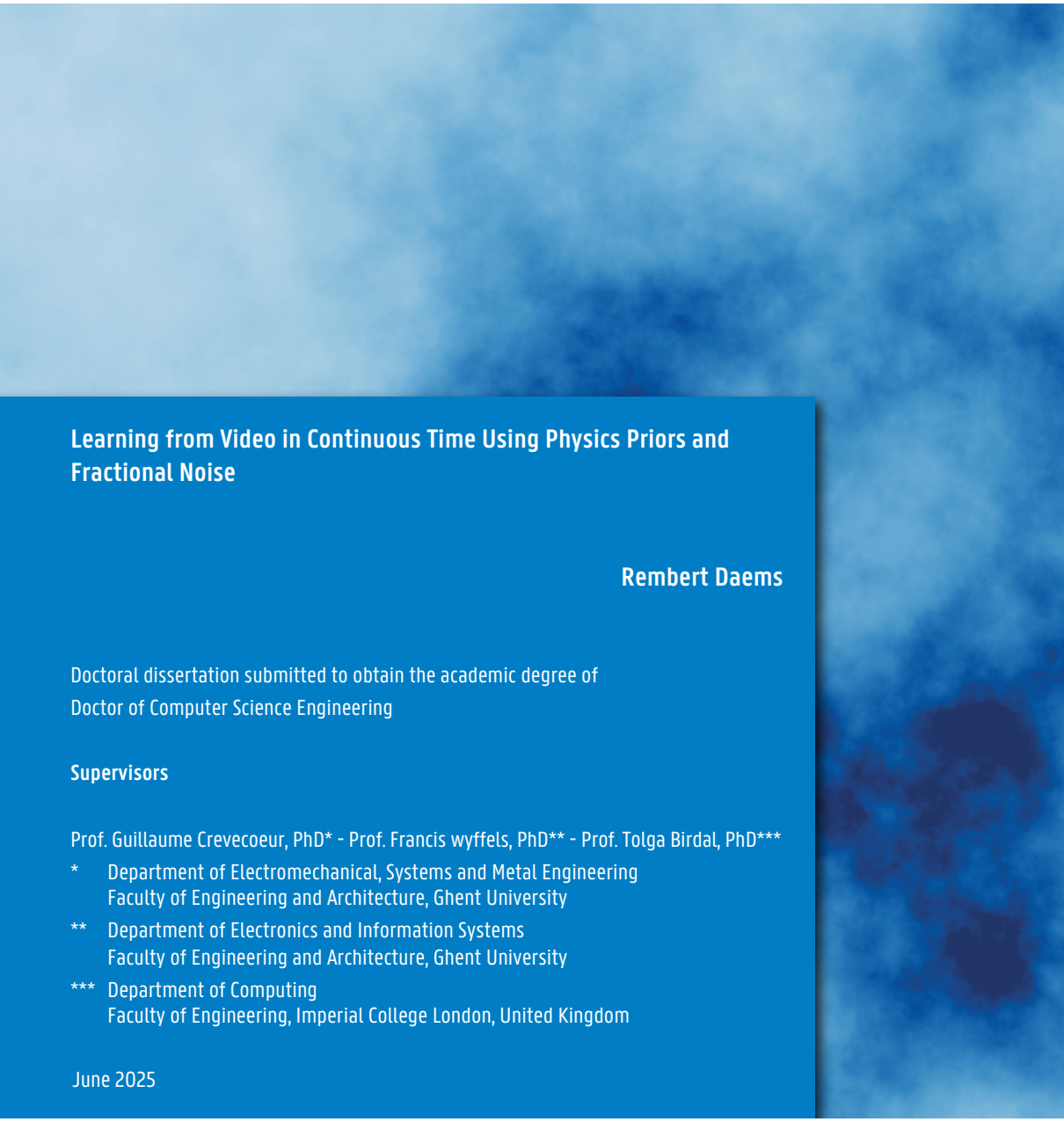
FACULTY OF ENGINEERING AND ARCHITECTURE



Learning from Video in Continuous Time Using Physics Priors and Fractional Noise

Rembert Daems

Doctoral dissertation submitted to obtain the academic degree of
Doctor of Computer Science Engineering

Supervisors

Prof. Guillaume Crevecoeur, PhD* - Prof. Francis wyffels, PhD** - Prof. Tolga Birdal, PhD***

* Department of Electromechanical, Systems and Metal Engineering
Faculty of Engineering and Architecture, Ghent University

** Department of Electronics and Information Systems
Faculty of Engineering and Architecture, Ghent University

*** Department of Computing
Faculty of Engineering, Imperial College London, United Kingdom

June 2025

GHENT UNIVERSITY

FACULTY OF ENGINEERING AND ARCHITECTURE

Learning from Video in Continuous Time Using Physics Priors and Fractional Noise

Rembert Daems

Doctoral dissertation submitted to obtain the academic degree of
Doctor of Computer Science Engineering

Supervisors

Prof. Guillaume Crevecoeur, PhD* - Prof. Francis wyffels, PhD** - Prof. Tolga Birdal, PhD***

* Department of Electromechanical, Systems and Metal Engineering
Faculty of Engineering and Architecture, Ghent University

** Department of Electronics and Information Systems
Faculty of Engineering and Architecture, Ghent University

*** Department of Computing
Faculty of Engineering, Imperial College London, United Kingdom

June 2025

GHENT UNIVERSITY

ISBN 978-94-6355-992-8
NUR 984
Wettelijk depot: D/2025/10.500/52

Members of the Examination Board

Chair

Prof. Em. Hendrik Van Landeghem, PhD, Ghent University

Other members entitled to vote

Jonas Degrave, PhD, Deepmind, United Kingdom
Prof. Bart Dhoedt, PhD, Ghent University
Tom Lefebvre, PhD, Ghent University
Prof. Manfred Opper, PhD, Technische Universität Berlin, Germany

Supervisors

Prof. Guillaume Crevecoeur, PhD, Ghent University
Prof. Francis wyffels, PhD, Ghent University
Prof. Tolga Birdal, PhD, Imperial College London, United Kingdom

We do these things not
because they are easy,
but because we thought
they were going to be easy.

— Maciej Cegłowski

Acknowledgements

This endeavor has come to an end, not in the least thanks to the many people that where involved, some directly, others tangentially, or rather serendipitously, maybe even without realizing their profound influences on my work. First and foremost, I thank my supervisors. Francis, thank you for being there from the start, always available and up for some tinkering and pondering. Guillaume, thank you for helping me and providing me with much needed peace of mind and the comfort of the fantastic D2Lab. My *physically remote*, albeit all the more *academically close* advisor Tolga, thank you for your guidance and continuous trust and support. I would like to especially highlight the astonishing amounts of intellectual freedom I have so deeply enjoyed at all times under all of your guidances. I sincerely thank the members of the jury for their efforts. The many thoughtful comments, questions and discussions were well appreciated and improved the quality of this work.

It is appropriate to begin at the very start of my academic journey. The first seeds where undoubtedly planted during my master thesis under the guidance of Matthias. I will always cherish these memories where I could first taste the unquenchable thirst of ultimate creativity, freedom and unparalleled intellectual growth. My next professional home was in the innovation team of Bart, at CNH Industrial. I was very lucky to have a 'boss' who above all did not really want to control my day-to-day activities, but rather put absolute trust in me and let me learn at an incredible speed. Thanks Bart for all the faith and trust in my abilities,

only later (because I had no comparison) did I fully realize how precious this was. Here my love for deep learning (what we call artificial intelligence now) took form and shape, never to leave again. Thanks to Rein for introducing me to this wonderful new world. My intellectual hunger seemed to reach the bounds of this multinational corporate environment and I thought I would be better off at the other extreme of the spectrum, at a very risky and even more ambitious start-up. I learned a lot about robots, linux and the critical need of having enough funding to build a good product. It was in this period I was finally able to start my PhD, at the IDLab-AIRO group. I have fond memories of this vibrant start to which I had looked forward for a long time.

I was fortunate enough to join this group when still one member from its illustrious ResLab times was present, though she mostly was away in London, visiting much more prestigious groups. When I finally had the chance to meet Ira in person, I immediately went to her desk to explain a new idea I was working on. Before long, I was interrupted–*Oh, but that's been done already.* Pointing me towards the relevant literature and crushing my naive hopes for a spectacular NIPS paper. Thanks Ira for guiding me from time to time while expecting nothing in return, and for getting me in touch with Jonas. Thank you Jonas for the many, many informal video chats we did over the years. It was always very elucidating to talk to you on a broad range of topics, and hearing your intuitions and advice on my research ideas. You were actually the first to draw my attention to generative diffusion models and their connection to SDEs.

Thanks Gabriel, Alexander, Matthias, Natacha, Olivier, Luthffi, Pieter (keep fighting the system!), Qiaoqiao, Tom, Mathieu, Matthijs, Andreas, Dries, Maria, Remco, Joni, Tony, and many others who I sadly did not really get to know very well due to my diminishing physical presence in the lab over time. Special thanks to the illustrious co-members of the *Keypoint Gang*: Peter, Maxim, Victor-Louis, Thomas and Jarne. Thank you for the very inspirational reading group sessions, hackathons, company visits, dinners and drinks. It has been, and still is, quite the honor.

I thank Wouter, Neel, Dieter, Didier and Sven for the pleasant

working environment and fun times in the *Roeselare* office. Thanks Jan for being my *industrial* advisor, I have fond memories of our many conversations where you tried to educate me in *classical* control and modeling. Sadly, this first part of my PhD did not work out, and I am very grateful to Guillaume to support me into the second phase of my PhD, and allowing me the freedom to work on research topics aligned with my interests.

In the midst of COVID lockdowns, I became a member of the D2Lab. Which felt a bit like coming home because I had some friends there from long-forgotten times. Thank you Frederik, for the friendship, academic advice and the weekly lunches during which I hope we can keep exchanging our dystopian ideas on the future of Europe's economy. Sander, thank you for the companionship on the train, bike and car rides between the two most beautiful cities in Western Europe, and for taking care of my kids one evening when the babysit had canceled. Thanks Tom S., Wannes, Tom L., Jolan, Victor and Cedric, Jeroen, Rikkert, Thomas N., Omid, Hendrik, Lisa, Thijs, Thomas D. C., Ahmed, Babak, Anatolli, Yentl, Robbie, Annelies, Mohammad, Nezmin, Joannes, Daan and Louis for many fun interactions over lunch, and the quiz, dinner and mario kart evenings or nights. Thank you Marilyn, Kathleen, Katrien, Vincent and Nick for the excellent technical and administrative support. Also thank you to Frederik and Wouter, my co-founders of the *Denktank Vlaamse Maakindustrie* for the inspirational technical sessions we did, as representatives (at the time) of three major Flemish manufacturing companies.

When the COVID restrictions where sufficiently lifted, I could finally go to an international conference: ECCV (2022) in Tel Aviv. Together with Peter, we had a great time getting to know a lot of very talented people. We presented our work at the *Recovering 6D Object Pose* workshop, where also a keynote presentation was given by Tolga Birdal. I remember hesitating to approach Tolga at the subsequent poster session, because I did not really have a concrete question but wanted to discuss the uses of stochastic differential equations in the works he presented. After only a few minutes, Tolga offered to collaborate on some of his ideas, which led to the work I am most proud of in this book. It quickly

became clear, we could not solve the problems we wanted to solve on our own. I just sent a quick email to Manfred, which was replied with a lengthy, very informative read. Manfred joined our weekly video calls and the three of us matched in a remarkably complementary way, not only in terms of our academic interests, but also on a personal level. I will forever feel blessed to have been able to experience this level of collaboration. Manfred, thanks for teaching me on stochastic calculus and many other topics. I greatly enjoyed working on the math together and my short visit to Berlin.

At NeurIPS (2023) in New Orleans, I met Gabriel, who happened to use the same approximation for fractional Brownian motion, for a different application. We ended up talking and literally collaborating in some quiet spot at the conference venue. This lead to a wonderful result at NeurIPS the next year, and our collaborations are still ongoing. Thank you Gabriel for the opportunities and involving me in your fantastic ideas and projects.

Thanks Raphaël for the nice collaboration on ODEs and SDEs for graphs. I also enjoyed collaborating with Simon and Axel, and later on Sebastiaan and David, at ILVO. It was always inspiring to see how you combine your versatile skills and experience into creative ideas for agricultural applications. Also thanks to Pieter and Marco at ORSI for the likewise very inspiring environment, where I briefly could help with some minor advice.

During my PhD I had the opportunity (thanks to Peter) to join a venture which would become Hippo Dx. I have very fond memories of tinkering on the first prototypes with Wim, Peter and Senne for this crazy idea where we automate skin prick allergy tests. Later on, I very much enjoyed working on the well-defined AI part of this project. Doing more applied stuff helped me to keep my sanity throughout the academic work of my PhD. Thank you Neel, Siebe, Rob, Sven, Dirk, Bert and the many others that grew this idea to a robust and successful company.

The attentive reader might have perceived that I have mentioned four Peters so far. While this is a relatively common name, all four are actually the same person! Peter, I am very grateful for sharing much of this PhD experience, as a true *compagnon de*

route you have been much more than a collaborator,–a trusted advisor, a friend. And I am very proud of the *CenderNet* work we did together.

I also want to thank Thomas for supporting the next steps in my academic career working on the exciting topic of machine learning for protein design. Thanks for the valuable guidance and already letting me join the project *before* my PhD was finished.

While growing up, there have been many people that have helped me reach this academic achievement. By being an example, believing in me, showing what is possible,–thank you *papa* and *mama*, my older brothers and sisters, my friends and teachers. I would also like to thank the Flemish taxpayer for supporting me–at least partially–through all of these career steps.

Over the years, I have undoubtedly complained in an incessant way about the–at times–seemingly impossible combination of having three children and doing a PhD. I want to sincerely apologize and thank all of you who have listened and encouraged me through this ordeal. I can honestly say, I have always felt very supported on this by anyone I have collaborated with. This was shown in small things like allowing my kids to interrupt video calls to say hi or show they can count to 10 in English. But just as well in the big picture, where I never felt any undue pressure and the utmost empathy for organizing and planning my work and life. Thank you Guillaume for the trust and flexibility.

Thank you Karel, Rozanne and Korneel for helping me understand the intricacies of life and showing me what is most important. Thank you for allowing me to gain perspective on my ambitions and learning me to work in a focused and much more productive way.

Life has not been what it is, without the love of my life. Berdieke, you give me meaning in the most real possible way. Thank you for the unconditional support and for being everything to me.

Rembert Daems, June 18, 2025

Summary

Cameras are ubiquitous in our daily lives, and the amount of video data generated is staggering. The ability to analyze and understand video content has far-reaching implications for various applications, including robotics, autonomous driving and generative art. A camera is a rich information source that, depending on the application, can be cheaper and more informative than a multitude of other sensors. For instance, in a robotic application, information about the environment, the robot's position, and the objects it interacts with can all be provided by a single camera.

Since around 2012, tremendous progress has been made in the field of computer vision. The introduction of deep learning has revolutionized the way we approach *e.g.* image classification, object detection and segmentation tasks. The development of powerful ideas such as convolutional neural networks (CNNs) and diffusion models have tremendously pushed the boundaries of what is possible in computer vision. The advent of large-scale datasets and the availability of pre-trained models have further accelerated research in this area. It has become relatively easy to solve certain practical computer vision problems using deep learning methods, compared to a decade ago. For many companies the main problem is not finding a good technical solution, but rather implementing and integrating this solution, or even more mundane, finding, labeling and cleaning the right data. This means, from an academic perspective, research directions have shifted to more difficult tasks, requiring more data, more complex models and more compute. In this context, we focus the

research in this thesis on methodological advances, rather than on scaling up data, models or compute.

In this thesis, we explore a number of ways to enhance the capabilities of models for video understanding. Learning models from video entails learning the underlying dynamics of a system from its visual representation. An accurate model is capable of predicting future video frames, given a sequence of past frames. This is useful for many applications, such as robotics, where the model can be used to control a robot based on its visual input.

One fundamental question is how we choose to approach the time aspect of these models. We can either choose a fixed discretization of time (*discretize-then-optimize*), or we can choose to model the dynamics in continuous time (*optimize-then-discretize*). The first option entails defining a *fixed grid* of time points and limiting our models to only operate on these predefined time points. The second option allows us to define models that are *continuous* in time, *i.e.*, we can use them at any time point, not just on a fixed grid of time points. Since nature is continuous, it makes sense to model the dynamics in continuous time. This is the approach we take in this thesis.

We identify three main research goals in this thesis, outlined below.

Physics Priors. Physics priors are a way to incorporate prior knowledge about the underlying dynamics of a system into the learning process. Specifically, we use Lagrangian dynamics to model the underlying dynamics of a system. This entails modeling the system explicitly using its potential energy and mass matrix. In this thesis we present KeyCLD, a framework to learn Lagrangian dynamics from images. Learned keypoints represent semantic landmarks in images and can directly represent state dynamics. We show that interpreting this state as Cartesian coordinates, coupled with explicit holonomic constraints, allows expressing the dynamics with a constrained Lagrangian. KeyCLD is trained unsupervised end-to-end on sequences of images. Our method explicitly models the mass matrix, potential energy and the input matrix, thus allowing energy based control. We demonstrate learning of Lagrangian dynamics from images on

the dm_control pendulum, cartpole and acrobot environments. KeyCLD can be learned on these systems, whether they are unactuated, underactuated or fully actuated. Trained models are able to produce long-term video predictions, showing that the dynamics are accurately learned. We compare with other recent methods from the literature, and investigate the benefit of the Lagrangian prior and the constraint function. KeyCLD achieves the highest valid prediction time on all benchmarks. Additionally, a very straightforward energy shaping controller is successfully applied on the fully actuated systems.

