2007) and (3) prediction Root Mean Square Error $RMSE = \frac{1}{N_D} \sum_{i=1}^{N_D} \|\dot{\mathbf{x}}_i^{ref} - f(\mathbf{x}_i^{ref})\|$. Table 2.5 shows the performance at reconstructing each of the demonstrations for each of the 12 dynamics shown in Figure 2.27.

The proposed implementation of the diffeomorphism appears more capable at shaping the streamlines of the vector field according to the demonstrated trajectories. It also has

	Angle	CShape	GShape	JShape	RShape	Trapezoid
RSME	23.92	16.55	18.50	16.11	14.36	15.13
DTWD	0.015	0.066	0.098	0.060	0.046	0.073
CS	0.679	0.992	0.979	0.861	0.504	0.749

Table 2.5: Performance evaluation at reconstructing the demonstrations for each of the 12 handdrawn examples of Figure 2.27. Performance is measured according to three metrics: root mean square error (RSMR), dynamic time warping distance (DTWD) and cosine similarity error (CS).

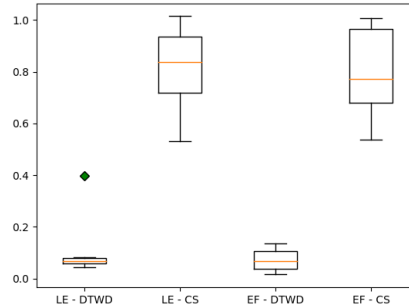


Figure 2.29: For Laplacian Embedding (LE) and Euclideanizing Flows (EF) approaches box plots comparison of DTWD and CS metrics.

good accuracy, with values of prediction cosine similarity error and average DTWD about 0.5 and 0.2, respectively, all of which are comparable to the current state-of-the-art (e.g. Figueroa and Billard (2018); Rana et al. (2020)) as shown in Fig.2.29. This comes at the cost of a poor RMSE, over the velocities, given that the velocity information has been discarded in the learning process. Reconstruction of the desired velocity profile can be achieved through proper re-scaling techniques of the reconstructed DS as shown, for instance, by Perrin and Schlehuber-Caissier (2016).

2.9 Conclusion

This paper showed that it is possible to automatically decompose a set of unlabelled data, stemming from a multiple-attractor DS, and to identify the number of underlying dynamics and their associated attractors. We further provided theoretical guarantees for DS linearization based on Manifold Learning reconstruction of multiple embedding spaces. We proved that, for a graph structure of the shape described in Sec. 2.4, the eigenvectors of the Laplacian matrix generate an embedding space where the sampled trajectories from a given DS are linearized. We introduced a novel velocity-augmented kernel to achieve the desired graph structure from real data. Relying on these theoretical results, we proposed an algorithm to cluster the sub-dynamics and identify the equilibria locations of multiple-attractor DS through eigendecomposition of a graph-based Laplacian matrix. In particular, we utilized the spectral properties of a Laplacian matrix applied to the particular graph proposed to reconstruct multiple embedding spaces, where each sub-dynamics is linearized while the others are compressed into a single point at zero. We showcased that, in such space, it is possible to identify the position of the attractor while, at the same time, clustering the sub-dynamics. Combined with state-of-the-art techniques for learning diffeomorphic maps, our method provides an algorithm for stable learning of multiple-attractor DS in a complete unsupervised learning scenario.

Limitations & Future Developments. Nevertheless, our algorithm relies heavily on the

Chapter 2. Linearization and Identification of Multiple-Attractor Dynamical Systems through Laplacian Eigenmaps

reconstruction of the correct graph. Whenever the velocity-augmented kernel is not able to reconstruct the correct graph, the clustering performance degrades drastically and it is not possible to locate the attractors anymore. This generally happens in extreme cases of considerable high curvature of the sampled trajectories. In addition we highlight that Spectral Clustering taking advantage of the kernel proposed in this work is capable of matching the clustering performance of our algorithm. However Spectral Clustering does not exploit the particular structure of the DS in the embedding space, and therefore, it is incapable of assessing the location of the attractors. In all examples used in this paper, the DS were 2-dimensional and represented single motion patterns. However, our algorithm is not limited theoretically to 2D and scales well to higher dimensions since both the proposed kernel and the Laplacian matrix are dimension-independent. The dimension of the embedding space is upper bounded by the number of trajectories sampled and it does not depend on the dimension of the original space.

3 Learning Dynamical Systems Encoding Non-Linearity within Space Curvature

3.1 Foreword

The work presented in this chapter is under review in the *International Journal of Robotics Research (IJRR)*, Fichera and Billard (2024).

3.2 Introduction

Learning from Demonstration (LfD) represents a powerful approach to derive global behavioral policies for high-level closed-loop control by observing demonstrated tasks. Such policies are represented using the mathematical framework of Dynamical Systems (DS), namely a vector field $\mathbf{f}: \mathbb{R}^d \rightarrow \mathbb{R}^d$, mapping the d -dimensional input state $\mathbf{x}(t) \in \mathbb{R}^d$ to its time-derivative $\dot{\mathbf{x}}(t) \in \mathbb{R}^d$, such that $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x})$.

In the field of robotics, this framework is commonly employed for describing and regulating various motion types, such as point-to-point motions characterized by fixed-point stable equilibrium DS, or periodic motions featuring stable limit-cycle DS. The stability of a learned DS becomes a significant concern when it is applied in closed-loop control systems. Using standard regression methods to learn the mapping \mathbf{f} provides no inherent guarantee of producing stable control policies. A wealth of learning approaches have been developed in the last decades to learn a DS with the stability guaranteed. They follow two fundamental paths: 1) constraint optimization; 2) learning of complex potential function via diffeomorphism.

In the first category, the most popular approach is to derive constraints via Lyapunov's second method for stability. In the beginning, Khansari-Zadeh and Billard (2011) utilized a quadratic Lyapunov function to establish stability conditions in a Gaussian Mixture Regression (GMR) problem for learning Dynamical Systems (DS). However, stability guarantee imposes a severe restriction on the learnable complexity of the DS and prevents learning highly non-linear DS containing high-curvature regions or non-monotonic motions (i.e., temporally moving away from the attractor). More recent approaches tried to alleviate this issue by im-

Chapter 3. Learning Dynamical Systems Encoding Non-Linearity within Space Curvature

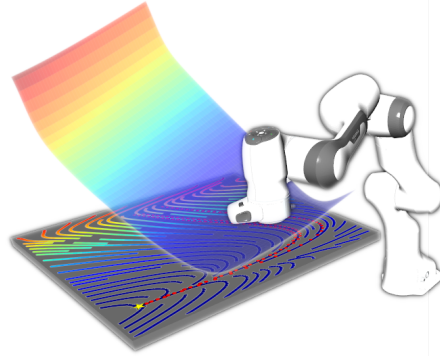


Figure 3.1: End-effector motion of a robotic arm guided by a 2D learned DS. The surface corresponds to the 3D Euclidean space embedded representation of the learnt 2D manifold. The color gradient represents the value of the potential function that drives the linear vector field taking place on the 2D manifold. The manifold's curvature induces "apparent" non-linearity in the 2D chart Euclidean space representation of the vector field taking place on the 2D manifold. During the learning process, the curvature of the manifold adapts so that the streamlines of the 2D chart Euclidean space representation of the vector field follow closely the demonstrated trajectories (red dots), preserving the stability towards a desired equilibrium point (yellow star).

proving the complexity of the Lyapunov function adopted as constraint, Figueroa and Billard (2018). Specifically, by moving towards an elliptic Lyapunov function, these approaches are capable of relaxing the constraints allowing for learning more complicated trajectories. Nevertheless, they still struggle in learning DS exhibiting high non-linearity and non-monotonic behavior in different radial directions with respect to the equilibrium point.

An alternative constraint optimization problem can be derived from Contraction Theory (CT), Lohmiller and Slotine (1998). Abstracting from the absolute position of the equilibrium point, CT follows a differential perspective. Conditions derived by CT impose local contraction of trajectories implying, as a consequence, global exponential stability towards the equilibrium point. Blocher et al. (2017) takes advantage of CT to derive a stabilizing controller that eliminates potential spurious attractor present in the DS learned without stability constraints. Sindhwani et al. (2018) uses CT to derive constraints for learning DS in a Support Vector Regression problem. Both Blocher et al. (2017) and Sindhwani et al. (2018) rely on non-generalized contraction analysis, which, in turn, results in overly conservative constraints. This is analogous to the adoption of a simplistic quadratic function in the Lyapunov approach to the stability problem. In their work, Ravichandar et al. (2017) introduced a GMR-based regression problem, incorporating stability constraints derived from generalized (CT) analysis. While this approach demonstrates superior performance by relaxing overly conservative constraints, it does so at the expense of achieving global stability, focusing solely on local stability.

Another approach to learning DS involves the existence of a *latent* space, in which either the Lyapunov function is quadratic or the DS is linear. In these approaches, the focus is on learning a diffeomorphism $\psi : \mathbb{R}^d \rightarrow \mathbb{R}^d$ between the original space and the latent one. A first example of this approach for solving the stability vs accuracy dilemma was proposed by Neumann and Steil (2015). This approach extends the applicability of SEDS by introducing a diffeomorphic mapping that transforms non-quadratic Lyapunov functions, ensuring point-wise stability of the demonstrated trajectory, into quadratic forms. After applying SEDS in this transformed space, the desired policy is obtained through the application of the inverse diffeomorphic mapping. More recent approaches concentrate on directly identifying latent spaces where the DS exhibits linear behavior. These methods make use of an approximation of the Large Deformation Diffeomorphic Metric Mapping (LDDMM) framework, Joshi and Miller (2000), to accommodate the required smoothness constraints in the mapping.

In Perrin and Schlehuber-Caissier (2016), the diffeomorphic learning DS is fundamentally geometric, focusing solely on the positions of the original and target points. To reconstruct the proper velocity profile, a rescaling of the learned DS is employed. The diffeomorphic map is learned as sequence locally weighted translations applied to the points in the original space. Additionally, more modern and network-based strategies, such as non-volume preserving transformation (NVP) Dinh et al. (2017), can be utilized to model the diffeomorphic map. Rana et al. (2020) adopts NVP transformations within an optimization framework that incorporates dynamic information, specifically velocity, into the process. This results in a one-step learning algorithm. Essentially, all these approaches involve learning a complex potential function whose gradient closely follows the target DS. While these methods demonstrate improved performance, they come with the trade-off of requiring sophisticated machinery for creating a function approximator capable of learning a mapping that exhibits the diffeomorphic property mandated by the proposed mathematical framework.

All the methods discussed above are confined to learning first-order conservative DS. Dissipative or second-order DS cannot be learned within this framework. Moreover, online local adaptation to environmental changes is not considered as part of the problem. In DS-based control, such issues are typically addressed a-posteriori and handled through a modulation matrix, as in Khansari-Zadeh and Billard (2012). These approaches are agnostic to the DS they aim to modulate, potentially leading to spurious attractors whenever the DS velocity direction aligns with the normal principal direction of the modulation matrix. A clever trick to partially address this problem involves breaking orthogonality between the modulation matrix's principal components, as proposed by Huber et al. (2019). However, these methods rely on manually designing modulation matrices for each local adaptation they aim to accommodate, resulting in increased complexity in both problem design and stability analysis. The application of modulation to second-order systems remains unclear.

These limitations cast a shadow over DS methods when compared to planning methods. Planning methods, armed with inherent adaptability and increasingly efficient sampling-based strategies, Williams et al. (2017), are gradually overcoming their reactivity challenges,

Chapter 3. Learning Dynamical Systems Encoding Non-Linearity within Space Curvature

fueled by advancements in computational hardware power, as shown in Bhardwaj et al. (2021). In response to this, geometry-based DS shaping approaches, drawing on tools from the field of differential geometry, emerge as a solution to reverse this trend. These approaches aim to achieve two crucial objectives: 1) enhance DS policies with greater expressivity and bolster their adaptability, and 2) broaden the modularity of the approach to tackle the growing complexity of real-world scenarios in which robotic systems must operate.

Drawing inspiration from Bullo and Lewis (2005), Ratliff et al. (2018) introduced the Riemannian Motion Policy (RMP), a modular mathematical framework for robotic motion generation. In contrast to prior works, this approach produces second-order DS, the specific behavior of which is inherently linked to a Riemannian metric. By carefully designing such metrics, a wide range of behaviors can be exhibited and combined.

With few exceptions, detailed in Section 3.3, this research line has not explicitly focused on Learning from Demonstration (LfD). Instead, the emphasis has been on expanding the capabilities of the mathematical framework to enhance the complexity and variety of reproducible behaviors. Building upon RMP, Cheng et al. (2020) proposed RMPflow, which effectively combines different tasks designed via RMPs, leveraging the sparsity of the structure for computational efficiency.

Summarizing the endeavors of previous works, Bylard et al. (2021) provides a principled and geometrically consistent description of the mathematical framework used for geometry-based policies. Current research in the field is shifting towards a more general formalism, extending beyond Riemannian differentiable manifolds to include Finsler structures, as discussed in Xie et al. (2021); Ratliff et al. (2021). This expansion aims to introduce velocities as a fundamental ingredient in shaping metrics that define policies' behavior.

Contribution

Our work bridges the gap between DS learning literature and the evolving field of geometry-based shaping of DS policies. From the DS learning literature, we draw inspiration from concepts related to the existence of a latent space. A notable distinction is that we do not enforce diffeomorphic constraints. On the other hand, we borrow from the Geometric DS literature the idea of a chart-based representation of DS occurring on a manifold, along with employing various tools from differential geometry to define the operators we utilize.

In this work, we introduce a novel approach to learning DS that aims to integrate LfD with modern geometric control techniques. Within our framework, the non-linearity of the DS is "encoded" within the curvature of a $d + 1$ -dimensional latent manifold, where d represents the dimension of the vector field being learned. This concept is illustrated in Figure 3.1.

Our framework naturally extends to second-order dissipative DS and easily adapts to potential online local non-linearity changes, such as those arising from the presence of obstacles. Additionally, we propose a variety of solutions, such as directional and exponential, to make

the usage second-order DS effective in LfD scenario. The proposed geometric framework, as indicated by standard metrics for evaluating learning performance, matches or outperforms the state-of-the-art while achieving notably lower computational costs during both training and query phases. These efficiency gains are obtained without compromising the performance or stability of the learned DS. Furthermore, such framework amplifies the expressivity of geometrical policies, shedding light on the relationship between DS non-linearity and manifold curvature. It also provides an explicit visualization of the Euclidean embedded representation of the latent manifold responsible for generating non-linearity.

To operationalize this work, we developed a fully differentiable PyTorch library¹, which can be used for both Learning from Demonstration (LfD) and manually shaping geometric policies. In second-order settings, such policies can exhibit either geodesic or damped harmonic behavior, expanding the variety of behaviors available. To preserve reactive control features, we developed a high-performance C++ library², that integrates our geometrical DS with fast one-step model-based or model-free Quadratic Programming control techniques. Additionally, the fully-templated nature of the library allows for the generalization of the controllers' suite to different non-Euclidean spaces, such as Lie Groups. In practical terms, for robot end-effector control, this represents a valuable feature for $\mathbb{R}^3 \times \text{SO}(3)$ control, where one controller operates in the three-dimensional Euclidean space, while the other one functions in the Special Orthogonal Group characterizing orientation space. The control strategy is fully modular and easily integrable in RMP frameworks as in Fichera and Billard (2023).

3.3 Related work

In contrast to the approach outlined in Rana et al. (2020), our method circumvents the necessity of learning a diffeomorphism between two chart representations of the underlying manifold. Instead, it focuses on learning an embedding that defines a higher-dimensional Euclidean representation of the manifold, offering two advantages: enhanced computational efficiency and greater model expressivity. From both a computational and mathematical perspective, our approach imposes fewer mathematical constraints. Specifically, we only need to ensure homeomorphic property between the manifold and its image through the embedding. In this context, an effective learning process can be achieved using any continuous function approximator, eliminating the need for a specific non-volume preserving (NVP) network to learn the diffeomorphism. Consequently, our learning process becomes more straightforward and faster in converging to the optimal solution. Furthermore, our approach naturally facilitates the construction of second-order DSs, although it can also yield first-order DSs as a sub-case. This increased mathematical structure provides greater expressivity, enabling us to replicate crossing trajectories and navigate around concave obstacles through hybrid geodesic/harmonic motion.

¹learn-embedding code available at: <https://github.com/nash169/learn-embedding>

²control-lib code available at: <https://github.com/nash169/control-lib>

Chapter 3. Learning Dynamical Systems Encoding Non-Linearity within Space Curvature

While not directly classified as part of the LfD literature, Mukadam et al. (2020) applies learning methods in RMPflow by introducing weight functions that hierarchically modify the Lyapunov functions associated with subtasks. This enhancement improves the overall performance of the combined policy. Contrastingly, our approach primarily focuses on learning the curvature of the manifold. This enables the generation of complex policies without incurring in stability problems caused by the lack of convexity of the Lyapunov function.

In the work presented in Rana et al. (2019), the emphasis lies in learning the Cholesky decomposition of the metric tensor. When compared with our approach, this method not only exhibits limitations by solely addressing first-order geometry but also mandates specifically designed and more complex function approximators to learn the lower diagonal matrix of the metric tensor Cholesky decomposition. This structure limits the flexibility to dynamically adjust the local geometry of the space to handle scenarios, such as obstacle avoidance.

Beik-Mohammadi et al. (2021) developed a method to generate geodesic motions aligned with observed trajectories by deriving a Riemannian metric from a map between the original and a latent space, learned through a Deep Autoencoder. This involves geodesic motions generated via an iterative optimization process that minimizes the curve length between two points. Conversely, our method focuses on constructing a DS formulation, ensuring stability, reactivity, and robustness against spatial and temporal disturbances typical in DS-based control. We directly access the manifold as a higher-dimensional Euclidean representation by approximating a single embedding component through a simple feedforward network, eliminating the need for complex Deep Learning structures. This not only enhances computational efficiency but also provides much greater expressivity, allowing a deeper understanding of why and how altering manifold's curvature to achieve the desired DS non-linearity. Our framework allows for a directly modifiable embedding designed to facilitate online local deformation.

3.4 Background

The presentation of the work relies heavily on concepts from differential geometry. Our notation follows do Carmo (1992). We employ the Einstein summation convention in which repeated indices are implicitly summed over.

Given a set, \mathcal{M} , and a Hausdorff and second-countable topology, \mathcal{O} , a topological space $(\mathcal{M}, \mathcal{O})$ is called a d -dimensional manifold if $\forall p \in \mathcal{M} : \exists \mathcal{U} \in \mathcal{O} : \exists x : \mathcal{U} \rightarrow x(\mathcal{U}) \subseteq \mathbb{R}^d$, with x and x^{-1} continuous maps. (\mathcal{U}, x) is a *chart* of the manifold $(\mathcal{M}, \mathcal{O})$. x is called the *chart map*; it maps $p \in \mathcal{M}$ to the point $x(p) = (x^1(p), \dots, x^d(p))$ into the \mathbb{R}^d Euclidean space. $(x^1(p), \dots, x^d(p))$ are known as the coordinate maps or local coordinates. With slight abuse of notation, we will refer to a point in \mathbb{R}^d using the bold vector notation $\mathbf{x} = x(p)$, dropping the explicit dependence on $p \in \mathcal{M}$. x^i will be i -th local coordinate of $\mathbf{x} \in \mathbb{R}^d$.

Throughout, we will denote with \mathcal{M} a *differentiable Riemannian* manifold, that is a mani-

fold endowed with a C^∞ -atlas, \mathcal{A} , and a $(0,2)$ -tensor field, g , with positive *signature*, satisfying symmetry and non-degeneracy properties. We refer to g as a Riemannian *metric*.

$T_p\mathcal{M}$ ($T_p^*\mathcal{M}$) denotes the tangent (resp. cotangent) space at $p \in \mathcal{M}$. We denote by $v_p \in T_p\mathcal{M}$ a vector in the tangent space at p . Given a set of local coordinates (x^1, \dots, x^d) , in a neighborhood \mathcal{U} of $p \in \mathcal{M}$, we denote by $\frac{\partial}{\partial x^i}$ (resp. dx^i) the i -th basis vector of $T_p\mathcal{M}$ (resp. $T_p^*\mathcal{M}$). The tangent bundle $T\mathcal{M}$ (resp. cotangent bundle $T^*\mathcal{M}$) is the disjoint union of these tangent (resp. cotangent) spaces over all $p \in \mathcal{M}$.

A vector field (resp. covector field) X on $\mathcal{U} \subset \mathcal{M}$ is a map assigning to each point $p \in \mathcal{U}$ a vector $X(p) \in T_p\mathcal{M}$ (resp. $X(p) \in T_p^*\mathcal{M}$). $\Gamma(T\mathcal{M})$ (resp. $\Gamma(T^*\mathcal{M})$) denotes the set of vector (resp. covector) fields on \mathcal{M} . Let $X, Y \in \Gamma(T\mathcal{M})$, the vector field $\nabla_Y X$ is the covariant derivative of X with respect to Y . In the context of dynamical systems subjected to external driving forces on manifolds, a force at a point $p \in \mathcal{M}$ is a covector, namely $f_d : T_p\mathcal{M} \times I \rightarrow T_p^*\mathcal{M}$.

The metric can be used to uniquely relate elements of $T\mathcal{M}$ and elements of $T^*\mathcal{M}$. For each $p \in \mathcal{M}$ we define the flat map $(\cdot)^\flat : T_p\mathcal{M} \rightarrow T_p^*\mathcal{M}$ and sharp map $(\cdot)^\sharp : T_p^*\mathcal{M} \rightarrow T_p\mathcal{M}$ as the inverse of $(\cdot)^\flat$.

$C^\infty(\mathcal{M})$ denotes the set of smooth functions $\varphi : \mathcal{M} \rightarrow \mathbb{R}$. The differential of a function $\varphi \in C^\infty(\mathcal{M})$ is the covector field $d\varphi \in \Gamma(T^*\mathcal{M})$. In local coordinates, $d\varphi = \frac{\partial\varphi}{\partial x^i} dx^i$. To express the partial derivative, we will adopt the contracted notation $\partial_i\varphi = \frac{\partial\varphi}{\partial x^i}$.

Let \mathcal{M} and \mathcal{N} be two differentiable Riemannian manifolds and $f : \mathcal{M} \rightarrow \mathcal{N}$ be a continuous map. The *pushforward* map f_* is the map $f_* : T\mathcal{M} \rightarrow T\mathcal{N}$ where $f_*(X)\varphi := X(f \circ \varphi) \quad \forall \varphi \in C^\infty(\mathcal{N}), \forall X \in \Gamma(T\mathcal{M})$. The *pullback* map f^* is the map $f^* : T^*\mathcal{N} \rightarrow T^*\mathcal{M}$, where $f^*(\omega)(X) := \omega(f_*(X)) \quad \forall X \in \Gamma(T\mathcal{M}), \forall \omega \in T^*\mathcal{N}$.

A curve γ on a given manifold \mathcal{M} is a mapping $\gamma : I \subset \mathbb{R} \rightarrow \mathcal{M}$. The curve can be expressed in local coordinate through the mapping $x_\gamma = x \circ \gamma : I \rightarrow \mathbb{R}^d$ such that $x(\gamma(t)) = x_\gamma(t) \in \mathbb{R}^d$ for each $t \in I$. We use $\dot{x}_\gamma(t)$ to express the *speed* $\frac{dx_\gamma}{dt}$. Wherever the explicit reference to the underlying curve is not needed, we use directly \dot{x}^i to indicate the i -th local coordinate of the velocity of the curve.

Given a curve $\gamma : I \rightarrow \mathcal{M}$, a vector field along γ , v_γ , is a map that assigns to each $t \in I$ an element $v_{\gamma(t)} \in T_{\gamma(t)}\mathcal{M}$. The covariant derivative of v_γ along v_γ in local coordinates is

$$(\nabla_{v_\gamma} v_\gamma)^k = \ddot{x}^k + \Gamma_{ij}^k \dot{x}^i \dot{x}^j, \quad (3.1)$$

with Γ_{ij}^k the Christoffel symbols. A Riemannian metric g induces a unique affine connection ∇ on \mathcal{M} , called the Levi-Civita connection. In this scenario the Christoffel symbols can be expressed as a function of the Riemannian metric g . In local coordinates the Christoffel symbols for the Levi-Civita connection are $\Gamma_{ij}^k = \frac{1}{2} g^{km} (\partial_i g_{mj} + \partial_j g_{mi} - \partial_m g_{ij})$ for $(i, j, k \in 1, \dots, d)$.

Chapter 3. Learning Dynamical Systems Encoding Non-Linearity within Space Curvature

A second-order linear DS on \mathcal{M} can be expressed, for $t \in I \subset \mathbb{R}$, in intrinsic formulation as

$$\nabla_{v_\gamma} v_\gamma = \mathcal{F}(\gamma, v_\gamma, t)^\# = -d\phi^\# - D(\cdot, v_\gamma)^\#, \quad (3.2)$$

where $\gamma : I \rightarrow \mathcal{M}$ is a curve on the manifold \mathcal{M} , v_γ is the vector field generated by the tangent velocities of curve γ . On the right-hand side of Equation (3.2), \mathcal{F} is the total forces covector. It can be further split into elastic and dissipative components. Given a potential function, $\phi \in C^\infty(\mathcal{M})$, $d\phi^\#$ represents the elastic gradient field, while $D(\cdot, v_\gamma) \in T^*\mathcal{M}$ is the dissipative covector field. In local coordinates we have

$$(\nabla_{v_\gamma} v_\gamma)^k = -g^{ak} \partial_a \phi - D_m^k \dot{x}^m. \quad (3.3)$$

Combining Equations (3.1) and (3.3), we have

$$\underbrace{\ddot{x}^k}_{\ddot{x}^k} + \underbrace{\Gamma_{ij}^k}_{\Xi} \underbrace{\dot{x}^i \dot{x}^j}_{\dot{x}^i \dot{x}^j} = - \underbrace{g^{ak}}_{\mathbf{G}^{-1}} \underbrace{\partial_a \phi}_{\nabla \phi} - \underbrace{D_m^k}_{\mathbf{D}} \underbrace{\dot{x}^m}_{\dot{x}^m}. \quad (3.4)$$

Note that $D_m^k = g^{ak} D_{am}$. Equation (3.4) can be expressed using vector notation as

$$\ddot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}) = -\mathbf{G}^{-1} \nabla \phi - \mathbf{D} \dot{\mathbf{x}} - \Xi \dot{\mathbf{x}}. \quad (3.5)$$

In the following sections, we will use capital bold letters to represent matrices in vector notation. When indices are used alongside a matrix, for the sake of clarity, the matrix itself will be denoted using capital letters without bold formatting.

3.5 Learning the Latent Manifold Embedding

Let \mathcal{M} and \mathcal{N} be two (non-compact) Riemannian manifolds, with respective charts (\mathcal{U}, x) and (\mathcal{V}, y) . We will indicate with g and h , the Riemannian metrics of \mathcal{M} and \mathcal{N} , respectively. In particular, for $\dim(\mathcal{M}) = d$, \mathcal{N} coincides with a $(d+1)$ -dimensional Euclidean space, \mathbb{R}^{d+1} . Therefore, $\mathcal{V} \equiv \mathbb{R}^{d+1}$ and $y \equiv id_{\mathbb{R}^{d+1}}$, where $id_{\mathbb{R}^{d+1}}$ is the identity map. Let $h = \delta_{ij}$ with respect to the chosen chart, where δ_{ij} is the Kronecker symbol, $\delta_{ij} = 1$ if $i = j$ otherwise $\delta_{ij} = 0$, and $i, j = 1, \dots, d+1$. With reference to Figure 3.2, $f : \mathcal{M} \hookrightarrow \mathbb{R}^{d+1}$ is a smooth isometric embedding into a $d+1$ Euclidean space. $(y \circ f \circ x^{-1})$ is the local coordinates expression of the embedding, i.e. the mapping between the two charts (\mathcal{U}, x) and (\mathcal{V}, y) .

$$\begin{array}{ccc} \mathcal{M} & \xrightarrow{f} & \mathcal{N} \equiv \mathbb{R}^{d+1} \\ \downarrow (\mathcal{U}, x) & & \downarrow (\mathcal{V} \equiv \mathbb{R}^{d+1}, y \equiv id_{\mathbb{R}^{d+1}}) \\ \mathbb{R}^d & \xrightarrow{y \circ f \circ x^{-1}} & \mathbb{R}^{d+1} \end{array}$$

Figure 3.2: Mapping structure across manifolds.

3.5 Learning the Latent Manifold Embedding

We propose to model the components of the local coordinates formulation of the embedding as follows

$$(y \circ f \circ x^{-1})^i = \begin{cases} x^i & \text{if } i \leq \dim(\mathcal{M}) \\ \psi(x^i; \mathbf{w}) & \text{otherwise} \end{cases}, \quad (3.6)$$

where $\psi: \mathbb{R}^d \rightarrow \mathbb{R}$ is a smooth function parametrized by the weights \mathbf{w} and $\psi \in C^r(\mathbb{R}^d)$ with $r \geq 1$. We will refer to the embedding expressed in local coordinates with the vector notation $\Psi: \mathbb{R}^d \rightarrow \mathbb{R}^{d+1}$, highlighting the implicit dependence on ψ .

Proposition 5. $f: \mathcal{M} \rightarrow \mathbb{R}^{d+1}$ is a smooth mapping with local coordinates as in Equation (3.6). $f: \mathcal{M} \hookrightarrow \mathbb{R}^{d+1}$ is an embedding.

Proof. See Appendix B.1.1. □

All the geometric operators, namely the metric and the Christoffel symbols, in Equation (3.5), are derived from the embedding defined in Equation (3.6) by leveraging on the pullback operation. To improve readability, in the following, we drop the explicit dependency of ψ on the local coordinates point \mathbf{x} and the weights \mathbf{w} . Nevertheless, all operators derived have to be considered dependent on \mathbf{x} and \mathbf{w} via ψ .

We first derive the metric as a function of the approximator ψ . The components of the pushforward map, $J_j^i = \partial_j (y \circ f \circ x^{-1})^i$, can be expressed in matrix notation as

$$\mathbf{J}(\mathbf{x}; \mathbf{w}) = \begin{bmatrix} \mathbf{I}^{\dim(\mathcal{M}) \times \dim(\mathcal{M})} \\ \nabla \psi^T \end{bmatrix}, \quad (3.7)$$

where $\nabla \psi = [\partial_1 \psi, \dots, \partial_{\dim(\mathcal{M})} \psi]^T$; \mathbf{J} is also known as the Jacobian. Isometry of the embedding implies that the metric on \mathcal{M} can be derived from the metric on \mathcal{N} via the pullback operation of the metric, $g = f^* h$. In local coordinates this can be expressed as

$$g_{ij} = \partial_i (y \circ f \circ x^{-1})^a h_{ab} \partial_j (y \circ f \circ x^{-1})^b. \quad (3.8)$$

Given $h_{ab} = \delta_{ab}$, Equation (3.8) can be written in matrix notation as

$$\mathbf{G}(\mathbf{x}; \mathbf{w}) = \mathbf{J}^T \mathbf{J} = \mathbf{I} + \nabla \psi \nabla \psi^T. \quad (3.9)$$

To derive the Christoffel symbols, we need to express the derivative of the metric. Given that the metric depends on \mathbf{x} only through the term $\nabla \psi \nabla \psi^T$, the Christoffel symbols (contracted with the velocity) can be expressed as

$$\Xi_j^q(\mathbf{x}, \dot{\mathbf{x}}; \mathbf{w}) = g^{qm} \frac{1}{2} (\partial_i (\partial_m \psi \partial_j \psi) + \partial_j (\partial_m \psi \partial_i \psi) - \partial_m (\partial_i \psi \partial_j \psi)) \dot{x}^i \quad (3.10)$$

The last two terms not yet defined in Equation (3.5) are the potential energy ϕ and the

Chapter 3. Learning Dynamical Systems Encoding Non-Linearity within Space Curvature

dissipative coefficients \mathbf{D} . We impose to the potential energy a classical quadratic structure $\phi = \frac{1}{2}\mathbf{x}^T \mathbf{K}\mathbf{x}$. Both the matrices \mathbf{K} and \mathbf{D} can be user-defined or left "open" for optimization.

Given the full state, $(\mathbf{x}, \dot{\mathbf{x}})$, of a DS, our training dataset is composed of position-velocity pairs $\{\mathbf{x}_m, \dot{\mathbf{x}}_m | m = 1, \dots, M\}$, with M the total number of sampled points. The ground truth is given by sampled acceleration $\{\ddot{\mathbf{x}}_m | m = 1, \dots, M\}$. We propose the following optimization problem

$$\min_{\mathbf{w}, \mathbf{K}, \mathbf{D}} \mathcal{F}_2(\mathbf{x}_m, \dot{\mathbf{x}}_m, \ddot{\mathbf{x}}_m | \mathbf{w}, \mathbf{K}, \mathbf{D}), \quad (3.11)$$

with

$$\mathcal{F}_2(\mathbf{x}_m, \dot{\mathbf{x}}_m, \ddot{\mathbf{x}}_m | \mathbf{w}, \mathbf{K}, \mathbf{D}) = \sum_{m=1}^M \|\ddot{\mathbf{x}}_m + \mathbf{G}^{-1}(\mathbf{x}_m; \mathbf{w}) (\mathbf{K}(\mathbf{x}_m - \mathbf{x}^*) + \mathbf{D}\dot{\mathbf{x}}_m) + \Xi(\mathbf{x}_m, \dot{\mathbf{x}}_m; \mathbf{w})\dot{\mathbf{x}}_m\|^2 + \lambda \|\mathbf{w}\|^2, \quad (3.12)$$

$$\Xi(\mathbf{x}_m, \dot{\mathbf{x}}_m; \mathbf{w})\dot{\mathbf{x}}_m\|^2 + \lambda \|\mathbf{w}\|^2, \quad (3.13)$$

where $\mathbf{K}, \mathbf{D} \in \mathcal{S}_{++}^d$, the manifold of Symmetric Positive Definite (SPD) matrices of dimension d . \mathbf{x}^* is the fixed stable equilibrium point, or attractor, of the DS. λ is a parameter weighing the regularization term. This parameter affects the manifold's curvature smoothness. Since the manifold's curvature translates into acceleration within the DS (via the Christoffel symbols), regularization plays a crucial role in containing high frequency change of curvature, avoiding high accelerations and potential overfitting.

SPD matrices optimization is achieved by parametrizing the generic SPD matrix as

$$\mathbf{M}(\boldsymbol{\alpha}, \boldsymbol{\xi}) = \mathbf{U}(\boldsymbol{\alpha})\Lambda(\boldsymbol{\xi})\mathbf{U}(\boldsymbol{\alpha})^T, \quad (3.14)$$

with

$$\Lambda(\boldsymbol{\xi}) = \begin{bmatrix} e^{\xi_1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & e^{\xi_d} \end{bmatrix} \quad (3.15)$$

and \mathbf{U} the orthogonal matrix resulting from QR decomposition of the matrix constructed so to have the first column equal to the vector $\boldsymbol{\alpha} \in \mathbb{R}^d$. $\boldsymbol{\alpha}$ and $\boldsymbol{\xi}$ are the learnable parameters.

Theorem 2. *Let $\mathbf{w} \in (-\infty, \infty)$. The dynamical system*

$$\ddot{\mathbf{x}} = -\mathbf{G}^{-1}(\mathbf{x}; \mathbf{w}) (\mathbf{K}(\mathbf{x} - \mathbf{x}^*) + \mathbf{D}\dot{\mathbf{x}}) - \Xi(\mathbf{x}, \dot{\mathbf{x}}; \mathbf{w})\dot{\mathbf{x}} \quad (3.16)$$

is globally asymptotically stable at the attractor \mathbf{x}^ , i.e. $\lim_{t \rightarrow \infty} \|\mathbf{x} - \mathbf{x}^*\| = 0$.*

Proof. See Appendix B.1.2. □

3.5.1 Gradient Systems & Incremental Learning

Our method can be applied to first order dynamics. Let $X : \mathcal{M} \rightarrow T\mathcal{M}$ be a vector field on \mathcal{M} . X is called a gradient system if

$$X = -d\phi^\# \xrightarrow{\text{local coordinates}} X^k = -g^{ik} \partial_i \phi. \quad (3.17)$$

As for the system in Equation (3.3), gradient systems are globally stable; moreover, they are globally exponentially stable. Such property follows from the fact that this type of systems are contracting on \mathcal{M} , Simpson-Porco and Bullo (2014), i.e. $g(\nabla_v X, v) \leq -\lambda g(v, v)$ for $p \in \mathcal{U}$ and $v \in T_p\mathcal{M}$. $\lambda > 0$ is the contraction rate. For strongly convex functions ϕ on \mathcal{U} , they satisfy the contraction condition $\text{Hess}(\phi) \geq \lambda g$, where $\text{Hess}(\phi)$ is the Riemannian Hessian, see Simpson-Porco and Bullo (2014); Wensing and Slotine (2020) for details.

We can minimize the Mean Square Error (MSE) loss, as in Equation (3.11), having as target the sampled velocities

$$\min_{\mathbf{w}, \mathbf{K}} \sum_{m=1}^M \mathcal{F}_1(\mathbf{x}_m, \dot{\mathbf{x}}_m | \mathbf{w}, \mathbf{K}). \quad (3.18)$$

with

$$\mathcal{F}_1(\mathbf{x}_m, \dot{\mathbf{x}}_m | \mathbf{w}, \mathbf{K}) = \sum_{m=1}^M \left\| \dot{\mathbf{x}}_m + \mathbf{G}^{-1}(\mathbf{x}_m; \mathbf{w}) \mathbf{K}(\mathbf{x}_m - \mathbf{x}^*) \right\|^2 + \lambda \|\mathbf{w}\|^2, \quad (3.19)$$

If for simpler scenarios, first-order DS might suffice, more complicated cases require the usage of second-order DS. Nevertheless, first-order DS optimization can be performed as a way of finding a good initial solution for the second-order DS. As shown in Section 3.7.4, given the simple structure of a first-order DS, solution to Equation (3.18) can be found in considerably lower time than Equation (3.11). For complicated problems, this situation suggests to perform a sort of incremental learning: first, find optimal embedding weights, \mathbf{w}^* and stiffness matrix, \mathbf{K}^* , by solving repeatedly Equation (3.18); second, solve Equation (3.11) starting from \mathbf{w}^* and \mathbf{K}^* .

3.6 Online Kernel-Based Local Space Deformation

Kernel-based space deformation is an effective method for generating localized curvature, which subsequently influences the behavior of chart-based DS. As detailed in Section 3.6.1, for obstacle avoidance scenarios, this approach is particularly relevant to our method, where an explicit representation of the latent manifold is available.

Nevertheless, kernel-based deformation can also be adopted to model any type of global manifold's curvature realized as a linear combination of point-wise sources of the deformation. The analytical formulation provides a simplified framework that would allow us to gain intuition on the effect of the metric tensor and Christoffel symbols in the chart space DS, as curvature starts to appear in the manifold. These concepts are explored in Sections 3.6.2 and 3.6.3. Starting from a flat space scenario, we analyze the effect of locally deforming the

Chapter 3. Learning Dynamical Systems Encoding Non-Linearity within Space Curvature

space via Radial Basis Function (RBF) kernels for first-order dynamics and second-order dynamics. In addition, for second-order dynamics, we show how it is possible to leverage on hybrid harmonic and geodesic motion to achieve concave obstacle avoidance, in fairly extreme scenarios, without recurring to planning strategies.

3.6.1 Obstacle Avoidance via Direct Space Deformation

One advantage of the proposed method is the direct extendibility to obstacle avoidance scenarios. We take advantage of the geometric obstacle avoidance techniques based on the local deformation or stretching of the space. In our approach, we encode the non-linearity of the DS within the curvature of the space. Without repeating the learning process, the non-linearity needed for the obstacle avoidance task can be encoded, locally, in the curvature of the space as well.

Geometry-based obstacle avoidance techniques rely on the definition of a metric that takes into account the presence of obstacles. The metric can be defined in the ambient space (and pulled back afterwards), Beik-Mohammadi et al. (2021), or directly in chart space, Cheng et al. (2020). The two approaches are equivalent. They work well if the original space on which the DS is taking place does not present pre-existent relevant curvature. In our approach, an explicit knowledge of the "shape" of the manifold is available. Given this knowledge, we show how it is possible to directly deform an already non-linear space to produce obstacle avoidance.

Let $k_{\text{chart}} : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ ($k_{\text{ambient}} : \mathbb{R}^{d+1} \times \mathbb{R}^{d+1} \rightarrow \mathbb{R}$) be a similarity measure in the chart (ambient) space. Let $\bar{\mathbf{x}} \in \mathbb{R}^d$ ($\bar{\mathbf{y}} \in \mathbb{R}^{d+1}$) be the location of an obstacle in chart (ambient) space. Given a current position $\mathbf{x} \in \mathbb{R}^d$ ($\mathbf{y} \in \mathbb{R}^{d+1}$), $k(\mathbf{x}, \bar{\mathbf{x}})$ ($k(\mathbf{y}, \bar{\mathbf{y}})$) informally expresses how close we are to the obstacle in the chart (ambient) space. Considering $\bar{\mathbf{x}}$ ($\bar{\mathbf{y}}$) fixed, we have $k_{\text{chart}}(\bar{\mathbf{x}}, \cdot) : \mathbb{R}^d \rightarrow \mathbb{R}$ ($k_{\text{ambient}}(\bar{\mathbf{y}}, \cdot) : \mathbb{R}^{d+1} \rightarrow \mathbb{R}$). We assume $k_{\text{chart}} \in C^r(\mathbb{R}^d)$ ($k_{\text{ambient}} \in C^r(\mathbb{R}^{d+1})$) with $r \geq 1$ and $k_{\text{chart}}(\mathbf{x}, \bar{\mathbf{x}}) \approx 0$ ($k_{\text{ambient}}(\mathbf{y}, \bar{\mathbf{y}}) \approx 0$) for $\|\mathbf{x} - \bar{\mathbf{x}}\| \geq \epsilon$ ($\|\mathbf{y} - \bar{\mathbf{y}}\| \geq \epsilon$) for some $\epsilon > 0$. The first condition imposes at least one time differentiability while the second one requires fast decay of the similarity measure away from the obstacle.

Recall in Equations (3.8) and (3.9), we assumed the ambient metric to be constant over all the space and equal to the identity, namely the Euclidean metric. We now use the following metric for the ambient Euclidean space

$$\mathbf{H} = \mathbf{I} + \nabla k_{\text{ambient}} \nabla k_{\text{ambient}}, \quad (3.20)$$

where $\nabla k_{\text{ambient}}$ expresses the derivative of the similarity measure k_{ambient} with respect to \mathbf{y} .

The metric in Equation (3.20) is implicitly defining a deformation of the ambient space. It can be derived via the pullback of the Euclidean metric of a $d+2$ -dimensional Euclidean space, embedding our ambient space $\Psi_{\text{ambient}} = [\mathbf{y}, k(\bar{\mathbf{y}}, \mathbf{y})]^T$. With $\mathbf{H} \in \mathbb{R}^{\dim(\mathcal{M})+1 \times \dim(\mathcal{M})+1}$, the chart space metric can be derived via the pullback of the ambient metric as in Equation (3.8),

$$\mathbf{G}(\mathbf{x}; \mathbf{w}) = \mathbf{J}^T \mathbf{H} \mathbf{J}.$$

The pullback operation is equivalent to adding the "obstacle" metric to the chart space metric encoding the non-linearity of the DS, $\mathbf{G}_{\text{total}} = \mathbf{G}(\mathbf{x}; \mathbf{w}) + \mathbf{G}_{\text{obs}}$, where $\mathbf{G}_{\text{obs}} = \mathbf{I} + \nabla k_{\text{chart}} \nabla k_{\text{chart}}$. Let $\mathbf{H} = \begin{bmatrix} \mathbf{I} + \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{1} + \mathbf{C} \end{bmatrix}$, with $\mathbf{A} \equiv \nabla k_{\text{chart}} \nabla k_{\text{chart}} \in \mathbb{R}^{d \times d}$, $\mathbf{B} \in \mathbb{R}^{1 \times d}$ and $\mathbf{C} \in \mathbb{R}^{1 \times 1}$. The pullback operation in Equation (3.20) becomes $\mathbf{G} = \mathbf{I} + \mathbf{A} + \mathbf{B} \nabla \psi^T + \nabla \psi \mathbf{B}^T + \nabla \psi (\mathbf{I} + \mathbf{C}) \nabla \psi^T$. For $\gamma(t) \in \mathcal{M} \quad \forall t \geq 0$ we have $\mathbf{B}, \mathbf{C} = 0$; therefore $\mathbf{G} = \mathbf{I} + \mathbf{A} + \nabla \psi \nabla \psi^T = \mathbf{I} + \nabla k_{\text{chart}} \nabla k_{\text{chart}} + \nabla \psi \nabla \psi^T = \mathbf{G}_{\text{obs}} + \mathbf{G}(\mathbf{x}; \mathbf{w})$.

Adding metrics linearly is akin to treating the deformations of space, due to the non-linearity of the DS and the presence of an obstacle, separately. $\mathbf{G}(\mathbf{x}; \mathbf{w})$ is derived as in Equation (3.9) from the embedding Ψ in Equation (3.6); \mathbf{G}_{obs} can be derived from an embedding of the type $\Psi_{\text{obstacle}} = [\mathbf{x}, k(\bar{\mathbf{x}}, \mathbf{x})]^T$. In other terms, the deformation of the space due to the presence of the obstacle is agnostic of the previous curvature in the manifold. Such a scenario still yields good results where the space is fairly flat.

The explicit formulation of the embedding allows us to directly deform the space while actively taking into consideration the curvature of the space. Let $\{\bar{\mathbf{x}}_i, i = 1, \dots, N\}$ be the location of N obstacles in chart space; $\bar{\mathbf{w}}$ are the weights after learning the DS. We model obstacles as a kernel-based local deformation of the spaces given by

$$\bar{\psi}(\mathbf{x}) = \sum_{i=1}^N \eta_i k(\bar{\mathbf{x}}_i, \mathbf{x}), \quad (3.21)$$

where η_i is a user-defined weight assigned to the local deformation at $\bar{\mathbf{x}}_i$. At query-time the embedding in Equation (3.6) can be expanded as follows

$$\Psi = \begin{bmatrix} \mathbf{x} \\ \psi(\mathbf{x}; \bar{\mathbf{w}}) + \bar{\psi}(\mathbf{x}). \end{bmatrix} \quad (3.22)$$

The embedding in Equation (3.22) leads to the following pullback metric

$$\mathbf{G}(\mathbf{x}; \mathbf{w}) = \mathbf{I} + \nabla \psi \nabla \psi^T + \overbrace{2 \nabla \psi \nabla \bar{\psi}(\mathbf{x})^T}^{\text{coupling term}} + \nabla \bar{\psi}(\mathbf{x}) \nabla \bar{\psi}(\mathbf{x})^T. \quad (3.23)$$

The coupling term in Equation (3.23) encodes the pre-existent curvature of the space. Given the imposed condition on the regularity of k , both Proposition 5 and Theorem 2 still hold. The overall DS generated with such embedding retains global asymptotical stability, independently from the number of obstacles present.

Second order dynamical systems cannot perform proximal obstacle avoidance. Indeed, the geometrical term given by the Christoffel symbols generates forces that lead the system to "climb up" regions of local high-curvature, penetrating the obstacle. This forces conflict with the potential one generating an overall motion that stagnates right after the obstacle,

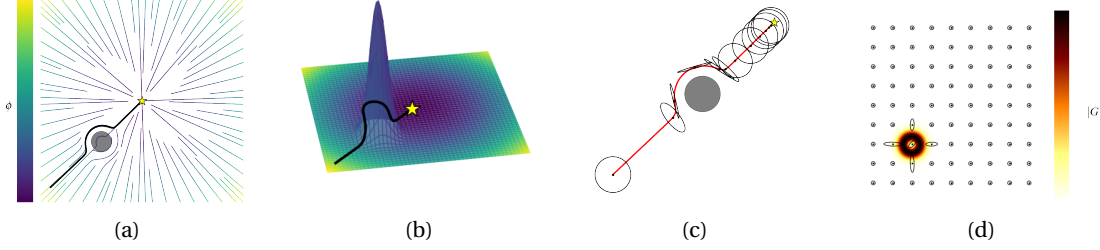


Figure 3.3: First-order DS in flat space with localized deformation in the obstacle area: (a) Vector field with one sampled streamline avoiding the obstacle; (b) 3D embedded representation of the manifold; (c) one sampled trajectory with eigenvalue decomposition ellipses of the inverse of the metric for selected location; (d) metric determinant function with eigenvalue decomposition ellipses of the metric.

see Section 3.6.3. This issue can be solved by introducing velocity-dependent local space deformations

$$\tilde{\psi}(\mathbf{x}, \dot{\mathbf{x}}) = \sum_{i=1}^N \eta_i(\mathbf{x} - \bar{\mathbf{x}}_i, \dot{\mathbf{x}}) k(\bar{\mathbf{x}}_i, \mathbf{x}), \quad (3.24)$$

where $\eta_i(\mathbf{x} - \bar{\mathbf{x}}_i, \dot{\mathbf{x}})^3$ is framed as generalized sigmoid function acting on the cosine kernel between $\mathbf{x} - \bar{\mathbf{x}}_i$ and $\dot{\mathbf{x}}$

$$\eta_i(\mathbf{x} - \bar{\mathbf{x}}_i, \dot{\mathbf{x}}) = \frac{1}{1 + e^{-\tau(k_{\cos}(\mathbf{x} - \bar{\mathbf{x}}_i, \dot{\mathbf{x}}) - \cos \theta_{ref})}}, \quad (3.25)$$

with $k_{\cos}(\mathbf{x} - \bar{\mathbf{x}}_i, \dot{\mathbf{x}}) = \frac{(\mathbf{x} - \bar{\mathbf{x}}_i)^T \dot{\mathbf{x}}}{\|\mathbf{x} - \bar{\mathbf{x}}_i\| \|\dot{\mathbf{x}}\|}$. τ regulates the growth rate and θ_{ref} defines the starting growth point. Safe option can be $\theta_{ref} = \frac{\pi}{2}$. In this scenario, the underlying manifold dynamically deforms (locally) whenever the motion is monotonically decreasing towards the obstacle. If the system is moving away from the obstacle, the manifold curvature goes back to the flat or nominal state. This allows to effectively turn off the local geometrical terms once crossed the obstacle, allowing the system to reach the desired equilibrium point.

3.6.2 Local Space Deformation In First Order DS

Let the $d + 1$ embedding component $(y \circ f \circ x^{-1})^{d+1} = \psi$ be defined as a weighted sum of exponentially decaying kernels

$$\psi(\mathbf{x}) = \sum_{i=1}^N \alpha_i k(\mathbf{x}, \bar{\mathbf{x}}_i) = \sum_{i=1}^N \alpha_i \exp\left(-\frac{\|\mathbf{x} - \bar{\mathbf{x}}_i\|^2}{2\sigma^2}\right), \quad (3.26)$$

where $\bar{\mathbf{x}}_i$ is the i -th kernel center and N is the number of kernel used. σ and α_i are user-defined parameters; the former controls till which distant the local deformation affects the space geometry, the latter defines the magnitude of the deformation. Consider $N = 1$ and $\alpha_1 = 1$.

³Note that this function, though dependent on \mathbf{x} , must be treated as constant in the derivation of the geometrical terms.

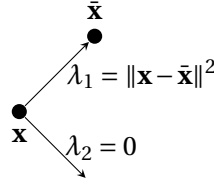


Figure 3.4: Kernel-based metric tensor eigenvalues and eigenvectors.

Via the pull-back of the embedding metric we recover the metric onto the manifold. In case of Euclidean (identity) metric for the ambient space we have

$$\mathbf{G}(\mathbf{x}) = \mathbf{I} + \frac{1}{\sigma^4} (\mathbf{x} - \bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}})^T k(\mathbf{x}, \bar{\mathbf{x}})^2. \quad (3.27)$$

See Appendix B.2.1 for derivation details. From Equation (3.27) we note that the metric tensor is composed by two terms: an identity term, independent from the location of the deformation source, and a term active only in the neighborhood of the deformation where $k(\mathbf{x}, \bar{\mathbf{x}})^2 \approx 1$. This second term is given by the outer product of the distance vector between the current location and the source of the deformation. Outer product matrices are rank deficient with the eigenvector related to the only non-zero eigenvalues, $\lambda_1 = \|\mathbf{x} - \bar{\mathbf{x}}\|^2$, directed as $\mathbf{x} - \bar{\mathbf{x}}$.

Consider a 2D space locally deformed in $\bar{\mathbf{x}}$. Figure 3.4 shows the eigenvalues and the eigenvectors of the second term in the sum of Equation (3.27). Recall that the metric tensor is used to measure lengths. In the direction of the deformation source, the space elongates of an amount proportional to $\|\mathbf{x} - \bar{\mathbf{x}}\|^2 k(\mathbf{x}, \bar{\mathbf{x}})^2$. In the direction perpendicular to the deformation source, as expected, the space does not elongate; indeed, $\lambda_2 = 0$. The space is only stretched in the direction of the deformation source. This stretch reaches its maximum in the neighborhood of the source to then decrease gradually towards zero at $\bar{\mathbf{x}}$ where the space goes back to be flat, Figure 3.3d. This explains clearly how obstacle avoidance is achieved for a gradient system as in Equation (3.17). The projection of the gradient system's velocity onto the inverse of the metric tensor decreases the velocity's component in the direction of the obstacle, located at the source of deformation, of an amount inversely proportional to the entity of space stretching. Figure 3.3c shows the ellipsoids generated by the inverse of the metric tensor. The velocity component perpendicular to the source of the deformation remains unchanged. This turns into an overall behavior of the gradient system that increases its velocity in the direction tangential to the obstacle. Note that, if the velocity of the gradient system, \mathbf{v} , points exactly towards the source of deformation, $\mathbf{v} \parallel \mathbf{x} - \bar{\mathbf{x}}$, the streamline will not be deflected at all. In such a case $\mathbf{v}^T (\mathbf{x} - \bar{\mathbf{x}})^\perp = 0$.

3.6.3 Local Space Deformation In Second Order DS

Second-order systems' behavior is affected by the Christoffel symbols. This term depends on the derivative of the metric tensor. As done for the differential of ψ , we can calculate the

Chapter 3. Learning Dynamical Systems Encoding Non-Linearity within Space Curvature

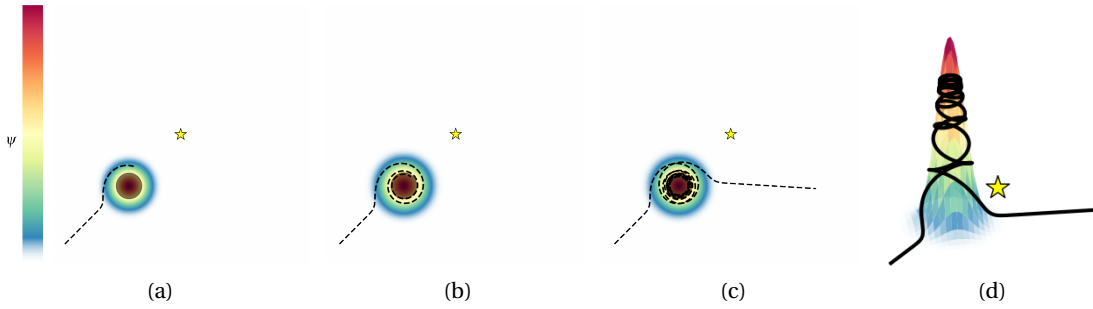


Figure 3.5: (a)-(c) Geodesic motion at time instants: 1s, 5s and 10s; (d) 3D embedded representation of case (c). In background the color gradient represents the $d+1$ embedding coordinate.

differential of the metric tensor

$$d\mathbf{G}(\mathbf{x})[\mathbf{v}] = \left((\mathbf{v}\tilde{\mathbf{x}}^T + \tilde{\mathbf{x}}\mathbf{v}^T)k(\tilde{\mathbf{x}}) - \frac{1}{\sigma^2}\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T(\tilde{\mathbf{x}}^T\mathbf{v}) \right) k(\tilde{\mathbf{x}}), \quad (3.28)$$

where $\tilde{\mathbf{x}} = \mathbf{x} - \bar{\mathbf{x}}$. Consider a geodesic motion. The Christoffel symbol generates a deceleration perpendicular to the source of deformation.

This, when approaching the source of deformation, deflects the streamlines avoiding the high curvature region. Nevertheless, if the streamline transits too close to the source of deformation, the geodesic gets captured by the high curvature region. Figures 3.5a to 3.5c show different frame of such geodesic motion. This is clear by analyzing the Christoffel symbols' principal components shown in Figure 3.6b. This components are perpendicular to the inverse metric ones, Figure 3.6a, and they generate an inward acceleration towards the obstacle that "captures" the geodesic motion leading the streamline to climb up and down the source of deformation as illustrated in Figure 3.6a.

When adopting second-order DS, the harmonic part conflicts with the Christoffel symbol, generating a stagnation of the DS right after the obstacle, Figure 3.7a. To alleviate this issue, as seen in Section 3.6.1, it is possible to define a velocity-dependent local deformation, Equa-

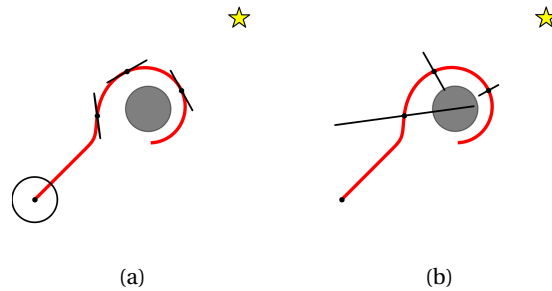


Figure 3.6: Geodesic motion after 1s with ellipses representing eigenvalues and eigenvectors of (a) the inverse of the metric and (b) the Christoffel symbols for selected locations.

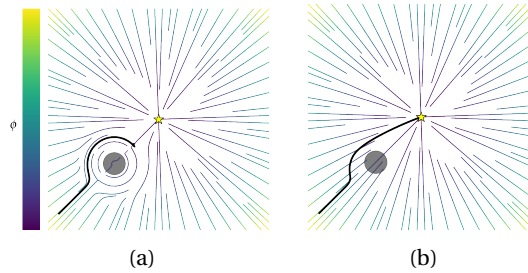


Figure 3.7: Second-order DS convex obstacle avoidance: (a) velocity independent local deformation; (b) velocity dependent deformation.

tion (3.24). This makes the curvature of the space "dynamic", so that it disappears when the obstacle is surpassed. In this scenario, the effect of the Christoffel symbols, due to the local deformation, is canceled; the second-order DS behaves like a standard Euclidean harmonic oscillator, successfully reaching the attractor, Figure 3.7b. As a byproduct of this strategy, we obtained a more effective obstacle avoidance behavior. The DS shows an asymmetrical behavior before and after the obstacle, following the most efficient trajectory to reach the attractor once surpassed the obstacle.

Concave obstacle avoidance represents a more challenging scenario where both first and second order DSs will stagnates in the middle of obstacle due to conflicting forces, Figure 3.8a. Nevertheless, as seen previously, second-order system geodesics exhibit the ability of navigating through the space deformation.

Similarly to what seen before, Figure 3.8b shows the geodesic motion in face of concave obstacle. When approaching the obstacle, the geodesic motion exhibits the ability of successfully avoid the deformed area. To perform concave obstacle avoidance, we propose an hybrid DS capable of leveraging on either geodesic or harmonic motion depending on the need

$$\ddot{\mathbf{x}} = -\Xi\dot{\mathbf{x}} - \sigma(\tilde{\psi}(\mathbf{x}, \dot{\mathbf{x}}))\mathbf{G}^{-1}(\mathbf{K}\mathbf{x} + \mathbf{D}\dot{\mathbf{x}}), \quad (3.29)$$

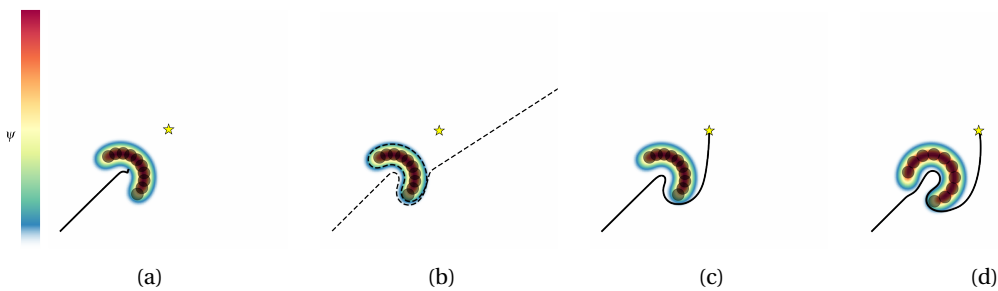


Figure 3.8: Concave obstacle avoidance: (a) harmonic motion; (b) geodesic motion; (c)-(d) hybrid motion for semicircle and horseshoe obstacles.