Long-range Stochastic Dynamics. The preceding paragraph considers physics priors in a deterministic manner. We also take a step further towards learning *stochastic* dynamical models from video. In this thesis we present a novel variational framework for performing inference in (neural) stochastic differential equations (SDEs) driven by Markov-approximate fractional Brownian motion (fBM). SDEs offer a versatile tool for modeling real-world continuous-time dynamic systems with inherent noise and randomness. Combining SDEs with the powerful inference capabilities of variational methods, enables the learning of representative distributions through stochastic gradient descent. However, conventional SDEs typically assume the underlying noise to follow a Brownian motion (BM), which hinders their ability to capture long-term dependencies. In contrast, fractional Brownian motion (fBM) extends BM to encompass non-Markovian dynamics, but existing methods for inferring fBM parameters are either computationally demanding or statistically inefficient. By building upon the Markov approximation of fBM, we derive the evidence lower bound essential for efficient variational inference of posterior path measures, drawing from the well-established field of stochastic analysis. Additionally, we provide a closed-form expression for optimal approximation coefficients and propose to use neural networks to learn the drift, diffusion and control terms within our variational posterior, leading to the variational training of neural-SDEs. In this framework, we also optimize the Hurst index, governing the nature of our fractional noise. Beyond validation on synthetic data, we contribute a novel architecture for variational latent video prediction,—an

approach that, to the best of our knowledge, enables the first variational neural-SDE application to video perception.

Efficient Learning of SDEs. Learning stochastic differential equations (SDEs) from data is computationally challenging. Due to the iterative nature of maximizing the evidence lower bound, which requires solving the SDE for each iteration, the training of neural SDEs is generally slow and unstable. How can we more efficiently learn SDEs from data? We present a hierarchical, control theory inspired method for variational inference for neural SDEs. In this chapter, we propose to decompose the control term into linear and residual non-linear components and derive an optimal control term for linear SDEs, using stochastic optimal control. Modeling the non-linear component by a neural network, we show how to efficiently train neural SDEs without sacrificing their expressive power. Since the linear part of the control term is optimal and does not need to be learned, the training is initialized at a lower cost and we observe faster convergence.

This dissertation answers the above three main goals to ultimately work towards *learning from video in continuous-time using physics priors and fractional noise*. Our research demonstrates that it is possible to simultaneously learn both Lagrangian dynamics and state estimator models from video in one end-to-end process. Secondly, our advances in variational inference for stochastic processes driven by fractional noise open new avenues for modeling complex temporal phenomena with long-range dependencies. Lastly, we propose a method inspired from optimal control theory to more efficiently learn SDEs from data.

Samenvatting

Camera's zijn alomtegenwoordig in ons dagelijks leven en de hoeveelheid gegenereerde videogegevens is enorm. De mogelijkheid om video-inhoud te analyseren en te interpreteren heeft verstrekkende gevolgen voor een veelheid aan toepassingen, zoals robotica, autonoom rijden en generatieve kunst. Een camera is een rijke informatiebron die, afhankelijk van de toepassing, goedkoper en informatiever kan zijn dan meerdere andere sensoren. In een roboticatoepassing bijvoorbeeld kan informatie over de omgeving, de positie van de robot en de objecten waarmee de robot interageert allemaal worden aangeleverd door één enkele camera.

Sinds ongeveer 2012 is er een enorme vooruitgang geboekt in het domein van computervisie. De introductie van *deep learning* heeft een revolutie teweeggebracht in de manier waarop we bijvoorbeeld beeldclassificatie, objectdetectie en segmentatietaken benaderen. De ontwikkeling van krachtige ideeën zoals convolutionele neurale netwerken en diffusiemodellen hebben de grenzen van wat mogelijk is in computervisie enorm uitgebreid. De komst van grootschalige hoeveelheden data en de beschikbaarheid van voorgetrainde modellen hebben het onderzoek in dit domein verder versneld. Het is relatief eenvoudig geworden om bepaalde praktische computervisie problemen op te lossen met behulp van diepe leermethoden, vergeleken met een decennium geleden. Voor veel bedrijven is het belangrijkste probleem niet om een goede technische oplossing te vinden, maar eerder om die oplossing te implementeren en te integreren, of zelfs meer

alledaags, om de meest bruikbare data te vinden, te annoteren en op te schonen. Vanuit academisch perspectief betekent dit dat de onderzoeksrichtingen zijn verschoven naar moeilijkere taken, waarvoor meer data, meer complexe modellen en meer rekenkracht nodig zijn. In deze context richten we het onderzoek in dit proefschrift op methodologische vooruitgang, in plaats van op het opschalen van data, modellen of rekenkracht.

In dit proefschrift verkennen we een aantal manieren om de capaciteit van modellen voor het begrijpen van video te verbeteren. Modellen leren van video houdt in dat je de onderliggende dynamiek van een systeem leert vanaf haar visuele representatie. Een nauwkeurig model is in staat om toekomstige videoframes te voorspellen, gegeven een reeks voorgaande frames. Dat is nuttig voor veel toepassingen, zoals robotica, waar het model kan worden gebruikt om een robot te besturen op basis van de visuele input.

Een fundamentele vraag is hoe we het aspect tijd bij deze modellen benaderen. We kunnen kiezen voor een vaste discretisatie van de tijd (*eerst discretiseren, dan optimaliseren*), of we kunnen ervoor kiezen om de dynamica in continue tijd te modelleren (*eerst optimaliseren, dan discretiseren*). De eerste optie houdt in dat we een vast raster van tijdpunten definiëren en onze modellen beperken tot deze vooraf gedefinieerde tijdpunten. De tweede optie stelt ons in staat om modellen te definiëren die continu zijn in de tijd, dat wil zeggen dat we ze op elk tijdstip kunnen gebruiken, niet alleen op een vast raster van tijdspunten. Aangezien de natuur continu is, is het zinvol om de dynamica in continue tijd te modelleren. Dat is de benadering die we in dit proefschrift hanteren.

We identificeren drie belangrijke onderzoeksdoelen in dit proefschrift, die hieronder worden beschreven.

Fysische voorkennis. Het gebruik van fysische voorkennis is een manier om eerdere kennis over de onderliggende dynamica van een systeem op te nemen in het leerproces. Specifiek gebruiken we Lagrangiaanse dynamica om de onderliggende dynamica van een systeem te modelleren. Dat houdt in dat het systeem expliciet gemodelleerd wordt met behulp van zijn potentiële energie

en massa matrix. In dit proefschrift presenteren we KeyCLD, een raamwerk om Lagrangiaanse dynamica vanaf afbeeldingen te leren. Geleerde sleutelpunten vertegenwoordigen semantische herkenningspunten in afbeeldingen en kunnen direct de toestand van de dynamica weergeven. We tonen aan dat het interpreteren van de toestand als Cartesische coördinaten, gekoppeld aan expliciete holonome nevenvoorwaarden, het mogelijk maakt om de dynamica uit te drukken met een beperkte Lagrangiaan. KeyCLD wordt zonder supervisie getraind op beeldsequenties. Onze methode modelleert expliciet de massamatrix, potentiële energie en de inputmatrix, waardoor energiegebaseerde controle mogelijk is. We demonstreren het leren van Lagrangiaanse dynamica uit afbeeldingen op de $\mathrm{dm\_control}$ slinger-, cartpole- en acrobotomgevingen. KeyCLD kan worden geleerd op deze systemen, of ze nu ongeactueerd, ondergeactueerd of volledig geactueerd zijn. Getrainde modellen zijn in staat om lange-termijn videovoorspellingen te doen, waaruit blijkt dat de dynamica nauwkeurig wordt geleerd. We vergelijken met andere recente methoden uit de literatuur en onderzoeken het voordeel van de Lagrangiaanse voorkennis en de holonome nevenvoorwaarden. KeyCLD behaalt de hoogste geldige voorspellingstijd op alle vergelijkingstesten. Daarnaast wordt een zeer eenvoudige regelaar voor energiegebaseerde controle met succes toegepast op de volledig geactueerde systemen.

Stochastische dynamica met lange-termijninteracties. De vorige paragraaf beschouwt het toevoegen van fysische voorkennis op een deterministische manier. We gaan ook een stap verder in de richting van het leren van stochastische dynamische modellen van video. In dit proefschrift presenteren we een nieuw variationeel raamwerk om inferentie uit te voeren van (neurale) stochastische differentiaalvergelijkingen (SDE's) aangedreven door een Markov-benadering van fractionele Browniaanse beweging (fBM). SDE's bieden een veelzijdig instrument voor het modelleren van echte continue-tijd dynamische systemen met inherente ruis en willekeurigheid. Door SDE's te combineren met de krachtige methode van variationele gevolgtrekking, maakt leren van representatieve distributies mogelijk via stochastische gradiëntafdaling. Conventionele SDE's gaan er echter typisch van uit dat het onderliggende ruis een Browniaanse beweging (BM) volgt,

wat hun vermogen om lange-termijn afhankelijkheden vast te leggen belemmert. De fractionele Browniaanse beweging (fBM) daarentegen breidt de BM uit met niet-Markoviaanse dynamiek. Bestaande methoden voor het afleiden van fBM-parameters zijn echter ofwel rekenkundig te veeleisend of statistisch inefficiënt. Door voort te bouwen op de Markov-benadering van fBM, leiden we de bewijsondergrens af die essentieel is voor efficiënte variationele gevolgtrekking van de a posteriori padmaten, waarbij we steunen op gevestigde methoden uit de stochastische analyse. Bovendien geven we een gesloten uitdrukking voor optimale benaderingscoëfficiënten en stellen we voor om neurale netwerken te gebruiken om de drift-, diffusie- en controletermen binnen onze variationele a posteriori distributie te leren, wat leidt tot de variationele training van neurale-SDE's. We optimaliseren ook de Hurst-index, die de aard van de fractionele ruis bepaalt. Naast validatie op synthetische data presenteren we een nieuwe architectuur voor variationele latente videovoorspelling, een benadering die, voor zover wij weten, de eerste variationele neurale-SDE toepassing op videoperceptie mogelijk maakt.

Efficiënt leren van SDE's. Het leren van stochastische differentiaalvergelijkingen (SDE's) van data is een computationele uitdaging. Door de iteratieve aard van het maximaliseren van de bewijsondergrens, die het oplossen van de SDE voor elke iteratie vereist, is het leren van neurale SDE's over het algemeen traag en onstabiel. Hoe kunnen we efficiënter SDE's van data leren? We presenteren een hiërarchische, op controletheorie geïnspireerde methode voor variationele gevolgtrekking voor neurale SDE's. In dit hoofdstuk stellen we voor om de controleterm te ontleden in lineaire en residuele niet-lineaire componenten, en een optimale controleterm af te leiden voor lineaire SDE's, met behulp van stochastische optimale controletheorie. Door de niet-lineaire component te modelleren met een neuraal netwerk, tonen we aan dat neurale SDE's efficiënt getraind kunnen worden zonder hun expressief vermogen te verminderen. Omdat het lineaire deel van de controleterm optimaal is en niet geleerd hoeft te worden, wordt de training geïnitialiseerd aan een lagere kost en observeren we snellere convergentie.

Dit proefschrift beantwoordt de bovenstaande drie hoofddoelen

om uiteindelijk toe te werken naar het *leren van video in continue tijd met behulp van fysische voorkennis en fractioneel ruis*. Ons onderzoek toont aan dat het mogelijk is om tegelijkertijd zowel Lagrangiaanse dynamica als toestandsschattingsmodellen te leren van beelden in één leerproces. Ten tweede opent onze vooruitgang in variationele gevolgtrekking voor stochastische processen gedreven door fractioneel ruis nieuwe mogelijkheden voor het modelleren van complexe temporele fenomenen met lange-termijnafhankelijkheden. Ten slotte stellen we een methode voor, geïnspireerd op optimale controletheorie, om efficiënter SDE's te leren uit data.

Contents

List of Figures

List of Tables

Acronyms

BM Brownian motion

BMI body mass index

BMSDE SDE driven by BM

CNNs convolutional neural networks

ELBO evidence lower bound

fBM fractional Brownian motion

fBMSDE SDE driven by fBM

fCIR fractional Cox-Ingersol-Ross

fOU fractional Ornstein-Uhlenbeck

HGN Hamiltonian Generative Networks

KeyCLD constrained Lagrangian dynamics in keypoint coordinates

KeyLD Lagrangian dynamics in keypoint coordinates

KeyODE2 Second order dynamics in keypoint coordinates

KL Kullback-Leibler

Lag-caVAE Lagrangian dynamics with coordinate-aware variational autoencoder

Lag-VAE Lagrangian dynamics with variational autoencoder

MA-fBM Markov approximated fBM

MA-fBMSDE SDE driven by MA-fBM

MLE maximum likelihood estimation

MNIST Modified National Institute of Standards and Technology

MSE mean squared error

NDEs neural differential equations

ODE ordinary differential equation

ODEs ordinary differential equations

OU Ornstein-Uhlenbeck

SDE stochastic differential equation

SDEs stochastic differential equations

SM-MNIST Stochastic Moving MNIST

VI variational inference

VPT valid prediction time

1

1

Introduction

1

Introduction

This chapter starts by explaining some preliminary concepts, and then presents the motivation and and main research contributions of this dissertation. These preliminary concepts will help the reader understand the later chapters. The author aims to provide some basic insights, intuitions and examples, that would have helped him at the time he was studying these topics himself. Whenever possible, references are provided for further reading.

1.1 Computer Vision in the Golden Age of Deep Learning

This section's title is possibly a bit hyperbolic and its longevity most probably quite limited, however this nicely reflects its contents. How can we indeed describe the current state of computer vision research with lasting relevance? At the time of writing, academic progress in the field of computer vision is at an astronomical pace. None can predict how this progress will evolve in the coming years. Still, we provide a brief history of this tremendous progress that might help us understand what is coming in the near future.

Many say the current *deep learning* era started with the so-called *ImageNet moment*, when Krizhevsky et al. (2012) presented vastly improved results on the 2012 ImageNet challenge. Though commonly thought to be the first major breakthrough of convolutional neural networks (CNNs), we can rely on Jürgen Schmidhu-

ber[1] to educate us otherwise. It was already in May 2011 that the first vision contest was won using a CNN (Ciregan et al. (2012)).

At the time of these breakthroughs, many computer vision researchers were skeptical of deep learning methods. They were seen as black box methods, prone to overfitting and unable to generalize to novel settings. After some years, even the most adamant critics had to concede to the superior results of learning based methods. This paradigm shift has been so profound one could say we are now in the opposite situation[2].

Deep learning methods outperformed traditional methods in various computer vision tasks such as image classification (He et al. (2016)), object detection (Redmon et al. (2016)), semantic segmentation (Long et al. (2015)), human pose estimation (Cao et al. (2017)) and image generation (Goodfellow et al. (2014)). I purposefully cite relatively old works, because these were big breakthroughs at the time and more importantly, in terms of *applied* computer vision, many practical applications are still largely based on these methods. Rightfully so, because in practice it is often more important to have a method that works, than to have the latest and greatest method that is only marginally better on some benchmarks.

One of the most important breakthroughs in computer vision was the introduction of *diffusion models*. Song and Ermon (2019) presented state-of-the-art result on image generation, using diffusion models. This led to a swift adoption of this approach and very fast progress towards higher resolution images (Rombach et al. (2022)) and many other applications (see Ling Yang et al. (2023) for an overview). The improvement in image generation crossed a capability threshold that sparked widespread public interest[3]. Tools such as Dall-E, Midjourney and Stable Diffusion quickly became popular and widely used.

1. For the full story, see https://people.idsia.ch/~juergen/2010-breakthrough-supervised-deep-learning.html.

2. When I attented the *European Conference on Computer Vision* (ECCV) in 2022, my first major conference visit after the COVID-19 pandemic, I was unable to find more than a handful of posters that did *not* use neural networks.

3. Coincidently around the same time large language models (LLMs) such as chatGPT became available, with arguably even larger societal impact.

It has become tremendously easy to solve certain practical computer vision problems using deep learning methods, compared to 10 years ago. For many companies the main problem is not finding a good technical solution, but rather implementing and integrating this solution, or even more mundane, finding, labeling and cleaning the right data. This means, from an academic perspective, research directions have shifted to more difficult tasks, requiring more data, more complex models and more compute. Along with this shift and a strongly growing community, the pressure and competition to publish has also increased. In this context, a deliberate choice was made to focus the research in this thesis to methodological advances, rather than on scaling up data, models or compute. We have sought to incorporate physics priors and fractional noise in our models in ways that had not been done before. Thinking about the underlying principles of our models and how they can be improved, rather than just scaling them up.

1.2 Continuous Time Dynamics

This thesis focusses on learning from video data. Working with video requires understanding and modelling the underlying dynamics of the system or scene that is captured. A fundamental question is how we choose to approach the time aspect of these models. We can either choose a fixed discretization of time (*discretize-then-optimize*), or we can choose to model the dynamics in continuous time (*optimize-then-discretize*). The first option entails defining a *fixed grid* of time points and limiting our models to only operate on these predefined time points. The second option allows us to define models that are *continuous* in time, *i.e.*, we can use them at any time point, not just the fixed grid of time points. Note that, since our computers are discrete machines, in the end we will still discretize time. But the model itself is agnostic of this and valid for any discretization choice. Since nature is continuous, it makes sense to model the dynamics in continuous time. This is the approach we take in this thesis.