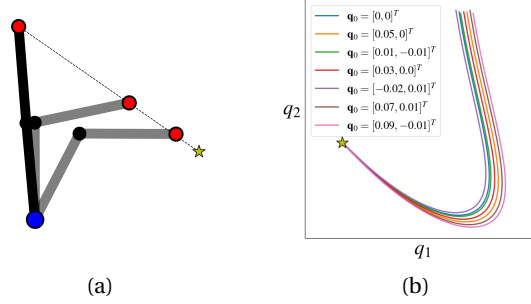


Figure 3.9: (a) 2-joints planar robotic arm performing a straight point-to-point motion of the end-effector; (b) corresponding trajectories in configuration space starting from different initial states.

where σ is a generalized sigmoid function, as in Equation (3.25), to smoothly transition between harmonic and geodesic motion. Figures 3.8c and 3.8d, respectively for semicircle and horseshoe obstacle shape, show the behavior of the DS in Equation (3.29). The DS in Equation (3.29) exhibits geodesic behavior near to obstacle, when approaching it, $\sigma(\bar{\psi}(\mathbf{x}, \dot{\mathbf{x}})) \approx 0$. When leaving the obstacle, thanks to the velocity dependency introduced before, the DS exhibits harmonic behavior, given that $\sigma(\bar{\psi}(\mathbf{x}, \dot{\mathbf{x}})) \approx 1$, allowing to reach the attractor without being "captured" by the local deformation of the space.

3.7 Synthetic Example

To gain intuition on how the proposed method operates, in this Section, we start by analyzing a synthetic dataset achieved by gathering configuration space non-linear motions of a 2-joints planar robotic arm.

Consider the 2-joints planar robotic arm in Figure 3.9a whose state is represented by the vector $\mathbf{q} = [q_1, q_2]^T$. Consider the generic equation of motion for a robotic arm, $\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} = -\mathbf{g}(\dot{\mathbf{q}}) + \boldsymbol{\tau}$, with $\mathbf{M}(\mathbf{q})$, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$, $\mathbf{g}(\dot{\mathbf{q}})$ and $\boldsymbol{\tau}$ being the inertia matrix, the Coriolis matrix, the gravity forces and input torques, respectively. We notice that classical mechanical systems are Riemannian geometries with the inertia matrix, $\mathbf{M}(\mathbf{q})$, playing the role of the metric tensor, $\mathbf{G}(\mathbf{q})$, and the fictitious (or Coriolis) forces, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$, representing the geometric forces derived by the product of the Christoffel symbols and the velocities, $\Xi(\mathbf{x}, \dot{\mathbf{x}})$. We elicit the non-linear dynamics of the robot by generating straight motions in the task (end-effector) space using operation space control, $\boldsymbol{\tau} = -\mathbf{J}(\mathbf{q})^T (\mathbf{K}(\mathbf{x} - \mathbf{x}^*) + \mathbf{D}\dot{\mathbf{x}})$, where $\mathbf{J}(\mathbf{q})$ is the Jacobian matrix relating joint space velocities to task space velocities, \mathbf{x}^* the equilibrium point in task space and \mathbf{K} and \mathbf{D} tunable gain matrices. Starting from different initial configurations, \mathbf{q}_0 , we generate in total 7 trajectories, Figure 3.9b, of which 4 are used for training and 3 for testing.

In order to learn the non-linear trajectories shown in Figure 3.9b, we approximate ψ , in Equation (3.6), with a feed-forward network composed by 2 hidden layers of 32 neurons each

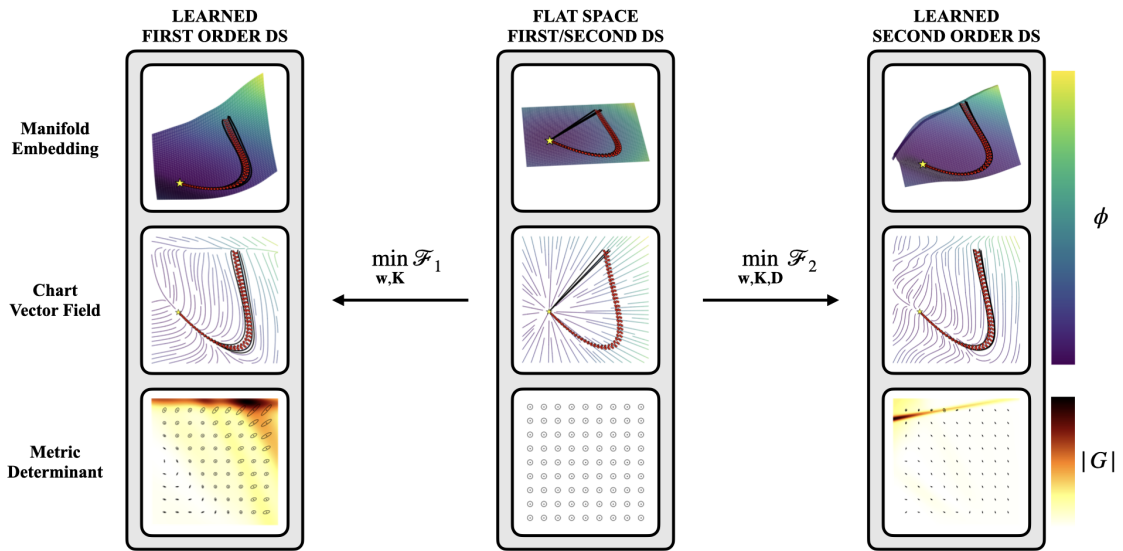


Figure 3.10: From top to bottom 3D embedded representation of the latent manifold, chart space representation of the DS field induced by the manifold curvature and metric tensor's determinant and ellipsoids for selected locations: (center) before learning; (left) learned first-order DS; (right) learned second-order DS.

with hyperbolic tangent activation function to guarantee at least C^1 regularity; see Appendix B.3 for details. The network's weights are randomly initialized very close to zero. This yields an almost flat manifold in the embedding space representation, Figure 3.10 central column in the top, with the metric tensor approaching the identity Euclidean metric. At the bottom of the central column, this is shown by the determinant of \mathbf{G} , the color gradient in background, and the ellipses representing the principal direction of the metric tensor. The determinant of the metric gives an absolute value of the local deformation of the space. In this case, the determinant of \mathbf{G} is constant and almost equal to 1 everywhere. The ellipses generated by the eigen-decomposition of the metric tensor are shown for a selected location. They provide an idea of the direction of the deformation.

For the Euclidean metric generated by the almost flat space condition, such ellipses approach the shape of a circle. For the first order DS learning, we start from a spherical stiffness matrix. For the second order DS, we additionally set the damping matrix to yield an initial critically damped behavior in flat space, $\mathbf{D} = 2(\mathbf{KM})^{1/2}$. These conditions, for both DSs, results in a linear vector field, in the chart space representation, where the sampled streamlines are straight lines towards the attractor, in the middle of the central column in Figure 3.10. Note that, for the second-order DS, the vector field is obtained by integrating one step forward Equation (3.4), considering an initial velocity of zero. During the training process, the stiffness and the damping matrices can be spherical, diagonal or Symmetric Positive Definite (SPD), as well as fixed and therefore not considered as an optimization variable. The stiffness and dissipation matrices, together with the curvature of the manifold, contribute to the non-linearity of the learned DS.

Chapter 3. Learning Dynamical Systems Encoding Non-Linearity within Space Curvature

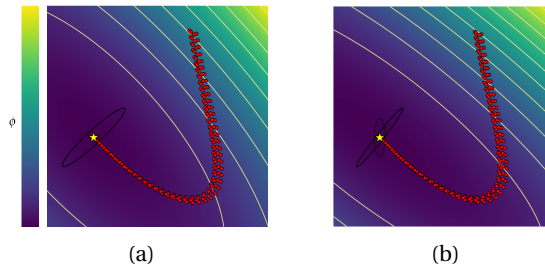


Figure 3.11: Learned potential function with stiffness and dissipation matrices principal direction ellipsoids for (a) first-order and (b) second-order DS.

Figures 3.11a and 3.11b show the isoline of the potential function, after the learning, for the first and second order DS. In this example, we opted for an SPD matrix for both the stiffness and the damping matrices. Observe that in both cases, first and second order DS, the learnt stiffness matrix "aligns" itself orthogonally to the direction of demonstrated trajectories in the neighborhoods of the attractor. Therefore, part of the contribution to the non-linearity of the streamlines is outsourced to the potential function gradient. The remaining non-linearity needed for learning the demonstrated DS is taken over by the curvature of the space.

The top row of Figure 3.10 shows the embedded representation of the learnt manifold for the first (left column) and second (right column) order DS. In the bottom line, the behavior of the determinant of the metric for both DSs. In the case of the first-order DS, the curvature of the space gives rise to an *energetic barrier* at the onset of the trajectory, guiding the flow downward. In the lower-right region of the space, it steers the streamlines towards the attractor. This phenomenon is further elucidated by examining the principal directions indicated by the ellipses of the metric tensor. The ellipses experience a compression perpendicular to the direction of maximal deformation. As a consequence, this leads to a projection of DS velocities tangentially to the deformation of the space, resulting in the desired non-linearity. In the case of the second-order DS shown at the bottom, we encounter a similar scenario, but this time with a more pronounced energetic barrier in the lower region of the space. This barrier, via the Christoffel symbols, induces directional deceleration, effectively guiding the streamlines towards the attractor.

The resulting chart space representation of the vector field along with 3 sampled testing trajectories is shown in the central row. The underlying deformation of the space induces an apparent non-linearity in the chart space representation of both DSs. The streamlines curve, adapting to the shape of the demonstrated trajectory for the first-order DS. Differently, in the second-order DS, the sampled streamlines do not evolve accordingly to the background vector field. Indeed, for the second-order DS, the background vector field assumes in each point zero initial velocity.

Consider the first order DS. Figure 3.12a shows how the principal directions of the inverse metric tensor vary along the one sampled trajectory due to deformation of the space. In the

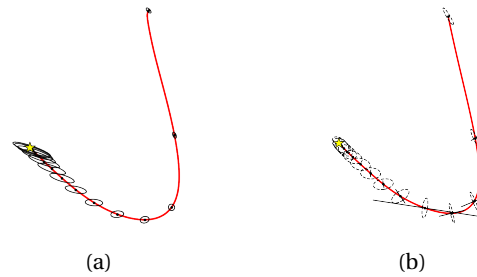


Figure 3.12: Evolution of (a) inverse metric and (b) Christoffel symbols principal directions along one sampled streamline.

first part of the trajectory the ellipses are considerably elongated in q_1 . By projecting the DS current velocity onto the inverse metric principal axis, see Equation (3.17), the velocity of the DS increases in the direction of q_1 , generating the non-linear behavior depicted. Similar consideration, Figure 3.12b, can be done for the second-order DS by analyzing the Christoffel symbols (solid line) and metric tensor inverse (dashed line) ellipses.

Notice that the DS is linear on the manifold and follows a mass-spring-damper system as in Equation (3.5). The curvature of the manifold makes the DS appear non-linear in the chart (Euclidean) space representation of the manifold.

3.7.1 Locally Active Space Deformation

Despite global stability guarantees, when using global function approximator such as neural networks, the behavior far away from demonstrations is not predictable. Figure 3.13a shows the contour of the $d+1$ embedding component, approximated by a neural network, that directly influences the curvature of the manifold. Close to the demonstrated trajectories, the function approximator ensures that the manifold's curvature would yield the desired DS

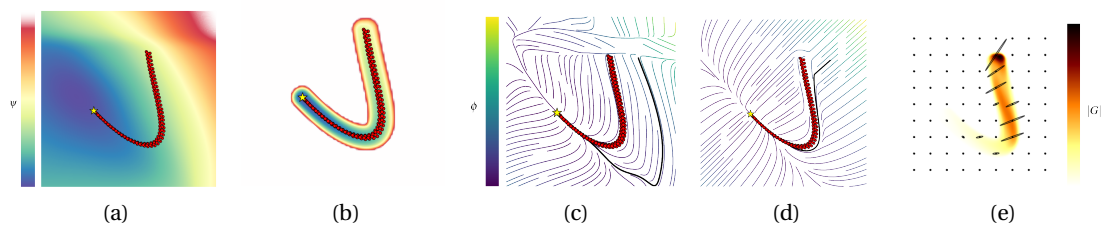


Figure 3.13: (a) Gradient of the $d+1$ embedding component; (b) gradient of the $d+1$ embedding component with bump function; (c) one sampled trajectory with initial position far away from the demonstrated trajectories; (d) one sampled trajectory with initial position far away from the demonstrated trajectories with bump function; (e) metric determinant with eigenvalue decomposition ellipses of the metric from selected location.

Chapter 3. Learning Dynamical Systems Encoding Non-Linearity within Space Curvature

non-linearity. Away from the demonstrated trajectories, the local curvature of the manifold produces a DS behavior that can be sub-optimal or even undesired, see Figure 3.13c.

To alleviate this problem, we propose to flatten the space far away from the demonstrations. This operation is performed at query time and it does not influence the training process. In this case, by flattening the space away from the demonstrations, the DS will exhibit global linear behavior except for the regions where training points are available. The nominal behavior far away from the demonstration is not limited to be linear. By choosing different types of nominal curvature, the DS will exhibit different behaviors.

To enforce flat space behavior in those areas where training data is not available, we deploy a distance dependent bump function. Let \mathcal{X} denoting the training set. We define

$$\psi_{\text{bumped}}(\mathbf{x}; \mathbf{w}) = \alpha(\mathbf{x})\psi(\mathbf{x}; \mathbf{w}), \quad (3.30)$$

where $\alpha(\mathbf{x})$ is a bump function defined as

$$\alpha(\mathbf{x}) = \begin{cases} \frac{1}{e} \exp\left(-\frac{r^2}{r^2 - \text{dist}(\mathbf{x}, \mathcal{X})^2}\right), & \text{dist}(\mathbf{x}, \mathcal{X}) \leq r \\ 0, & \text{otherwise.} \end{cases} \quad (3.31)$$

$\text{dist}(\mathbf{x}, \mathcal{X})$ is the distance between \mathbf{x} and its nearest neighborhood $\mathbf{x}_i \in \mathcal{X}$ ⁴. r is a user-defined parameter that regulates how far from the demonstrated trajectories the manifold should start to have nominal zero curvature.

Figure 3.13b shows the third embedding component behavior when pre-multiplied by the bumped function. The manifold region in the neighborhood of the demonstration preserves its original curvature given by the learning procedure. Away from the demonstration the manifold becomes increasingly flat. This type of manifold structure yields a linear behavior away from the demonstrations that smoothly transitions towards nonlinear behavior when approaching the area of the demonstrated trajectories, see Figure 3.13d. By analyzing the determinant of the metric tensor, Figure 3.13e, we can clearly notice how this type of embedding structure affects the curvature only in localized portion of the space. The ellipses of the metric tensor away from the demonstrated trajectories converge to circles; constant and equal eigenvalues of value 1. Close to the demonstrations the ellipses deform, yielding the correct geometric accelerations to follow the demonstrated trajectories.

⁴In order to increase regularity of the $\psi_{\text{bumped}}(\mathbf{x}; \mathbf{w})$, $\text{dist}(\mathbf{x}, \mathcal{X})$ can be computed as the average distance between \mathbf{x} and its K nearest neighborhoods. This is still computationally cheap by adopting modern efficient implementation of KNN. Distribution based bump function may represent an alternative way to KNN. In this case the distribution can be learned offline (for instance via GMM) yielding almost zero cost at query time. In addition joint position-velocity distribution could be learned to enforce flat-space behavior away from the demonstrated velocities.

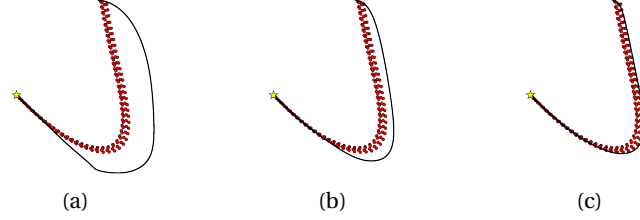


Figure 3.14: Second-order DS (a) without, (b)-(c) with ($\lambda = 10$ and $\lambda = 20$) directional dissipation.

3.7.2 Directional & Exponential Dissipation

Second-order DSs offer a richer and more versatile way to articulate high-level policies compared to first-order DS. However, their application in LfD is uncommon. Primarily, user demonstrations tend to emphasize position over velocity. For instance, during a kinaesthetic demonstration with a robot—manually guiding the robot’s end-effector along a desired path—focus is primarily on the sequence of positions rather than the end-effector’s velocity. Consequently, this oversight can result in unintended behaviors near the demonstrated positions, especially when the state velocity differs from the demonstrated one. In this section, we present two potential strategies aimed at mitigating this issue when deploying second-order DSs.

Figure 3.14a shows the streamline sampled from the initial position of one of the testing trajectories. In this case the initial velocity is not equal to zero, differently from what we had during the training. As it is possible to see in this scenario, the sampled streamline does not follow at all the demonstrations taking an alternative path to reach the attractor. When controlling, for instance, on real robot’s end-effector, we cannot ensure the current velocity of the end-effector will always lie to the demonstrated ones for each specific position, especially when the controlled system has to answer to compliance requisite in the interaction with humans. In order to make second-order DS reliable in this type of scenario and alleviate undesired behaviors whenever we have initial velocities far away from the demonstrated ones, we propose to add, without loss of stability, an additional directional dissipation

$$\begin{aligned} \ddot{\mathbf{x}} = & -\mathbf{G}^{-1}(\mathbf{x}; \mathbf{w}) (\mathbf{K}(\mathbf{x} - \mathbf{x}^*) + \mathbf{D}\dot{\mathbf{x}}) - \Xi(\mathbf{x}, \dot{\mathbf{x}}; \mathbf{w})\dot{\mathbf{x}} \\ & - \lambda_{\text{dir}} \left(\frac{\dot{\mathbf{x}}}{\|\dot{\mathbf{x}}\|} - \frac{\dot{\mathbf{x}}^*}{\|\dot{\mathbf{x}}^*\|} \right). \end{aligned} \quad (3.32)$$

$\dot{\mathbf{x}}^*$ is a first-order reference DS, providing the desired velocity field nominal behavior. Whenever first-order DS following the demonstrated trajectories is available we can take advantage of it to steer our second-order DS within the demonstrated velocities. Figures 3.14b and 3.14c show the resulting streamline for increasing values of λ_{dir} .

Another issue encountered in adopting second-order DS is the undesired under-damped

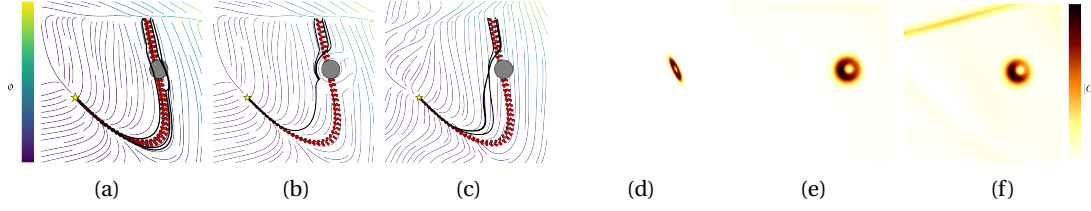


Figure 3.15: Vector field and metric determinant for (a)-(d) first-order DS with classic method; (b)-(e) first-order DS with direct deformation method; (c)-(f) second-order DS with direct deformation method.

behavior. If for classical harmonic damped oscillator we can easily set the damping term so to have a critically-damped behavior, this is not possible for oscillators on manifolds whereas the Christoffel symbols acts as a non-linear damping term not under our control. This leads to trajectory overshoot in the attractor area. In order to alleviate this problem, an additional dissipative force can be considered

$$\begin{aligned} \ddot{\mathbf{x}} = & -\mathbf{G}^{-1}(\mathbf{x}; \mathbf{w}) (\mathbf{K}(\mathbf{x} - \mathbf{x}^*) + \mathbf{D}\dot{\mathbf{x}}) - \Xi(\mathbf{x}, \dot{\mathbf{x}}; \mathbf{w})\dot{\mathbf{x}} \\ & - \lambda_{\text{exp}} \exp(-\tau \|\mathbf{x} - \mathbf{x}^*\|^2). \end{aligned} \quad (3.33)$$

This new dissipative force acts only locally and it grows exponentially approaching the attractor. This term effectively arrest the DS streamline at the attractor removing the problem of overshooting.

3.7.3 Obstacle Avoidance Online Direct Deformation

We now show how the learned DS can adapt to the presence of obstacles. As analyzed in Section 3.6.1 we have two ways of performing obstacle avoidance leveraging on the geometrical structure of the space.

The first method, as in Beik-Mohammadi et al. (2021), consists in designing a specific metric that acts in the ambient space. All the geometric terms are then derived as shown in Section 3.5 with $\mathbf{H} = \hat{\mathbf{H}} \neq \mathbf{I}$. We will refer to this approach as *classical method*.

In our framework, we have the additional and more intuitive option of directly deforming the space by altering locally the embedding map post-training, $\hat{\psi}(\mathbf{x}; \bar{\mathbf{w}}) = \psi(\mathbf{x}; \bar{\mathbf{w}}) + \bar{\psi}(\mathbf{x})$. We will refer to our approach as *direct deformation method*.

For the classical method we opt, as barrier function, for the Gaussian kernels given by $k(\bar{\mathbf{y}}, \mathbf{y}) = \exp\left(-\frac{\|\bar{\mathbf{y}} - \mathbf{y}\|^2}{2\sigma^2}\right)$, with kernel width σ . $\bar{\mathbf{y}} = \Psi(\bar{\mathbf{x}})$ is the obstacle position in the ambient space. The ambient metric is then constructed as in Equation (3.20).

For the direct deformation of the space the same Gaussian kernel acting in the chart space

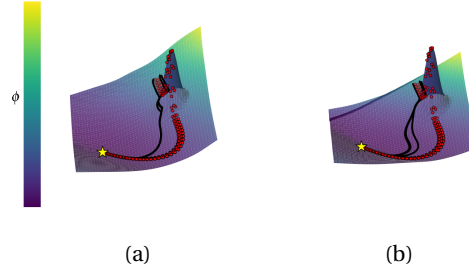


Figure 3.16: Embedding space visualization of the local deformation due to the obstacle presence for (a) first-order DS and (b) second-order DS.

$k(\bar{\mathbf{x}}, \mathbf{x})$ can be used, with no need to query the ambient space location of the obstacle. The embedding is then altered as in Equation (3.22).

σ is a user-defined parameter regulating the speed of the decay of the local deformation. One straightforward way of setting this parameter is imposing the desired decay at the border of the obstacle. For instance $\sigma = \sqrt{-\frac{1}{2} \frac{r^2}{\log \epsilon}}$, where r is the local radius of the obstacle and ϵ is value of the kernel at r ; typically $\epsilon \leq 1e-3$. Barrier functions of the type $k(\bar{\mathbf{y}}, \mathbf{y}) = \exp\left(\frac{a}{b(\|\bar{\mathbf{y}} - \mathbf{y}\| - r)^b}\right)$ can be used as well. a and b are user-defined parameters that regulate the entity of the local deformation of the space and r is the radius of the obstacle as before.

The training process is carried out using $\mathbf{H} = \mathbf{I}$ and $\hat{\psi}(\mathbf{x}; \bar{\mathbf{w}}) = \psi(\mathbf{x}; \bar{\mathbf{w}})$. The ambient metric, for the classical method, and the embedding map, for the direct deformation method, will be locally modified at test time to take into account the presence of obstacles.

Figure 3.15a shows, for the first order DS, the resulting vector field and test sampled trajectories, adopting the classical method for performing obstacle avoidance. In this scenario, it is possible to notice how the streamline do not avoid the obstacle properly. Moreover, by looking at Figure 3.15d, we can notice how the space has been deformed in a asymmetric way. Despite the isotropic kernel used, the local deformation of the space spans a tilted elliptic area. This is due to the presence of prior curvature in the embedded space given by the learnt DS. Whenever considerable curvature is present in the manifold, the lack of the coupling term, see Equation (3.23), can lead to undesired behavior in the neighborhood of the obstacle.

By considering the obstacle as a direct local deformation of the manifold, we recover the expected behavior of the vector field in the neighborhood of the obstacle, Figure 3.15b. The metric determinant, Figure 3.15e, displays space deformation consonant with the isotropic kernel used. The same happens for a second-order DS, Figures 3.15c and 3.15f. Differently from the first-order DS, as already observed already in Section 3.6.3, we notice how the streamlines detach more quickly from the obstacle showing asymmetric behavior before and after the local deformation.

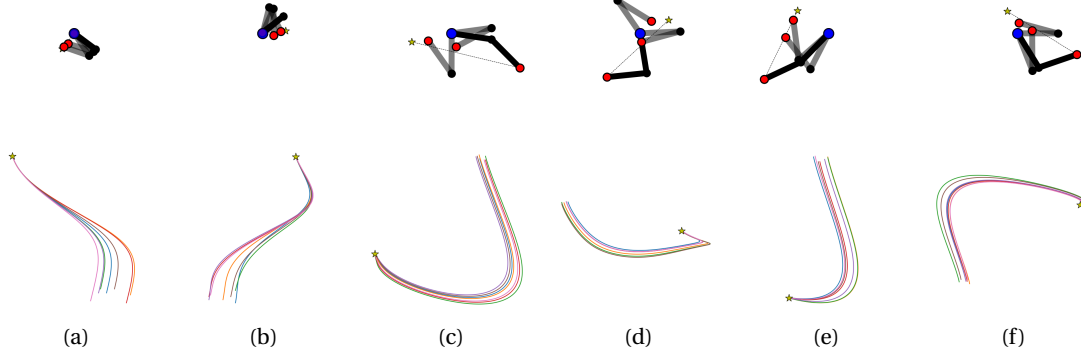


Figure 3.17: Evaluation: (top) Motion frames of 2-joints planar robotic manipulator; (bottom) sampled DSs in configuration space, corresponding to the above depicted robotic motion.

In this case, we can directly visualize how the presence of the obstacle affects the manifold's curvature, Figures 3.16a and 3.16b. The manifold structure is not re-learned and it remains globally consistent with the embedded representation shown in Figure 3.10. The obstacle affects the geometry only locally allowing the streamlines to follow the demonstrated trajectories away from the obstacle.

3.7.4 Evaluation

To evaluate the performance of the learned DS we employ three metrics: 1) Root Mean Square Error, $RMSE = \sqrt{\frac{1}{M} \sum_{i=1}^M \|\dot{\mathbf{x}}_i^{\text{ref}} - \dot{\mathbf{x}}_i^{\text{DS}}\|^2}$, 2) Cosine Similarity, $CS = \frac{1}{M} \sum_{i=1}^M \left| 1 - \frac{\langle \dot{\mathbf{x}}_i^{\text{ref}}, \dot{\mathbf{x}}_i^{\text{DS}} \rangle}{\|\dot{\mathbf{x}}_i^{\text{ref}}\| \|\dot{\mathbf{x}}_i^{\text{DS}}\|} \right|$, and 3) Dynamic Time Warping Distance, DTWD. (1) and (2) are point-wise metrics that measure the similarity between two vector fields; in the first case, magnitude and direction are considered while, in the second case, only direction influences the score.

For the first order system we have $\dot{\mathbf{x}}_i^{\text{DS}} = \mathbf{f}(\mathbf{x}_i^{\text{ref}})$. To compare first and second order systems with metric (1) and (2), for the second order system, we sample one step forward from the learned DS with initial condition given by $(\mathbf{x}_i^{\text{ref}}, \dot{\mathbf{x}}_i^{\text{ref}})$. Using the same sampling frequency, h , of the testing trajectories we have

$$\dot{\mathbf{x}}_{i+1}^{\text{DS}} = \dot{\mathbf{x}}_i^{\text{ref}} + \frac{1}{h} \mathbf{f}(\mathbf{x}_i^{\text{ref}}, \dot{\mathbf{x}}_i^{\text{ref}}) \quad \text{for } i = 1, \dots, N-1, \quad (3.34)$$

where N the number of sampled points per testing trajectory.

(3) measures the dissimilarity between the shape of a reference trajectory and its corresponding reproduction from the same initial points (and velocity for second order systems). In this case, for each testing trajectory, we sample a streamline starting from initial condition $\mathbf{x}_1^{\text{ref}}$, for the first order DS, and $(\mathbf{x}_1^{\text{ref}}, \dot{\mathbf{x}}_1^{\text{ref}})$, for the second order DS, with sampling frequency h . The sampling frequency is coincident with the one used to sample the testing trajectories.

3.7 Synthetic Example

Demonstration	RMSE [rad/s]			CS [rad]			DTWD [rad ²]		
	Baseline	1st - DS	2nd - DS	Baseline	1st - DS	2nd - DS	Baseline	1st - DS	2nd - DS
(a)	1.35 ± 0.66	0.40 ± 0.20	0.02 ± 0.00	0.04 ± 0.03	0.01 ± 0.00	0.01 ± 0.00	1.04 ± 0.89	0.20 ± 0.16	0.15 ± 0.06
(b)	6.81 ± 3.10	0.30 ± 0.07	0.02 ± 0.00	0.37 ± 0.25	0.01 ± 0.00	0.01 ± 0.00	2.57 ± 1.10	0.28 ± 0.07	0.70 ± 0.24
(c)	> 10.00	1.31 ± 0.13	0.04 ± 0.01	0.49 ± 0.44	0.01 ± 0.00	0.00 ± 0.00	> 10.00	1.59 ± 0.31	1.19 ± 0.26
(d)	> 10.00	1.25 ± 0.07	0.05 ± 0.01	0.71 ± 0.31	0.02 ± 0.00	0.00 ± 0.00	> 10.00	3.15 ± 0.57	2.33 ± 0.18
(e)	4.17 ± 2.79	0.72 ± 0.11	0.02 ± 0.00	0.03 ± 0.01	0.00 ± 0.00	0.00 ± 0.00	1.21 ± 0.48	0.26 ± 0.11	0.31 ± 0.10
(f)	> 10.00	0.88 ± 0.11	0.03 ± 0.00	0.49 ± 0.33	0.01 ± 0.0	0.00 ± 0.00	7.45 ± 5.28	5.17 ± 5.65	1.04 ± 0.45

Table 3.1: Evaluation results: for each sampled DS in Figure 3.17 we evaluate 1st and 2nd order learning DS against the Baseline with respect to the three metrics RMSE, Cosine Similarity Kernel in velocity space, DTWD.