The general approach for modeling continuous time dynamics

is to use *differential equations*. Ordinary differential equations (ODEs) allow modeling a system by its derivative function:

$$\frac{\mathrm{d}x}{\mathrm{d}t}(t) = f(x), \quad x(0) = x_0\,. \tag{1.1}$$

In this example, $f(x)$ governs the dynamics of this system, *i.e.*, describes the derivative of the state x. Its solution at time t is given by

$$x(t) = x_0 + \int_0^t f(s)\,\mathrm{d}s\,, \tag{1.2}$$

which (for simple cases) can be solved analytically, or for most applications, numerically. The field of differential equations is vast and has been studied for centuries. There exist many types of ODE solvers, the most basic one is probably the Euler method. It consists of discretizing the time with a fixed timestep Δt and updating the discretized state accordingly:

$$x_{n+1} = x_n + f(x_n)\Delta t\,. \tag{1.3}$$

Within machine learning, the field of neural differential equations (NDEs) has seen a surge in popularity. The idea to bridge the worlds of neural networks and differential equations was popularized by R. T. Chen et al. (2018). In their 2018 NIPS paper, recognized with a *best paper award*, they show how to learn ODEs by backpropagating through a black-box ODE solver, *i.e.*, using the *adjoint sensitivity method*. Concretely, $f(x)$ is parameterized as a neural network, and the solution $x(t)$ at some time t is ultimately used in a loss function. This means the ODE can be learned end-to-end, within a bigger framework, and the ODE solver is part of this computation. By using the adjoint sensitivity method, the solver can be black-box, *i.e.*, does not need to be differentiable. The *implicit function theorem* is used to calculate the gradient of the solution with respect to the parameters of the ODE. See the excellent tutorial by Kolter et al. (2020) for a more detailed explanation and various examples and methodologies.

This approach was not novel in itself, as it is a common technique in control theory (Andersson et al. (2019)). However, the application to neural networks led to a significant surge in popularity within the field of machine learning. Neural ODEs have

been applied to many problems, including problems with no explicit dynamical aspects or time dynamics. For a more complete overview of the field of neural differential equations, we refer to Kidger (2021).

1.3 Physics Priors

This thesis investigates the use of *physics priors* in the context of learning from video data. The world behaves mostly in certain, predictable ways, governed by the laws of physics. Can we learn more efficiently, robustly or accurately to understand the world from video data, by incorporating specific prior knowledge about its underlying physics? This is explored in Chapter 2, where we use the Lagrangian mechanics formalism to infuse physics priors into our models.

In general, the dynamics of a physical system can be described by a second-order ordinary differential equation, we call the *equations of motion*:

$$\ddot{q}(t) = f(q(t), \dot{q}(t), t)\,, \tag{1.4}$$

where $q(t)$ is a vector describing the current positions of all degrees of freedom of a system, $\dot{q}(t)$ its first time derivative (velocities) and $\ddot{q}(t)$ its second time derivative (accelerations). Thus, the full state of a physical system is described by $(q(t), \dot{q}(t))$, its positions and velocities. This means we can solve the equation above, starting at some initial $(q_0, \dot{q}_0)$, and predict future states of the system. As an example, let us think about a small particle that can be approximated as a point mass. We can use Newton's second law of motion to write the equations of motion as $f(q(t), \dot{q}(t), t) = F(t)/m$, where m is its mass and $F(t)$ is the sum of all forces acting on the particle. For more complicated mechanical systems with coupled degrees of freedom, such as the two-link pendulum given as an example below, we can write down all forces and torques acting on the system, and inbetween all the rigid bodies that constitute the system. One can work out the equations of motion for such a system, but it requires practice and is not trivial, or even impossible for more complicated systems with many degrees of freedom.

On the other hand, *Lagrangian mechanics* offers an elegant and powerful way to describe the dynamics of a physical system, starting from the *Lagrangian*, which is the difference between the kinetic energy T and the potential energy V of a system:

$$L(q, \dot{q}, t) = T(q, \dot{q}, t) - V(q, t)\,. \tag{1.5}$$

Note that for any system, even when it has many degrees of freedom, L is a scalar function. The equations of motion can be derived from the Lagrangian in a general fashion, using the Euler-Lagrange equations. The underlying princple is the *principle of least action*[4]. To explain this principle, we consider all smooth paths $q(t)$ that a system can take from some initial state q_0 at initial time t_0 to some final state q_1 at final time t_1. For a given physical system, there exist infinite possible paths that this system can take to go from q_0 to q_1, and we want to find the *actual path taken by the system*. For this, we first define the *action* as as functional of the path $q(t)$, *i.e.*, the action is a function of the function $q(t)$:

$$S[q(t)] = \int_{t_0}^{t_1} L(q(t), \dot{q}(t), t)\,\mathrm{d}t\,. \tag{1.6}$$

Theorem 1 (Principle of Least Action (Tong (2005)))**.** *The actual path taken by the system is an extremum of S.*

Sketch of the proof. For a one-dimensional system, consider a small variation $\delta q(t)$ of the path $q(t)$, with $\delta q(t_0) = 0$ and $\delta q(t_1) = 0$ such that the path still goes from q_0 to q_1. The action of the path $q(t) + \delta q(t)$ is then given by

$$S[q(t) + \delta q(t)] = \int_{t_0}^{t_1} L(q(t) + \delta q(t), \dot{q}(t) + \delta\dot{q}(t), t)\,\mathrm{d}t\,. \tag{1.7}$$

4. For an excellent, approachable explanation of this principle and its history, see the video by *Veritasium* and Steven Strogatz: https://youtu.be/Q10_srZ-pbs

We express the change of action as

$$\delta S = \delta \left[\int_{t_0}^{t_1} L \, \mathrm{d}t \right] \tag{1.8}$$

$$= \int_{t_0}^{t_1} \delta L \, \mathrm{d}t \tag{1.9}$$

$$= \int_{t_0}^{t_1} \left(\frac{\partial L}{\partial q} \delta q + \frac{\partial L}{\partial \dot{q}} \delta \dot{q} \right) \mathrm{d}t \, . \tag{1.10}$$

Using integration by parts on the second term we arrive at

$$\delta S = \int_{t_0}^{t_1} \left(\frac{\partial L}{\partial q} - \frac{\mathrm{d}}{\mathrm{d}t} \left(\frac{\partial L}{\partial \dot{q}} \right) \right) \delta q \, \mathrm{d}t + \left[\frac{\partial L}{\partial \dot{q}} \delta q \right]_{t_0}^{t_1} , \tag{1.11}$$

where the final term vanishes because $\delta q(t_0) = \delta q(t_1) = 0$. Because the actual path taken by the system is an extremum of S, we require that $\delta S = 0$ for all variations $\delta q(t)$:

$$0 = \int_{t_0}^{t_1} \left(\frac{\partial L}{\partial q} - \frac{\mathrm{d}}{\mathrm{d}t} \left(\frac{\partial L}{\partial \dot{q}} \right) \right) \delta q \, \mathrm{d}t \, . \tag{1.12}$$

This can only be true when the first term in the integrand is zero:

$$\frac{\mathrm{d}}{\mathrm{d}t} \left(\frac{\partial L}{\partial \dot{q}} \right) - \frac{\partial L}{\partial q} = 0 \, . \tag{1.13}$$

□

For the multi-dimensional case $q(t) = (q_1(t), \ldots, q_D(t))$, we simply need to solve for every dimension and Eq. (1.14) becomes:

$$\frac{\mathrm{d}}{\mathrm{d}t} \left(\frac{\partial L}{\partial \dot{q}_i} \right) - \frac{\partial L}{\partial q_i} = 0, \quad i = 1, \ldots, D \, . \tag{1.14}$$

We have now arrived at the Euler-Lagrange equations. These equations give us the equations of motion for any physical system, given its Lagrangian. Note that this result is not limited to mechanical systems! In fact, (nearly) all the fundamental laws of physics can be written using just one Lagrangian:

$$L = \sqrt{g} \left(R - \frac{1}{2} F_{\mu\nu} F^{\mu\nu} + \bar{\psi} \not{D} \psi \right) , \tag{1.15}$$

which describes gravity, the forces of nature (electromagnetism, and weak and strong nuclear forces) and the dynamics of subatomic particles. Still, the use of *Lagrangian mechanics* is limited to mechanical systems in this work, primarily relevant for Chapter 2. For a broader overview we refer to Tong (2005).

1.3.1 Non-conservative forces

The derivations above assume that the forces acting on the system are conservative, *i.e.*, they can be described by a potential energy. This means, the sum of the potential and kinetic energy $T + V$ is conserved, *i.e.*, stays constant over time. When we want to model *non-conservative forces*, such as friction or external motor inputs, we can add a term to the Lagrangian:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = \frac{\partial W}{\partial q_i}, \quad i = 1, \ldots, D\,, \tag{1.16}$$

where W is the work done by these non-conservative forces. This is a generalization of the Euler-Lagrange equation, and allows us to model a wider range of physical systems.

1.3.2 Example for a two-link pendulum

To offer the reader some intuition on how to derive the equations of motion for a mechanical system using Lagrangian mechanics, we consider a two-link pendulum, see Fig. 1.1. This system has two degrees of freedom, thus we can use the angles of the links with respect to the vertical y-axis as coordinates:

$$q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}, \quad \dot{q} = \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix}, \tag{1.17}$$

The links have no mass and lengths of respectively l_1 and l_2, and at the end of the links are point masses of respectively m_1 and m_2. We start by writing the Cartesian coordinates of the point masses,

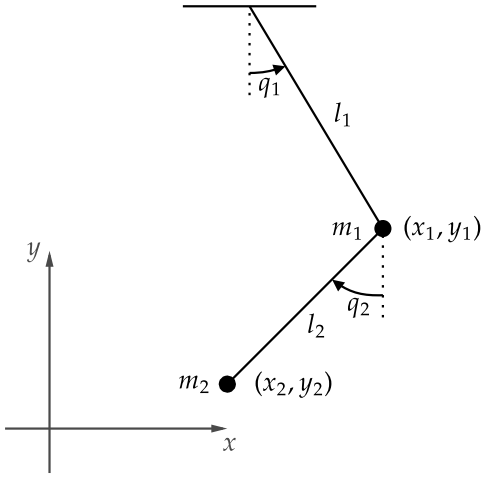


Figure 1.1 Two-link pendulum.

because it allows a more convenient way to describe the potential and kinetic energy:

$$x_1 = l_1 \sin(q_1)\,, \tag{1.18}$$
$$y_1 = -l_1 \cos(q_1)\,, \tag{1.19}$$
$$x_2 = l_1 \sin(q_1) + l_2 \sin(q_2)\,, \tag{1.20}$$
$$y_2 = -l_1 \cos(q_1) - l_2 \cos(q_2)\,, \tag{1.21}$$

and their derivatives with respect to time:

$$\dot{x}_1 = l_1 \cos(q_1)\dot{q}_1\,, \tag{1.22}$$
$$\dot{y}_1 = l_1 \sin(q_1)\dot{q}_1\,, \tag{1.23}$$
$$\dot{x}_2 = l_1 \cos(q_1)\dot{q}_1 + l_2 \cos(q_2)\dot{q}_2\,, \tag{1.24}$$
$$\dot{y}_2 = l_1 \sin(q_1)\dot{q}_1 + l_2 \sin(q_2)\dot{q}_2\,. \tag{1.25}$$

The kinetic energy of the system is given by

$$T = \frac{1}{2} m_1 \left(\dot{x}_1^2 + \dot{y}_1^2\right) + \frac{1}{2} m_2 \left(\dot{x}_2^2 + \dot{y}_2^2\right) \tag{1.26}$$
$$= \frac{1}{2} m_1 l_1^2 \dot{q}_1^2 + \frac{1}{2} m_2 \left(l_1^2 \dot{q}_1^2 + l_2^2 \dot{q}_2^2 + 2 l_1 l_2 \dot{q}_1 \dot{q}_2 \cos(q_1 - q_2)\right) \tag{1.27}$$
$$= \frac{1}{2}(m_1 + m_2) l_1^2 \dot{q}_1^2 + \frac{1}{2} m_2 l_2^2 \dot{q}_2^2 + m_2 l_1 l_2 \cos(q_1 - q_2)\dot{q}_1 \dot{q}_2 \tag{1.28}$$

where we used the trigonometric identity $\cos(q_1)\cos(q_2) + \sin(q_1)\sin(q_2) = \cos(q_1 - q_2)$. The potential energy is

$$V = m_1 g y_1 + m_2 g y_2 \tag{1.29}$$
$$= -m_1 g l_1 \cos(q_1) - m_2 g (l_1 \cos(q_1) + l_2 \cos(q_2)) \tag{1.30}$$
$$= -(m_1 + m_2) g l_1 \cos(q_1) - m_2 g l_2 \cos(q_2)\,, \tag{1.31}$$

where g is the acceleration of gravity. We first derive the sub parts of Eq. (1.14):

$$\frac{\partial L}{\partial q_1} = \frac{\partial T}{\partial q_1} - \frac{\partial V}{\partial q_1} \tag{1.32}$$
$$= -m_2 l_1 l_2 \sin(q_1 - q_2)\dot{q}_1\dot{q}_2 - (m_1 + m_2) g l_1 \sin(q_1)\,, \tag{1.33}$$
$$\frac{\partial L}{\partial q_2} = \frac{\partial T}{\partial q_2} - \frac{\partial V}{\partial q_2} \tag{1.34}$$
$$= m_2 l_1 l_2 \sin(q_1 - q_2)\dot{q}_1\dot{q}_2 - m_2 g l_2 \sin(q_2)\,, \tag{1.35}$$
$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{q}_1}\right) = \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T}{\partial \dot{q}_1} - \frac{\partial V}{\partial \dot{q}_1}\right) \tag{1.36}$$
$$= \frac{\mathrm{d}}{\mathrm{d}t}\left((m_1 + m_2) l_1^2 \dot{q}_1 + m_2 l_1 l_2 \cos(q_1 - q_2)\dot{q}_2\right) \tag{1.37}$$
$$= (m_1 + m_2) l_1^2 \ddot{q}_1 + m_2 l_1 l_2 \cos(q_1 - q_2)\ddot{q}_2 - m_2 l_1 l_2 \sin(q_1 - q_2)(\dot{q}_1 - \dot{q}_2)\dot{q}_2\,, \tag{1.38}$$
$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{q}_2}\right) = \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial T}{\partial \dot{q}_2} - \frac{\partial V}{\partial \dot{q}_2}\right) \tag{1.39}$$
$$= \frac{\mathrm{d}}{\mathrm{d}t}\left(m_2 l_2^2 \dot{q}_2 + m_2 l_1 l_2 \cos(q_1 - q_2)\dot{q}_1\right) \tag{1.40}$$
$$= m_2 l_2^2 \ddot{q}_2 + m_2 l_1 l_2 \cos(q_1 - q_2)\ddot{q}_1 - m_2 l_1 l_2 \sin(q_1 - q_2)(\dot{q}_1 - \dot{q}_2)\dot{q}_1\,, \tag{1.41}$$

leading to an equation for each degree of freedom:

$$0 = (m_1 + m_2) l_1^2 \ddot{q}_1 + m_2 l_1 l_2 \cos(q_1 - q_2)\ddot{q}_2 + m_2 l_1 l_2 \sin(q_1 - q_2)\dot{q}_2^2 + (m_1 + m_2) g l_1 \sin(q_1)\,, \tag{1.42}$$
$$0 = m_2 l_2^2 \ddot{q}_2 + m_2 l_1 l_2 \cos(q_1 - q_2)\ddot{q}_1 - m_2 l_1 l_2 \sin(q_1 - q_2)\dot{q}_1^2 + m_2 g l_2 \sin(q_2)\,. \tag{1.43}$$

We can write this system of equations in vector form:

$$0 = M(q)\ddot{q} + C(q, \dot{q})\dot{q} - F(q)\,, \tag{1.44}$$

where

$$M(q) = \begin{bmatrix} (m_1 + m_2)l_1^2 & m_2 l_1 l_2 \cos(q_1 - q_2) \\ m_2 l_1 l_2 \cos(q_1 - q_2) & m_2 l_2^2 \end{bmatrix}, \tag{1.45}$$

$$C(q, \dot{q}) = \begin{bmatrix} 0 & m_2 l_1 l_2 \sin(q_1 - q_2)\dot{q}_2 \\ -m_2 l_1 \sin(q_1 - q_2)\dot{q}_1 & 0 \end{bmatrix}, \tag{1.46}$$

$$F(q) = \begin{bmatrix} -(m_1 + m_2) g l_1 \sin(q_1) \\ -m_2 g l_2 \sin(q_2) \end{bmatrix}. \tag{1.47}$$

In fact, we could have derived these equations directly in this general vector form. This is possible because we can write the kinetic energy of any mechanical system as

$$T = \frac{1}{2}\dot{q}^T M(q)\dot{q}, \tag{1.48}$$

and both $M(q)$ and $V(q)$ are only a function of the positions q, not of the velocities $\dot{q}$. One can see that $F(q) = -\frac{\partial V}{\partial q}$, thus representing the forces acting on the system due to the potential energy. Finally, $C(q, \dot{q})$ can be directly obtained from $M(q)$ by

$$C_{k,j}(q, \dot{q}) = \sum_{i=1}^{D} c_{ijk}(q)\dot{q}_i \,, \tag{1.49}$$

$$c_{ijk} = \frac{1}{2}\left(\frac{\partial M_{k,j}(q)}{\partial q_j} + \frac{\partial M_{k,i}(q)}{\partial q_j} - \frac{\partial M_{i,j}(q)}{\partial q_k} \right) \dot{q}_i \,, \tag{1.50}$$

where c_{ijk} are the so-called *Christoffel symbols*, see Spong and Vidyasagar (2008) for a more elaborate explanation. The $C(q, \dot{q})\dot{q}$ term represents the *centrifugal and Coriolis forces*. If this term would be zero (*i.e.*, if M would be independent of q, and thus static), our equations would match the well-known Newton's second law of motion: force equals mass times acceleration. However, the system has couplings between its degrees of freedom, *i.e.*, the dynamics of the second link depend on the position and velocity of the first link. The *Lagrangian mechanics* approach naturally leads to these terms, without the need to take local and rotating reference frames explicitly into account.

Finally, we can add an external input to the system, *e.g.*, a motor that applies a torque to the first link. This system is known as an *acrobot*. The work done by this moter with torque τ is given by

$$W = \tau \dot{q}_1 \,. \tag{1.51}$$

Thus, the full equations of motion for the acrobot are (using Eq. (1.16))

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} - F(q) = \begin{bmatrix}\tau \\ 0\end{bmatrix} . \tag{1.52}$$

The full state of the system is described by its positions and velocities:

$$x = \begin{bmatrix} q_1 \\ q_2 \\ \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} . \tag{1.53}$$

When we know x at time t, the equations of motion allow us to predict the state of the system at some future time $T > t$, since the system is deterministic. We can easily write the second order differential equations as a system of first order differential equations:

$$\frac{\mathrm{d}x}{\mathrm{d}t}(t) = \begin{bmatrix} & \dot{q}_1 \\ & \dot{q}_2 \\ M^{-1}(q)\left(\begin{bmatrix}\tau \\ 0\end{bmatrix} - C(q,\dot{q})\dot{q} + F(q)\right) & \end{bmatrix} , \tag{1.54}$$

such that it is compatible with Sec. 1.2. There exists no analytical solution for a two-link pendulum. Its equation of motion should be solved numerically, *e.g.*, using the Euler method mentioned above or more advanced methods, such as Runge-Kutta, that achieve a higher accuracy. Typically, one can use implementations of ODE solvers available in scientific computing libraries, such as SciPy (Virtanen et al. (2020))[5].

1.4 Stochastic Dynamics

While the previous sections handle *deterministic* dynamics, the world is not always deterministic. At least not in the sense that we have full knowledge of the multitude of different variables

5. See https://scipython.com/blog/the-double-pendulum/ for an example of an implementation of this system.

that influence the behaviour of a given system. As an example, the equations of motion of the double pendulum in Sec. 1.3.2 can be elegantly derived, but a number of assumptions were made. *e.g.*, the influence of the air on the motion of the pendulum was ignored. While it is theoretically possible to take this into account in a deterministic way, this is not at all practical. Let alone, how can one measure the full state of the air surrounding this system, all the pressure differences and airflows at a fine-grained scale? Alternatively, we can model the influence of the air and other unknown effects as *noise*. Concretely, this means our equations of motion are no longer a deterministic ordinary differential equation (ODE), but rather a *stochastic differential equation (SDE)* (see Sec. 1.6.1).

A second focus of this thesis is thus learning *stochastic* models from video. But what exactly does it mean to *learn* from data? Inference is the process of estimating the parameters of a model from data. In the case of ODEs, we can estimate the parameters of the model by minimizing the difference between the predicted and observed data. This is typically done using *maximum likelihood estimation (MLE)*. But in the case of SDEs, the situation is more complicated. The SDE is a stochastic process, which means it is not possible to predict the future state of the system exactly. Instead, we can only predict the probability distribution of the future state of the system. This means we need to use a different approach to estimate the parameters of the model: *variational inference (VI)*.

1.5 Inference

Let us say we happen to be interested in modeling the chest circumference of Scottish soldiers from the early nineteenth century. At least, that was what Adolphe Quetelet[6] was interested in, when he did some calculations on this data. He found that the distribution of the chest circumference was approximately *normally distributed*, which spurred his interest in the concept

6. Adolphe Quetelet was born in Ghent in 1796, and received the first doctorate in mathematics ever awarded by Ghent University, in 1819.

of the *average man* (Quetelet (1842)). He started collecting and investigating all kinds of human measurements and believed perfection was represented by the *average*, where any deviation from this should be considered an error. Among these ideas was the *Quetelet Index*, which is nowadays better know as the *body mass index (BMI)*.

What we are interested in, is how to find the parameters of a model that best describes some given data. If, as Quetelet did for the Scottish soldiers, we use a Gaussian distribution, it is generally feasible to find its parameters. There are only two: the mean μ and the variance σ^2, and we can straightforwardly estimate them from the data. A bit more formally, if we denote the circumference of the chest of a Scottish soldier as X, we describe the probability distribution of X as

$$p(X) = \mathcal{N}(\mu, \sigma^2)\,. \tag{1.55}$$

While many statistical methods assume Gaussian distributions due to their mathematical convenience, real-world phenomena often deviate significantly from this idealized framework. Many processes in nature, society, and complex systems exhibit non-Gaussian behavior, heavy tails, multimodality, or discrete structures that cannot be adequately captured by simple parametric distributions. Moreover, the quantities we wish to understand are frequently not directly observable or measurable, as the fundamental processes that govern a system's behavior remain hidden from direct measurement.

1.5.1 Latent variable models

When the true variables of interest are hidden, we work with latent variable models where the variables we want to understand, denoted as Z, are latent and we can only observe their indirect effects through noisy, often non-linear measurements denoted as X. This framework naturally leads us to consider both our prior beliefs about the latent variables and how these beliefs are updated through observed evidence.

The *prior distribution* $p(Z)$ represents our initial beliefs or assumptions about the latent variables before observing any data.

This prior represents our domain knowledge, theoretical understanding, or simply our assumptions about reasonable values for the hidden variables. In Bayesian inference, the prior serves as a regularizing force that prevents overfitting and incorporates external knowledge into the inference process.

Using Bayes' theorem, the prior is combined with the likelihood of the observations to yield the *posterior distribution*:

$$p(Z \mid X) = \frac{p(X \mid Z)p(Z)}{p(X)} \,. \tag{1.56}$$

The posterior $p(Z \mid X)$ represents our updated beliefs about the latent variables after incorporating the observed evidence X. It balances the information from the data, captured by the likelihood $p(X \mid Z)$, with our prior knowledge $p(Z)$. The denominator $p(X)$ is the marginal likelihood or *evidence*. Note that the posterior is proportional to the product of the likelihood and the prior. When the prior is informative, it can significantly improve inference quality, particularly in settings with limited data. On the other hand, when data is abundant, the likelihood dominates and the posterior becomes less sensitive to the choice of prior.

1.5.2 Inference as optimization

When the relationship between X and Z is complicated (*e.g.*, X can be an image, or a sequence of words, or even a Ph.D. dissertation), it is often infeasible (*intractable*) to calculate the posterior distribution $p(Z \mid X)$ exactly. This arises because the posterior often involves high-dimensional integrals that cannot be solved analytically.

Variational inference (VI) is one of the ways to handle this problem. We approximate the posterior distribution by a so-called *variational distribution* $q(Z)$:

$$q(Z) \approx p(Z \mid X) \,. \tag{1.57}$$

By choosing the form of $q(Z)$ wisely, it will have a structure we can work with. The goal of VI is to find the parameters of $q(Z)$ that best approximate the posterior distribution $p(Z \mid X)$. This is

done by minimizing a dissimilarity metric, such as the *Kullback-Leibler (KL) divergence*, between $q(Z)$ and $p(Z \mid X)$:

$$\arg\min_{q(Z)} D_{\mathrm{KL}}\left(q(Z) \,||\, p(Z \mid X)\right) . \tag{1.58}$$

The KL divergence is not directly computable, because it involves the posterior distribution $p(Z \mid X)$. However, we define the evidence lower bound (ELBO) as

$$\mathrm{ELBO} = \mathbb{E}\left[\log p(X \mid Z)\right] - D_{\mathrm{KL}}\left(q(Z) \,||\, p(Z)\right) . \tag{1.59}$$

This means the ELBO is the sum of the expected log likelihood of the model $p(X \mid Z)$ and the negative KL divergence between the variational distribution $q(Z)$ and the prior distribution $p(Z)$. The first term measures how well the model explains the observed data when the latent variables are distributed according to $q(Z)$. The second terms acts as a regularization loss that prevents the variational distribution from deviating too much from the prior distribution $p(Z)$. This regularization term is important in the context of balancing prior knowledge and observed evidence. A high value indicates that the variational distribution substantially contradicts the prior beliefs. The ELBO thus naturally incorporates both the quality of data fit and the consistency with prior assumptions.

One can show that maximizing the ELBO is equivalent to minimizing the above KL divergence (Blei et al. (2017)). Finally, since the ELBO is a function of the parameters of the variational distribution $q(Z)$ we can optimize it using gradient-based optimization methods. This is the essence of *variational inference*. See Bishop and Nasrabadi (2006), Ganguly and Earp (2021), and Goodfellow et al. (2016) for more detailed introductions and Alemi et al. (2018) and Poole et al. (2019) for a more information theoretic perspective.

1.6 Stochastic Calculus

In this section we explain some basic concepts from stochastic calculus, or more specifically Itô calculus. No attempt is made to

provide a general overview, but rather to provide the necessary background for the rest of this dissertation. Comprehensive introductions can be found in Särkkä and Solin (2019) and Øksendal (2003).

1.6.1 Stochastic differential equations

Stochastic differential equations (SDEs) are a way to describe the dynamics of a stochastic process. For simplicity, we consider a one-dimensional stochastic process $x(t) \in \mathbb{R}$. It can be described as a stochastic differential equation (SDE) of the form:

$$\mathrm{d}x(t) = b\left(x(t), t\right) \mathrm{d}t + \sigma\left(x(t), t\right) \mathrm{d}W(t)\,, \tag{1.60}$$

where $b : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is the drift function, $\sigma : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is the diffusion function and $W(t) \in \mathbb{R}$ is a Wiener process. The Wiener process is a continuous-time stochastic process with independent increments and normally distributed increments:

$$\mathrm{d}W(t) \sim \mathcal{N}(0, \mathrm{d}t)\,. \tag{1.61}$$

In the context of this dissertation it is synonymous with Brownian motion (BM), but we will mostly use the term Wiener process to differentiate more clearly with *fractional Brownian motion (fBM)* (see Chapter 3).

One can see that if the diffusion $\sigma(x(t), t)$ would be zero, the process defined by Eq. (1.60) falls back to an ordinary differential equation (ODE, see Sec. 1.2). The difussion term is driven by the stochastic Wiener process, introducing randomness in the process. This means, solving an SDE actually means sampling from a distribution of possible paths, *i.e.*, everytime you solve an SDE you will have a different result.

Similar to ODEs, SDEs can be solved numerically using various methods. The simplest method is the Euler-Maruyama method, which is a generalization of the Euler method for ODEs:

$$x_{n+1} = x_n + b(x_n, t_n)\Delta t + \sigma(x_n, t_n)\Delta W_n\,, \tag{1.62}$$

where $\Delta W_n \sim \mathcal{N}(0, \Delta t)$.

1.6.2 Itô formula

A crucial difference between ordinary calculus and Itô calculus is described by the Itô formula. This formula is the Itô calculus equivalent of the chain rule. Consider an arbitrary scalar function $\phi(x(t), t)$ of the process above, then the differential of ϕ (or the Itô SDE for ϕ) is given as

$$\mathrm{d}\phi = \frac{\partial \phi}{\partial t}\,\mathrm{d}t + \frac{\partial \phi}{\partial x}\,\mathrm{d}x + \frac{1}{2}\frac{\partial^2 \phi}{\partial x^2}\,\mathrm{d}x^2\,. \tag{1.63}$$

This contrasts with ordinary calculus, where the last term is not present. When working this out, the following rules are observed:

$$\mathrm{d}W\,\mathrm{d}t = 0\,, \tag{1.64}$$

$$\mathrm{d}W^2 = \mathrm{d}t\,. \tag{1.65}$$

Example

As an example, we will transform the SDE defined by Eq. (1.60) using $y = \log(x)$:

$$\begin{aligned}
\mathrm{d}y &= x^{-1}\,\mathrm{d}x - \frac{1}{2}x^{-2}\,\mathrm{d}x^2 && (1.66)\\
&= x^{-1}\left(b\left(x,t\right)\mathrm{d}t + \sigma\left(x,t\right)\mathrm{d}W\right) - \frac{1}{2}x^{-2}\sigma\left(x,t\right)^2\,\mathrm{d}t && (1.67)\\
&= e^{-y}\left(b\left(e^y,t\right)\mathrm{d}t + \sigma\left(e^y,t\right)\mathrm{d}W\right) - \frac{1}{2}e^{-2y}\sigma\left(e^y,t\right)^2\,\mathrm{d}t && (1.68)\\
&= \left(e^{-y}b\left(e^y,t\right) - \frac{1}{2}e^{-2y}\sigma\left(e^y,t\right)^2\right)\mathrm{d}t + e^{-y}\sigma\left(e^y,t\right)\mathrm{d}W\,. && (1.69)
\end{aligned}$$

1.6.3 Itô isometry

We will often calculate variances of stochastic processes given as Itô integrals. Itô isometry is a useful tool to calculate these

variances. Given two stochastic processes $X(t)$ and $Y(t)$ defined by

$$X(t) = \int_0^t f(s)\,\mathrm{d}W(s)\,, \tag{1.70}$$

$$Y(t) = \int_0^t g(s)\,\mathrm{d}W(s)\,. \tag{1.71}$$

Itô isometry states that

$$\mathbb{E}\left[X(t_1)Y(t_2)\right] = \mathbb{E}\left[\left(\int_0^{t_1} f(s)\,\mathrm{d}W(s)\right)\left(\int_0^{t_2} g(s)\,\mathrm{d}W(s)\right)\right] \tag{1.72}$$

$$= \int_0^{\min(t_1,t_2)} f(s)g(s)\,\mathrm{d}s\,. \tag{1.73}$$

Furthermore, since the expectation of a stochastic integral is always zero, this directly gives us the the covariance between $X(t)$ and $Y(t)$:

$$\mathbb{E}\left[X(t_1)\right] = 0, \tag{1.74}$$

$$\mathbb{E}\left[Y(t_2)\right] = 0, \tag{1.75}$$

$$\mathrm{Cov}\left(X(t_1), Y(t_2)\right) = \int_0^{\min(t_1,t_2)} f(s)g(s)\,\mathrm{d}s\,. \tag{1.76}$$

1.7 Gamma Function

Here an introduction to the gamma function and some of its properties is provided, since it will be used in Chapter 3. The gamma function $\Gamma(z)$ is an extension of the factorial function. It is defined for all complex numbers $z \in \mathbb{C}$, except for the non-positive integers. For positive integers $n = z$, it reduces to the factorial function:

$$\Gamma(n) = (n-1)!\,. \tag{1.77}$$

For complex numbers z with a positive real part it can be defined by this integral:

$$\Gamma(z) = \int_0^\infty t^{z-1}e^{-t}\,\mathrm{d}t\,. \tag{1.78}$$

One useful property of the gamma function is the recursion relation, which shows the connection between the gamma function and the factorial function:

$$\Gamma(z+1) = z\Gamma(z)\,. \tag{1.79}$$

We can also define incomplete gamma functions, where the integral is truncated. The upper incomplete gamma function:

$$\Gamma(z,x) = \int_x^\infty t^{z-1}e^{-t}\,\mathrm{d}t, \tag{1.80}$$

and the lower incomplete gamma function[7]:

$$\gamma(z,x) = \int_0^x t^{z-1}e^{-t}\,\mathrm{d}t\,. \tag{1.81}$$

Sometimes these are easier to work with in their normalized versions:

$$Q(z,x) = \frac{\Gamma(z,x)}{\Gamma(z)} = \frac{1}{\Gamma(z)}\int_x^\infty t^{z-1}e^{-t}\,\mathrm{d}t, \tag{1.82}$$

$$P(z,x) = \frac{\gamma(z,x)}{\Gamma(z)} = \frac{1}{\Gamma(z)}\int_0^x t^{z-1}e^{-t}\,\mathrm{d}t\,. \tag{1.83}$$

These are some of their properties useful in this work:

$$\Gamma(z,0) = \Gamma(z), \tag{1.84}$$

$$\gamma(z,0) = 0, \tag{1.85}$$

$$\Gamma(z,x) + \gamma(z,x) = \Gamma(z), \tag{1.86}$$

$$Q(z,0) = 1, \tag{1.87}$$

$$P(z,0) = 0, \tag{1.88}$$

$$Q(z,x) + P(z,x) = 1\,. \tag{1.89}$$

While the functions presented in this section have no closed-form solutions, implementations of their numerical evaluation are readily available in standard scientific computing libraries such as SciPy (Virtanen et al. (2020)), or in specialized machine learning libraries such as PyTorch (Paszke et al. (2019)) and JAX (Bradbury et al. (2018)).

7. Outside of this section, the use of γ is reserved for the MA-fBM parameters (Chapter 3), to avoid any possible confusion.

1.8 Motivation and Goals of this Dissertation

Learning models from video entails learning the underlying dynamics of a system from its visual representation. An accurate model is capable of predicting future video frames, given a sequence of past frames. This is useful for many applications, such as robotics, where the model can be used to control a robot based on its visual input. However, learning models from video is a challenging task, as it requires the model to learn the underlying dynamics of the system from its visual representation. One fundamental choice in this regard is whether we model time discretely (*discretize-then-optimize*) or in a continuous way (*optimize-then-discretize*). In this thesis we choose the continuous-time approach (see Sec. 1.2).

Physics Priors. Can the learning of video models benefit from introducing physics priors? Physics priors are a way to incorporate prior knowledge about the underlying dynamics of a system into the learning process. Specifically, we can use Lagrangian dynamics to model the underlying dynamics of a system, as introduced in Sec. 1.3. This entails modeling the system explicitly using its potential energy and mass matrix. Our contributions towards this research question are presented in Chapter 2.