We compare our approach against the learning dynamical systems via diffeomorphism in Rana et al. (2020). We refer to this approach as *Euclideanizing Flows* (EF)⁵. This approach adopts an NVP transformation structure, Dinh et al. (2017), where the diffeomorphism is achieved by a sequence of the so-called *coupling layers*. Each coupling layers is composed by operations of scaling and translation approximated by weighted sum of Random Fourier Features (RFF), Rahimi and Recht (2007b), kernels.

In each scenario, we conduct Adam optimization until convergence with a dynamic learning rate starting from a value of 0.01. We use a single NVIDIA GeForce 3090-24GB GPU for the experiments. Each trajectory is constituted by triplets $\{\mathbf{x}_i, \dot{\mathbf{x}}_i, \ddot{\mathbf{x}}_i\}_{i=1,\dots,T}$ with $T = 1000$.

Figure 3.17 the different DSs on which we conducted our evaluation. For each DS we performed 5 training repetitions, each time randomizing training and testing trajectories. Table 3.1 reports the comparison of our approach for first and second order DS against the baseline. Our method for the first-order DS achieves on average better performance than the baseline. For the second-order DS, our approach outperforms by a considerable margin the baseline and the first-order DS in RMSE.

Table 3.2 reports, over 2000 iterations, the training loss and time for each of the tested approaches. Due to the simplicity of our architecture, our method can be up to 5 times faster than the baseline for the first-order DS. Despite the increased complexity of the learning problem with respect to the first-order case, our method still manages to be up to two times faster at training time with respect to the baseline.

⁵Efficient PyTorch implementation of EF available at: <https://github.com/nash169/learn-diffeomorphism>

	Baseline	First DS	Second DS
Loss	6.8e-4 ± 9.6e-4	8.7e-5 ± 2.2e-5	1.1e-3 ± 8.5e-4
Time/Epoch	0.017 ± 8.7e-5	0.003 ± 1.9e-4	0.011 ± 1.2e-4
Train Epochs	4404.3 ± 1744.9	4168.3 ± 797.3	13257.0 ± 2280.3

Table 3.2: Training results.

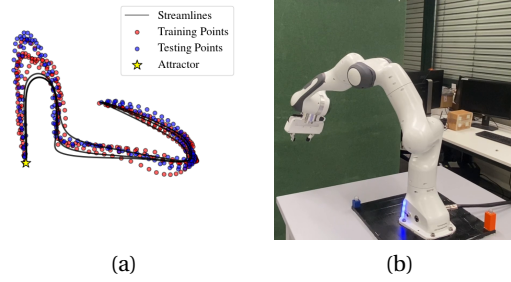


Figure 3.18: Learning three dimensional Dynamical Systems for Robotic Arm Control.

3.8 Robotics Experiments

We evaluate our method on learning 3D robotic end-effector motions in real-world settings⁶. The robotic platform used consists of the 7-DOF Franka Emika, Figure 3.18b. We gather 7 samples of the desired task space behavior by manually driving the robot’s end-effector. 4 trajectories are selected for training; the remaining trajectories are used as testing set. Figure 3.18a shows, qualitatively, the 3D learnt DS. The red dots represent the the sampled observations from the demonstrated trajectories converging towards the attractor represented by the yellow star. The blue trajectories represents the streamlines obtained by sampling, till convergence, from the learnt 3D DS starting from two different initial locations.

The equations of motion for an articulated robot system can be described as

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) = \boldsymbol{\tau}_c + \boldsymbol{\tau}_e \quad (3.35)$$

where $\mathbf{M}(\mathbf{q})$ is the inertia matrix, $\mathbf{h}(\mathbf{q}, \dot{\mathbf{q}})$ is the sum of gravitational, centrifugal and Coriolis forces. $\boldsymbol{\tau}_c$ and $\boldsymbol{\tau}_e$ are the vector of controlled and external joint torques, respectively.

Robustness to spatial and temporal perturbation as well as compliancy in the event of human interactions are the desired features of our control strategy. In this work, we test three different torque-based control strategies. Figure 3.19a illustrates the generic control strategy where robot configuration space state is fed directly to the controller module while task space information are pre-processed by the learned DS before going inside the controller. Depending on the DS order we have different control strategies illustrated in details in Figure 3.19b. The left block shows the two DS tested. Both the first-order and second-order DSs are split in two submodules, one operating in \mathbb{R}^3 and the other one in $SO(3)$, respectively used to control end-effector’s position and orientation. The DS operating in \mathbb{R}^3 is learned based on the demonstrated trajectory. The DS operating in $SO(3)$ generates a linear DS (critically damped for the second-order space) with the velocities taking place in the Lie Algebra $\mathfrak{so3}$, Solà et al. (2021).

⁶Code to reproduce simulation and real-robot results available at: <https://github.com/nash169/demo-learn-embedding>

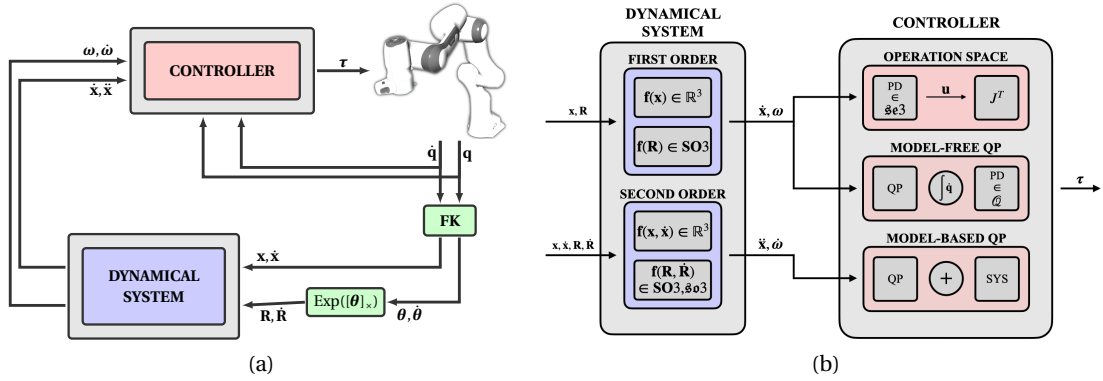


Figure 3.19: Control Structure.

Passive Interaction Control. Model-free control strategy that naturally adapts DS-based control, Kronander and Billard (2016). It is constitute by a feedback controller with solely the damping term

$$\tau_c = \mathbf{J}^T \mathbf{D}(\dot{\mathbf{x}} - \mathbf{f}(\mathbf{x})) + \mathbf{g}(\mathbf{q}). \quad (3.36)$$

$\mathbf{f}(\mathbf{x})$ is the target velocity generated by the first-order DS, top-left in Figure 3.19b.

Model-Free Quadratic Programming Control. In model-free QP control we first find the desired joint velocities as a solution of the optimization problem

$$\begin{aligned} \min_{\dot{\mathbf{q}}, \xi} \quad & \frac{1}{2} \dot{\mathbf{q}}^T \mathbf{Q} \dot{\mathbf{q}} + \xi^T \mathbf{W} \xi \\ \text{s.t.} \quad & \mathbf{J} \dot{\mathbf{q}} = \mathbf{f}(\mathbf{x}) + \xi \\ & \mathbf{q}^- \leq \mathbf{q}_{t-1} + dt \dot{\mathbf{q}} \leq \mathbf{q}^+, \quad \dot{\mathbf{q}}^- \leq \dot{\mathbf{q}} \leq \dot{\mathbf{q}}^+. \end{aligned} \quad (3.37)$$

The target task space velocity generate by the first-order DS is imposed as relaxed inverse kinematics constraint in the optimization problem. After prior integration of the desired joint velocities, a feedback controller composed by a proportional and a derivative terms generates the control torques

$$\tau_c = -\mathbf{K}(\mathbf{q} - \mathbf{q}^*) - \mathbf{D} \dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}). \quad (3.38)$$

Model-Based Quadratic Programming Control. This control strategy is suited when adopting our learned second-order DS. In this case the control torques are generated as solution of the optimization problem

$$\begin{aligned} \min_{\ddot{\mathbf{q}}, \tau, \xi} \quad & \frac{1}{2} \ddot{\mathbf{q}}^T \mathbf{Q} \ddot{\mathbf{q}} + \tau^T \mathbf{R} \tau + \xi^T \mathbf{W} \xi \\ \text{s.t.} \quad & \mathbf{M}(\mathbf{q}) \ddot{\mathbf{q}} = \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) + \tau + \xi, \quad \mathbf{J} \ddot{\mathbf{q}} = \mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}) + \xi, \quad \mathbf{q}^- \leq \mathbf{q}_{t-1} + dt \dot{\mathbf{q}}_{t-1} + 1/2 dt^2 \ddot{\mathbf{q}} \leq \mathbf{q}^+ \\ & \dot{\mathbf{q}}^- \leq \dot{\mathbf{q}}_{t-1} + dt \ddot{\mathbf{q}} \leq \dot{\mathbf{q}}^+ \\ & \ddot{\mathbf{q}}^-, \tau^- \leq \ddot{\mathbf{q}}, \tau \leq \ddot{\mathbf{q}}^+, \tau^+. \end{aligned} \quad (3.39)$$

Chapter 3. Learning Dynamical Systems Encoding Non-Linearity within Space Curvature

Metric	1st Order DS		2n Order DS
	Operation Space	QP Inverse Kinematics	QP Inverse Dynamics
DTWD	> 5.00	2.46 ± 0.09	2.06 ± 0.27
Frequency [Hz]	≈ 1300	≈ 1300	≈ 500

Table 3.3: Experimental results.

In this case we impose track the desired acceleration generated by our second-order DS by imposing a relaxed inverse dynamics constraint in the quadratic optimization problem. The control torques are

$$\boldsymbol{\tau}_c = \boldsymbol{\tau}^*, \quad (3.40)$$

where $\boldsymbol{\tau}^*$ is extracted from the solution of the optimization problem in Equation (3.39).

3.8.1 Trajectory Tracking Evaluation

In the case of first-order DS-based control, we deploy a standard proportional controller to generate an Euclidean space linear first-order DS that drives the end-effector to the starting point of each testing trajectory. Afterwards, we switch to the learned first-order DS to perform the desired motion.

When adopting second-order DS-based control, a critically damped proportional and derivative feedback is used to generate a standard Euclidean space linear second-order DS that drives the end-effector to the starting point of each testing trajectory. Afterwards, we switch to the learned second-order DS to perform the desired motion.

Table 3.3 reports the simulation⁷ results. Operation Space control yields considerably worse performance in terms of DTW but achieves much higher control frequency making it suitable for application where reactivity is major concern rather than precision. The combination of model-based QP and second-order DS demonstrates the best DTW. This comes at the cost of a reduced control frequency. Nevertheless the response time of such control makes it suitable for a large variety of high frequency applications.

3.9 Conclusion

In this study, we presented an approach for learning non-linear DS grounded in a purely geometrical framework. Our approach involves defining a harmonic damped oscillator on a latent manifold. The inherent non-linearity of the DS is intrinsically captured by the curvature of this manifold so that the chart space representation of the DS’s vector field accurately replicates target trajectories. Our method ensures global asymptotic stability, which is maintained irrespective of the manifold’s curvature. Additionally, our method’s explicit embedded manifold’s

⁷beautiful-bullet simulator available at: <https://github.com/nash169/beautiful-bullet>

representation grants direct control over the curvature of the space. This feature is particularly advantageous in integrating the learning of non-linear DS with scenarios involving obstacle avoidance. Our approach is characterized by relatively minimal constraints. Specifically, the function approximator is required to learn a multi-scalar function from \mathbb{R}^d to \mathbb{R} , maintaining only C^1 -regularity. This constraint significantly reduces computational demands during both the training and query phases.

In conventional robotic motion generation, the process typically involves initially planning trajectories at a kinematic level, followed by the development of controllers for accurate trajectory tracking. Our approach aims at reincorporating dynamics information within LfD framework by integrating two elements: (1) the utilization of high-level policies represented as more expressive second-order DSs, and (2) the application of model-based QP control for efficient one-step inverse dynamics.

Limitations & Future Developments. We demonstrated the application of learned DS. We believe that our approach can be adapted for joint space learning without violating joint limits. This involves initially learning a DS in high-dimensional joint space, followed by the application of local deformations to create energetic barriers, ensuring the robot remains within joint limits. In order to learn the embedding, we adopted a simple feed-forward network. Nevertheless, the use of controlled-smoothness kernels like the Matern kernel, coupled with probabilistic embedding, enhances noise robustness. Specifically, Gaussian Process Regression models can improve precision and query time, making our approach viable for online and adaptive learning. It is important to note that not every d -dimensional manifold can be isometrically embedded in a $d + 1$ dimensional Euclidean space. This limitation constrains the extent of non-linearity that can be effectively learned. A potential avenue for future research involves exploring embedding strategies for the manifold in a $d + n$ dimensional Euclidean space. However, this approach would necessitate the use of n function approximators, potentially leading to a decrease in model interpretability. Currently, our methods cannot handle vector fields with limit-cycles or non-zero curl components. To address the first limitation, one could consider embedding *compact* Riemannian manifolds into higher-dimensional Euclidean spaces. For the second limitation, involving the manifold topology and the generation of non-zero curl vector fields, extending the theory to include the embedding of pseudo-Riemannian manifolds into higher-dimensional Minkowski spaces may offer a solution.

4 Implicit Manifold Gaussian Process Regression

4.1 Foreword

The work presented in this chapter has been published and presented in *Fichera, B., Borovitskiy, V., Krause, A., and Billard, A. (2023). Implicit Manifold Gaussian Process Regression. In Advances in Neural Information Processing Systems 37th (NeurIPS 2023)*. The author of this thesis expresses gratitude to Viacheslav Borovitskiy for his invaluable assistance and support throughout the development of this project. His inputs were essential to its completion. Additionally, special thanks are owed for his contributions in Appendix C.1, which greatly enriched this work.

4.2 Introduction

Gaussian processes are among the most adopted models for learning unknown functions within the Bayesian framework. Their data efficiency and aptitude for uncertainty quantification make them appealing for modeling and decision-making applications in the fields like robotics Deisenroth and Rasmussen (2011), geostatistics Chilès and Delfiner (2012), numerics Hennig et al. (2015), etc.

The most widely used Gaussian process models, like squared exponential and Matérn Rasmussen and Williams (2006), impose the simple assumption of differentiability of the unknown function while also respecting the geometry of \mathbb{R}^d by virtue of being stationary or isotropic. Such simple assumptions make uncertainty estimates *reliable*, albeit too conservative at times. The same simplicity makes these models struggle from the *curse of dimensionality*. We hypothesize that it is still possible to leverage these simple priors for real world high-dimensional problems granted that they are adapted to the implicit *low-dimensional submanifolds* where the data actually lies, as illustrated by Figure 4.1.

Recent works in machine learning generalized Matérn Gaussian processes for modeling

Code available at <https://github.com/nash169/manifold-gp>.

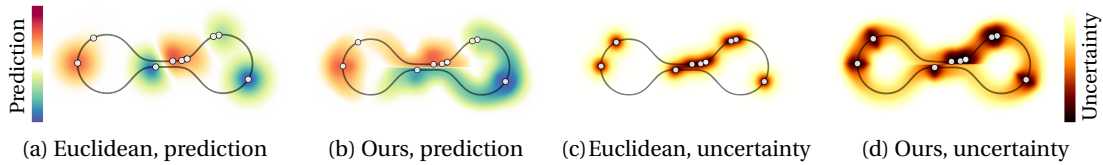


Figure 4.1: *Euclidean* (standard Matérn- $5/2$ kernel) vs *ours* (implicit manifold) Gaussian process regression for data that lies on a dumbbell-shaped curve (1-dimensional manifold) assumed unknown. The data contains a small set of labeled points and a large set of unlabeled points. Our technique recognizes that the two lines in the middle are intrinsically far away from each other, giving a much better model on and near the manifold. Far away from the manifold it reverts to the Euclidean model.

functions on non-Euclidean domains such as manifolds or graphs Borovitskiy et al. (2020, 2021, 2023); Azangulov et al. (2024a,b). Crucially, this line of work assumes *known* geometry (e.g. a manifold or a graph) beforehand. In this work we aim to widen the applicability of Gaussian processes for higher dimensional problems by *automatically learning* the implicit low-dimensional manifold upon which the data lies, the existence of which is suggested by the *manifold hypothesis*. We propose a new model which learns this structure and approximates the Matérn kernel on the implicit manifold.

Our approach can operate in both supervised and semi-supervised settings, with the emphasis on the latter: uncovering the implicit manifold may require a lot of samples from it, however these samples need not be labeled, and unlabeled data is usually more abundant. Taking inspiration in the manifold learning results of Coifman and Lafon (2006), Dunson et al. (2021) and others we approximate the unknown manifold by an appropriately weighted nearest neighbor graph. Then we use graph Matérn kernels thereon as approximations to the manifold Matérn kernels of Borovitskiy et al. (2020), extending them to the vicinity of the manifold in the ambient \mathbb{R}^d in an appropriate way.

4.2.1 Related Work and Contribution

High-dimensional Gaussian process regression is an area of active research, primarily motivated by decision making applications like Bayesian optimization. There are three main directions in this area: (1) selecting a small subset of input dimensions, (2) learning a small number of new features by linearly projecting the inputs and (3) learning non-linear features. Our technique belongs to the third direction. Further details on the area can be found in the recent review by Binois and Wycoff (2022).

The closest relative of our technique in the literature is described in Dunson et al. (2022). It targets the low-dimensional setting where the inputs are densely sampled on the underlying surface. It is based on the heat (diffusion) kernels on graphs as in Kondor and Lafferty (2002) and uses the Nyström method to extend kernels to \mathbb{R}^d , both of which may incur a high

computational cost.

We are targeting the high-dimensional setting. Here, larger datasets of partly labeled points are often needed to *infer* geometry. Because of this, we emphasize computational efficiency by leveraging sparse precision matrix structure of Matérn kernels (as opposed to the heat kernels) and use KNN for sparsifying the graph and accelerating the Nyström method. This results in linear computational complexity with respect to the number of data points. Furthermore, the model we propose is fully differentiable, which may be used to find both kernel and geometry hyperparameters by maximizing the marginal likelihood. Finally, to get reasonable predictions on the whole ambient space \mathbb{R}^d , we combine the prediction of the geometric model with the prediction of a classical Euclidean Gaussian process, weighting these by the relative distance to the manifold.

The geometric model is differentiable with respect to its kernel-, likelihood- and geometry-related hyperparameters, with gradient evaluation cost being linear with respect to the number of data points. After training, we can efficiently compute the predictive mean and kernel as well as sample the predictive model, providing the basic computational primitives needed for the downstream applications like Bayesian optimization. We evaluate our technique on a synthetic low-dimensional example and test it in a high-dimensional large dataset setting of predicting rotation angles of rotated MNIST images, improving over the standard Gaussian process regression.

4.3 Gaussian Processes

A Gaussian process $f \sim \text{GP}(m, k)$ is a distribution over functions on a set \mathcal{X} . It is determined by the mean function $m(\mathbf{x}) = \mathbb{E} f(\mathbf{x})$ and the covariance function (kernel) $k(\mathbf{x}, \mathbf{x}') = \text{Cov}(f(\mathbf{x}), f(\mathbf{x}'))$.

Given data \mathbf{X}, \mathbf{y} , where $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_n)^\top$ and $\mathbf{y} = (y_1, \dots, y_n)^\top$ with $\mathbf{x}_i \in \mathcal{X}^d, y_i \in \mathbb{R}$, one usually assumes $y_i = f(\mathbf{x}_i) + \varepsilon_i$ where $\varepsilon_i \sim \text{N}(0, \sigma_\varepsilon^2)$ is IID noise and $f \sim \text{GP}(0, k)$ is some *prior* Gaussian process, whose mean is assumed to be zero in order to simplify notation. The posterior distribution $f \mid \mathbf{y}$ is then another Gaussian process $f \mid \mathbf{y} \sim \text{GP}(\hat{m}, \hat{k})$ with Rasmussen and Williams (2006)

$$\hat{m}(\cdot) = \mathbf{K}_{(\cdot)\mathbf{X}}(\mathbf{K}_{\mathbf{X}\mathbf{X}} + \sigma_\varepsilon^2 \mathbf{I})^{-1} \mathbf{y}, \quad \hat{k}(\cdot, \cdot') = \mathbf{K}_{(\cdot, \cdot')} - \mathbf{K}_{(\cdot)\mathbf{X}}(\mathbf{K}_{\mathbf{X}\mathbf{X}} + \sigma_\varepsilon^2 \mathbf{I})^{-1} \mathbf{K}_{\mathbf{X}(\cdot')}, \quad (4.1)$$

where the matrix $\mathbf{K}_{\mathbf{X}\mathbf{X}}$ has entries $k(\mathbf{x}_i, \mathbf{x}_j)$, the vector $\mathbf{K}_{\mathbf{X}(\cdot)} = \mathbf{K}_{(\cdot)\mathbf{X}}^\top$ has components $k(\mathbf{x}_i, \cdot)$. If needed, one can efficiently sample $f \mid \mathbf{y}$ using *pathwise conditioning* Wilson et al. (2020, 2021)

Matérn Gaussian processes—including the limiting $\nu \rightarrow \infty$ case, squared exponential Gaussian processes—are the most popular family of models for $\mathcal{X} = \mathbb{R}^d$. These have zero

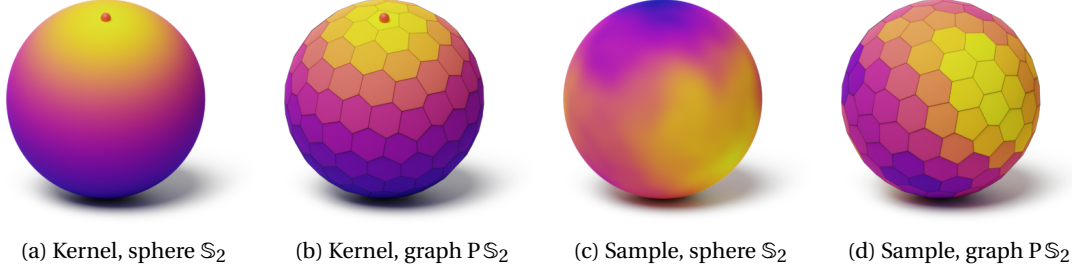


Figure 4.2: Kernel values $k(\cdot, \cdot)$ and samples for the Matérn- $3/2$ Gaussian processes on the sphere manifold \mathbb{S}_2 and for the approximating Matérn- $5/2$ process on a geodesic polyhedron graph $P\mathbb{S}_2$.

mean and kernels

$$k_{\nu, \kappa, \sigma^2}(\mathbf{x}, \mathbf{x}') = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} \frac{\|\mathbf{x} - \mathbf{x}'\|}{\kappa} \right)^\nu K_\nu \left(\sqrt{2\nu} \frac{\|\mathbf{x} - \mathbf{x}'\|}{\kappa} \right) \quad (4.2)$$

where K_ν is the modified Bessel function of the second kind Gradshteyn and Ryzhik (2014) and ν, κ, σ^2 are the hyperparameters responsible for smoothness, length scale and variance, respectively. We proceed to describe how Matérn processes can be generalized to inputs \mathbf{x} lying on *explicitly given* manifolds or graphs instead of the Euclidean space \mathbb{R}^d .

4.3.1 Matérn Gaussian Processes on Explicit Manifolds and Graphs

For a domain which is a Riemannian manifold, an obvious and natural idea for generalizing Matérn Gaussian processes could be to substitute the Euclidean distances $\|x - x'\|$ in Equation (4.2) with the geodesic distance. However, this approach results in ill-defined kernels that fail to be positive semi-definite Feragen et al. (2015b); Gneiting (2013).

Another direction for generalization is based on the stochastic partial differential equation (SPDE) characterization of Matérn processes first described by Whittle (1963b): $f \sim \text{GP}(0, k_{\nu, \kappa, \sigma^2})$ solves

$$\left(\frac{2\nu}{\kappa^2} - \Delta_{\mathbb{R}^d} \right)^{\frac{\nu}{2} + \frac{d}{4}} f = \mathcal{W}, \quad (4.3)$$

where $\Delta_{\mathbb{R}^d}$ is the standard Laplacian operator and \mathcal{W} is the Gaussian white noise with variance proportional to σ^2 . If taken to be the definition, this characterization can be easily extended to general Riemannian manifolds $\mathcal{X} = \mathcal{M}$ by substituting $\Delta_{\mathbb{R}^d}$ with the Laplace–Beltrami operator $\Delta_{\mathcal{M}}$, taking $d = \dim \mathcal{M}$ and substituting \mathcal{W} with the appropriate generalization of the Gaussian white noise Lindgren et al. (2011). Based on this idea, Borovitskiy et al. (2020) showed that on *compact* Riemannian manifolds, Matérn Gaussian processes are the zero-mean processes with kernels

$$k_{\nu, \kappa, \sigma^2}(x, x') = \frac{\sigma^2}{C_{\nu, \kappa}} \sum_{l=0}^{\infty} \left(\frac{2\nu}{\kappa^2} + \lambda_l \right)^{-\nu - d/2} f_l(x) f_l(x'), \quad (4.4)$$

4.4 Implicit Manifolds and Gaussian Processes on Them

where $-\lambda_l, f_l$ are eigenvalues and eigenfunctions of the Laplace–Beltrami operator and $C_{v,\kappa}$ is the normalizing constant ensuring that $\frac{1}{\mathcal{X}} \int_{\mathcal{X}} k_{v,\kappa,\sigma^2}(x, x) dx = \sigma^2$. This, alongside with considerations from Azangulov et al. (2024a) allows one to practically compute k_{v,κ,σ^2} for many compact manifolds.

If the domain \mathcal{X} is a weighted undirected graph \mathcal{G} , we can also use Equation (4.3) to define Matérn Gaussian processes on \mathcal{G} Borovitskiy et al. (2021). In this case, $\Delta_{\mathbb{R}^d}$ is substituted with the minus graph Laplacian $-\Delta_{\mathcal{G}}$ and $\mathcal{W} \sim \mathcal{N}(0, \sigma_{\mathcal{W}}^2 \mathbf{I})$ is the vector of IID Gaussians. Here, SPDE transforms into a stochastic linear system, whose solution is of the same form as Equation (4.4) but with a finite sum instead of the infinite series, with $d = 0$ because there is no canonical notion of dimension for graphs and with λ_l, f_l being the eigenvalues and eigenvectors—as functions on the node set—of the matrix $\Delta_{\mathcal{G}}$. These processes are illustrated on Figure 4.2.

4.4 Implicit Manifolds and Gaussian Processes on Them

Consider a dataset $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)^\top$, $\mathbf{x}_i \in \mathbb{R}^d$ partially labeled with labels $y_1, \dots, y_n \in \mathbb{R}$, $n \leq N$. Assume that \mathbf{x}_i are IID randomly sampled from a compact Riemannian submanifold $\mathcal{M} \subseteq \mathbb{R}^d$. As by Section 4.3.1, the manifold \mathcal{M} is associated to a family of Matérn Gaussian processes tailored to its geometry. We do not assume to know \mathcal{M} , only the fact that it exists, hence the question is: how can we recover the kernels of the aforementioned geometry-aware processes from the observed dataset?

It is clear from Equation (4.4) that to recover k_{v,κ,σ^2} we need to get the eigenpairs $-\lambda_l, f_l$ of the Laplace–Beltrami operator on \mathcal{M} . Naturally, for a finite dataset this can only be done approximately. We proceed to discuss the relevant theory of Laplace–Beltrami eigenpair approximation.

4.4.1 Background on Approximating the Eigenpairs of the Laplace–Beltrami Operator

There exists a number of theoretical and empirical results on eigenpair approximation. Virtually all of them study approximating the implicit manifold by some kind of a weighted undirected graph¹ with node set $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ and weights that are somehow determined by the Euclidean distances $\|\mathbf{x}_i - \mathbf{x}_j\|$. The eigenvalues of the *graph Laplacian* on this graph are supposed to approximate the eigenvalues of the Laplace–Beltrami operator, while the eigenvectors—regarded as functions on the node set—approximate the values of the eigenfunctions of the Laplace–Beltrami operator at $\mathbf{x}_i \in \mathcal{M}$. To approximate eigenfunctions elsewhere, any sort of continuous (smooth) interpolation suffices.

There are three popular notions of graph Laplacian. Let us denote the adjacency matrix

¹Other possibilities include linear (classical PCA) or quadratic Pavutnitskiy et al. (2022) approximations. See the books by Ma and Fu (2011); Lee and Verleysen (2007) for additional context.

Chapter 4. Implicit Manifold Gaussian Process Regression

of the weighted graph by \mathbf{A} and define \mathbf{D} to be the diagonal degree matrix with $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$. Then

$$\underbrace{\Delta_{\text{un}} = \mathbf{D} - \mathbf{A}}_{\text{unnormalized}}, \quad \underbrace{\Delta_{\text{sym}} = \mathbf{I} - \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2}}_{\text{symmetric normalized}}, \quad \underbrace{\Delta_{\text{rw}} = \mathbf{I} - \mathbf{D}^{-1} \mathbf{A}}_{\text{random walk normalized}}. \quad (4.5)$$

The first two of these are symmetric positive semi-definite matrices, the third is, generally speaking, non-symmetric. However, from the point of view of linear operators, all of them can be considered symmetric (self-adjoint) positive semi-definite: the first two with respect to the standard Euclidean inner product $\langle \cdot, \cdot \rangle$, and the third one with respect to the modified inner product $\langle \mathbf{v}, \mathbf{u} \rangle_{\mathbf{D}} = \langle \mathbf{D}\mathbf{v}, \mathbf{u} \rangle$. Thus for each there exists an orthonormal basis of eigenvectors and eigenvalues are non-negative.²

The most common way to define the graph is by setting $\mathbf{A}_{ij} = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/4\alpha^2)$ for an $\alpha > 0$. If \mathbf{x}_i are IID samples from the *uniform* distribution on the manifold \mathcal{M} , then all of the graph Laplacians, each multiplied by an appropriate power of α , converge to the Laplace–Beltrami operator, both pointwise Hein and Audibert (2007) and *spectrally* García Trillos et al. (2020), i.e. in the sense of eigenpair convergence, at least at the node set of the graph.³ However, if the inputs \mathbf{x}_i are sampled non-uniformly, graph Laplacians, at best, converge to different continuous limits, none of which coincides with the Laplace–Beltrami operator Hein and Audibert (2007).