Long-range Stochastic Dynamics. Modeling stochastic dynamics in continuous-time is typically done using stochastic differential equations (SDEs) driving by Brownian motion (BM) (see Sec. 1.6). The increments of BM are independent, which means this noise process has no *memory*. Can the learning of video models benefit from using long-range stochastic dynamics? Stochastic differential equations (SDEs) driven by fractional Brownian motion (fBM) are a powerful tool to model stochastic systems with long-range dependence. However, the inference of the parameters of these models is challenging, due to the non-Markovian nature of fBM. This is precisely the research question we address in Chapter 3, where we aim to learn the parameters of SDEs driven by a Markov approximation of fBM.

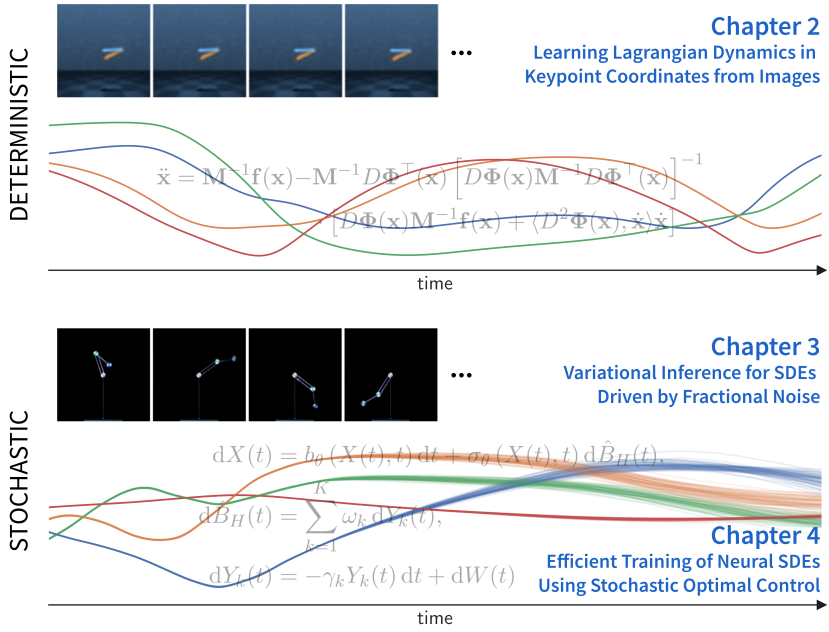


Figure 1.2 Overview of the research topics in this dissertation.

Efficient Learning of SDEs. Within the field of machine learning, the use of neural networks to model SDEs has gained popularity. However, the training of these models is computationally expensive, due to the iterative nature of maximizing the evidence lower bound (ELBO). How can we more efficiently train neural SDEs? We address this question in Chapter 4, where a method is proposed to train neural SDEs leveraging stochastic optimal control.

This dissertation answers the above three main goals to ultimately work towards *learning from video in continuous-time using physics priors and fractional noise.*

1.9 Research Contributions and Overview of this Dissertation

We summarize the following main research contributions of this dissertation:

1. In Chapter 2, we introduce KeyCLD, a framework to learn constrained Lagrangian dynamics from images. We are the first to use learned keypoint representations from images to learn Lagrangian dynamics. We show that keypoint representations derived from images can directly be used as positional state vector for constrained Lagrangian dynamics, expressed in Cartesian coordinates.

2. We show how to control constrained Lagrangian dynamics in Cartesian coordinates with energy shaping, where the state is estimated from images.

3. KeyCLD is empirically validated on the pendulum, cartpole and acrobot systems from dm_control (Tunyasuvunakool et al. (2020)). We show that KeyCLD can be learned on these systems, whether they are unactuated, underactuated or fully actuated. We compare quantitatively with Lag-caVAE, Lag-VAE (Zhong and Leonard (2020)) and HGN (Toth et al. (2020)), and investigate the benefit of the Lagrangian prior and the constraint function. KeyCLD achieves the highest valid prediction time on all benchmarks (see Tab. 2.2).

4. In Chapter 3, we make accessible the relatively uncharted Markovian embedding of fractional Brownian motion (fBM) and its strong approximation, to the machine learning community. This allows us to employ the traditional machinery of stochastic differential equations (SDEs) in working with non-Markovian systems.

5. We show how to balance the contribution of Markov processes by optimising for the combination coefficients in closed form. We further estimate the (time-dependent) Hurst index from data.

6. We derive the evidence lower bound (ELBO) for SDEs driven by approximate fBM of both Types I and II.
7. We model the drift, diffusion and control terms in our framework by neural networks, and propose a novel architecture for video prediction.
8. In Chapter 4, we present a method for efficient training of neural SDEs using stochastic optimal control.

Chapter 2: KeyCLD: Learning Constrained Lagrangian Dynamics in Keypoint Coordinates from Images

We present KeyCLD, a framework to learn Lagrangian dynamics from images. Learned keypoints represent semantic landmarks in images and can directly represent state dynamics. We show that interpreting this state as Cartesian coordinates, coupled with explicit holonomic constraints, allows expressing the dynamics with a constrained Lagrangian. KeyCLD is trained unsupervised end-to-end on sequences of images. Our method explicitly models the mass matrix, potential energy and the input matrix, thus allowing energy based control. We demonstrate learning of Lagrangian dynamics from images on the dm_control pendulum, cartpole and acrobot environments. KeyCLD can be learned on these systems, whether they are unactuated, underactuated or fully actuated. Trained models are able to produce long-term video predictions, showing that the dynamics are accurately learned. We compare with Lag-VAE, Lag-caVAE and HGN, and investigate the benefit of the Lagrangian prior and the constraint function. KeyCLD achieves the highest valid prediction time on all benchmarks. Additionally, a very straightforward energy shaping controller is successfully applied on the fully actuated systems.

Chapter 3: Variational Inference of SDEs Driven by Fractional Noise

We present a novel variational framework for performing inference in (neural) stochastic differential equations (SDEs)

driven by Markov-approximate fractional Brownian motion (fBM). SDEs offer a versatile tool for modeling real-world continuous-time dynamic systems with inherent noise and randomness. Combining SDEs with the powerful inference capabilities of variational methods, enables the learning of representative distributions through stochastic gradient descent. However, conventional SDEs typically assume the underlying noise to follow a Brownian motion (BM), which hinders their ability to capture long-term dependencies. In contrast, fractional Brownian motion (fBM) extends BM to encompass non-Markovian dynamics, but existing methods for inferring fBM parameters are either computationally demanding or statistically inefficient. In this chapter, building upon the Markov approximation of fBM, we derive the evidence lower bound essential for efficient variational inference of posterior path measures, drawing from the well-established field of stochastic analysis. Additionally, we provide a closed-form expression for optimal approximation coefficients and propose to use neural networks to learn the drift, diffusion and control terms within our variational posterior, leading to the variational training of neural-SDEs. In this framework, we also optimize the Hurst index, governing the nature of our fractional noise. Beyond validation on synthetic data, we contribute a novel architecture for variational latent video prediction,—an approach that, to the best of our knowledge, enables the first variational neural-SDE application to video perception.

Chapter 4: Efficient Training of Neural SDEs Using Stochastic Optimal Control

We present a hierarchical, control theory inspired method for variational inference (VI) for neural stochastic differential equations (SDEs). While VI for neural SDEs is a promising avenue for uncertainty-aware reasoning in time-series, it is computationally challenging due to the iterative nature of maximizing the ELBO. In this work, we propose to decompose the control term into linear and residual non-linear components and derive an optimal control term for linear SDEs, using stochastic optimal control. Modeling the non-linear component by a neural network, we

show how to efficiently train neural SDEs without sacrificing their expressive power. Since the linear part of the control term is optimal and does not need to be learned, the training is initialized at a lower cost and we observe faster convergence.

Chapter 5: Conclusion and Future Work

The final chapter presents a conclusion of this dissertation. Future work directions are discussed, most notably a relatively detailed exposition of a novel approach for an improved Markov approximation of fractional Brownian motion.

1.10 Publication List

1. **Daems, R.**, Seys, S. F., Hox, V., Chaker, A., De Greve, G., Lemmens, W., Poirrier, A.-L., Beckers, E., Diamant, Z., Dierickx, C., Hellings, P. W., Huart, C., Jerin, C., Jorissen, M., Oscé, H., Roux, K., Thompson, M., Tombu, S., Uyttebroek, S., Zarowski, A., Gorris, S., Van Gerven, L., Loeckx, D., and Demeester, T. (2025). Improved Allergy Wheal Detection for the Skin Prick Automated Test Device. Artificial Intelligence in Medicine (AIME) 2025.
   The work done in this publication is outside the scope of this dissertation.

2. Nobis, G., Belova, A., Springenberg, M., **Daems, R.**, Knochenhauer, C., Opper, M., Birdal, T., and Samek, W. (2025). Fractional Brownian Bridges for Aligned Data. Learning Meaningful Representations of Life (LMRL) Workshop at ICLR 2025.
   The work done in this publication is outside the scope of this dissertation.

3. **Daems, R.**, Opper, M., Crevecoeur, G., and Birdal, T. (2025). Efficient Training of Neural SDEs Using Stochastic Optimal Control. European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN) 2025.

4. Nobis, G., Springenberg, M., Aversa, M., Detzel, M., **Daems, R.**, Murray-Smith, R., Nakajima, S., Lapuschkin, S., Ermon, S., Birdal, T., Opper, M., Knochenhauer, C., Oala, L., and Samek, W. (2024). Generative Fractional Diffusion Models. Neural Information Processing Systems (NeurIPS) 2024.
   The work done in this publication is outside the scope of this dissertation.

5. **Daems, R.**, Opper, M., Crevecoeur, G., and Birdal, T. (2024). Variational Inference for SDEs Driven by Fractional Noise. International Conference on Learning Representations (ICLR) 2024.

6. Seys, S. F., Gorris, S., Uyttebroek, S., Backaert, W., Jorissen, M., Schrijvers, R., **Daems, R.**, Loeckx, D., Van Gerven, L., and Hellings, P. W. (2023). Evaluation of skin prick location on the forearm using a novel skin prick automated test device. Frontiers in Allergy, 4, 1289031.
   The work done in this publication is outside the scope of this dissertation.

7. **Daems, R.**, Taets, J., wyffels, F., and Crevecoeur, G. (2022). KeyCLD: Learning Constrained Lagrangian Dynamics in Keypoint Coordinates from Images. Neurocomputing, 573 (2024): 127175.

8. De Roovere, P., **Daems, R.**, Croenen, J., Bourgana, T., de Hoog, J., and wyffels, F. (2022). CenDerNet: Center and Curvature Representations for Render-and-Compare 6D Pose Estimation. Proceedings of the European Conference on Computer Vision (ECCV) Workshops 2022.
   The work done in this publication is outside the scope of this dissertation.

9. **Daems, R.**, and wyffels, F. (2020). Unsupervised Orientation Learning Using Autoencoders. Differential Geometry meets Deep Learning Workshop at NeurIPS 2020.
   The work done in this publication is outside the scope of this dissertation.

10. **Daems, R.**, Victor, J., De Baets, P., Van Onsem, S., and Verstraete, M. Validation of three-dimensional total knee replacement kinematics measurement using single-plane fluoroscopy. International Journal of Sustainable Construction and Design, 7.1 (2016).
The work done in this publication is outside the scope of this dissertation.

2

2

KeyCLD: Learning Constrained Lagrangian Dynamics in Keypoint Coordinates from Images

2

KeyCLD: Learning Constrained Lagrangian Dynamics in Keypoint Coordinates from Images

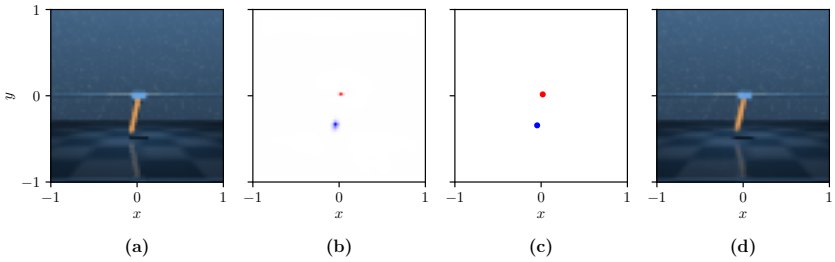


Figure 2.1 KeyCLD learns Lagrangian dynamics from images. **(a)** An observation of a dynamical system is processed by a keypoint estimator model. **(b)** The model represents the positions of the keypoints with a set of spatial probability heatmaps. **(c)** Cartesian coordinates are extracted using spatial softmax and used as state representations to learn Lagrangian dynamics. **(d)** The information in the keypoint coordinates bottleneck suffices for a learned renderer model to reconstruct the original observation, including background, reflections and shadows. The keypoint estimator model, Lagrangian dynamics models and renderer model are jointly learned unsupervised on sequences of images.

We present KeyCLD, a framework to learn Lagrangian dynamics from images. Learned keypoints represent semantic landmarks in images and can directly represent state dynamics. We show that interpreting this state as Cartesian coordinates, coupled with explicit holonomic constraints, allows expressing the

dynamics with a constrained Lagrangian. KeyCLD is trained unsupervised end-to-end on sequences of images. Our method explicitly models the mass matrix, potential energy and the input matrix, thus allowing energy based control. We demonstrate learning of Lagrangian dynamics from images on the dm_control pendulum, cartpole and acrobot environments. KeyCLD can be learned on these systems, whether they are unactuated, underactuated or fully actuated. Trained models are able to produce long-term video predictions, showing that the dynamics are accurately learned. We compare with Lag-VAE, Lag-caVAE and HGN, and investigate the benefit of the Lagrangian prior and the constraint function. KeyCLD achieves the highest valid prediction time on all benchmarks. Additionally, a very straightforward energy shaping controller is successfully applied on the fully actuated systems.

This chapter addresses the first research goal, namely how to incorporate physics priors in the form of Lagrangian dynamics, while learning from video (see Sec. 1.8). The models are defined in continuous-time, as explained and motivated in Sec. 1.2. For a more comprehensive introduction to Lagrangian dynamics, see Sec. 1.3. The work in this chapter is based on Daems, Taets, et al. (2024).

2.1 Introduction and Related Work

Learning dynamical models from data is a crucial aspect while striving towards intelligent agents interacting with the physical world. Understanding the dynamics and being able to predict future states is paramount for controlling autonomous systems or robots interacting with their environment. For many dynamical systems, the equations of motion can be derived from scalar functions such as the Lagrangian or Hamiltonian. This strong physics prior enables more data-efficient learning and holds energy conserving properties. Greydanus et al. (2019) introduced Hamiltonian neural networks. By using Hamiltonian mechanics as inductive bias, the model respects exact energy conservation laws. Michael Lutter et al. (2018) and Lutter et al. (2019) pioneered

the use of Lagrangian mechanics as physics priors for learning dynamical models from data. Cranmer et al. (2020) expanded this idea to a more general setting. By modelling the Lagrangian itself with a neural network instead of explicitly modelling mechanical kinetic energy, they can model physical systems beyond classical mechanics. Finzi et al. (2020) introduced learning of Lagrangian or Hamiltonian dynamics in Cartesian coordinates, with explicit constraints. This enables more data efficient models, at the cost of providing extra knowledge about the system in the form of a constraint function.

Zhong et al. (2020) included external input forces and energy dissipation, and introduced energy-based control by leveraging the learned energy models.

2.1.1 Learning Lagrangian dynamics from images

It is often not possible to observe the full state of a system directly. Cameras provide a rich information source, containing the full state when properly positioned. However, the difficulty lies in interpreting the images and extracting the underlying state. As was recently argued by Lutter and Peters (2021), learning Lagrangian or Hamiltonian dynamics from realistic images remains an open challenge. The majority of related work (Greydanus et al. (2019), Toth et al. (2020), Saemundsson et al. (2020), Allen-Blanchette et al. (2020), and Botev et al. (2021)) use a variational autoencoder (VAE) framework to represent the state in a latent space embedding. The dynamics model is expressed in this latent space. Zhong and Leonard (2020) use interpretable coordinates, however their state estimator module needs full knowledge of the kinematic chain, and the images are segmented per object. Tab. 2.1 provides an overview of closely related work in literature.

2.1.2 Keypoints

Instead of using abstract latent embeddings, our method leverages fully convolutional keypoint estimator models to observe the

Table 2.1 An overview of closely related Lagrangian or Hamiltonian models. Lag-caVAE (Zhong and Leonard (2020)) is capable of modelling external forces and learning from images, but individual moving bodies need to be segmented in the images, on a black background. It additionally needs full knowledge of the kinematic chain, which is more prior information than the constraint function for our method (see Section 2.2.3). HGN (Toth et al. (2020)) needs no prior knowledge of the kinematic chain, but is unable to model external forces. CHNN (Finzi et al. (2020)) expresses Lagrangian or Hamiltonian dynamics in Cartesian coordinates, but can not be learned from images. Our method, KeyCLD, is capable of learning Lagrangian dynamics with external forces, from unsegmented images with shadows, reflections and backgrounds.

	HGN	Lag-caVAE	CHNN	KeyCLD
External forces (control)		✓		✓
Interpretable coordinates		✓	✓	✓
Cartesian coordinates			✓	✓
Learns from images	✓	✓		✓
Learns from unsegmented images	✓			✓
Needs kinematic chain prior		✓		
Needs constraint prior			✓	✓

state from images. Objects can be represented with one or more keypoints, fully capturing the position and orientation. Because the model is fully convolutional, it is also translation equivariant, hence exhibiting higher data efficiency. Zhou et al. (2019) used keypoints for object detection, with great success. Keypoint detectors are commonly used for human pose estimation (Zheng et al. (2020)). More closely related to this work, keypoints can be learned for control and robotic manipulation (B. Chen et al. (2021) and Vecerik et al. (2021)). Minderer et al. (2019) learn unsupervised keypoints from videos to represent objects and dynamics. Jaques et al. (2021) leverage keypoints for system identification

and dynamic modelling. Jakab et al. (2018) learn a keypoint representation unsupervised by using it as an information bottleneck for reconstructing images. The keypoints represent semantic landmarks in the images and generalize well to unseen data. It is the main inspiration for the use of keypoints in our work.

2.1.3 Contributions

Concretely, our work makes the following contributions. **(1)** We introduce KeyCLD, a framework to learn constrained Lagrangian dynamics from images. We are the first to use learned keypoint representations from images to learn Lagrangian dynamics. We show that keypoint representations derived from images can directly be used as positional state vector for constrained Lagrangian dynamics, expressed in Cartesian coordinates. **(2)** We show how to control constrained Lagrangian dynamics in Cartesian coordinates with energy shaping, where the state is estimated from images. **(3)** KeyCLD is empirically validated on the pendulum, cartpole and acrobot systems from $\mathrm{dm\_control}$ (Tunyasuvunakool et al. (2020)). We show that KeyCLD can be learned on these systems, whether they are unactuated, underactuated or fully actuated. We compare quantitatively with Lag-caVAE, Lag-VAE (Zhong and Leonard (2020)) and HGN (Toth et al. (2020)), and investigate the benefit of the Lagrangian prior and the constraint function. KeyCLD achieves the highest valid prediction time on all benchmarks (see Tab. 2.2).

2.2 Constrained Lagrangian Dynamics

2.2.1 Lagrangian Dynamics

For a dynamical system with m degrees of freedom, a set of independent generalized coordinates $\mathbf{q} \in \mathbb{R}^m$ represents all possible kinematic configurations of the system. The time derivatives $\dot{\mathbf{q}} \in \mathbb{R}^m$ are the velocities of the system. If the system is fully deterministic, its dynamics can be described by the equations

of motion, a set of second order ordinary differential equations (ODEs):

$$\ddot{\mathbf{q}} = \mathbf{f}(\mathbf{q}(t), \dot{\mathbf{q}}(t), t, \mathbf{u}) \,, \tag{2.1}$$

where $\mathbf{u}$ are the external forces acting on the system. From a known initial value $(\mathbf{q}, \dot{\mathbf{q}})$, we can integrate $\mathbf{f}$ through time to predict future states of the system. It is possible to model $\mathbf{f}$ with a neural network, and train the parameters with backpropagation through an ODE solver (R. T. Chen et al. (2018)).

However, by expressing the dynamics with a Lagrangian we introduce a strong physics prior (Lutter et al. (2019)):

$$\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}) = T(\mathbf{q}, \dot{\mathbf{q}}) - V(\mathbf{q}) \,, \tag{2.2}$$

where T is the kinetic energy and V is the potential energy of the system. For any mechanical system the kinetic energy is defined as:

$$T(\mathbf{q}, \dot{\mathbf{q}}) = \frac{1}{2} \dot{\mathbf{q}}^\top \mathbf{M}(\mathbf{q}) \dot{\mathbf{q}} \,, \tag{2.3}$$

where $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{m \times m}$ is the positive semi-definite mass matrix. Ensuring that $\mathbf{M}(\mathbf{q})$ is positive semi-definite can be done by expressing $\mathbf{M}(\mathbf{q}) = \mathbf{L}(\mathbf{q})\mathbf{L}(\mathbf{q})^\top$, where $\mathbf{L}(\mathbf{q})$ is a lower triangular matrix. It is now possible to describe the dynamics with two neural networks, one for the mass matrix and one for the potential energy. Since both are only in function of $\mathbf{q}$ and not $\dot{\mathbf{q}}$, and expressing the mass matrix and potential energy is more straightforward than expressing the equations of motion, it is generally much more simple to learn dynamics with this framework. In other words, adding more physics priors in the form of Lagrangian mechanics, makes learning the dynamics more robust and data-efficient (Michael Lutter et al. (2018), Lutter et al. (2019), Cranmer et al. (2020), and Lutter and Peters (2021)).

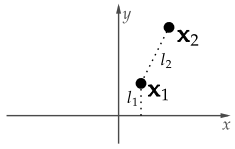
The Euler-Lagrange equations (2.4) allow transforming the Lagrangian into the equations of motion by solving for $\ddot{\mathbf{q}}$:

$$\frac{\mathrm{d}}{\mathrm{d}t} \nabla_{\dot{\mathbf{q}}} \mathcal{L} - \nabla_{\mathbf{q}} \mathcal{L} = \nabla_{\mathbf{q}} W \,, \tag{2.4}$$

$$\nabla_{\mathbf{q}} W = \mathbf{g}(\mathbf{q}) \mathbf{u} \,, \tag{2.5}$$

where W is the external work done on the system, *e.g.* forces applied for control. The input matrix $\mathbf{g} \in \mathbb{R}^{m \times l}$ allows introducing external forces $\mathbf{u} \in \mathbb{R}^l$ for modelling any control-affine system. If the external forces and torques are aligned with the degrees of freedom $\mathbf{q}$, $\mathbf{g}$ can be a diagonal matrix or even an identity matrix. More generally, if no prior knowledge is present about the relationship between $\mathbf{u}$ and the generalized coordinates $\mathbf{q}$, $\mathbf{g}(\mathbf{q}) : \mathbb{R}^m \to \mathbb{R}^{m \times l}$ is a function of $\mathbf{q}$ and can be modelled with a third neural network Zhong et al. 2020. If the system is fully actuated, the number of actuaters equals the number of degrees of freedom ($l = m$), if it is underactuated not all degrees of freedom are actuated ($l < m$).

2.2.2 Cartesian coordinates



$$\boldsymbol{\Phi}(\mathbf{x}) = \begin{bmatrix} \mathbf{e}_y \mathbf{x}_1 - l_1 \\ \|\mathbf{x}_2 - \mathbf{x}_1\|^2 - l_2 \end{bmatrix}$$

Figure 2.2 Example of a constraint function $\boldsymbol{\Phi}(\mathbf{x})$ to express the cartpole system with Cartesian coordinates. The cartpole system has 2 degrees of freedom, but is expressed in $\mathbf{x} \in \mathbb{R}^4$. Valid configurations of the system in $\mathbb{R}^4$ are constrained on a manifold defined by $\mathbf{0} = \boldsymbol{\Phi}(\mathbf{x})$. The first constraint only allows horizontal movement of $\mathbf{x}_1$, and the second constraint enforces a constant distance between $\mathbf{x}_1$ and $\mathbf{x}_2$. Although unknown l_1 and l_2 constants are present in $\boldsymbol{\Phi}(\mathbf{x})$, their values are irrelevant, since only the Jacobian of $\boldsymbol{\Phi}(\mathbf{x})$ is used in our framework (see equation (2.14)). See 2.7.4 for more examples of constraint functions.

Finzi et al. (2020) showed that expressing Lagrangian mechanics in Cartesian coordinates $\mathbf{x} \in \mathbb{R}^k$ instead of independent generalized coordinates $\mathbf{q} \in \mathbb{R}^m$ has several advantages:

$$\mathcal{L}(\mathbf{x}, \dot{\mathbf{x}}) = \frac{1}{2} \dot{\mathbf{x}}^\top \mathbf{M} \dot{\mathbf{x}} - V(\mathbf{x}) \,. \tag{2.6}$$

The mass matrix $\mathbf{M}$ no longer changes in function of the state, and is thus static. This means that a neural network is no longer required to model the mass matrix, simply the values in the matrix itself are optimized. The potential energy $V(\mathbf{x})$ is now in function of $\mathbf{x}$. Expressing the potential energy in Cartesian coordinates can often be simpler than in generalized coordinates. *e.g.*, for gravity, this is simply a linear function. Also the input matrix $\mathbf{g}(\mathbf{x})$ is now in function of $\mathbf{x}$, and the shape is $k \times l$. Multiplying the input matrix with the input vector $\mathbf{g}(\mathbf{x})\mathbf{u}$ results in force vectors acting in Cartesian space. The input matrix is modelled with a fully connected neural network, of which the output vector is reshaped in the correct shape.

Because we are now expressing the system in Cartesian coordinates we additionally need a set of n holonomic constraint functions $\mathbf{\Phi}(\mathbf{x}) : \mathbb{R}^k \rightarrow \mathbb{R}^n$. These guarantee a valid configuration of the system, and a correct number of degrees of freedom: $m = k-n$ (see Fig. 2.2). The constrained Euler-Lagrange equations are expressed with a vector $\boldsymbol{\lambda} \in \mathbb{R}^n$ containing Lagrange multipliers for the constraints (Finzi et al. (2020) and Lanczos (2020)):

$$\frac{\mathrm{d}}{\mathrm{d}t}\nabla_{\dot{\mathbf{x}}}\mathcal{L}(\mathbf{x},\dot{\mathbf{x}}) - \nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x},\dot{\mathbf{x}}) = \mathbf{g}(\mathbf{x})\mathbf{u} + D\mathbf{\Phi}^\top(\mathbf{x})\boldsymbol{\lambda}\,, \tag{2.7}$$

with D being the Jacobian operator. Because the mass matrix is static[1], this is simplified to:

$$\mathbf{M}\ddot{\mathbf{x}} + \nabla_{\mathbf{x}}V(\mathbf{x}) = \mathbf{g}(\mathbf{x})\mathbf{u} + D\mathbf{\Phi}^\top(\mathbf{x})\boldsymbol{\lambda}\,, \tag{2.8}$$

$$\ddot{\mathbf{x}} = \mathbf{M}^{-1}\mathbf{f}(\mathbf{x}) + \mathbf{M}^{-1}D\mathbf{\Phi}^\top(\mathbf{x})\boldsymbol{\lambda}\,,\ \mathbf{f}(\mathbf{x}) = -\nabla_{\mathbf{x}}V(\mathbf{x}) + \mathbf{g}(\mathbf{x})\mathbf{u}\,. \tag{2.9}$$

Calculating twice the time derivative of the constraint conditions yields:

$$\begin{aligned}
\mathbf{0} &\equiv \mathbf{\Phi}(\mathbf{x}) \\
\mathbf{0} &= \dot{\mathbf{\Phi}}(\mathbf{x}) \\
\mathbf{0} &= D\mathbf{\Phi}(\mathbf{x})\dot{\mathbf{x}} \\
\mathbf{0} &= D\dot{\mathbf{\Phi}}(\mathbf{x})\dot{\mathbf{x}} + D\mathbf{\Phi}(\mathbf{x})\ddot{\mathbf{x}}\,.
\end{aligned} \tag{2.10}$$

1. In other words, the centrifugal and Coriolis forces are zero because $\dot{\mathbf{M}} = \mathbf{0}$ and $\nabla_{\mathbf{x}}\mathbf{M} = \mathbf{0}$.

The Lagrange multipliers $\boldsymbol{\lambda}$ are solved by substituting $\ddot{\mathbf{x}}$ from equation (2.9) in equation (2.10):

$$-D\dot{\boldsymbol{\Phi}}(\mathbf{x})\dot{\mathbf{x}} = D\boldsymbol{\Phi}(\mathbf{x})\mathbf{M}^{-1}\mathbf{f}(\mathbf{x}) + D\boldsymbol{\Phi}(\mathbf{x})\mathbf{M}^{-1}D\boldsymbol{\Phi}^{\top}(\mathbf{x})\boldsymbol{\lambda}, \tag{2.11}$$

$$\boldsymbol{\lambda} = \left[D\boldsymbol{\Phi}(\mathbf{x})\mathbf{M}^{-1}D\boldsymbol{\Phi}^{\top}(\mathbf{x})\right]^{-1}\left[D\boldsymbol{\Phi}(\mathbf{x})\mathbf{M}^{-1}\mathbf{f}(\mathbf{x}) + D\dot{\boldsymbol{\Phi}}(\mathbf{x})\dot{\mathbf{x}}\right] \tag{2.12}$$

We use the chain rule a second time to get rid of the time derivative of $D\boldsymbol{\Phi}(\mathbf{x})$:

$$D\dot{\boldsymbol{\Phi}}(\mathbf{x})\dot{\mathbf{x}} = \langle D^2\boldsymbol{\Phi}(\mathbf{x}), \dot{\mathbf{x}}\rangle\dot{\mathbf{x}}\,. \tag{2.13}$$

Substituting $\boldsymbol{\lambda}$ in (2.9) we finally arrive at:

$$\ddot{\mathbf{x}} = \mathbf{M}^{-1}\mathbf{f}(\mathbf{x}) - \mathbf{M}^{-1}D\boldsymbol{\Phi}^{\top}(\mathbf{x})\left[D\boldsymbol{\Phi}(\mathbf{x})\mathbf{M}^{-1}D\boldsymbol{\Phi}^{\top}(\mathbf{x})\right]^{-1} \cdot \left[D\boldsymbol{\Phi}(\mathbf{x})\mathbf{M}^{-1}\mathbf{f}(\mathbf{x}) + \langle D^2\boldsymbol{\Phi}(\mathbf{x}), \dot{\mathbf{x}}\rangle\dot{\mathbf{x}}\right]\,. \tag{2.14}$$

Since time derivatives of functions modelled with neural networks are no longer present, equation (2.14) can be implemented in an autograd library which handles the calculation of gradients and Jacobians automatically. See 2.7.1 for details and the implementation of equation (2.14) in JAX (Bradbury et al. (2018)).

Note that in equation (2.14) only the Jacobian of $\boldsymbol{\Phi}(\mathbf{x})$ is present. This means that there is no need to learn explicit constants in $\boldsymbol{\Phi}(\mathbf{x})$, such as lengths or distances between points. Rather that constant distances and lengths through time are enforced by $D\boldsymbol{\Phi}(\mathbf{x})\dot{\mathbf{x}} = \mathbf{0}$. We use this property to our advantage since this simplifies the learning process. See Fig. 2.2 for an example of the cartpole system expressed in Cartesian coordinates and a constraint function.

2.2.3 Constraints as prior knowledge

The given constraint function $\boldsymbol{\Phi}(\mathbf{x})$ adds extra prior information to our model. Alternatively, we could use a mapping function $\mathbf{x} = \mathbf{F}(\mathbf{q})$. This leads directly to an expression of the Lagrangian in Cartesian coordinates using $\dot{\mathbf{x}} = D\mathbf{F}(\mathbf{q})\dot{\mathbf{q}}$:

$$\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}) = \frac{1}{2}\dot{\mathbf{q}}^{\top}D\mathbf{F}(\mathbf{q})^{\top}\mathbf{M}D\mathbf{F}(\mathbf{q})\dot{\mathbf{q}} - V(\mathbf{F}(\mathbf{q}))\,, \tag{2.15}$$

from which the equations of motion can be derived using the Euler-Lagrange equations, similar to equation (2.14). In terms of explicit knowledge about the system, the mapping $\mathbf{x} = \mathbf{F}(\mathbf{q})$ is equivalent to the kinematic chain as required for the method of Zhong and Leonard (2020). Using the constraint function is however more general. Some systems, such as systems with closed loop kinematics, can not be expressed in generalized coordinates $\mathbf{q}$, and thus have no mapping function (Betsch (2005)). Furthermore, learning the constraint manifold $\mathbf{0} = \mathbf{\Phi}(\mathbf{x})$ from data with geometric manifold learning algorithms (B. Chen et al. (2022) and Levina and Bickel (2004)) could be a future research direction. We therefore argue that adopting the constraint function $\mathbf{\Phi}(\mathbf{x})$ is more general and requires less explicit knowledge injected in the model.

2.2.4 Relationship between Lagrangian and Hamiltonian

Both Lagrangian and Hamiltonian mechanics ultimately express the dynamics in terms of kinetic and potential energy. The Hamiltonian expresses the total energy of the system $H(\mathbf{q}, \mathbf{p}) = T(\mathbf{q}, \mathbf{p}) + V(\mathbf{q})$ (Greydanus et al. (2019) and Toth et al. (2020)). It is expressed in the position and the generalized momenta $(\mathbf{q}, \mathbf{p})$, instead of generalized velocities. Using the Legendre transformation it is possible to transform L into H or back. We focus in our work on Lagrangian mechanics because it is more general (Cranmer et al. (2020)) and observing the momenta $\mathbf{p}$ is impossible from images. See also Botev et al. (2021) for a short discussion on the differences.

2.3 Learning Lagrangian Dynamics from Images

2.3.1 Keypoints as state representations

We introduce the use of keypoints to learn Lagrangian dynamics from images. KeyCLD is trained unsupervised on sequences of n images $\{\mathbf{z}^i\}, i \in \{1, \dots, n\}$ and a constant input vector $\mathbf{u}$. See Fig. 2.3 for a schematic overview.