Coifman and Lafon (2006) proposed a clever trick to handle non-uniformly sampled data $\mathbf{x}_1, \dots, \mathbf{x}_N$. Starting with $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{D}}$ defined in the same way as \mathbf{A} and \mathbf{D} before, they define $\mathbf{A} = \tilde{\mathbf{D}}^{-1} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-1}$. Intuitively, this corresponds to normalizing by the kernel density estimator to cancel out the unknown density. The corresponding Δ_{rw} then converges pointwise to the Laplace–Beltrami operator Hein and Audibert (2007), though Δ_{un} and Δ_{sym} do not: they converge to different continuous limits. (Dunson et al., 2021, Theorem 2) show that, under technical regularity assumptions, eigenvalues λ_k and renormalized eigenvectors of Δ_{rw} converge to the respective eigenvalues and eigenfunctions of the Laplace–Beltrami operator, regardless of the sampling density of \mathbf{x}_i .

Both in the simple case and in the sampling density independent case, the graphs and their respective Laplacians turn out to be dense, requiring a lot of memory to store and being inefficient to operate with. To make computations efficient, sparse graphs such as KNN graphs are much more preferable over the dense graphs. Spectral convergence for KNN graphs is studied, for example, in Calder and Trillos (2022), for $\mathbf{A}_{ij} = h(\|\mathbf{x}_i - \mathbf{x}_j\|/\alpha)$ with a compactly supported regular function h , and with limit depending on the sampling density. Unfortunately, we are unaware of any spectral convergence results in the literature that hold for KNN graphs and are independent of the data sampling density.

²Naturally, for the random walk normalized Laplacian Δ_{rw} the orthonormality is with respect to $\langle \cdot, \cdot \rangle_{\mathbf{D}}$.

³citegarcia2020 do not explicitly study Δ_{sym} . Since Δ_{sym} and Δ_{rw} are *similar* matrices, they share eigenvalues, so eigenvalue convergence for Δ_{sym} is trivial, the eigenvector convergence, however, is not.

4.4.2 Approximating Matérn Kernels on Manifolds

Here we incorporate various convergence results, including but not limited to the ones described in Section 4.4.1, proving that all spectral convergence results imply the convergence of graph Matérn kernels to the respective manifold Matérn kernels.

Proposition 6. *Denote the eigenpairs by λ_l, f_l for a graph Laplacian and by $\lambda_l^{\mathcal{M}}, f_l^{\mathcal{M}}$ for the Laplace–Beltrami operator. Fix $\delta > 0$. Assume that, with probability at least $1 - \delta$, for all $\varepsilon > 0$, for α small enough and for N large enough we have $|\lambda_l - \lambda_l^{\mathcal{M}}| < \varepsilon$ and $|f_l(\mathbf{x}_i) - f_l^{\mathcal{M}}(\mathbf{x}_i)| < \varepsilon$. Then, with probability at least $1 - \delta$, we have $k_{v,\kappa,\sigma^2}^{N,\alpha,L}(\mathbf{x}_i, \mathbf{x}_j) \rightarrow k_{v,\kappa,\sigma^2}(\mathbf{x}_i, \mathbf{x}_j)$ as $\alpha \rightarrow 0, N, L \rightarrow \infty$, where*

$$k_{v,\kappa,\sigma^2}^{N,\alpha,L}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sigma^2}{C_{v,\kappa}} \sum_{l=0}^{L-1} \left(\frac{2v}{\kappa^2} + \lambda_l \right)^{-v - \dim(\mathcal{M})/2} f_l(\mathbf{x}_i) f_l(\mathbf{x}_j). \quad (4.6)$$

Proof. First prove that the tail of the series in Equation (4.4) converges uniformly to zero, then combine this with eigenpair bounds. See details in Appendix C.1. \square

Remark. The convergence in $\mathbf{x}_i \in \mathcal{M}$ can be lifted to pointwise convergence for all $\mathbf{x} \in \mathcal{M}$ if eigenvectors are interpolated Lipschitz-continuously, simply because the eigenfunctions are smooth.

Inspired by this theory, we proceed to present the implicit manifold Gaussian process model.

4.4.3 Implicit Manifold Gaussian Process

Guided by the theory we described in the previous section we are now ready to formulate the implicit manifold Gaussian process model. Given the dataset $\mathbf{x}_1, \dots, \mathbf{x}_N \in \mathbb{R}^d$ and $y_1, \dots, y_n \in \mathbb{R}$, we put

$$\mathbf{A} = \tilde{\mathbf{D}}^{-1} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-1}, \quad \tilde{\mathbf{A}}_{ij} = S_K(\mathbf{x}_i, \mathbf{x}_j) \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{4\alpha^2}\right), \quad \tilde{\mathbf{D}}_{ij} = \begin{cases} \sum_m \tilde{\mathbf{A}}_{im} & i = j, \\ 0 & i \neq j. \end{cases} \quad (4.7)$$

Here $S_K(\mathbf{x}_i, \mathbf{x}_j) = 1$ if \mathbf{x}_i is one of the K nearest neighbors of \mathbf{x}_j or vice versa and $S_K(\mathbf{x}_i, \mathbf{x}_j) = 0$ otherwise; all matrices are of size $N \times N$ and depend on α and K as hyperparameters. Thanks to the coefficient $S_K(\mathbf{x}_i, \mathbf{x}_j)$ that performs KNN sparsification, the matrix \mathbf{A} is sparse when $K \ll N$.⁴

Then we consider the operator $\Delta_{\text{rw}} = \mathbf{I} - \mathbf{D}^{-1} \mathbf{A}$ defined by Equation (4.5), whose matrix is also sparse. Denoting its eigenvalues—ordered from the smallest to the largest—by $0 = \lambda_0 \leq \lambda_1 \leq \dots \leq \lambda_{N-1}$, and its eigenvectors—orthonormal under the modified inner product $\langle \cdot, \cdot \rangle_D$

⁴Note: $\tilde{\mathbf{A}}_{ii} = 1$, as if the graph has loops. Assuming $\tilde{\mathbf{A}}_{ii} = 0$ would lead to discontinuities at later stages.

Chapter 4. Implicit Manifold Gaussian Process Regression

and regarded as functions on the node set of the graph—by f_0, f_1, \dots, f_{N-1} , we define Matérn kernel on graph nodes \mathbf{x}_i by

$$k_{\nu, \kappa, \sigma^2}^{\mathbf{X}}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sigma^2}{C_{\nu, \kappa}} \sum_{l=0}^{L-1} \Phi_{\nu, \kappa}(\lambda_l) f_l(\mathbf{x}_i) f_l(\mathbf{x}_j), \quad \Phi_{\nu, \kappa}(\lambda) = \left(\frac{2\nu}{\kappa^2} + \lambda \right)^{-\nu}, \quad (4.8)$$

where L does not need to be equal to the actual number N of eigenpairs. Doing so means truncating the high frequency eigenvectors (f_l for l large), which always contribute less to the sum because they correspond to smaller values of $\Phi_{\nu, \kappa}(\lambda_l)$. This can massively reduce the computational costs.

By Proposition 6, Equation (4.8) approximates the manifold Matérn kernel with smoothness $\nu' = \nu - \dim(\mathcal{M})/2$. We adopt such a reparametrization because it does not require estimating the a priori unknown $\dim(\mathcal{M})$. This, however, makes the typical assumption of $\nu \in \{1/2, 3/2, 5/2\}$ inadequate. We chose a particular graph Laplacian normalization, namely the random walk normalized graph Laplacian Δ_{rw} , to approximate the true Laplace-Beltrami operator regardless of the potential non-uniform sampling of $\mathbf{x}_1, \dots, \mathbf{x}_N$, based on the theoretical insights described in Section 4.4.1.

The kernel in Equation (4.8) is only defined on the set of nodes \mathbf{x}_i , next step is to extend it to the whole space \mathbb{R}^d . Extending kernels is usually a difficult problem because one has to worry about positive semi-definiteness. To work around it, we extend the features f_l . For this, we use Nyström method: we allow the first argument of S_K to be an arbitrary vector from \mathbb{R}^d , defining $S_K(\mathbf{x}, \mathbf{x}_j) = 1$ if \mathbf{x}_j is one of the K nearest neighbors of \mathbf{x} among $\mathbf{x}_1, \dots, \mathbf{x}_N$. This allows us to extend $\tilde{\mathbf{A}}, \tilde{\mathbf{D}}, \mathbf{A}$ and \mathbf{D} as

$$\tilde{A}(\mathbf{x}, \mathbf{x}_j) = S_K(\mathbf{x}, \mathbf{x}_j) \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_j\|^2}{4\alpha^2}\right), \quad \tilde{D}(\mathbf{x}) = \sum_{j=1}^N \tilde{A}(\mathbf{x}, \mathbf{x}_j) = \sum_{\mathbf{x}_j \in \text{KNN}(\mathbf{x})} \tilde{A}(\mathbf{x}, \mathbf{x}_j), \quad (4.9)$$

$$A(\mathbf{x}, \mathbf{x}_j) = \frac{\tilde{A}(\mathbf{x}, \mathbf{x}_j)}{\tilde{D}(\mathbf{x})}, \quad D(\mathbf{x}) = \sum_{j=1}^N A(\mathbf{x}, \mathbf{x}_j) = \sum_{\mathbf{x}_j \in \text{KNN}(\mathbf{x})} A(\mathbf{x}, \mathbf{x}_j), \quad (4.10)$$

where $\text{KNN}(\mathbf{x})$ is the set of the K nearest neighbors from \mathbf{x} among $\mathbf{x}_1, \dots, \mathbf{x}_N$. With this, we define

$$f_l(\mathbf{x}) = \frac{1}{1 - \lambda_l} \sum_{j=1}^N \frac{A(\mathbf{x}, \mathbf{x}_j)}{D(\mathbf{x})} f_l(\mathbf{x}_j) = \frac{1}{1 - \lambda_l} \sum_{\mathbf{x}_j \in \text{KNN}(\mathbf{x})} \frac{A(\mathbf{x}, \mathbf{x}_j)}{D(\mathbf{x})} f_l(\mathbf{x}_j). \quad (4.11)$$

It is easy to check that this extension perfectly reproduces the values $f_l(\mathbf{x}_j)$, simply because $A(\mathbf{x}, \mathbf{x}_j)$ coincides with \mathbf{A}_{ij} when $\mathbf{x} = \mathbf{x}_i$ and because $(f_l(\mathbf{x}_1), \dots, f_l(\mathbf{x}_N))^{\top}$ is the eigenvector of \mathbf{A} corresponding to the eigenvalue $1 - \lambda_l$. It also allows us to extend the kernel $k_{\nu, \kappa, \sigma^2}^{\mathbf{X}}$ as well, such that $k_{\nu, \kappa, \sigma^2}^{\mathbf{X}}(\mathbf{x}, \mathbf{x}')$ is defined for arbitrary $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$ and Equation (4.8) still holds for the nodes $\mathbf{x}_1, \dots, \mathbf{x}_N$. We visualize an extended eigenvector and an extended kernel in Figures 4.3a and 4.3b.

For \mathbf{x} far away from the nodes \mathbf{x}_j the values of $\tilde{D}(\mathbf{x})$ become very small, making the

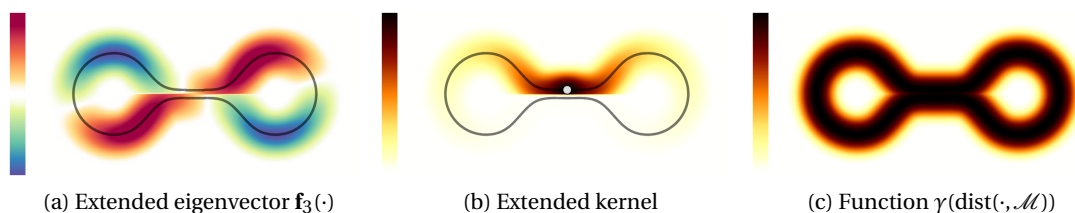


Figure 4.3: Different quantities connected to kernel extension. Notice that the values on subfigures (a) and (b) are artificially restricted to the set $\text{dist}(\cdot, \mathcal{M}) < 3\alpha$ to maintain numerical stability.

extension procedure numerically unstable. Furthermore, the geometric model is generally not so relevant far away from the manifold. Because of this, the final predictive model $f^{(p)} \sim \text{GP}(m^{(p)}, k^{(p)})$ combines the geometric model $f^{(m)} \sim \text{GP}(m^{(m)}, k^{(m)})$ —the posterior under the kernel $k_{\nu, \kappa, \sigma^2}^{\mathbf{x}}$ we defined just above—with the standard Euclidean model $f^{(e)} \sim \text{GP}(m^{(e)}, k^{(e)})$ —the posterior under the standard Euclidean Gaussian process in \mathbb{R}^d , for instance with the squared exponential kernel:

$$f^{(p)}(\mathbf{x}) = \gamma(\mathbf{x})f^{(m)}(\mathbf{x}) + (1 - \gamma(\mathbf{x}))f^{(e)}(\mathbf{x}), \quad \gamma(\mathbf{x}) = \exp\left(1 - \frac{(3\alpha)^2}{(3\alpha)^2 - \text{dist}(\mathbf{x}, \mathcal{M})^2}\right), \quad (4.12)$$

where γ is a bump function that is zero outside the 3α neighborhood of the manifold, illustrated in Figure 4.3c. Here $\text{dist}(\mathbf{x}, \mathcal{M})$ can be computed as the distance from \mathbf{x} to its nearest neighbor node \mathbf{x}_j .⁵

4.5 Efficient Training of the Implicit Manifold Gaussian Processes

Here we describe how to perform the implicit manifold Gaussian process regression efficiently, being able to handle hundreds of thousands of points, in both supervised and semi-supervised regimes.

In all cases we need efficient (approximate) KNN to build a graph, extend the kernel beyond the nodes \mathbf{x}_i and combine the geometric model with the standard Euclidean one. For this we use FAISS Johnson et al. (2019). The resulting sparse matrices, such as the Laplacian Δ_{rw} , we represent as black box functions capable of performing matrix-vector multiplications for any given input vector.

After hyperparameters are found—we will return to their search later—we need to compute the eigenpairs λ_l, \mathbf{f}_l of Δ_{rw} . For this we run Lanczos algorithm Meurant (2006) to evaluate the eigenpairs $\lambda_l^{\text{sym}}, \mathbf{f}_l^{\text{sym}}$ of the symmetric matrix Δ_{sym} , putting $\lambda_l = \lambda_l^{\text{sym}}$ and $\mathbf{f}_l = \mathbf{D}^{-1/2} \mathbf{f}_l^{\text{sym}}$ because the matrices Δ_{rw} and Δ_{sym} are similar, i.e. $\Delta_{\text{rw}} = \mathbf{D}^{-1/2} \Delta_{\text{sym}} \mathbf{D}^{1/2}$. Importantly, Lanczos

⁵In practice it makes sense to compute $\text{dist}(\mathbf{x}, \mathcal{M})$ as the average of distances between \mathbf{x} and its K nearest neighbors to smoothen the resulting $\gamma(\mathbf{x})$ —this is computationally cheap given an efficient KNN implementation.

Chapter 4. Implicit Manifold Gaussian Process Regression

algorithm only relies on matrix-vector products with Δ_{sym} . We only compute a few hundred of eigenpairs, asking Lanczos to provide twice as many and disregarding the rest.

When there is a lot of labeled data, we approximate the classical Euclidean (e.g. squared exponential) kernel using random Fourier features approximation Rahimi and Recht (2007c), this allows linear computational complexity scaling with respect to the number of data points. The number of Fourier features is taken to be equal to L , with the same L as in Equation (4.8).

As it was already mentioned, we need to find hyperparameters $\hat{\theta} = (\hat{\alpha}, \hat{\kappa}, \hat{\sigma}^2, \hat{\sigma}_\varepsilon^2)$ that determine the graph, Gaussian process prior and the noise variance that fit the observations \mathbf{y} best. The ν parameter we assume manually fixed. To avoid nonsensical parameter values—a common difficulty often occurring when the data is scarce—one might want to assume a prior $p(\theta)$ on θ . Some specific choices of which are discussed in Appendix C.3. Then $\hat{\theta}$ is a maximum a posteriori (MAP) estimate:

$$\hat{\theta} = \arg \max_{\theta} \log p(\mathbf{y} | \theta, \mathbf{X}) + \log p(\theta). \quad (4.13)$$

To simplify hyperparameter initialization and align with zero prior mean assumption it makes sense to preprocess y_i to be centered and normalized.

To solve the optimization problem in Equation (4.13) we use restarted gradient descent. Repeatedly evaluating the gradient of $\log p(\mathbf{y} | \theta, \mathbf{X})$ is the main computational bottleneck. The key idea for doing this efficiently—viable for integer values of ν —is to reduce matrix-vector products with Matérn kernels' precision to iterated matrix-vector products with the Laplacian, which is *sparse*. First, we describe this in detail in the noiseless supervised setting, where the idea is most directly applicable.

4.5.1 Noiseless Supervised Learning

Here we assume that all inputs are labeled, i.e. $N = n$, and all observations are noiseless, i.e. $\sigma_\varepsilon^2 = 0$.

Denoting by $\mathbf{P}_{\mathbf{XX}} = \mathbf{K}_{\mathbf{XX}}^{-1}$ the precision matrix, the log-likelihood $\log p(\mathbf{y} | \theta, \mathbf{X})$, up to a multiplicative constant and an additive constant irrelevant for optimization, is given by

$$L(\theta) = -\log \det(\mathbf{K}_{\mathbf{XX}}) - \mathbf{y}^\top \mathbf{K}_{\mathbf{XX}}^{-1} \mathbf{y} = \log \det(\mathbf{P}_{\mathbf{XX}}) - \mathbf{y}^\top \mathbf{P}_{\mathbf{XX}} \mathbf{y}. \quad (4.14)$$

Its gradient may be given and then subsequently approximated Hutchinson (1989) by

$$\frac{\partial L(\theta)}{\partial \theta} = \text{tr} \left(\mathbf{P}_{\mathbf{XX}}^{-1} \frac{\partial \mathbf{P}_{\mathbf{XX}}}{\partial \theta} \right) - \mathbf{y}^\top \frac{\partial \mathbf{P}_{\mathbf{XX}}}{\partial \theta} \mathbf{y} \approx \mathbf{z}^\top \mathbf{P}_{\mathbf{XX}}^{-1} \frac{\partial \mathbf{P}_{\mathbf{XX}}}{\partial \theta} \mathbf{z} - \mathbf{y}^\top \frac{\partial \mathbf{P}_{\mathbf{XX}}}{\partial \theta} \mathbf{y}, \quad (4.15)$$

where \mathbf{z} is a random vector consisting of IID variables that are either 1 or -1 with probability $1/2$. Since the kernels from Section 4.4.3 coincide with graph Matérn kernels on the nodes x_i ,

we have

$$\mathbf{K}_{\mathbf{X}\mathbf{X}} = \frac{\sigma^2}{C_{v,\kappa}} \sum_{l=0}^{L-1} \Phi_{v,\kappa}(\lambda_l) \mathbf{f}_l \mathbf{f}_l^\top, \quad \Delta_{\text{rw}} \mathbf{f}_l = \lambda_l \mathbf{f}_l, \quad \mathbf{f}_l^\top \mathbf{D} \mathbf{f}_m = \delta_{lm}. \quad (4.16)$$

However, as the graph bandwidth α is one of the hyperparameters we optimize over, using Equation (4.16) would entail repeated eigenpair computations and differentiating through this procedure. Because of this, *we use an alternative way* to compute matrix-vector products $\mathbf{P}_{\mathbf{X}\mathbf{X}} \mathbf{u}$ detailed below.⁶

Proposition 7. *Assuming $v \in \mathbb{N}$, the precision matrix $\mathbf{P}_{\mathbf{X}\mathbf{X}}$ of $k_{v,\kappa,\sigma^2}^{\mathbf{X}}(\mathbf{x}_i, \mathbf{x}_j)$ can be given by*

$$\mathbf{P}_{\mathbf{X}\mathbf{X}} = \frac{\sigma^{-2}}{C_{v,\kappa}^{-1}} \mathbf{D} \underbrace{\left(\frac{2v}{\kappa^2} \mathbf{I} + \Delta_{\text{rw}} \right) \cdots \left(\frac{2v}{\kappa^2} \mathbf{I} + \Delta_{\text{rw}} \right)}_{v \text{ times}}. \quad (4.17)$$

Proof. See Appendix C.1. □

Using Proposition 7 to evaluate matrix-vector products $\mathbf{P}_{\mathbf{X}\mathbf{X}} \mathbf{u}$ and conjugate gradients Meurant (2006) to solve $\mathbf{z}^\top \mathbf{P}_{\mathbf{X}\mathbf{X}}^{-1}$ using only the matrix-vector products, we can efficiently evaluate the right-hand side of Equation (4.15), with linear costs with respect to N , assuming that the graph is sparse.

4.5.2 Noiseless Semi-Supervised Learning

Here we assume that inputs are partly unlabeled, i.e. $N \neq n$, while observations are still noiseless, i.e. $\sigma_\varepsilon^2 = 0$. Denote \mathbf{Z} to be the labeled part of \mathbf{X} . Then the matrix $\mathbf{K}_{\mathbf{X}\mathbf{X}}$ in the log-likelihood given by Equation (4.14) should be substituted with $\mathbf{K}_{\mathbf{Z}\mathbf{Z}}$. However, while $\mathbf{P}_{\mathbf{X}\mathbf{X}}$ can be represented using Equation (4.17), the precision $\mathbf{P}_{\mathbf{Z}\mathbf{Z}} = \mathbf{K}_{\mathbf{Z}\mathbf{Z}}^{-1}$ cannot. We thus compute it as the Schur complement:

$$\mathbf{P}_{\mathbf{Z}\mathbf{Z}} = \mathbf{A} - \mathbf{B} \mathbf{D}^{-1} \mathbf{C}, \quad \mathbf{P}_{\mathbf{X}\mathbf{X}} = \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix}, \quad (4.18)$$

where partitioning of $\mathbf{P}_{\mathbf{X}\mathbf{X}}$ corresponds to partitioning \mathbf{X} into \mathbf{Z} and the rest. Evaluating a matrix-vector product $\mathbf{P}_{\mathbf{Z}\mathbf{Z}} \mathbf{u}$ requires a solve of $\mathbf{D}^{-1}(\mathbf{C}\mathbf{u})$. This solve can also be performed using conjugate gradients, keeping the computational complexity linear in N but increasing the constants.

⁶Though automatic differentiability could in principle work for iterative methods like the Lanczos algorithm, the amount of memory required for storing the gradients of the intermediate steps quickly becomes prohibitive.

4.5.3 Handling Noisy Observations

Finally, we assume noisy observations, i.e. $\sigma_\varepsilon^2 > 0$. The inputs can be partially unlabeled, i.e. $N \neq n$.

In this case, matrix $\mathbf{K}_{\mathbf{X}\mathbf{X}}$ in the log-likelihood given by Equation (4.14) should be substituted with $\mathbf{K}_{\mathbf{Z}\mathbf{Z}} + \sigma_\varepsilon^2 \mathbf{I}$. To reduce this to the previously considered cases, we use the Taylor expansion

$$(\mathbf{K}_{\mathbf{Z}\mathbf{Z}} + \sigma_\varepsilon^2 \mathbf{I})^{-1} \approx \mathbf{K}_{\mathbf{Z}\mathbf{Z}}^{-1} - \sigma_\varepsilon^2 \mathbf{K}_{\mathbf{Z}\mathbf{Z}}^{-2} + \sigma_\varepsilon^4 \mathbf{K}_{\mathbf{Z}\mathbf{Z}}^{-3} - \dots = \mathbf{P}_{\mathbf{Z}\mathbf{Z}} - \sigma_\varepsilon^2 \mathbf{P}_{\mathbf{Z}\mathbf{Z}}^2 + \sigma_\varepsilon^4 \mathbf{P}_{\mathbf{Z}\mathbf{Z}}^3 - \dots \quad (4.19)$$

In practice, we only use the first two terms on the right-hand side as an approximation. This allows to retain linear computational complexity scaling with respect to N but increases the constants.

4.5.4 Resulting Algorithm

Here we provide a concise summary of the *implicit manifold Gaussian process regression* algorithm.

Step 1: KNN-index. Construct the KNN index on the points $\mathbf{x}_1, \dots, \mathbf{x}_N$. This allows linear time evaluation of any matrix-vector product with $\tilde{\mathbf{A}}$, and thus also with \mathbf{A} , Δ_{rw} , $\mathbf{P}_{\mathbf{X}\mathbf{X}}$ for $v \in \mathbb{N}$, etc.

Step 2: hyperparameter optimization. Find the hyperparameters $\hat{\boldsymbol{\theta}}$ that solve Equation (4.13). Assuming $v = \hat{v} \in \mathbb{N}$ is manually fixed, this relies only on matrix-vector products with Δ_{rw} .

Step 3: computing the eigenpairs. Fixing the graph bandwidth $\hat{\alpha}$ found on Step 2, compute the eigenpairs λ_l, f_l corresponding to the L smallest eigenvalues λ_l . For large N , use Lanczos algorithm.

After the steps above are finished, Equations (4.8) and (4.11) define the geometric kernel $k_{\hat{v}, \hat{\kappa}, \hat{\sigma}^2}^{\mathbf{X}}(\mathbf{x}, \mathbf{x}')$ for arbitrary $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$. Then the respective prior $\text{GP}(0, k_{\hat{v}, \hat{\kappa}, \hat{\sigma}^2}^{\mathbf{X}})$ can be conditioned by the labeled data in the standard way, yielding the posterior $f^{(m)} \sim \text{GP}(m^{(m)}, k^{(m)})$. To get sensible predictions far away from the data, the geometric model $f^{(m)}$ is convexly combined with an independently trained classical Gaussian process model, as given by Equation (4.12). The resulting predictive model is still a Gaussian process, sum of two appropriately weighted independent Gaussian processes.

Remark. The number of neighbors K , the number of eigenpairs L and the smoothness v are assumed to be manually fixed parameters. Higher values of K and L improve the quality of approximation of the manifold kernel, which is often linked to better predictive performance, but requires more computational resources. The parameter v can be picked using cross validation or prior knowledge. Small integer values of v reduce computational costs, but may

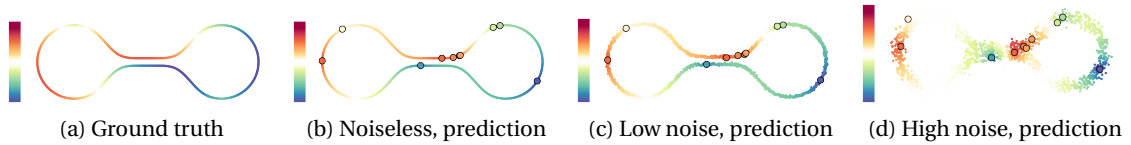


Figure 4.4: The ground truth function on the dumbbell manifold and the predictions of the implicit manifold Gaussian process regression (IMGP) under different levels of noise.

be inadequate for higher dimensions of the assumed manifold due to the $\nu' = \nu - \dim(\mathcal{M})/2$ link with the manifold kernel smoothness ν' .

4.6 Experiments

We start in Section 4.6.1 by examining a simple synthetic example to gain intuition on how noise-sensitive the technique is. Then in Section 4.6.2 we consider real datasets, showing improvements in higher dimensions. More experiments, results, and additional discussion can be found in Appendix C.2

4.6.1 Synthetic Examples

We consider a one dimensional manifold resembling the shape of a *dumbbell* which already appeared in Figures 4.1 and 4.3. The unknown function f_* is defined by fixing a point x^* in the top left part of the dumbbell, and computing $\sin(d(x^*, \cdot))$ where $d(\cdot, \cdot)$ denotes the geodesic (intrinsic) distance between a pair of points on the manifold. This function is illustrated in Figure 4.4a.

To measure performance we primarily rely on measuring negative log-likelihood (NLL) on the dense mesh of test locations. We do this because such metric is able to combine accuracy and calibration simultaneously. Additionally, we present the root mean square error (RMSE).

We investigate a semi-supervised setting where the number of unlabeled points is large ($N - n = 1546$) and the number of labeled points is small ($n = 10$). We contaminate the inputs with noise, putting $\mathbf{X} = \mathbf{X}_{\text{noiseless}} + \mathbf{N}(0, \sigma_{\mathbf{X}}^2 \mathbf{I})$ and do the same with the outputs, putting $\mathbf{y} = f_*(\mathbf{X}) + \mathbf{N}(0, \sigma_{\mathbf{y}}^2 \mathbf{I})$ for various values of $\sigma_{\mathbf{X}}, \sigma_{\mathbf{y}} > 0$. Specifically, we consider $\sigma_{\mathbf{X}} = \sigma_{\mathbf{y}} = \beta \in \{0, 0.01, 0.05\}$ to which we refer to as the noiseless setting, the low noise setting and the high noise setting, respectively.

The results for these are visualized in Figures 4.4b to 4.4d with performance metrics reported in Table 4.1. The implicit manifold Gaussian process regression is referred to as IMGP (we use $\nu = 1$) and it is compared with the standard Euclidean Matérn- $5/2$ Gaussian process. IMGP performs much better in the noiseless and the low noise settings. The high noise is enough to damage the calibration of IMGP, as it ties with the baseline model: NLL is slightly

worse and RMSE is slightly better.

In Appendix C.2.1 we show how performance depends on the fraction n/N of labeled data points, the truncation level L and we discuss the choice of the K parameter in KNN. Additionally, in Appendix C.2.2 we consider noise-sensitivity for a 2D manifold.

4.6.2 High Dimensional Datasets

For the high-dimensional setting, we considered predicting rotation angles for MNIST-based datasets. Additionally, we examined a high-dimensional dataset from the UCI ML Repository, CT slices.

Setup

Datasets. We consider two MNIST-based datasets. The first one is created by extracting a single image per digit from the complete MNIST dataset. By randomly rotating these 10 images we obtained $N = 10000$ training samples and 1000 testing samples. We call it *Single Rotated MNIST (SR-MNIST)*. For the second dataset, we select 100 random samples from MNIST. By randomly rotating these, we generate $N = 100000$ training samples, most will be unlabeled, and 10000 testing samples. We call it *Multiple Rotated MNIST (MR-MNIST)*. The last dataset, *CT slices*, has dimensionality of $d = 385$, we split it to have $N = 24075$ training samples and 24075 testing samples. Dataset names can be complemented by the fraction of labeled samples, e.g. MR-MNIST-10% refers to $n = 10\%N$.

Methods. We consider implicit manifold Gaussian processes in the supervised regime (*S-IMGP*) and in the semi-supervised regime (*SS-IMGP*). We compare them to the GPyTorch implementation of the Euclidean Matérn-5/2 Gaussian Process. We refer to it as the *Euclidean Gaussian Process (EGP)*.

Additional details. We run 100 iterations of hyperparameter optimization using Adam with a fixed learning rate of 0.01. For MNIST, with use IMGP with $\nu = 2$; for CT slices—with $\nu = 3$.