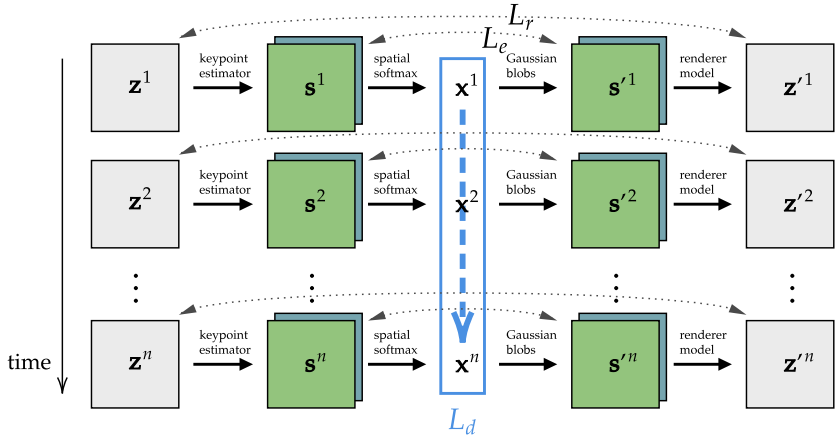


Figure 2.3 Schematic overview of training KeyCLD. A sequence of n images $\{\mathbf{z}^i\}, i \in \{1, \ldots, n\}$, is processed by the keypoint estimator model, returning heatmaps $\{\mathbf{s}^i\}$ representing spatial probabilities of the keypoints. $\mathbf{s}^i$ consists of m heatmaps $\mathbf{s}_k^i$, one for every keypoint $\mathbf{x}_k^i, k \in \{1, \ldots, m\}$. Spatial softmax is used to extract the Cartesian coordinates of the keypoints, and all keypoints are concatenated in the state vector $\mathbf{x}^i$. $\mathbf{x}^i$ is transformed back to a spatial representation $\mathbf{s}'^i$ using Gaussian blobs. This prior is encouraged on the keypoint estimator model by a binary cross-entropy loss L_e between $\mathbf{s}^i$ and $\mathbf{s}'^i$. The renderer model reconstructs images $\mathbf{z}'^i$ based on $\mathbf{s}'^i$, with reconstruction loss L_r. The dynamics loss L_d is calculated on the sequence of state vectors $\mathbf{x}^i$. Keypoint estimator model, renderer model and the dynamics models (mass matrix, potential energy and input matrix) are trained jointly with a weighted sum of the losses $L = L_r + L_e + \lambda L_d$.

All images $\mathbf{z}^i$ in the sequence are processed by the keypoint estimator model, returning each a set of heatmaps $\mathbf{s}^i$ representing the spatial probabilities of keypoint positions. $\mathbf{s}^i$ consists of m heatmaps $\mathbf{s}_k^i$, one for every keypoint $\mathbf{x}_k^i, k \in \{1, \ldots, m\}$. The keypoint estimator model is a fully convolutional neural network, maintaining a spatial representation from input to output (see Fig. 2.4 for the detailed architecture). This contrasts with a model ending in fully connected layers regressing to the coordinates

directly, where the spatial representation is lost (Toth et al. (2020) and Zhong and Leonard (2020)). Because a fully convolutional model is equivariant to translation, it can better generalize to unseen states that are translations of seen states. Another advantage is the possibility of augmenting $\mathbf{z}$ with random transformations of the D_4 dihedral group to increase robustness and data efficiency. Because $\mathbf{s}$ can be transformed back with the inverse transformation, this augmentation is confined to the keypoint estimator model and has no effect on the dynamics.

To distill keypoint coordinates from the heatmaps, we define a Cartesian coordinate system in the image (see for example Fig. 2.1). Based on this definition, every pixel $\mathbf{p}$ corresponds to a point $\mathbf{x}_\mathbf{p}$ in the Cartesian space. The choice of the Cartesian coordinate system is arbitrary but is equal to the space of the dynamics $\ddot{\mathbf{x}}(\dot{\mathbf{x}}, \mathbf{x}, t, \mathbf{u})$ and the constraint function $\mathbf{\Phi}(\mathbf{x})$ (see Section 2.2). We use spatial softmax over all pixels $\mathbf{p} \in \mathcal{P}$ to distill the coordinates of keypoint $\mathbf{x}_k$ from its probability heatmap:

$$\mathbf{x}_k = \frac{\sum_{\mathbf{p}\in\mathcal{P}} \mathbf{x}_\mathbf{p} e^{\mathbf{s}_k(\mathbf{p})}}{\sum_{\mathbf{p}\in\mathcal{P}} e^{\mathbf{s}_k(\mathbf{p})}} \,. \tag{2.16}$$

Spatial softmax is differentiable, and the loss will backpropagate through the whole heatmap since $\mathbf{x}_k$ depends on all the pixels. Cartesian coordinates $\mathbf{x}_k$ of the different keypoints are concatenated in vector $\mathbf{x}$ which serves as the state representation of the system. *This compelling connection between image keypoints and Cartesian coordinates forms the basis of this work.* The keypoint estimator model serves directly as state estimator to learn constrained Lagrangian dynamics from images.

Similar to Jakab et al. (2018), $\mathbf{x}$ acts as an information bottleneck, through which only the Cartesian coordinates of the keypoints flow to reconstruct the image with the renderer model. First, all $\mathbf{x}_k$ are transformed back to spatial representations $\mathbf{s}'_k$ using unnormalized Gaussian blobs, parameterized by a hyperparameter σ.

$$\mathbf{s}'_k = \exp\left(-\frac{\|\mathbf{x}_\mathbf{p} - \mathbf{x}_k\|^2}{2\sigma^2}\right) \,. \tag{2.17}$$

A binary cross-entropy loss L_e is formulated over $\mathbf{s}$ and $\mathbf{s}'$ to encourage this Gaussian prior. The renderer model can more easily

interpret the state in this spatial representation, as it lies closer to its semantic meaning of keypoints as semantic landmarks in the reconstructed image. The renderer model learns a constant feature tensor (inspired by Nguyen-Phuoc et al. (2019)), which provides it with positional information. Since the model itself is translation equivariant, it needs positional information to reconstruct background information or specific appearances that depend on the positions of objects. See Fig. 2.4 for the detailed architecture.

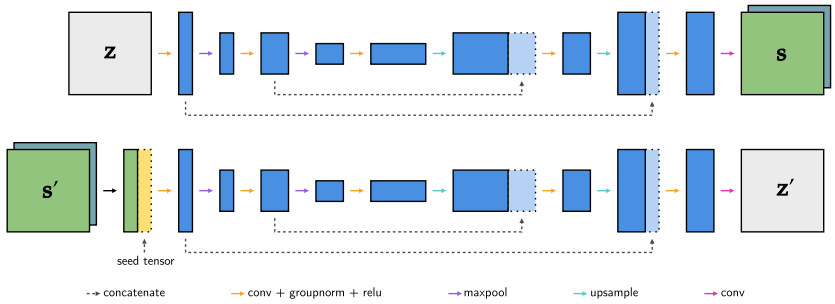


Figure 2.4 Visualization of the keypoint estimator (top) and renderer (bottom) model architectures. The keypoint estimator model and renderer model have similar architectures, utilizing down- and upsampling and skip connections which help increasing the receptive field Gu et al. 2019; Newell et al. 2016. The renderer model learns a constant feature tensor that is concatenated with the input $\mathbf{s}'$. The feature tensor provides positional information since the fully-convolutional model is translation equivariant.

Finally, a reconstruction loss is formulated over the reconstructed images $\mathbf{z}'^i$ and original images $\mathbf{z}^i$:

$$L_r = \sum_{i=1}^{n} \|\mathbf{z}'^i - \mathbf{z}^i\|^2 . \tag{2.18}$$

2.3.2 Dynamics loss function

The sequence $\{\mathbf{x}^i\}$, corresponding to the sequence of given images $\{\mathbf{z}^i\}$, and the constant input $\mathbf{u}$ is used to calculate the dy-

namics loss. A fundamental aspect in learning dynamics from images is that velocities can not be directly observed. A single image only captures the position of a system, and contains no information about its velocities[2]. Other work uses sequences of images as input to a model (Toth et al. (2020)) or a specific velocity estimator model trained to estimate velocities from a sequence of positions (Jaques et al. (2019)). Zhong and Leonard (2020) demonstrate that for estimating velocities, finite differencing simply works better.

We use a central first order finite difference estimation, and project the estimated velocity on the constraints, so that the constraints are not violated:

$$\dot{\mathbf{x}}^i = \left[\boldsymbol{I} - D\boldsymbol{\Phi}(\mathbf{x}^i)^+ D\boldsymbol{\Phi}(\mathbf{x}^i)\right] \frac{\mathbf{x}^{i+1} - \mathbf{x}^{i-1}}{2h},\ i = 2, \ldots, n-1\,, \quad (2.19)$$

where $(\cdot)^+$ signifies the Moore-Penrose pseudo-inverse and h the timestep. We can now integrate future timesteps $\hat{\mathbf{x}}$ starting from initial values $(\mathbf{x}^i, \dot{\mathbf{x}}^i)$ using an ODE solver. The equations of motion (2.14) are solved starting from all initial values in parallel, for ν timesteps This maximizes the learning signal obtained to learn the dynamics and leads to overlapping sequences of length ν:

$$\{\hat{\mathbf{x}}^{i+1}, \ldots, \hat{\mathbf{x}}^{i+\nu}\}, \quad i = 2, \ldots, n - \nu\,. \quad (2.20)$$

Thus, $\hat{\mathbf{x}}^{i+j}$ is obtained by integrating j timesteps forward in time, starting from initial value $\mathbf{x}^i$, which was derived by the keypoint estimator model. All $\{\hat{\mathbf{x}}^{i+j}\}$ in all sequences are compared with their corresponding keypoint states $\{\mathbf{x}^{i+j}\}$ in an L_2 loss:

$$L_d = \sum_{i=2}^{n-\nu} \sum_{j=1}^{\nu} \left\| \mathbf{x}^{i+j} - \hat{\mathbf{x}}^{i+j} \right\|^2 . \quad (2.21)$$

2.3.3 Total loss

The total loss is the weighted sum of L_r, L_e and L_d, with a weighing hyperparameter λ: $L = L_r + L_e + \lambda L_d$.

2. Neglecting side-effects such as motion blur, which are not very useful for this purpose.

To conclude, the keypoint estimator model, renderer model and dynamics models (mass matrix, potential energy and input matrix) are jointly trained end-to-end on sequences of images $\{\mathbf{z}^i\}$ and constant inputs $\mathbf{u}$ with stochastic gradient descent.

2.3.4 Rigid bodies as rigid sets of point masses

By interpreting a set of keypoints as a set of point masses, we can represent any rigid body and its corresponding kinetic and potential energy. Additional constraints are added for the pairwise distances between keypoints representing a single rigid body (Finzi et al. (2020)). For 3D systems, at least four keypoints are required to represent any rigid body (Laus and Selig (2020)). We focus in our work on 2D systems in a plane parallel to the camera plane. 2D rigid bodies can be expressed with a set of 2 point masses, which can further be reduced depending on the constraints and connections between bodies (see 2.7.2 for more detail and proof). In our framework, the keypoint model is free to choose the relative placement of keypoints on the different moving parts of the dynamic system, enabling the choice of distinct landmarks that also express the state accurately, *e.g.* the endpoint of a beam.

The interpretation of rigid bodies as sets of point masses allows expressing the kinetic energy as the sum of the kinetic energies of the point masses. Corresponding to equation (2.3), the mass matrix for a 2D system is defined as a diagonal matrix with masses m_k for every keypoint $\mathbf{x}_k$:

$$T(\dot{\mathbf{x}}) = \frac{1}{2}\dot{\mathbf{x}}^\top \mathbf{M}\dot{\mathbf{x}} \tag{2.22}$$

$$= \frac{1}{2}\begin{bmatrix}\dot{\mathbf{x}}_1 & \dots & \dot{\mathbf{x}}_n\end{bmatrix}\begin{bmatrix} m_1 & 0 & \dots & 0 & 0 \\ 0 & m_1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & m_n & 0 \\ 0 & 0 & \dots & 0 & m_n \end{bmatrix}\begin{bmatrix}\dot{\mathbf{x}}_1 \\ \vdots \\ \dot{\mathbf{x}}_n\end{bmatrix}. \tag{2.23}$$

To enforce positive values, the masses are parameterized by their square root and squared.

2.4 Experiments

We adapted the pendulum, cartpole and acrobot environments from dm_control (Tunyasuvunakool et al. (2020) and Todorov et al. (2012)) for our experiments. See 2.7.4 for details about the environments, their constraint functions and the data generation procedure. The exact same model architectures, hyperparameters and control parameters were used for all environments (see 2.7.5 for more details). This further demonstrates the generality and robustness of our method.

Since KeyCLD is trained directly on image observations, quantitative metrics can only be expressed in the image domain. The mean squared error (MSE) in the image domain is not a good metric of long term prediction accuracy (Minderer et al. (2019) and Zhong and Leonard (2020)). A model that trivially learns to predict a static image, which is the average of the dataset, learns no dynamics at all yet this model could report a lower MSE than a model that did learn the dynamics but started drifting from the groundtruth after some time. Therefore, we use the valid prediction time (VPT) score (Botev et al. (2021) and Jin et al. (2020)) which measures how long the predicted images stay close to the groundtruth images of a sequence:

$$\text{VPT} = \text{argmin}_i[\text{MSE}(\mathbf{z'}^i, \mathbf{z}^i) > \epsilon] \,, \tag{2.24}$$

where $\mathbf{z}^i$ are the groundtruth images, $\mathbf{z'}^i$ are the predicted images and ϵ is the error threshold. ϵ is determined separately for the different environments because it depends on the relative size in pixels of moving parts. We define it as the MSE of the averaged image of the respective validation dataset. Thus it is the lower bound for a model that would simply predict a static image. We present evaluations with the following ablations and baselines:

KeyCLD The full framework as described in Sections 2.2 and 2.3.

KeyLD The constraint function is omitted.

KeyODE2 A second order neural ODE modelling the acceleration is used instead of the Lagrangian prior. The

keypoint estimator and renderer model are identical to KeyCLD.

Lag-caVAE The model presented by Zhong and Leonard (2020). We adapted the model to the higher resolution and color images.

Lag-VAE The model presented by Zhong and Leonard (2020). We adapted the model to the higher resolution and color images.

HGN Hamiltonian Generative Network presented by Toth et al. (2020).

2.4.1 Future frame predictions

We generate predictions of 50 frames, given the first 3 frames of the ground truth sequences to estimate the initial velocity according equation (2.19). The VPT metric is calculated for the 50 sequences in the validation set (see 2.7.4 for details) and averaged. See Tab. 2.2 for an overview of results. Lag-caVAE is unable to model data with background (see also Fig. 2.5). Despite our best efforts for implementation and training, Lag-VAE and HGN perform very poorly. The models are not capable of handling the relatively more challenging visual structure of $\mathrm{dm\_control}$ environments. Removing the constraint function (KeyLD) has a detrimental effect on the ability to make long-term predictions. Results are comparable to removing the Lagrangian prior altogether (KeyODE2). This suggests that modeling dynamics in Cartesian coordinates coupled with keypoint representations is in itself a very strong prior and that the effect of the Langrangian prior without constraints is minimal. This is consistent with recent findings by Gruver et al. (2021). However, using a Lagrangian formulation allows leveraging a constraint function, since a general neural ode model can not make use of a constraint function. Thus, if a constraint function is available, the Lagrangian prior becomes much more powerful.

One sequence of each experiment is visualized in the paper. Please compare these qualitative results for the unactuated and actuated pendulum environment (Fig. 2.5, 2.6), unactuated,

Table 2.2 Valid prediction time (VPT) (higher is better, equation (2.24)) in number of predicted frames (mean $\pm$ std) for the different models evaluated on the 50 sequences in the validation set. Lag-caVAE and Lag-VAE are only reported on the pendulum environment, since they are unable to model more than one moving body without segmented images. HGN is only reported on non-actuated systems, since it is incapable of modelling external forces and torques. KeyCLD achieves the best results on all benchmarks.

	# actuators	**KeyCLD**	KeyLD	KeyODE2	Lag-caVAE	Lag-VAE	HGN
Pendulum	0 (Fig. 2.5)	**43.1±9.7**	16.4±11.3	19.1±6.2	0.0±0.0	10.8±13.8	0.2±1.4
	1 (Fig. 2.6)	**39.3±9.8**	14.9±7.9	12.0±4.1	0.0±0.1	8.0±10.2	-
Cartpole	0 (Fig. 2.7)	**39.9±7.4**	29.8±11.2	29.5±9.5	-	-	0.0±0.0
	1 (Fig. 2.8)	**38.4±8.7**	28.0±9.7	24.4±7.9	-	-	-
	2 (Fig. 2.9)	**30.2±10.7**	23.9±9.6	17.7±8.2	-	-	-
Acrobot	0 (Fig. 2.10)	**47.0±6.0**	40.0±7.9	34.3±9.5	-	-	2.2±6.9
	1 (Fig. 2.11)	**46.8±4.6**	29.5±6.3	33.0±7.4	-	-	-
	2 (Fig. 2.12)	**47.0±3.5**	39.1±9.9	30.8±9.3	-	-	-

underactuated and fully actuated cartpole environment (Fig. 2.7, 2.8, 2.9) and unactuated, underactuated and fully actuated acrobot environment (Fig. 2.10, 2.11, 2.12). Every third frame of the sequence is shown.

2.4.2 Learned potential energy models

Since the potential energy V is explicitly modelled, we can plot values throughout sequences of the state space. A sequence of images is processed by the learned keypoint estimator model, and the states are then used to calculate the potential energy with the learned potential energy model. Absolute values of the potential energy are irrelevant, since the potential is relative, but we gain insights by moving parts of the system separately. See Fig. 2.13 for results for the pendulum, Figs. 2.14 and 2.15 for the cartpole and Figs. 2.16 and 2.17 for the acrobot.