	RMSE			NLL		
	$\beta = 0$	$\beta = 0.01$	$\beta = 0.05$	$\beta = 0$	$\beta = 0.01$	$\beta = 0.05$
Euclidean Matérn-5/2	0.98	0.99 ± 0.02	1.02 ± 0.03	-2.17	-2.09 ± 0.03	-1.91 ± 0.10
IMGP	0.33	0.34 ± 0.02	1.00 ± 0.02	-5.02	-4.19 ± 0.1	-1.91 ± 1.88

Table 4.1: Performance metrics for the dumbbell manifold with varying magnitude of noise β .

Method	MNIST			CT slices		
	SR - 10%	MR - 1%	MR - 10%	5%	10%	25%
EGP	-0.54 ± 0.01	-0.20 ± 0.01	-0.43 ± 0.01	-0.80 ± 0.02	-0.96 ± 0.00	-1.20 ± 0.09
S-IMGP	-1.42 ± 0.01	2.24 ± 0.20	-0.68 ± 0.08	0.47 ± 0.06	-0.59 ± 0.08	-0.08 ± 0.01
SS-IMGP	-1.52 ± 0.01	-0.59 ± 0.01	-0.79 ± 0.00	26.1 ± 12.7	1.03 ± 0.09	-0.72 ± 0.68
S-IMGP (full)	-	-	-	0.64 ± 0.83	0.88 ± 0.29	-0.42 ± 0.10
SS-IMGP (full)	-	-	-	-2.48 ± 0.08	-2.35 ± 0.04	-1.99 ± 0.04

Table 4.2: Negative log likelihood on test samples for real datasets. For RMSE see Tables C.2 and C.3.

Results

Table 4.2 shows the negative log-likelihood metric for different datasets and methods on the test set. The respective RMSEs are presented in Appendix C.2.3. On SR-MNIST, IMGP outperforms EGP in both supervised and semi-supervised scenarios. MR-MNIST is more challenging. In the supervised setting for $n = 1\%N$, S-IMGP is incapable of inferring the underlying manifold structure, performing worse than EGP. However, SS-IMGP, with more data to infer manifold from, performs best. For $n = 10\%N$, IMGP gets a better grip of the dataset’s geometry, outperforming EGP in both regimes.

For CT slices, regardless of n , both S-IMGP and SS-IMGP performed poorly. Looking for an explanation, we considered two modifications. First, we fixed the graph bandwidth $\hat{\alpha}$ found in the algorithm’s Step 2 (cf. Section 4.5.4), and re-optimized the other hyperparameters $\kappa, \sigma^2, \sigma_\epsilon^2$ by maximizing the likelihood of the eigenpair-based model (truncated to $L = 2000$ eigenpairs) computed in the algorithm’s Step 3. This resulted in limited improvement but did not change the big picture—in fact, values for S-IMGP and SS-IMGP in Table 4.2 for CT slices already include this modification.

Second, on top of this hyperparameter re-optimization, we tried computing the eigenpairs using `torch.linalg.eigh` instead of the Lanczos implementation in GPyTorch, taking the same number $L = 2000$ of eigenpairs. The resulting methods S-IMGP (full) and SS-IMGP (full) showed considerable improvement over the baseline, as shown in Table 4.2. This indicated an issue with the quality of eigenpairs derived from the Lanczos method which requires further investigation. We discuss this in Appendix C.2.3, together with the aforementioned hyperparameter re-optimization procedure.

4.7 Conclusion

In this work, we propose the *implicit manifold Gaussian process regression* technique. It is able to use unlabeled data to improve predictions and uncertainty calibration by learning the implicit manifold upon which the data lies, being inspired by the convergence of graph Matérn Gaussian processes to their manifold counterparts. This helps building better probabilistic

Chapter 4. Implicit Manifold Gaussian Process Regression

models in higher dimensional settings where the standard Euclidean Gaussian processes usually struggle. This is supported by our experiments in a synthetic low-dimensional setting and for high-dimensional datasets. Leveraging sparse structure of graph Matérn precision matrices and efficient approximate KNN, the technique is able to scale to large datasets of hundreds of thousands points, which is especially important in high dimension, where a large number of unlabeled points is often needed to learn the implicit manifold. The model is fully differentiable, making it possible to infer hyperparameters in the usual way.

Limitations. The quality of the constructed graph significantly influences the technique's performance. When dealing with data from complex manifolds or exhibiting highly non-uniform density, simplistic KNN strategies might fail to capture the manifold structure due to their reliance on a single graph bandwidth. In such scenarios, larger values of parameters K and L , or in high dimensions, of parameter ν , may be beneficial but could substantially increase computational costs. Furthermore, larger datasets coupled with high parameter values can lead to numerical stability issues, for instance, in the Lanczos algorithm, calling for further improvements and research. Despite these challenges, our method shows promise for advancing probabilistic modeling in higher dimensions.

5 Conclusion and Future Developments

This chapter offers a comprehensive review of the research undertaken in this thesis, providing a concise summary of the key contributions made. Following this, we delve into a critical discussion of the limitations encountered during the research process. This sets the stage for outlining promising avenues for future exploration and development, thereby contextualizing the work within the broader scope of ongoing academic inquiry and potential real-world applications.

5.1 Main Contributions

Initially, we explored the potential of decomposing unlabelled data from multiple-attractor DS, focusing on identifying the number of underlying dynamics and their associated attractors. Our approach, grounded in Manifold Learning, provided theoretical guarantees for DS linearization through the reconstruction of multiple embedding spaces. By introducing a novel velocity-augmented kernel, we achieved a graph structure that allowed for the clustering of sub-dynamics and identification of equilibria locations of multiple-attractor DS. Subsequently, the use of eigendecomposition of a graph-based Laplacian matrix in combination with diffeomorphic strategies effectively allows to learn each of the identified sub-dynamics.

Further, we extended our investigation to improving complexity and adaptability of single-attractor non-linear learned DS, employing a purely geometrical framework. Here, we defined a harmonic damped oscillator on a latent manifold, capturing the inherent non-linearity of the DS through the curvature of this manifold. This approach not only ensured global asymptotic stability but also provided direct control over the space's curvature, which is advantageous in scenarios like obstacle avoidance. Our method's minimal constraints and efficient computational demands represent a significant advancement in conventional robotic motion generation, integrating dynamics information within a LfD framework. This integration encompasses high-level policies through expressive second-order DSs and model-based QP control for efficient inverse dynamics.

Chapter 5. Conclusion and Future Developments

Lastly, we proposed the Implicit Manifold Gaussian process Regression technique, for probabilistic learning on unknown manifolds or higher-dimensional settings. By learning the implicit manifold on which data lies, our technique enhances predictions and uncertainty calibration. It capitalizes on the convergence of graph Matérn Gaussian processes to their manifold counterparts, demonstrating significant scalability to large datasets, a crucial factor in high-dimensional contexts where a large volume of unlabeled data is often necessary. The fully differentiable nature of this model allows for conventional hyperparameter inference.

5.2 Limitations & Future Developments

In this thesis, we have presented novel methodologies and algorithms for DS identification and learning. However, each approach comes with its own set of limitations and avenues for future development. This section synthesizes these limitations and potential future directions, providing a cohesive view of the next steps in this field.

A critical limitation in our first study involves the dependency on accurate graph reconstruction. In cases where the velocity-augmented kernel fails to reconstruct the correct graph, such as in scenarios with high curvature in sampled trajectories, the performance of our algorithms for clustering and attractor location identification degrades significantly. Additionally, while our algorithm demonstrates scalability to higher dimensions, only its efficacy in 2-dimensional DS and single motion patterns has been thoroughly tested and confirmed. The extension to more complex and higher-dimensional spaces remains a theoretical possibility yet to be fully explored.

In the context of learning DS, our approach shows potential for adaptation to configuration space DS learning while respecting potential state limits. Our current method faces challenges in modeling certain type of non-linearities effectively, as not every d -dimensional manifold can be embedded isometrically in a $d + 1$ dimensional Euclidean space. Future research could explore embedding strategies in higher-dimensional spaces, albeit at the potential cost of reduced model interpretability. Furthermore, our methods currently do not accommodate vector fields with limit cycles or non-zero curl components, which presents an opportunity for future exploration, possibly through embedding compact Riemannian manifolds, for the former, and Minkowski manifolds, for the latter, into higher-dimensional spaces.

Lastly, the performance of our Implicit Manifold Gaussian Process Regression technique is closely tied to the quality of the constructed graph. The technique's efficacy is compromised when dealing with complex manifolds or data with highly non-uniform density, as simplistic KNN strategies may not adequately capture manifold structures. Adjusting parameters like K , L , and ν can improve performance in high-dimensional settings, but this comes with increased computational costs and potential numerical stability issues. These challenges underscore the need for further research and development in probabilistic modeling, particularly for large datasets and in higher-dimensional spaces.

5.2 Limitations & Future Developments

In conclusion, while our approaches have demonstrated promising results in their respective areas, the future of DS analysis and learning is poised for significant advancements. Addressing the limitations in graph reconstruction, extending the applicability to higher-dimensional and more complex DS, enhancing the embedding strategies for non-linear manifolds, and improving the stability and efficiency of probabilistic modeling techniques are key areas for future research. These developments will not only deepen our understanding of dynamical systems but also expand their practical applications in various fields.

A Appendix of Chapter 1

A.1 Proof of Lem. 1 from Sec 2.4

For each m -th row L_m of $L(G)$, we have $D + 1$ non-zero entries, where D is the degree of the m -th node with $L_{m,n} = D$ for $m = n$ and $L_{m,n} = -1$ for $m \neq n$. In the following, for the sake of clarity, we will drop the upper index k for the eigenvector's entries.

The first node, $n = 1$ has degree $D = 1$, as it is connected only to its direct neighbor. Using the monotonic ordered labelling of the nodes, $L_1 \mathbf{u} = \lambda u_1$ yields $u_1 - u_2 = \lambda u_1$, that simplifies into:

$$u_2 = (1 - \lambda)u_1. \quad (\text{A.1})$$

The second node of the path graph is connected to the previous and next node. It has hence degree $D = 2$. $L_2 \mathbf{u} = \lambda u_2$ yields $2u_2 - u_1 - u_3 = \lambda u_2$. The same holds for all the other nodes in the path. Hence, we have $2u_n - u_{n-1} - u_{n+1} = \lambda u_n$. This recurrence can be rewritten into the recursion:

$$u_{n+1} = (2 - \lambda)u_n - u_{n-1} \quad \text{for } n = 2, \dots, p_k - 1, \quad (\text{A.2})$$

or, equivalently,

$$u_n = (2 - \lambda)u_{n-1} - u_{n-2} \quad \text{for } n = 3, \dots, p_k. \quad (\text{A.3})$$

A.2 Proof of Lem. 2 from Sec 2.4

Consider the following Chebyshev polynomial of first kind T :

$$\begin{aligned} T_1(\lambda) &= 1 \\ T_2(\lambda) &= 1 - \frac{\lambda}{2} \\ T_n(\lambda) &= 2 \left(1 - \frac{\lambda}{2}\right) T_{n-1}(\lambda) - T_{n-2}(\lambda) \quad \text{for } n \geq 3, \end{aligned} \quad (\text{A.4})$$

Appendix A. Appendix of Chapter 1

equivalent to the following trigonometric expression:

$$T_n(\lambda) = \cos\left((n-1)\cos^{-1}\left(1 - \frac{\lambda}{2}\right)\right) \quad \text{for } n \geq 1. \quad (\text{A.5})$$

Consider the following Chebyshev polynomial of the second kind V :

$$\begin{aligned} V_1(\lambda) &= 1 \\ V_2(\lambda) &= 2\left(1 - \frac{\lambda}{2}\right) \\ V_n(\lambda) &= 2\left(1 - \frac{\lambda}{2}\right)V_{n-1}(\lambda) - V_{n-2}(\lambda) \quad \text{for } n \geq 3, \end{aligned} \quad (\text{A.6})$$

equivalent to the following trigonometric expression:

$$V_n(\lambda) = \frac{\sin\left(n\cos^{-1}\left(1 - \frac{\lambda}{2}\right)\right)}{\sin\left(\cos^{-1}\left(1 - \frac{\lambda}{2}\right)\right)} \quad \text{for } n \geq 1. \quad (\text{A.7})$$

The combination of the Chebyshev polynomials in Eq. A.5 and A.7, as indicated in Eq. 2.9, yields the trigonometric expression in Eq. 2.10.

A.3 Proof of Prop. 2 from Sec 2.4

For simplicity, we remove the superscript k in u_n^k and denote it as u_n . To preserve monotonicity, we must determine the conditions for which the entries do not change sign along one path. From Lem. 2, we know that u_n follows periodic functions that are expressed as a combination of trigonometric functions given by Eq. 2.10 for $n \geq 1$. We study the stationary points of Eq. 2.10 with respect to the index n :

$$\begin{aligned} \frac{\partial}{\partial n} u_n &= u_1 \left[-\theta \sin((n-1)\theta) - \theta \frac{\lambda \cos((n-1)\theta)}{2 \sin(\theta)} \right] \\ &= \sin((n-1)\theta) + \frac{\lambda \cos((n-1)\theta)}{2 \sin(\theta)} = 0. \end{aligned} \quad (\text{A.8})$$

Let the eigenvalue λ be

$$\lambda = 2(1 - \cos(\theta - 2\pi j)), \quad j \in \mathbb{N}. \quad (\text{A.9})$$

This expression of λ is sufficient to represent all eigenvalues in $[0, 4]$. Replacing Eq. A.9 in Eq. A.8, we obtain

$$\begin{aligned} \sin((n-1)\theta) + \frac{2(1 - \cos(\theta - 2\pi j)) \cos((n-1)\theta)}{2 \sin(\theta - 2\pi j)} &= 0 \\ \sin((n-1)\theta) + \tan\left(\frac{\theta}{2} - \pi j\right) \cos((n-1)\theta) &= 0 \\ \tan((n-1)\theta) + \tan\left(\frac{\theta}{2} - \pi j\right) &= 0. \end{aligned} \quad (\text{A.10})$$

The stationary points correspond to all n , such that:

$$n = \frac{\pi j}{\theta} + \frac{1}{2}. \quad (\text{A.11})$$

Expressing the stationary points in term of λ , we have:

$$n = \frac{\pi j}{\cos^{-1}(1 - \frac{\lambda}{2})} + \frac{1}{2}. \quad (\text{A.12})$$

The monotonicity of the entries u_n is hence preserved until one reaches a stationary point.

Observe, from Eq. A.12, that the smaller λ , the larger the number of nodes in the path with monotonicity preserved. If p_k is the number of nodes within each path graph k , we can determine an upper bound on λ for which all u_n within one path would evolve monotonically :

$$\lambda \leq 2 \left[1 - \cos\left(\frac{\pi}{p_k - \frac{1}{2}}\right) \right]. \quad (\text{A.13})$$

From (A.13), it is clear that for $p_k \geq 3$ one has additionally $\lambda < 1$. From Prop. 1, we have $u_2 = (1 - \lambda)u_1$, hence $u_2 < u_1$ if u_1 positive and $u_2 > u_1$, otherwise.

A.4 Proof of Prop. 3 from Sec 2.4

In the following we will drop the reference to the underlying graph structure G . Given the circulant block structure of the matrix J in Eq. 2.13, as shown by Tee (2007), the eigenvalues and corresponding eigenvectors of such matrix are determined by the following K equations, each giving us N eigenvalues and eigenvectors

$$H_j v = \lambda_j v \quad j \in \{0, \dots, K-1\}, \quad (\text{A.14})$$

with the matrix H_j defined as

$$H_j := B_0 + B_1 \left(\rho_j + \rho_j^{K-1} \right), \quad (\text{A.15})$$

Appendix A. Appendix of Chapter 1

where $\rho_j = \exp\left(i\frac{2\pi j}{K}\right)$. Observe that $\rho_j^{K-1} = \rho_j^{-1}$ for all j . Further,

$$\rho_j + \rho_j^{-1} = 2\cos\left(\frac{2\pi j}{K}\right) = 2\cos\left(\frac{2\pi(K-j)}{K}\right) = \rho_{K-j} + \rho_{K-j}^{-1}. \quad (\text{A.16})$$

Therefore the matrices H_j and H_{K-j} , for $j \geq 1$, yield the same eigenvalues. This shows how in the matrix J has at least $\frac{(K-1)}{2}$, if K is odd, or $\frac{K}{2} - 1$, if K is even, eigenvalues with algebraic multiplicity equal to 2.

A.5 Proof of Prop 4 from Sec 2.4

In the following we will drop the reference to the underlying graph structure G . The eigenvalues of the Laplacian matrix L are given by

$$\Lambda_L = 2I - \Lambda_J, \quad (\text{A.17})$$

where Λ_L and Λ_J are the diagonal matrices containing the eigenvalues of L and J , respectively. Therefore the smallest non-zero eigenvalue of L corresponds to the largest non-zero eigenvalue of J . The matrices H_j in Eq. A.15 can be expressed as

$$H_j = I + M_j, \quad (\text{A.18})$$

where M_j is the matrix

$$M_j = \begin{bmatrix} 0 & 1 & & & \\ 1 & -1 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & -1 & 1 \\ & & & 1 & \alpha_j - 2 \end{bmatrix} = M_{1,j} + M_2, \quad (\text{A.19})$$

wherein

$$M_{1,j} = \begin{bmatrix} M_0 & v \\ v^T & \alpha_j - 2 \end{bmatrix}, \quad M_2 = \begin{bmatrix} 0 & & & & \\ & -1 & & & \\ & & \ddots & & \\ & & & -1 & \\ & & & & 0 \end{bmatrix} \quad (\text{A.20})$$

with $\alpha_j = 2\cos\left(\frac{2\pi j}{K}\right)$ and

$$M_0 = \begin{bmatrix} 0 & 1 & & & \\ 1 & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ & & & 1 & 0 \end{bmatrix}, \quad v = [0, \dots, 0, 1]^T. \quad (\text{A.21})$$

M_0 is the adjacency matrix of a single path graph with $N - 1$ vertices. From Thm. 3.7 in Bapat (2010) the eigenvalues of M_0 are given by

$$\lambda_n(P) = 2\cos\left(\frac{\pi n}{N}\right), \quad n = 1, \dots, N - 1. \quad (\text{A.22})$$

Let λ_{max} and λ_{min} the largest and the smallest eigenvalues in the spectrum, respectively. From Thm 4.3.17 in Horn and Johnson (2012), the largest eigenvalue of each $M_1(i)$ matrices is equal or larger than the largest eigenvalue of M_0 . The largest eigenvalue of M_0 is found for $n = 1$. Hence, we have

$$2\cos\left(\frac{\pi}{N}\right) \leq \lambda_n(M_{1,j}), \quad \forall j. \quad (\text{A.23})$$

From Cor. 4.3.15 in Horn and Johnson (2012) the largest eigenvalue among all M_j matrices is bounded above and below by

$$2\cos\left(\frac{\pi}{N}\right) - 1 = \lambda_{max}(M_{1,j}) + \lambda_{min}(M_2) < \lambda_{max}(M_j) < 2\cos\left(\frac{\pi}{N}\right) \quad \forall j. \quad (\text{A.24})$$

The inequality is strict as $M_{1,j}$ and M_2 do not have a common eigenvector. Recalling that $\lambda(L) = 1 - \lambda(M_j)$, this is equivalent to

$$1 - 2\cos\left(\frac{\pi}{N}\right) < \lambda_{min}(L) < 2\left(1 - \cos\left(\frac{\pi}{N}\right)\right), \quad (\text{A.25})$$

being $\lambda_{min}(L)$ the smallest non-zero eigenvalue of the Laplacian matrix. Since $\cos\left(\frac{\pi}{N}\right) > \cos\left(\frac{\pi}{N-\frac{1}{2}}\right)$ for $N \geq 2$, we obtain the inequality in Eq. 2.16.

B Appendix of Chapter 2

B.1 Proofs of Prop. 5 & Thm. 2

In this appendix, we provide proofs for Prop. 5 and Thm. 2.

B.1.1 Proof of Prop. 5

A smooth embedding is an injective *immersion* $f : \mathcal{M} \rightarrow \mathcal{N}$ that is also a *topological embedding*. f , in order to be a topological embedding, has to yield a homeomorphism between \mathcal{M} and $f(\mathcal{M})$. Every map that is injective and continuous is an embedding in the topological sense.

Consider the local representative function in Eq. 3.6. Continuity follows directly from imposing $\psi \in C^r(\mathbb{R}^d)$ with $r \geq 1$. Also the injectivity property follows by the construction of the embedding. Let $\mathbf{x} \in \mathbb{R}^d$ and $\tilde{\mathbf{x}} \in \mathbb{R}^d$ two different points on \mathcal{M} , expressed in local coordinates, satisfying $\Psi(\mathbf{x}) = \Psi(\tilde{\mathbf{x}})$. Therefore

$$\begin{bmatrix} \mathbf{x} \\ \psi(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{x}} \\ \psi(\tilde{\mathbf{x}}) \end{bmatrix}. \quad (\text{B.1})$$

It is clearly that in Eq. B.1 the equality holds only if $\mathbf{x} = \tilde{\mathbf{x}}$.

$f : \mathcal{M} \rightarrow \mathcal{N}$ is an immersion if $f_* : T_p\mathcal{M} \rightarrow T_p\mathcal{N}$ is injective for every point $p \in \mathcal{M}$; equivalently $\text{rank}(f_*) = \dim(\mathcal{M})$. In order to analyze the rank of the pushforward map f_* we can look at its local coordinates components

$$f_{*j}^i = \partial_j(y \circ f \circ x^{-1})^i. \quad (\text{B.2})$$

It is clear from Eq. 3.6 that for $i \leq \dim(\mathcal{M})$ we have $f_{*j}^i \equiv \delta_j^i$ where δ_j^i is the Kronecker symbol. Therefore $\text{rank}(f_*) = \dim(\mathcal{M}) = d$.

B.1.2 Proof of Thm. 2

Stability of the harmonic linearly damped oscillator in Eq. 3.2 can be proved via Lyapunov's second method for stability. Let $\gamma : I \rightarrow \mathcal{M}$ a curve on the manifold \mathcal{M} . We adopt the following energetic function in intrinsic notation

$$V(\gamma(t), v_{\gamma(t)}) = \frac{1}{2} g_{\gamma(t)}(v_{\gamma(t)}, v_{\gamma(t)}) + \phi(\gamma(t)), \quad (\text{B.3})$$

where $t \in I$ and $\gamma(t) \equiv p \in \mathcal{M}$. The time derivative, D_t , of V is given by

$$D_t V(p, v_p) = \nabla_{v_p} V(p, v_p) \quad (\text{B.4})$$

$$= \frac{1}{2} \nabla_{v_p} (g_{(p)}(v_p, v_p)) + \nabla_{v_p} \phi(p). \quad (\text{B.5})$$

For the compatibility of the Levi-Civita connection with the metric, $\nabla_{v_p} g = 0$, we can simplify Eq. B.5 in

$$D_t V(p, v_p) = g_{(p)}(\nabla_{v_p} v_p, v_p) + \nabla_{v_p} \phi(p). \quad (\text{B.6})$$

Eq. B.6 can be expressed in local coordinates as

$$\begin{aligned} D_t V(p, v_p) &= g_{(p)} \left(-g^{ik} (\partial_i \phi + D_{ij} \dot{x}^j) \frac{\partial}{\partial x^k}, \dot{x}^i \frac{\partial}{\partial x^i} \right) \\ &\quad + \partial_i \phi \dot{x}^i \\ &= g_{(p)} \left(\frac{\partial}{\partial x^k}, \frac{\partial}{\partial x^i} \right) \left(-g^{ik} (\partial_i \phi + D_{ij} \dot{x}^j) \dot{x}^i \right) \\ &\quad + \partial_i \phi \dot{x}^i \\ &= -g_{ki} g^{ik} (\partial_i \phi + D_{ij} \dot{x}^j) \dot{x}^i + \partial_i \phi \dot{x}^i \end{aligned} \quad (\text{B.7})$$

Given the symmetry of the metric tensor $g^{ik} = g^{ki}$ we have

$$\begin{aligned} D_t V(p, v_p) &= -\partial_i \phi \dot{x}^i - D_{ij} \dot{x}^j \dot{x}^i + \partial_i \phi \dot{x}^i \\ &= -D_{ij} \dot{x}^j \dot{x}^i \end{aligned} \quad (\text{B.8})$$

We assumed $D \in \mathcal{S}_{++}^d$. Hence it follows $D > 0$ and $D_t V(p, v_p) < 0$.

B.2 Kernel-Based Space Deformation

In this appendix we analyze kernel based deformation. In Sec. 3.6.1 we saw how this technique to effectively achieve obstacle avoidance.

B.2.1 Derivation of the Metric Tensor

The differential of the deformation function in the direction \mathbf{v} is given by

$$\begin{aligned} d\psi(\mathbf{x})[\mathbf{v}] &= \lim_{t \rightarrow 0} \frac{\exp - \frac{\|\mathbf{x} + t\mathbf{v} - \bar{\mathbf{x}}\|^2}{2\sigma^2} - \exp - \frac{\|\mathbf{x} - \bar{\mathbf{x}}\|^2}{2\sigma^2}}{t} \\ &= k(\mathbf{x}, \bar{\mathbf{x}}) \cdot \lim_{t \rightarrow 0} \frac{1}{t} (e^z - 1), \end{aligned} \quad (\text{B.9})$$

where $z = -\frac{1}{\sigma^2} (t(\mathbf{x} - \bar{\mathbf{x}})^T \mathbf{v} + t^2 \mathbf{v}^T \mathbf{v})$. Dividing and multiplying Eq. B.9 by z and using the property $\lim_{z \rightarrow 0} \left(\frac{e^z - 1}{z} \right) = 1$ we have

$$\begin{aligned} d\psi(\mathbf{x})[\mathbf{v}] &= k(\mathbf{x}, \bar{\mathbf{x}}) \cdot \lim_{z \rightarrow 0} \left(\frac{e^z - 1}{z} \right) \cdot \lim_{t \rightarrow 0} \frac{z}{t} \\ &= k(\mathbf{x}, \bar{\mathbf{x}}) \cdot \lim_{t \rightarrow 0} -\frac{1}{\sigma^2} ((\mathbf{x} - \bar{\mathbf{x}})^T \mathbf{v} + t\mathbf{v}^T \mathbf{v}) \\ &= \left\langle -\frac{1}{\sigma^2} (\mathbf{x} - \bar{\mathbf{x}}) k(\mathbf{x}, \bar{\mathbf{x}}), \mathbf{v} \right\rangle = \langle \nabla \psi(\bar{\mathbf{x}}), \mathbf{v} \rangle. \end{aligned} \quad (\text{B.10})$$

Via the pull-back of the embedding metric we recover the metric onto the manifold. In case of Euclidean (identity) metric for the ambient space we have

$$\mathbf{G}(\mathbf{x}) = \mathbf{I} + \frac{1}{\sigma^4} (\mathbf{x} - \bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}})^T k(\mathbf{x}, \bar{\mathbf{x}})^2. \quad (\text{B.11})$$

The generic sum of kernels formulation is given by

$$\mathbf{G}(\mathbf{x}) = \mathbf{I} + \frac{1}{\sigma^4} \sum_{i=1}^N \alpha_i (\mathbf{x} - \bar{\mathbf{x}}_i)(\mathbf{x} - \bar{\mathbf{x}}_i)^T k(\mathbf{x}, \bar{\mathbf{x}}_i)^2. \quad (\text{B.12})$$

B.2.2 Derivative of the Metric Tensor

$$d\mathbf{G}(\mathbf{x})[\mathbf{v}] = \left((\mathbf{v}\bar{\mathbf{x}}^T + \bar{\mathbf{x}}\mathbf{v}^T) k(\bar{\mathbf{x}}) - \frac{1}{\sigma^2} \bar{\mathbf{x}}\bar{\mathbf{x}}^T (\bar{\mathbf{x}}^T \mathbf{v}) \right) k(\bar{\mathbf{x}}). \quad (\text{B.13})$$

B.3 Ablation Study

In order to select the correct model for the neural network used to learn the manifold embedding we performed an ablation study. In order to assess properly the model's ability of learning the embedding, for the ablation study, we fixed the stiffness matrix to be spherical and we do not optimize for it. For the second-order DS, the damping matrix is set to be spherical and fixed as well, with diagonal values such that the systems exhibits critically damped behavior in flat space. In this case the non-linearity is achieved solely via the manifold's curvature. For different number of layers and neurons within each layer we train each model till convergence. The training and testing dataset are given by 4 and 3 demonstrated trajectories, respectively.

Appendix B. Appendix of Chapter 2

MSE - 1e-3							
NEURONS PER LAYER							
	8	16	32	64	128	256	
LAYERS	1	1.935 ± 2.144	0.839 ± 0.798	1.317 ± 0.652	0.632 ± 0.306	1.066 ± 0.528	0.543 ± 0.314
	2	0.209 ± 0.074	1.195 ± 2.370	0.133 ± 0.020	0.183 ± 0.083	0.159 ± 0.096	2.092 ± 2.073
	3	0.335 ± 0.325	0.192 ± 0.061	1.093 ± 1.990	0.193 ± 0.115	0.477 ± 0.480	3.507 ± 3.777
	4	0.232 ± 0.080	0.161 ± 0.077	0.121 ± 0.040	0.718 ± 1.282	2.094 ± 1.984	1.972 ± 2.235
	5	1.226 ± 2.111	0.844 ± 1.507	1.221 ± 2.296	0.702 ± 0.766	2.564 ± 1.451	2.598 ± 2.406
	6	0.686 ± 0.554	1.191 ± 2.356	2.318 ± 2.860	2.542 ± 2.516	3.887 ± 1.637	7.678 ± 4.950

Table B.1: Ablation Study for the Neural Network function approximator.

Each trajectory is composed by 1000 sampled points. Results are averaged over 10 runs of ADAM optimization on an NVIDIA RTX4090 24GB.

Tab. B.1 reports the MSE results for different configurations. As it possible to see from the results, a 2-layers configuration with 32 neurons per layer is enough to achieve good performance with a high level of repeatability. By increasing the number of layers to 4, we observe an improvement of the performance. Nevertheless, we did not consider this marginal improvement sufficient to justify the additional overhead in term of computational cost at training and query time. Therefore, for both the 2D and 3D experiments we opted for a neural network model composed by 2 hidden layers each of the composed by 32 neurons.

C Appendix of Chapter 3

C.1 Theory

Proposition 6. Denote the eigenpairs by λ_l, f_l for a graph Laplacian and by $\lambda_l^{\mathcal{M}}, f_l^{\mathcal{M}}$ for the Laplace–Beltrami operator. Fix $\delta > 0$. Assume that, with probability at least $1 - \delta$, for all $\varepsilon > 0$, for α small enough and for N, L large enough we have $|\lambda_l - \lambda_l^{\mathcal{M}}| < \varepsilon$ and $|f_l(\mathbf{x}_i) - f_l^{\mathcal{M}}(\mathbf{x}_i)| < \varepsilon$. Then, with probability at least $1 - \delta$, we have $k_{v,\kappa,\sigma^2}^{N,\alpha,L}(\mathbf{x}_i, \mathbf{x}_j) \rightarrow k_{v,\kappa,\sigma^2}(\mathbf{x}_i, \mathbf{x}_j)$ as $\alpha \rightarrow 0, N, L \rightarrow \infty$, where

$$k_{v,\kappa,\sigma^2}^{N,\alpha,L}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sigma^2}{C_{v,\kappa}} \sum_{l=0}^{L-1} \left(\frac{2v}{\kappa^2} + \lambda_l \right)^{-v - \dim(\mathcal{M})/2} f_l(\mathbf{x}_i) f_l(\mathbf{x}_j). \quad (4.6)$$

Proof. Fix small $\varepsilon > 0$. We will prove that for α small enough and N, L large enough we have $|k_{v,\kappa,\sigma_f^2}(\mathbf{x}_i, \mathbf{x}_j) - k_{v,\kappa,\sigma_f^2}^{N,\alpha,L}(\mathbf{x}_i, \mathbf{x}_j)| < C\varepsilon$ for some $C > 0$ with probability at least $1 - \delta$. Since the assumption holds on the same event of probability $1 - \delta$ for all ε , this directly translates to the convergence on the same event. In fact, a probabilistic narrative is nonessential for what we actually prove, and we do not use it below. To simplify notation, we replace $\sum_{l=0}^{L-1}$ by $\sum_{l=0}^L$.

First, for any $L \in \mathbb{Z}_{>0}$ define the truncated version $k_{v,\kappa,\sigma_f^2}^L$ of the manifold kernel k_{v,κ,σ_f^2} by

$$k_{v,\kappa,\sigma_f^2}^L(\mathbf{x}, \mathbf{x}') = \frac{\sigma_f^2}{C_{v,\kappa}} \sum_{l=0}^L \left(\frac{2v}{\kappa^2} + \lambda_l^{\mathcal{M}} \right)^{-v - \dim(\mathcal{M})/2} f_l^{\mathcal{M}}(\mathbf{x}) f_l^{\mathcal{M}}(\mathbf{x}'). \quad (C.1)$$

The manifold Matérn kernels are the reproducing kernels of Sobolev spaces, if the latter are defined appropriately Borovitskiy et al. (2020). These are Mercer kernels De Vito et al. (2021), hence, by Mercer's theorem, $k_{v,\kappa,\sigma_f^2}^L \rightarrow k_{v,\kappa,\sigma_f^2}$ uniformly on \mathcal{M} , i.e. for L large enough we have

$$\left| k_{v,\kappa,\sigma_f^2}^L(\mathbf{x}_i, \mathbf{x}_j) - k_{v,\kappa,\sigma_f^2}(\mathbf{x}_i, \mathbf{x}_j) \right| \leq \sup_{\mathbf{x}, \mathbf{x}' \in \mathcal{M}} \left| k_{v,\kappa,\sigma_f^2}^L(\mathbf{x}, \mathbf{x}') - k_{v,\kappa,\sigma_f^2}(\mathbf{x}, \mathbf{x}') \right| < \varepsilon. \quad (C.2)$$

Appendix C. Appendix of Chapter 3

Now suppose that α is small enough and N is large enough so that

$$\left| \lambda_l - \lambda_l^{\mathcal{M}} \right| < \varepsilon', \quad |f_l(\mathbf{x}_i) - f_l^{\mathcal{M}}(\mathbf{x}_i)| < \varepsilon', \quad \varepsilon' = \min\left(1, \left(\lambda_L^{\mathcal{M}}\right)^{-\frac{d-1}{4}}, \left(\lambda_L^{\mathcal{M}}\right)^{-\frac{d-1}{2}}\right) \frac{\varepsilon}{L} \quad (\text{C.3})$$

for all $l \in \{1, \dots, L\}$ and for all $i \in \{1, \dots, N\}$ with probability at least $1 - \delta$.

Assuming the manifold is connected, by we have $|f_l^{\mathcal{M}}| \leq C_\lambda (\lambda_l^{\mathcal{M}})^{\frac{d-1}{4}}$ for $l > 0$ Donnelly (2006). The case $l = 0$ is special because $\lambda_0^{\mathcal{M}} = 0$. Since $f_0^{\mathcal{M}}$ is a constant function, we have

$$|f_l^{\mathcal{M}}| \leq C_\lambda \max\left((\lambda_l^{\mathcal{M}})^{\frac{d-1}{4}}, 1\right) \quad (\text{C.4})$$

for all $l \geq 0$, where C_λ here is potentially different from the C_λ before. Assuming, without loss of generality, $\varepsilon < 1$, we have

$$\left| f_l(\mathbf{x}_i) f_l(\mathbf{x}_j) - f_l^{\mathcal{M}}(\mathbf{x}_i) f_l^{\mathcal{M}}(\mathbf{x}_j) \right| \leq |f_l(\mathbf{x}_i)| \left| f_l(\mathbf{x}_j) - f_l^{\mathcal{M}}(\mathbf{x}_j) \right| \quad (\text{C.5})$$

$$+ \left| f_l^{\mathcal{M}}(\mathbf{x}_j) \right| \left| f_l(\mathbf{x}_i) - f_l^{\mathcal{M}}(\mathbf{x}_i) \right| \quad (\text{C.6})$$

$$\leq \varepsilon' \cdot \left(|f_l(\mathbf{x}_i)| + |f_l^{\mathcal{M}}(\mathbf{x}_j)| \right) \quad (\text{C.7})$$

$$\leq \varepsilon' \cdot \left(|f_l(\mathbf{x}_i) - f_l^{\mathcal{M}}(\mathbf{x}_i)| + |f_l^{\mathcal{M}}(\mathbf{x}_i)| + |f_l^{\mathcal{M}}(\mathbf{x}_j)| \right) \quad (\text{C.8})$$

$$\leq \varepsilon' \cdot \left(\varepsilon' + 2C_\lambda \max\left((\lambda_l^{\mathcal{M}})^{\frac{d-1}{4}}, 1\right) \right) \leq \frac{(1 + 2C_\lambda)\varepsilon}{L}. \quad (\text{C.9})$$

The function $\Phi(\lambda) = \left(\frac{2\nu}{\kappa^2} + \lambda\right)^{-\nu - \dim(\mathcal{M})/2}$ is Lipschitz: $|\Phi(\lambda) - \Phi(\lambda')| \leq C_\Phi |\lambda - \lambda'|$ where

$$C_\Phi = \sup_{\lambda \geq 0} |\Phi'(\lambda)| = \sup_{\lambda \geq 0} (\nu + \dim(\mathcal{M})/2) \left(\frac{2\nu}{\kappa^2} + \lambda\right)^{-\nu - \dim(\mathcal{M})/2 - 1} = (\nu + \dim(\mathcal{M})/2) \left(\frac{2\nu}{\kappa^2}\right)^{-\nu - \dim(\mathcal{M})/2 - 1}. \quad (\text{C.10})$$

Define an auxiliary kernel with manifold eigenvalues and graph eigenfunctions by

$$\tilde{k}_{\nu, \kappa, \sigma_f^2}^L(\mathbf{x}, \mathbf{x}') = \frac{\sigma_f^2}{C_{\nu, \kappa}} \sum_{l=0}^L \left(\frac{2\nu}{\kappa^2} + \lambda_l^{\mathcal{M}}\right)^{-\nu - \dim(\mathcal{M})/2} f_l(\mathbf{x}) f_l(\mathbf{x}'). \quad (\text{C.11})$$

Then

$$\frac{C_{\nu, \kappa}}{\sigma_f^2} \left| k_{\nu, \kappa, \sigma_f^2}^L(\mathbf{x}_i, \mathbf{x}_j) - \tilde{k}_{\nu, \kappa, \sigma_f^2}^L(\mathbf{x}_i, \mathbf{x}_j) \right| \leq \sum_{l=0}^L \left(\frac{2\nu}{\kappa^2} + \lambda_l^{\mathcal{M}}\right)^{-\nu - \dim(\mathcal{M})/2} \frac{(1 + 2C_\lambda)\varepsilon}{L} \quad (\text{C.12})$$

$$\leq \Phi(0)(1 + 2C_\lambda)\varepsilon. \quad (\text{C.13})$$

Also, noting that $|f_l(\mathbf{x}_i)| \leq |f_l^{\mathcal{M}}(\mathbf{x}_i) - f_l(\mathbf{x}_i)| + |f_l^{\mathcal{M}}(\mathbf{x}_i)| \leq \varepsilon' + C_\lambda \max((\lambda_l^{\mathcal{M}})^{\frac{d-1}{4}}, 1)$, write

$$\frac{C_{v,\kappa}}{\sigma_f^2} \left| \tilde{k}_{v,\kappa,\sigma_f^2}^L(\mathbf{x}_i, \mathbf{x}_j) - k_{v,\kappa,\sigma_f^2}^{N,\alpha,L}(\mathbf{x}_i, \mathbf{x}_j) \right| \leq \sum_{l=0}^L \left| \Phi(\lambda_l^{\mathcal{M}}) - \Phi(\lambda_l) \right| |f_l(\mathbf{x}_i)| |f_l(\mathbf{x}_j)| \quad (\text{C.14})$$

$$\leq \sum_{l=0}^L C_\Phi \left| \lambda_l^{\mathcal{M}} - \lambda_l \right| |f_l(\mathbf{x}_i)| |f_l(\mathbf{x}_j)| \quad (\text{C.15})$$

$$\leq \sum_{l=0}^L 2C_\Phi \varepsilon' \left((\varepsilon')^2 + C_\lambda^2 \max((\lambda_l^{\mathcal{M}})^{\frac{d-1}{2}}, 1) \right) \quad (\text{C.16})$$

$$\leq 2C_\Phi (1 + C_\lambda^2) \varepsilon. \quad (\text{C.17})$$

Finally,

$$\left| k_{v,\kappa,\sigma_f^2}(\mathbf{x}_i, \mathbf{x}_j) - k_{v,\kappa,\sigma_f^2}^{N,\alpha,L}(\mathbf{x}_i, \mathbf{x}_j) \right| \leq \left| k_{v,\kappa,\sigma_f^2}(\mathbf{x}_i, \mathbf{x}_j) - k_{v,\kappa,\sigma_f^2}^L(\mathbf{x}_i, \mathbf{x}_j) \right| \quad (\text{C.18})$$

$$+ \left| k_{v,\kappa,\sigma_f^2}^L(\mathbf{x}_i, \mathbf{x}_j) - \tilde{k}_{v,\kappa,\sigma_f^2}^L(\mathbf{x}_i, \mathbf{x}_j) \right| \quad (\text{C.19})$$

$$+ \left| \tilde{k}_{v,\kappa,\sigma_f^2}^L(\mathbf{x}_i, \mathbf{x}_j) - k_{v,\kappa,\sigma_f^2}^{N,\alpha,L}(\mathbf{x}_i, \mathbf{x}_j) \right| \quad (\text{C.20})$$

$$\leq \varepsilon + \frac{\sigma_f^2}{C_{v,\kappa}} (\Phi(0)(1 + 2C_\lambda) + 2C_\Phi(1 + C_\lambda^2)) \varepsilon. \quad (\text{C.21})$$

This proves the claim. \square

Proposition 7. Assuming $v \in \mathbb{N}$, the precision matrix $\mathbf{P}_{\mathbf{X}\mathbf{X}}$ of $k_{v,\kappa,\sigma^2}^{\mathbf{X}}(\mathbf{x}_i, \mathbf{x}_j)$ can be given by

$$\mathbf{P}_{\mathbf{X}\mathbf{X}} = \frac{\sigma^{-2}}{C_{v,\kappa}^{-1}} \mathbf{D} \underbrace{\left(\frac{2v}{\kappa^2} \mathbf{I} + \Delta_{\text{rw}} \right) \cdots \left(\frac{2v}{\kappa^2} \mathbf{I} + \Delta_{\text{rw}} \right)}_{v \text{ times}}. \quad (\text{4.17})$$

Proof. The covariance matrix $\mathbf{K}_{\mathbf{X}\mathbf{X}}$ is given by

$$\mathbf{K}_{\mathbf{X}\mathbf{X}} = \frac{\sigma^2}{C_{v,\kappa}} \sum_{l=0}^{L-1} \Phi(\lambda_l) \mathbf{f}_l \mathbf{f}_l^\top, \quad \Phi(\lambda) = \left(\frac{2v}{\kappa^2} + \lambda \right)^{-v} \quad (\text{C.22})$$

where $\mathbf{f}_l = \mathbf{D}^{-1/2} \mathbf{f}_l^{\text{sym}}$ and $\mathbf{f}_l^{\text{sym}}$ are the orthonormal eigenvectors of the symmetric normalized Laplacian Δ_{sym} . Denote

$$\mathbf{K}_{\mathbf{X}\mathbf{X}}^{\text{sym}} = \frac{\sigma^2}{C_{v,\kappa}} \sum_{l=0}^{L-1} \Phi(\lambda_l) \mathbf{f}_l^{\text{sym}} (\mathbf{f}_l^{\text{sym}})^\top \quad (\text{C.23})$$

then $(\mathbf{K}_{\mathbf{X}\mathbf{X}}^{\text{sym}})^{-1} = \frac{\sigma^{-2}}{C_{v,\kappa}^{-1}} \sum_{l=0}^{L-1} \left(\frac{2v}{\kappa^2} + \lambda_l \right)^v \mathbf{f}_l^{\text{sym}} (\mathbf{f}_l^{\text{sym}})^\top = \frac{\sigma^{-2}}{C_{v,\kappa}^{-1}} \left(\frac{2v}{\kappa^2} \mathbf{I} + \Delta_{\text{sym}} \right)^v$. We have

$$\mathbf{K}_{\mathbf{X}\mathbf{X}} = \frac{\sigma^2}{C_{v,\kappa}} \sum_{l=0}^{L-1} \Phi(\lambda_l) \mathbf{D}^{-1/2} \mathbf{f}_l^{\text{sym}} (\mathbf{f}_l^{\text{sym}})^\top \mathbf{D}^{-1/2} = \mathbf{D}^{-1/2} \mathbf{K}_{\mathbf{X}\mathbf{X}}^{\text{sym}} \mathbf{D}^{-1/2}. \quad (\text{C.24})$$

On the other hand,

$$\frac{\sigma_f^{-2}}{C_{v,\kappa}^{-1}} \mathbf{D} \left(\frac{2\nu}{\kappa^2} \mathbf{I} + \Delta_{\text{rw}} \right)^v = \frac{\sigma_f^{-2}}{C_{v,\kappa}^{-1}} \mathbf{D} \left(\frac{2\nu}{\kappa^2} \mathbf{I} + \mathbf{D}^{-1/2} \Delta_{\text{sym}} \mathbf{D}^{1/2} \right)^v \quad (\text{C.25})$$

$$= \frac{\sigma_f^{-2}}{C_{v,\kappa}^{-1}} \mathbf{D}^{1/2} \left(\frac{2\nu}{\kappa^2} \mathbf{I} + \Delta_{\text{sym}} \right)^v \mathbf{D}^{1/2} = \mathbf{D}^{1/2} (\mathbf{K}_{\text{XX}}^{\text{sym}})^{-1} \mathbf{D}^{1/2} \quad (\text{C.26})$$

$$= \mathbf{K}_{\text{XX}}^{-1} = \mathbf{P}_{\text{XX}}. \quad (\text{C.27})$$

□

C.2 Additional Experimental Results and Details

Here we provide additional results and details for synthetic examples and high-dimensional datasets.

C.2.1 1D Dumbbell Manifold

In this section, we analyze our method's sensitivity to the number of labeled points, spectrum truncation, and neighbor count in graph construction, offering deeper insights into its behavior.

Eigenpairs Truncation. From a theoretical standpoint, utilizing the complete set of eigenpairs to construct the kernel should be beneficial. However, this assumption may not hold true in cases where the optimization problem is not fully converged. The trends of RMSE and NLL, as depicted in Figures C.1a and C.1b, illustrate the impact of increasing the number of eigenpairs for 100, 200, and 1000 iterations. In situations where hyperparameter optimization does not converge fully, the length scale parameter κ fails to reach sufficiently high values necessary to generate appropriate spectral density decay, which in turn would properly weigh higher frequency eigenpairs. In such scenarios, truncating the spectrum is similar to increasing the length scale, which might improve results in certain cases. In the 1D scenario, due to its simplicity, we opted for a modest number of eigenpairs, namely $L = 50$. Exceeding this count did not yield any discernible improvements.

Dataset Size. We analyze the sensitivity to dataset size of our method (IMGP) against the Euclidean case (EGP) in the semi-supervised learning scenario. For fixed number of eigenpairs, $L = 50$, Figures C.1c and C.1d show performance metrics (RMSE and NLL) depending on the percentage $n\%N$ of labeled points. In a typical scenario where we have at our disposal fewer labeled points compared to the number of unlabeled points, our method outperforms EGP in both accuracy (RMSE) and uncertainty quantification (NLL). As n increases EGP starts to match the performance of IMGP.

Number of neighbors. For the one dimensional case considered in this section, only three

C.2 Additional Experimental Results and Details

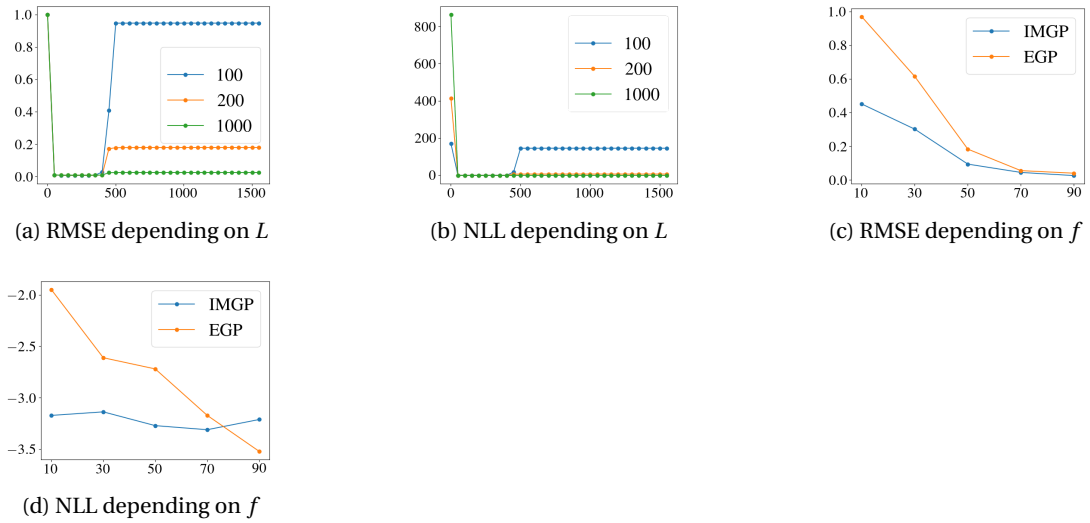


Figure C.1: Root Mean Square Error (RMSE) and Negative Log-Likelihood (NLL) for increasing number of eigenpairs L (left panels) and increasing fraction $f = n\%N$ of labeled points (right panels). The legend in (a) and (b) refers to the number of hyperparameter optimization iterations.

neighbors are necessary to capture the essential features of the manifold's geometry. Choosing a number less than three would hinder the algorithm's ability to capture the underlying manifold structure. On the other hand, increasing the number of neighbors beyond this threshold does not affect the solution, provided that sufficient time is allocated for hyperparameter optimization to converge and the graph bandwidth becomes small enough to "correct" for all undesired edges in the graph structure (for instance in the central region of the dumbbell). In high-dimensional problems where the dimension of the underlying manifold is unknown, this suggests to incrementally increase the number of neighbors until the loss function stops improving, if it is computationally feasible to try multiple values of K .

C.2.2 2D Dragon Manifold

We consider another synthetic setting, a complex 2D manifold, depicted in Figure C.2. The ground truth function is visualized in Figure C.2a, it is the same as in the one-dimensional case, the sine of the geodesic distance to a point, which is located in the green area.

	RMSE			NLL		
	$\beta = 0$	$\beta = 0.01$	$\beta = 0.05$	$\beta = 0$	$\beta = 0.01$	$\beta = 0.05$
Euclidean Matérn- $5/2$	0.24 ± 0.02	0.24 ± 0.01	0.26 ± 0.00	-0.85 ± 0.46	0.16 ± 1.58	1.26 ± 1.80
IMG	0.12 ± 0.01	0.21 ± 0.01	0.22 ± 0.01	-2.14 ± 0.13	-1.51 ± 0.03	-1.30 ± 0.09

Table C.1: Results for a complex 2D manifold with varying magnitude of sampling noise.

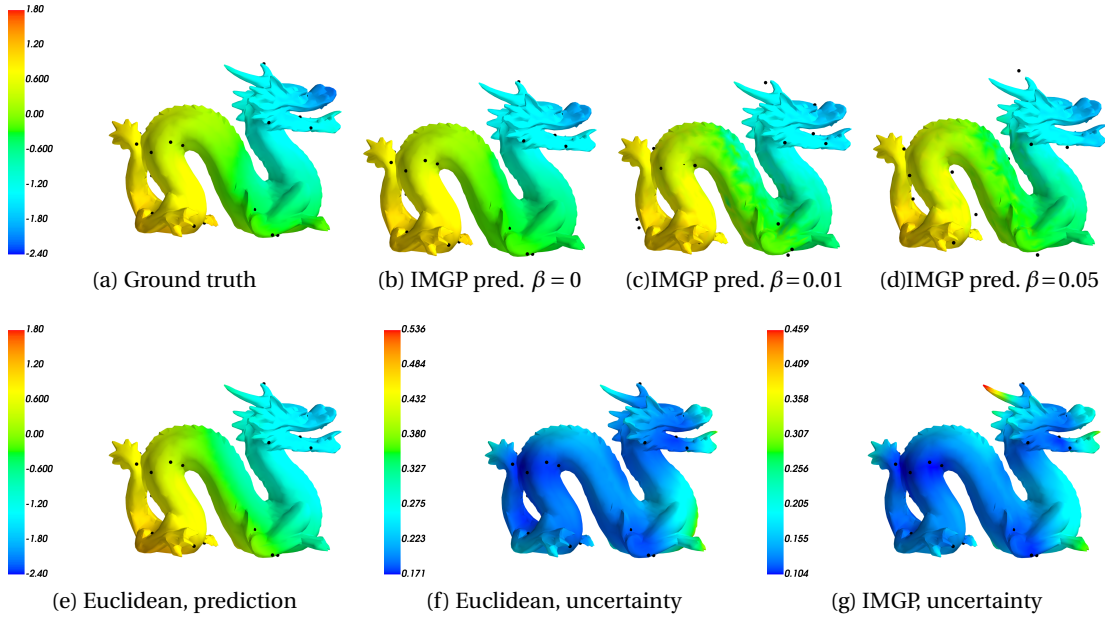


Figure C.2: (a) Ground truth function on the complex 2D manifold and (b)-(d) predictions of the implicit manifold Gaussian process regression (IMGP) for increasing level of sampling noise β . (e)-(d) Euclidean GP prediction and uncertainty and (g) IMGP uncertainty in noiseless scenario.

In the semi-supervised learning scenario, Figures C.2e and C.2b offer a comparison of the posterior mean between the Euclidean GP and IMGP, while Figures C.2f and C.2g illustrate the posterior standard deviation for both models—quite similar to each other, in this regime. Similar to what we did for the 1D compact manifold in Section 4.6.1, we evaluated the performance of IMGP under varying levels of sampling noise. Figures C.2b to C.2d display the IMGP predictions in the semi-supervised learning scenario as sampling noise increases. Similar to the 1D case, our approach consistently outperforms the standard Euclidean GP, as evidenced in Table C.1.

Remark. We observed that for higher levels of sampling noise, the linear combination of posteriors, as described by Equation (4.12), significantly outperform the single geometric model.

C.2.3 High Dimensional Datasets

Rotated MNIST. For IMGP, we use $\nu = 2$. As discussed in Appendix C.2.1, the optimal number of eigenpairs L varies considerably depending on the convergence of the optimization problem. Given the limited number of iterations per run we opted to fix the number of eigenpairs at $20\%N$ and $2\%N$, for SR-MNIST and MR-MNIST, respectively. Table C.2 reports the complete results obtained for the rotated MNIST dataset, including both RMSE and NLL. Notably, these emphasize the importance of unlabeled points, as S-IMGP performs worse than both SS-IMGP

C.2 Additional Experimental Results and Details

and EGP on MR-MNIST-1%.

CT slices. For IMGP, we use $\nu = 3$. In Table C.3 we report results for IMGP-S (full) and IMGP-SS (full). These are based on the "exact" eigenpairs, computed by `torch.linalg.eigh`, as opposed to the standard Lanczos implementation we use by default. Additionally, these include the hyperparameter re-optimization step, as described in Section 4.6.2 and discussed below. Regarding RMSE, all three compared methods—IMGP-S (full) and IMGP-SS (full) and EGP—exhibit similar performance, with a slight advantage for SS-IMGP (full) as the number of training points increases. When considering NLL, as previously observed with MNIST, SS-IMGP performs best in all settings. However, NLL decreases for larger values of n/N , indicating a probable overfit in this regime.

Hyperparameter re-optimization. We observed this step to serve two important purposes: (1) fixing overly small values of the signal variance σ^2 , potentially caused by the absence of covariance normalization in the optimization process and poor convergence; (2) adjusting the length scale parameter to take into account the loss of high-frequency components due to truncation.

Implementation. Currently, the implementation faces two significant limitations that might restrict its usability to high-memory hardware setups. Firstly, due to the absence of rich enough sparse matrix routines in PyTorch, we had to develop our own custom implementation of differentiable sparse operators. We kept to high-level routines, which forced us to strike a balance between performance and memory efficiency. In particular, for matrix-vector sparse product operations, our approach relies on highly optimized vectorized code, delivering high performance on GPU at the expense of increased memory allocation. Secondly, we faced challenges with sparse eigen-solvers in the PyTorch ecosystem. Our attempts of using the PyTorch implementation of the LOBPCG algorithm, `torch.lobpcg`, yielded relatively poor results. We had similar experience with the Lanczos implementation from GPyTorch Gardner et al. (2018). In light of this, when extracting higher-frequency components was needed, we resorted to two alternative solutions: PyTorch dense matrix eigen-decomposition `torch.linalg.eigh` and the SciPy Arpack wrapper `scipy.linalg.eigsh`. The first approach, while benefiting from GPU acceleration, can be infeasible because of limited GPU memory. The second approach, known for its efficiency in memory usage due to its Krylov subspace-based nature, is constrained to CPU utilization, significantly impacting the algorithm’s overall performance.

Dataset	n/N	RMSE			NLL		
		S-IMGP	SS-IMGP	EGP	S-IMGP	SS-IMGP	EGP
SR-MNIST	10%	0.04 ± 0.01	0.01 ± 0.00	0.12 ± 0.01	-1.42 ± 0.01	-1.52 ± 0.01	-0.54 ± 0.01
MR-MNIST	1%	0.74 ± 0.02	0.43 ± 0.02	0.74 ± 0.02	2.24 ± 0.20	-0.59 ± 0.01	-0.20 ± 0.01
MR-MNIST	10%	0.14 ± 0.05	0.03 ± 0.00	0.13 ± 0.00	-0.68 ± 0.08	-0.79 ± 0.00	-0.43 ± 0.01

Table C.2: Results for the rotated MNIST dataset.

Appendix C. Appendix of Chapter 3

n/N	RMSE			NLL		
	S-IMGP (full)	SS-IMGP (full)	EGP	S-IMGP (full)	SS-IMGP (full)	EGP
5%	0.27 ± 0.02	0.39 ± 0.04	0.24 ± 0.02	0.64 ± 0.83	-2.48 ± 0.08	-0.80 ± 0.02
10%	0.22 ± 0.02	0.19 ± 0.01	0.16 ± 0.00	0.88 ± 0.29	-2.35 ± 0.04	-0.96 ± 0.00
25%	0.14 ± 0.01	0.08 ± 0.02	0.09 ± 0.01	-0.42 ± 0.10	-1.99 ± 0.04	-1.20 ± 0.09
50%	0.08 ± 0.01	0.07 ± 0.01	0.08 ± 0.04	-0.96 ± 0.11	-2.04 ± 0.02	-1.02 ± 0.04

Table C.3: Results for Relative location of CT slices on axial axis ($d = 385$, $N = 48150$) from UCI Machine Learning Repository.

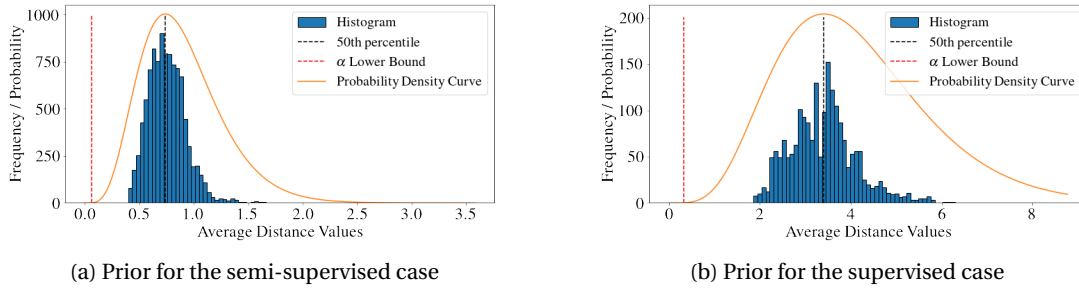


Figure C.3: The histogram of \mathcal{D} and the prior for the bandwidth hyperparameter α .

C.3 Hyperparameter Priors and Initialization

Here we describe hyperparameter priors which might be of help when using implicit manifold Gaussian process regression.

Graph Bandwidth α . Graph Laplacian converges to the Laplace–Beltrami operator when α tends to zero, motivating smaller α . However, in a non-asymptotic setting it is impractical to have α overly small as it will render the graph effectively disconnected and cause numerical instabilities. One appropriate prior could thus be *gamma distribution*, whose right tail discourages high values of α , and which, given an appropriate choice of the parameters, encourages α to align with the scale of pairwise distances between graph nodes. Specifically, we choose the parameters of the gamma distribution so as to (1) match its mode with the median (Q_2) of pairwise average distance between K -th nearest neighbors because such an α would give reasonable weights in the KNN graph and (2) so as to place its standard 0.95 confidence interval to the right of a certain lower bound $\bar{\alpha}$ we define further that ensures the graph is numerically not disconnected. Let

$$\mathcal{D} = \{\max_{\mathbf{x}_i \in \text{KNN}(\mathbf{x}_j)} \|\mathbf{x}_i - \mathbf{x}_j\| \text{ where } j = 1, \dots, N\}. \quad (\text{C.28})$$

The bandwidth's lower bound we compute as

$$\bar{\alpha} = \min_{d \in \mathcal{D}} \sqrt{-\frac{d^2}{4 \log \tau}}, \quad (\text{C.29})$$

C.3 Hyperparameter Priors and Initialization

where τ is a user-defined parameter.

Let η and β the shape and rate parameters of the gamma distribution. In order to achieve (1), we define

$$\eta = \rho Q_2 + 1 \quad \text{and} \quad \beta = \rho \quad (\text{C.30})$$

where ρ is used to achieve (2) and is given by

$$\rho \approx \frac{4Q_2}{(Q_2 - \bar{\alpha})^2}. \quad (\text{C.31})$$

Considering the S-MNIST dataset, Figure C.3 shows the bandwidth prior distribution for the semi-supervised (labeled and unlabeled points) and the supervised (labeled points) scenarios.

Signal Variance σ_f^2 . Assuming normalized y_i , the signal variance σ_f^2 should be close to 1. Because of this a natural prior for σ^2 is the truncated normal (onto the set of positive reals $\sigma^2 > 0$), with mode at 1. The pre-truncation variance can be chosen, for example, to have 0 at 3 standard deviations away from the mode, i.e. it can be chosen to be 1/9. Note that setting a prior over the signal variance parameter requires evaluating the normalization constant $C_{v,\kappa}$ ¹ for the kernel at each hyperparameter optimization step, something that can be avoided otherwise.

Noise Variance σ_ε^2 . Choosing a prior for the noise variance σ_ε^2 is heavily problem-dependent. Truncated normal (onto the set $\sigma_\varepsilon^2 > 0$) with mode at 0 could be a reasonable option. The pre-truncation variance can be chosen, for example, to be 1, 1/4 or 1/9, placing the value 1, which is the variance of the normalized observations y_i , at 1, 2 or 3 standard deviations away from the mode.

Length scale. The interpretation and the scale of the length scale parameter is manifold-specific. This makes it very difficult to come up with any reasonable prior. Because of this, we suggest actually leaving the length scale parameter free.

Parameter initialization When doing MAP estimation, one can initialize parameters randomly, sampling them from respective priors.

¹Covariance matrix normalization constant $C_{v,\kappa}$ can be approximated as $C_{v,\kappa} \approx \frac{1}{M} \sum_{i=1}^M \mathbf{e}_i^T \mathbf{P}_{\mathbf{X}\mathbf{X}}^{-1} \mathbf{e}_i$, where M is relatively small, \mathbf{e}_i are random standard basis vectors and the solve is performed by running the conjugate gradients. Differentiability of the model is preserved. Note however that we did not use this normalization in our tests since the performance improvement we observed was not enough to justify the additional computation overhead. This trade-off can vary considerably from case to case, which is why in our implementation, the covariance normalization is optional.

Bibliography

- Absil, P.-A., Mahony, R., and Sepulchre, R. (2008). *Optimization Algorithms on Matrix Manifolds*. Princeton University Press.
- Albu-Schäffer, A. and Della Santina, C. (2020). A review on nonlinear modes in conservative mechanical systems. *Annual Reviews in Control*, 50:49–71.
- Albu-Schäffer, A. and Sachtler, A. (2023). What Can Algebraic Topology and Differential Geometry Teach Us About Intrinsic Dynamics and Global Behavior of Robots? In *Robotics Research*, Springer Proceedings in Advanced Robotics, pages 468–484. Springer Nature Switzerland.
- Antonova, R., Rai, A., and Atkeson, C. G. (2017). Deep kernels for optimizing locomotion controllers. In *Conference on Robot Learning*, pages 47–56. PMLR.
- Azangulov, I., Smolensky, A., Terenin, A., and Borovitskiy, V. (2024a). Stationary kernels and gaussian processes on lie groups and their homogeneous spaces i: the compact case. *Journal of Machine Learning Research*.
- Azangulov, I., Smolensky, A., Terenin, A., and Borovitskiy, V. (2024b). Stationary kernels and gaussian processes on lie groups and their homogeneous spaces ii: non-compact symmetric spaces. *Journal of Machine Learning Research*.
- Bapat, R. B. (2010). *Graphs and matrices*, volume 27. Springer.
- Beik-Mohammadi, H., Hauberg, S., Arvanitidis, G., Neumann, G., and Rozo, L. (2021). Learning Riemannian Manifolds for Geodesic Motion Skills. In *Proceedings of Robotics: Science and Systems*.
- Belkin, M. and Niyogi, P. (2002). Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering. In *Advances in Neural Information Processing Systems 14*, pages 585–591. MIT Press.
- Belkin, M. and Niyogi, P. (2003). Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. *Neural Computation*, 15(6):1373–1396.
- Belkin, M. and Niyogi, P. (2004). Semi-Supervised Learning on Riemannian Manifolds. *Machine Learning*, 56(1):209–239.

Bibliography

- Belkin, M., Niyogi, P., and Sindhvani, V. (2006). Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, 7(Nov):2399–2434.
- Bengio, Y., Paiement, J.-f., Vincent, P., Delalleau, O., Roux, N. L., and Ouimet, M. (2004). Out-of-Sample Extensions for LLE, Isomap, MDS, Eigenmaps, and Spectral Clustering. In *Advances in Neural Information Processing Systems 16*, pages 177–184. MIT Press.
- Bhardwaj, M., Sundaralingam, B., Mousavian, A., Ratliff, N. D., Fox, D., Ramos, F., and Boots, B. (2021). STORM: An Integrated Framework for Fast Joint-Space Model-Predictive Control for Reactive Manipulation. In *5th Annual Conference on Robot Learning*.
- Billard, A., Calinon, S., Dillmann, R., and Schaal, S. (2008). Survey: Robot programming by demonstration. Technical report, Springer.
- Billard, A. G., Calinon, S., and Dillmann, R. (2016). Learning from humans. In *Springer handbook of robotics*, pages 1995–2014. Springer.
- Binois, M. and Wycoff, N. (2022). A survey on high-dimensional gaussian process modeling with application to bayesian optimization. *ACM Transactions on Evolutionary Learning and Optimization*, 2(2):1–26.
- Bjelonic, F., Sachtler, A., Albu-Schäffer, A., and Santina, C. D. (2022). Experimental Closed-Loop Excitation of Nonlinear Normal Modes on an Elastic Industrial Robot. *IEEE Robotics and Automation Letters*, 7(2):1689–1696.
- Blocher, C., Saveriano, M., and Lee, D. (2017). Learning stable dynamical systems using contraction theory. In *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 124–129.
- Bonalli, R., Bylard, A., Cauligi, A., Lew, T., and Pavone, M. (2019). Trajectory Optimization on Manifolds: A Theoretically-Guaranteed Embedded Sequential Convex Programming Approach. In *booktitle = Proceedings of Robotics: Science and Systems*.
- Borovitskiy, V., Azangulov, I., Terenin, A., Mostowsky, P., Deisenroth, M., and Durrande, N. (2021). Matérn Gaussian Processes on Graphs. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, pages 2593–2601. PMLR.
- Borovitskiy, V., Karimi, M. R., Somnath, V. R., and Krause, A. (2023). Isotropic gaussian processes on finite spaces of graphs. In *International Conference on Artificial Intelligence and Statistics*.
- Borovitskiy, V., Terenin, A., Mostowsky, P., and Deisenroth (he/him), M. (2020). Matérn Gaussian Processes on Riemannian Manifolds. In *Advances in Neural Information Processing Systems*, volume 33, pages 12426–12437. Curran Associates, Inc.
- Boumal, N. (2023). *An Introduction to Optimization on Smooth Manifolds*. Cambridge University Press, 1 edition.

- Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. (2017). Geometric Deep Learning: Going beyond Euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42.
- Bullo, F. and Lewis, A. D. (2005). *Geometric Control of Mechanical Systems: Modeling, Analysis, and Design for Simple Mechanical Control Systems*. Texts in Applied Mathematics. Springer-Verlag.
- Bylard, A., Bonalli, R., and Pavone, M. (2021). Composable Geometric Motion Policies using Multi-Task Pullback Bundle Dynamical Systems. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7464–7470.
- Calder, J. and Trillos, N. G. (2022). Improved spectral convergence rates for graph laplacians on ϵ -graphs and k-nn graphs. *Applied and Computational Harmonic Analysis*, 60:123–175.
- Calinon, S. (2020). Gaussians on riemannian manifolds: Applications for robot learning and adaptive control. *IEEE Robotics & Automation Magazine*, 27(2):33–45.
- Cheng, C.-A., Mukadam, M., Issac, J., Birchfield, S., Fox, D., Boots, B., and Ratliff, N. (2020). RMPflow: A Computational Graph for Automatic Motion Policy Generation. In *Algorithmic Foundations of Robotics XIII*, Springer Proceedings in Advanced Robotics, pages 441–457. Springer International Publishing.
- Chilès, J.-P. and Delfiner, P. (2012). *Geostatistics: Modeling Spatial Uncertainty*. John Wiley & Sons.
- Coifman, R. R. and Lafon, S. (2006). Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5–30.
- Coifman, R. R., Lafon, S., Lee, A. B., Maggioni, M., Nadler, B., Warner, F., and Zucker, S. W. (2005). Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *Proceedings of the National Academy of Sciences*, 102(21):7426–7431.
- Corteville, B., Aertbeliën, E., Bruyninckx, H., De Schutter, J., and Van Brussel, H. (2007). Human-inspired robot assistant for fast point-to-point movements. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3639–3644. IEEE.
- De Vito, E., Mücke, N., and Rosasco, L. (2021). Reproducing kernel hilbert spaces on manifolds: Sobolev and diffusion spaces. *Analysis and Applications*, 19(03):363–396.
- Deisenroth, M. and Rasmussen, C. E. (2011). Pilco: A model-based and data-efficient approach to policy search. In *International Conference on Machine Learning*.
- Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2017). Density estimation using Real NVP. In *The International Conference on Learning Representations (ICLR)*.
- do Carmo, M. (1992). *Riemannian Geometry*. Mathematics: Theory & Applications. Birkhäuser Basel.

Bibliography

- Donnelly, H. (2006). Eigenfunctions of the laplacian on compact riemannian manifolds. *Asian Journal of Mathematics*, 10(1):115–126.
- Dsilva, C. J., Talmon, R., Coifman, R. R., and Kevrekidis, I. G. (2018). Parsimonious representation of nonlinear dynamical systems through manifold learning: A chemotaxis case study. *Applied and Computational Harmonic Analysis*, 44(3):759–773.
- Dunson, D. B., Wu, H.-T., and Wu, N. (2021). Spectral convergence of graph laplacian and heat kernel reconstruction in L^∞ from random samples. *Applied and Computational Harmonic Analysis*, 55:282–336.
- Dunson, D. B., Wu, H.-T., and Wu, N. (2022). Graph based gaussian processes on restricted domains. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 84(2):414–439.
- Erem, B., Orellana, R. M., Hyde, D. E., Peters, J. M., Duffy, F. H., Stovicek, P., Warfield, S. K., MacLeod, R. S., Tadmor, G., and Brooks, D. H. (2016). Extensions to a manifold learning framework for time-series analysis on dynamic manifolds in bioelectric signals. *Physical Review E*, 93(4):042218.
- Feragen, A., Lauze, F., and Hauberg, S. (2015a). Geodesic exponential kernels: When curvature and linearity conflict. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3032–3042.
- Feragen, A., Lauze, F., and Hauberg, S. (2015b). Geodesic exponential kernels: When curvature and linearity conflict. In *Conference on Computer Vision and Pattern Recognition*.
- Fichera, B. and Billard, A. (2022). Linearization and Identification of Multiple-Attractor Dynamical Systems through Laplacian Eigenmaps. *Journal of Machine Learning Research*, 23(294):1–35.
- Fichera, B. and Billard, A. (2023). Hybrid Quadratic Programming - Pullback Bundle Dynamical Systems Control. In *Robotics Research*, Springer Proceedings in Advanced Robotics, pages 387–394. Springer Nature Switzerland.
- Fichera, B. and Billard, A. (2024). Learning dynamical systems encoding non-linearity within space curvature. *Under Review*.
- Fichera, B., Borovitskiy, V., Krause, A., and Billard, A. (2023). Implicit Manifold Gaussian Process Regression. In *Advances in Neural Information Processing Systems 37th (NeurIPS 2023)*.
- Figueroa, N. and Billard, A. (2018). A Physically-Consistent Bayesian Non-Parametric Mixture Model for Dynamical System Learning. In *Conference on Robot Learning (CoRL)*, pages 927–946.
- Forni, F. and Sepulchre, R. (2014). A Differential Lyapunov Framework for Contraction Analysis. *IEEE Transactions on Automatic Control*, 59(3):614–628.

- García Trillos, N., Gerlach, M., Hein, M., and Slepčev, D. (2020). Error estimates for spectral convergence of the graph laplacian on random geometric graphs toward the laplace–beltrami operator. *Foundations of Computational Mathematics*, 20(4):827–887.
- Gardner, J., Pleiss, G., Weinberger, K. Q., Bindel, D., and Wilson, A. G. (2018). GPyTorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration. In *Advances in Neural Information Processing Systems*.
- Gneiting, T. (1998). Simple tests for the validity of correlation function models on the circle. *Statistics & probability letters*, 39(2):119–122.
- Gneiting, T. (2013). Strictly and non-strictly positive definite functions on spheres. *Bernoulli*, 19(4):1327–1349.
- Gradshteyn, I. S. and Ryzhik, I. M. (2014). *Table of Integrals, Series, and Products*. Academic Press, 7 edition.
- Gupta, S., Nayak, A., and Billard, A. (2022). Learning high dimensional demonstrations using laplacian eigenmaps. *arXiv preprint arXiv:2207.08714*.
- Hein, M. and Audibert, J.-Y. (2007). Graph Laplacians and their Convergence on Random Neighborhood Graphs. *Journal of Machine Learning Research*, 8:1325–1368.
- Heinzmann, J. and Zelinsky, A. (2003). Quantitative safety guarantees for physical human-robot interaction. *The International Journal of Robotics Research*, 22(7-8):479–504.
- Hennig, P., Osborne, M. A., and Girolami, M. (2015). Probabilistic numerics and uncertainty in computations. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 471(2179):20150142.
- Hensman, J., Durrande, N., and Solin, A. (2018). Variational fourier features for gaussian processes. *Journal of Machine Learning Research*, 18(151):1–52.
- Horn, R. A. and Johnson, C. R. (2012). *Matrix analysis*. Cambridge university press.
- Huang, J., Wu, F., Precup, D., and Cai, Y. (2018). Learning Safe Policies with Expert Guidance. In *Advances in neural information processing systems*.
- Huber, L., Billard, A., and Slotine, J.-J. (2019). Avoidance of convex and concave obstacles with convergence ensured through contraction. *IEEE Robotics and Automation Letters*, 4(2):1462–1469.
- Hutchinson, M. F. (1989). A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076.
- Jaquier, N., Borovitskiy, V., Smolensky, A., Terenin, A., Asfour, T., and Rozo, L. (2022). Geometry-aware bayesian optimization in robotics using riemannian matern kernels. In *Conference on Robot Learning*, pages 794–805. PMLR.

Bibliography

- Jaquier, N., Rozo, L., Calinon, S., and Bürger, M. (2020). Bayesian optimization meets riemannian manifolds in robot learning. In *Conference on Robot Learning*, pages 233–246. PMLR.
- Jayasumana, S., Hartley, R., Salzmann, M., Li, H., and Harandi, M. (2013). Kernel Methods on the Riemannian Manifold of Symmetric Positive Definite Matrices. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 73–80.
- Jayasumana, S., Hartley, R., Salzmann, M., Li, H., and Harandi, M. (2015). Kernel Methods on Riemannian Manifolds with Gaussian RBF Kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(12):2464–2477.
- Johnson, J., Douze, M., and Jégou, H. (2019). Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.
- Joshi, S. C. and Miller, M. I. (2000). Landmark matching via large deformation diffeomorphisms. *IEEE transactions on image processing*, 9(8):1357–1370.
- Khansari-Zadeh, S. M. and Billard, A. (2011). Learning Stable Nonlinear Dynamical Systems With Gaussian Mixture Models. *IEEE Transactions on Robotics*, 27(5):943–957.
- Khansari-Zadeh, S. M. and Billard, A. (2012). A dynamical system approach to realtime obstacle avoidance. *Autonomous Robots*, 32:433–454.
- Kondor, R. I. and Lafferty, J. (2002). Diffusion kernels on graphs and other discrete structures. In *International Conference on Machine Learning*.
- Kronander, K. and Billard, A. (2016). Passive Interaction Control With Dynamical Systems. *IEEE Robotics and Automation Letters*, 1(1):106–113.
- Lee, J. A. and Verleysen, M. (2007). *Nonlinear Dimensionality Reduction*. Springer Science & Business Media.
- Lindgren, F., Rue, H., and Lindström, J. (2011). An explicit link between Gaussian fields and Gaussian Markov random fields: The stochastic partial differential equation approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(4):423–498.
- Lohmiller, W. and Slotine, J.-J. E. (1998). On Contraction Analysis for Non-linear Systems. *Automatica*, 34(6):683–696.
- Ma, Y. and Fu, Y. (2011). *Manifold Learning Theory and Applications*. CRC press.
- Manschitz, S., Gienger, M., Kober, J., and Peters, J. (2018). Mixture of Attractors: A Novel Movement Primitive Representation for Learning Motor Skills From Demonstrations. *IEEE Robotics and Automation Letters*, 3(2):926–933.
- Medina, J. R. and Billard, A. (2017). Learning Stable Task Sequences from Demonstration with Linear Parameter Varying Systems and Hidden Markov Models. In *Conference on Robot Learning*, pages 175–184.

- Meurant, G. (2006). *The Lanczos and conjugate gradient algorithms: from theory to finite precision computations*. SIAM.
- Mirrazavi Salehian, S. S. (2018). Compliant control of uni/multi-robotic arms with dynamical systems. Technical report, EPFL.
- Mohammad Khansari-Zadeh, S. and Billard, A. (2014). Learning control Lyapunov function to ensure stability of dynamical system-based robot reaching motions. *Robotics and Autonomous Systems*, 62(6):752–765.
- Mukadam, M., Cheng, C.-A., Fox, D., Boots, B., and Ratliff, N. (2020). Riemannian motion policy fusion through learnable Lyapunov function reshaping. In *Conference on robot learning*, pages 204–219. PMLR.
- Nadler, B., Lafon, S., Coifman, R. R., and Kevrekidis, I. G. (2006a). Diffusion maps, spectral clustering and reaction coordinates of dynamical systems. *Applied and Computational Harmonic Analysis*, 21(1):113–127.
- Nadler, B., Lafon, S., Kevrekidis, I., and Coifman, R. R. (2006b). Diffusion Maps, Spectral Clustering and Eigenfunctions of Fokker-Planck Operators. In *Advances in Neural Information Processing Systems*, volume 18, pages 955–962. MIT Press.
- Nadler, B., Srebro, N., and Zhou, X. (2009). Statistical analysis of semi-supervised learning: The limit of infinite unlabelled data. In *Advances in neural information processing systems*, pages 1330–1338.
- Nash, J. (1956). The imbedding problem for Riemannian manifolds. *Annals of Mathematics*, pages 20–63.
- Neumann, K. and Steil, J. J. (2015). Learning robot motions with stable dynamical systems under diffeomorphic transformations. *Robotics and Autonomous Systems*, 70:1–15.
- Pavutnitskiy, F., Ivanov, S. O., Abramov, E., Borovitskiy, V., Klochkov, A., Vyalov, V., Zaikovskii, A., and Petiushko, A. (2022). Quadric hypersurface intersection for manifold learning in feature space. In *International Conference on Artificial Intelligence and Statistics*.
- Perrin, N. and Schlehuber-Caissier, P. (2016). Fast diffeomorphic matching to learn globally asymptotically stable nonlinear dynamical systems. *Systems & Control Letters*, 96:51–59.
- Rahimi, A. and Recht, B. (2007a). Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20:1177–1184.
- Rahimi, A. and Recht, B. (2007b). Random Features for Large-Scale Kernel Machines. In *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc.
- Rahimi, A. and Recht, B. (2007c). Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*.

Bibliography

- Rai, A., Antonova, R., Song, S., Martin, W., Geyer, H., and Atkeson, C. (2018). Bayesian optimization using domain knowledge on the atrias biped. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1771–1778. IEEE.
- Rana, M., Li, A., Ravichandar, H., Mukadam, M., Chernova, S., Fox, D., Boots, B., and Ratliff, N. (2019). Learning Reactive Motion Policies in Multiple Task Spaces from Human Demonstrations. In *Conference: Conference on Robot Learning (CoRL)*.
- Rana, M. A., Li, A., Fox, D., Boots, B., Ramos, F., and Ratliff, N. (2020). Euclideanizing Flows: Diffeomorphic Reduction for Learning Stable Dynamical Systems. In *Learning for Dynamics and Control*, pages 630–639. PMLR.
- Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press.
- Ratliff, N. D., Issac, J., Kappler, D., Birchfield, S., and Fox, D. (2018). Riemannian Motion Policies. *arXiv preprint arXiv:1801.02854*.
- Ratliff, N. D., Van Wyk, K., Xie, M., Li, A., and Rana, M. A. (2021). Generalized Nonlinear and Finsler Geometry for Robotics. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10206–10212.
- Ravichandar, H., Salehi, I., and Dani, A. (2017). Learning Partially Contracting Dynamical Systems from Demonstrations. In *Conference on Robot Learning*, pages 369–378.
- Robert-Nicoud, D., Krause, A., and Borovitskiy, V. (2024). Intrinsic gaussian vector fields on manifolds. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*.
- Rosen, D. M., Doherty, K. J., Terán Espinoza, A., and Leonard, J. J. (2021). Advances in inference and representation for simultaneous localization and mapping. *Annual Review of Control, Robotics, and Autonomous Systems*, 4:215–242.
- Salvador, S. and Chan, P. (2007). Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580.
- Schölkopf, B. and Smola, A. J. (2002). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Adaptive Computation and Machine Learning. MIT Press.
- Shnitzer, T., Talmon, R., and Slotine, J.-J. (2020). Diffusion maps kalman filter for a class of systems with gradient flows. *IEEE Transactions on Signal Processing*, 68:2739–2753.
- Shukla, A. and Billard, A. (2012). Augmented-SVM: Automatic space partitioning for combining multiple non-linear dynamics. In *Advances in Neural Information Processing Systems 25*, pages 1016–1024. Curran Associates, Inc.
- Simpson-Porco, J. W. and Bullo, F. (2014). Contraction theory on Riemannian manifolds. *Systems & Control Letters*, 65:74–80.

- Sindhwani, V., Chu, W., and Keerthi, S. S. (2007). Semi-supervised gaussian process classifiers. In *IJCAI*, pages 1059–1064.
- Sindhwani, V., Tu, S., and Khansari, M. (2018). Learning Contracting Vector Fields For Stable Imitation Learning. *arXiv preprint arXiv:1804.04878*.
- Smola, A. J. and Kondor, R. (2003). Kernels and regularization on graphs. In *Learning theory and kernel machines*, pages 144–158. Springer.
- Solà, J., Deray, J., and Atchuthan, D. (2021). A micro Lie theory for state estimation in robotics. (arXiv preprint arXiv:1812.01537).
- Steinke, F. and Schölkopf, B. (2008). Kernels, regularization and differential equations. *Pattern Recognition*, 41(11):3271–3286.
- Sulam, J., Romano, Y., and Talmon, R. (2017). Dynamical system classification with diffusion embedding for ecg-based person identification. *Signal Processing*, 130:403–411.
- Tee, G. J. (2007). Eigenvectors of block circulant and alternating circulant matrices. *New Zealand Journal of Mathematics*, 36(8):195–211.
- Tenenbaum, J. B., de Silva, V., and Langford, J. C. (2000). A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500):2319–2323.
- Wabersich, K. P. and Zeilinger, M. N. (2021). A predictive safety filter for learning-based control of constrained nonlinear dynamical systems. *Automatica*, 129:109597.
- Watterson, M., Liu, S., Sun, K., Smith, T., and Kumar, V. (2018). Trajectory Optimization On Manifolds with Applications to SO(3) and R3XS2. In *Robotics: Science and Systems XIV*, volume 14.
- Wensing, P. M. and Slotine, J.-J. (2020). Beyond convexity—contraction and global convergence of gradient descent. *Plos one*, 15(8):e0236661.
- Whittle, P. (1963a). Stochastic-processes in several dimensions. *Bulletin of the International Statistical Institute*, 40:974–994.
- Whittle, P. (1963b). Stochastic processes in several dimensions. *Bulletin of the International Statistical Institute*, 40:974–994.
- Williams, G., Aldrich, A., and Theodorou, E. A. (2017). Model Predictive Path Integral Control: From Theory to Parallel Computation. *Journal of Guidance, Control, and Dynamics*, 40(2):344–357.
- Wilson, J. T., Borovitskiy, V., Terenin, A., Mostowsky, P., and Deisenroth, M. P. (2020). Efficiently Sampling Functions from Gaussian Process Posteriors. In *International Conference on Machine Learning*.

Bibliography

- Wilson, J. T., Borovitskiy, V., Terenin, A., Mostowsky, P., and Deisenroth, M. P. (2021). Pathwise conditioning of gaussian processes. *The Journal of Machine Learning Research*, 22(1):4741–4787.
- Xie, M., Van Wyk, K., Li, A., Rana, M. A., Wan, Q., Fox, D., Boots, B., and Ratliff, N. (2021). Geometric Fabrics for the Acceleration-based Design of Robotic Motion. *arXiv preprint arXiv:2010.14750*.
- Zeestraten, M. J. A., Havoutis, I., Silvério, J., Calinon, S., and Caldwell, D. G. (2017). An Approach for Imitation Learning on Riemannian Manifolds. *IEEE Robotics and Automation Letters*, 2(3):1240–1247.
- Zhou, X. and Belkin, M. (2011). Semi-supervised learning by higher order regularization. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 892–900.
- Zhu, X., Lafferty, J., and Ghahramani, Z. (2003). *Semi-supervised learning: From Gaussian fields to Gaussian processes*. School of Computer Science, Carnegie Mellon University.

Bernardo Fichera

bernardo.fichera@epfl.ch
<https://bernardofichera.com>

EDUCATION

Ph.D. in Robotics, Control, and Intelligent Systems 2024
École Polytechnique Fédérale de Lausanne
Thesis: *Geometric Learning: Leveraging Differential Geometry for Learning and Control*
Supervised by Prof. A. Billard

M.Sc. in Aeronautical Engineering 2016
Politecnico di Milano
Thesis: *Spiking Neural Network Controller For Flight Stabilization Of The Harvard Microrobotic Bee*
Supervised by Prof. M. Lovera and Prof. S. Ferrari

B.Sc. in Aerospace Engineering 2012
Politecnico di Milano

EXPERIENCE

Research Assistant 2018
École Polytechnique Fédérale de Lausanne

Intern 2017
École Polytechnique Fédérale de Lausanne

Visiting Student 2016
Cornell University

SKILLS

Research Interests

diffusion processes, differential geometry, Bayesian optimization, Gaussian processes, online learning, reinforcement learning, optimal control

Programming

C++ (advanced), Python (advanced), MATLAB (good), C (good)
<https://github.com/nash169>

Languages

English (fluent), French (basic), Italian (native)

PUBLICATIONS

B. Fichera, V. Borovitskiy, A. Krause, A. Billard. Implicit Manifold Gaussian Process Regression. In *Neural Information Processing Systems (NeurIPS)*, 2023.

B. Fichera, A. Billard. Hybrid Quadratic Programming-Pullback Bundle Dynamical Systems Control. In *International Symposium of Robotics Research (ISRR)*, 2022.

B. Fichera, A. Billard. Linearization and Identification of Multiple Attractors Dynamical Systems through Laplacian Eigenmaps. In *Journal of Machine Learning Research (JMLR)*, 2022.

K. Chatzilygeroudis, **B. Fichera**, I. Lauzana, F. Bu, K. Yao, F. Khadivar, A. Billard. Benchmark for bimanual robotic manipulation of semi-deformable objects. In *IEEE Robotics and Automation Letters (RAL)*, 2020.

K. Yao, **B. Fichera**, A. B. Haget, I. Lauzana, A. Billard. Integrating Multisensory Information for Modeling Human Dexterous Bimanual Manipulation Skills. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

PREPRINTS

B. Fichera, A. Billard. Learning Dynamical Systems Encoding Non-Linearity within Space Curvature. *arXiv:2403.11948*, 2024. Under Review in *International Journal of Robotics Research (IJRR)*.

N. Jaquier, M. C. Welle, A. Gams, K. Yao, **B. Fichera**, A. Billard, A. Ude, T. Asfour, D. Kragić. Transfer Learning in Robotics: An Upcoming Breakthrough? A Review of Promises and Challenges. *arXiv:2311.18044*, 2024. Accepted in *International Journal of Robotics Research (IJRR)*.

TEACHING

Teaching Assistant 2018—2023
École Polytechnique Fédérale de Lausanne
Courses taught: Applied Machine Learning (200+ students), Advanced Machine Learning (50+ students)
Leading TA and member of the exam preparation team.

SUPERVISION

Master's Thesis 2024
École Polytechnique Fédérale de Lausanne
B. Bosc. *Configuration Space Obstacle Avoidance with Geometrical Dynamical Systems*.

Master's Thesis 2024
 École Polytechnique Fédérale de Lausanne
 K. Vaudaux. *Distance Geometric Inverse Kinematics for Motion Planing.*

Semester Project 2022
 École Polytechnique Fédérale de Lausanne
 A. Guntli. *Optimal Tracking of Geometrical Dynamical Systems.*

Semester Project 2022
 École Polytechnique Fédérale de Lausanne
 R. Uebersax. *Incrementally Learning and Exploiting Inverse Dynamics.*

Master's Thesis 2020
 École Polytechnique Fédérale de Lausanne
 G. Coppola. *Planar target tracking in a moving multi-obstacles scenario.*

SERVICE

Reviewer
 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)

FUNDING

Fellowship from The College of Engineering (\approx \$30k) 2015
 Cornell University