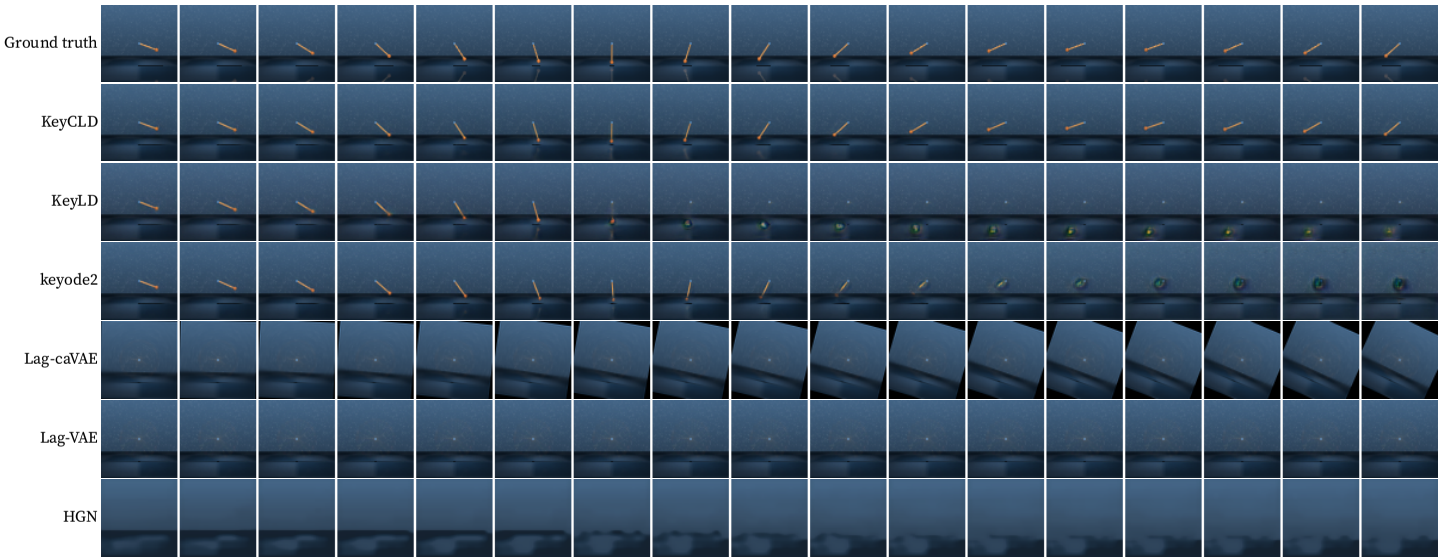


Figure 2.5 Future frame predictions of the unactuated pendulum. These correspond to the first row in Tab. 2.2. 50 frames are predicted based on the first three frames of the ground truth sequence to estimate the velocity. Every third frame of every sequence is shown. KeyCLD is capable of making accurate long-term predictions with minimal drift of the dynamics. Without constraint function, KeyLD is not capable of making long-term predictions. Similarly, keyode2 is unable of making long-term predictions. Lag-caVAE is fundamentally incapable of modelling data with background information, since the reconstructed images are explicitly rotated. Lag-VAE does not succeed in modelling moving parts in the data, and simply learns to predict static images. HGN also does not capture the dynamics and only learns the background.

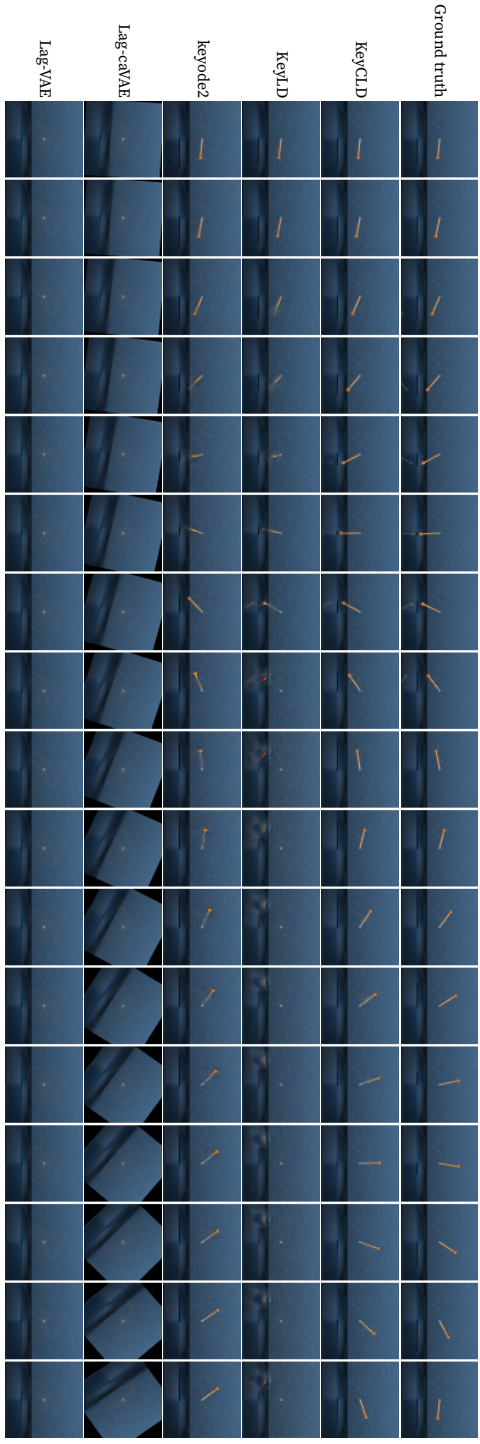


Figure 2.6 Future frame predictions of the actuated pendulum.

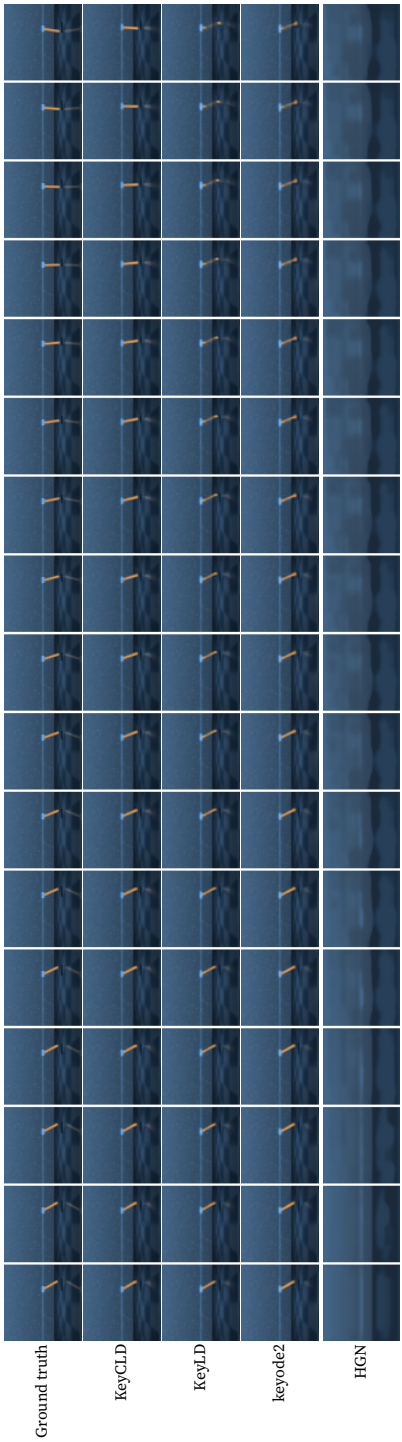


Figure 2.7 Future frame predictions of the unactuated cartpole.

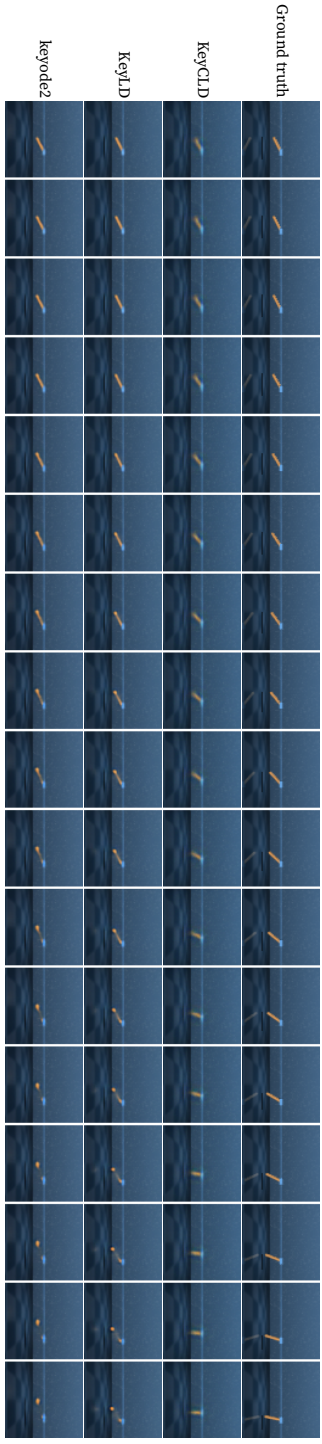


Figure 2.8 Future frame predictions of the underactuated cartpole.

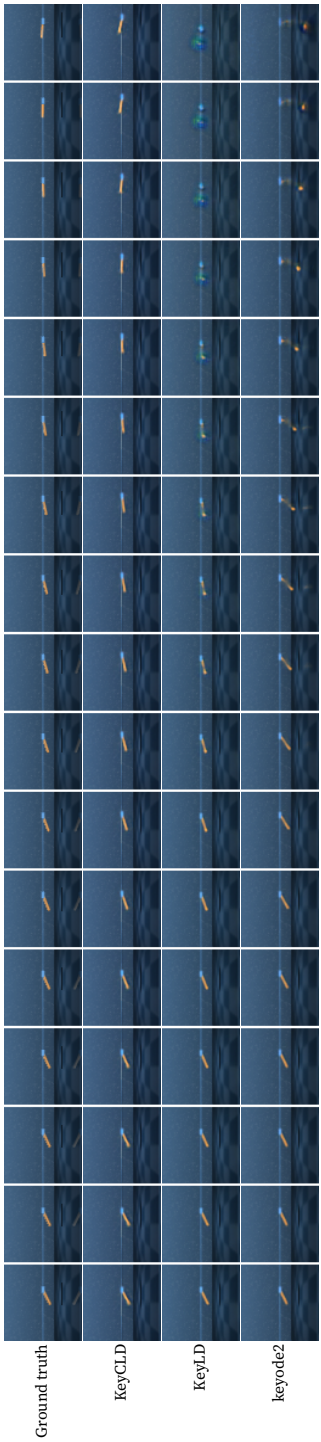


Figure 2.9 Future frame predictions of the fully actuated cartpole.

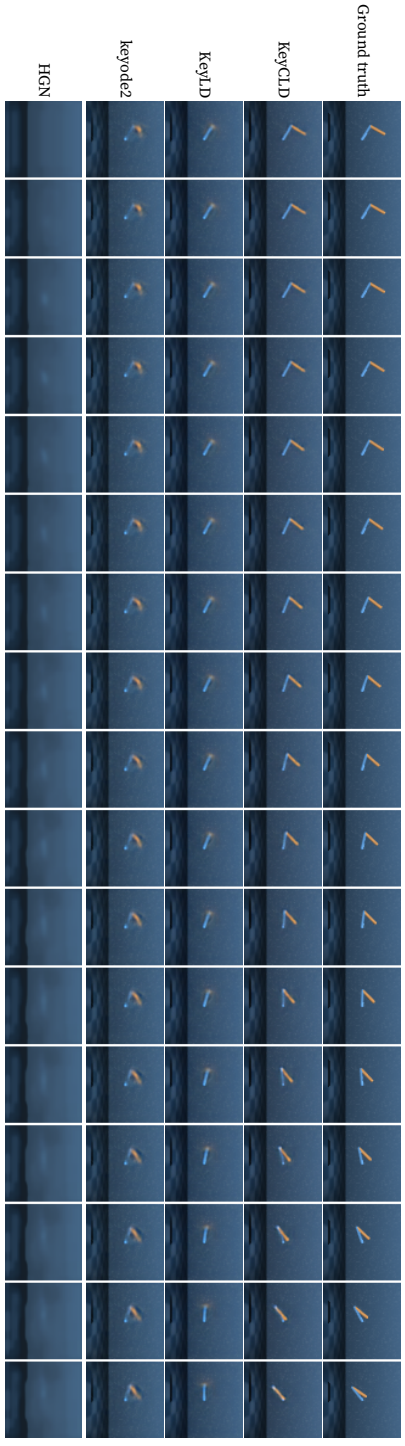


Figure 2.10 Future frame predictions of the unactuated acrobot.

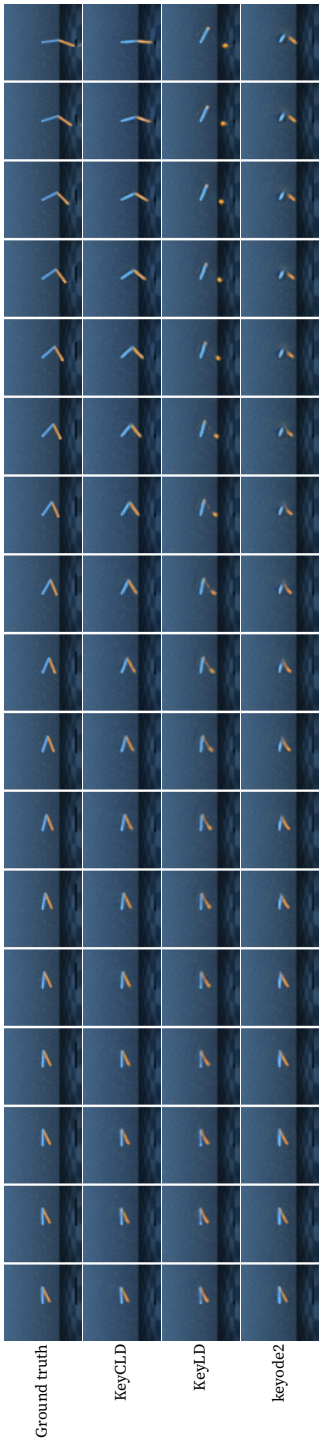


Figure 2.11 Future frame predictions of the underactuated acrobot.

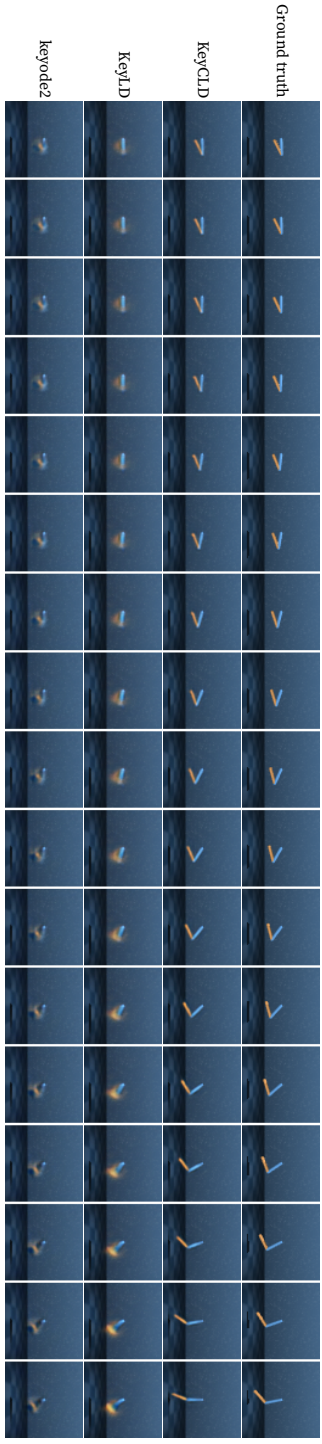


Figure 2.12 Future frame predictions of the fully actuated acrobot.

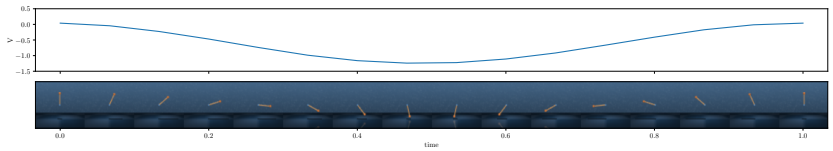


Figure 2.13 Potential energy of the trained KeyCLD model of the pendulum environment. The pendulum makes a full rotation. As expected, the potential energy follows a smooth sinusoidal path throughout this sequence. The maximum value is reached when the pendulum is upright, and the minimum value is reached when the pendulum is down.

2.4.3 Learned input matrix models

Learning the input matrix $\mathbf{g}(\mathbf{x})$ is crucial for learning dynamics models with external inputs $\mathbf{u}$. We can visualize the vector basis that is represented by the input matrix, by drawing the vectors originating on their respective keypoints. See Fig. 2.18, 2.19 and 2.20 for the input matrices that are learned with our model. Each input corresponds to a vector base, visualized in different colors. The vectors multiplied by their respective input, can be interpreted as forces acting on the keypoints. These qualitative results allow further insight in our method.

2.4.4 Energy shaping control

A major argument in favor of expressing dynamics in terms of a mass matrix and potential energy is the straightforward control design via passivity based control and energy shaping (Ortega et al. (2001)). Our framework allows design of a simple energy shaping controller, see 2.7.3 for details and derivation. Fig. 2.21 shows results of successful swing-up of the pendulum, cartpole and acrobot system. The same control parameters $k_p = 5.0$ and $k_d = 2.0$ are used for all systems, demonstrating the generality of the control method.

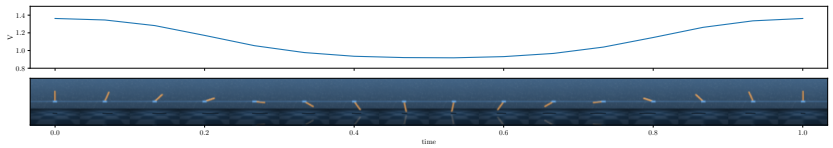


Figure 2.14 Potential energy of the trained KeyCLD model of the cartpole environment. The position of the cart is fixed, and the pole makes a full rotation. As expected, the potential energy follows a smooth sinusoidal path throughout this sequence.

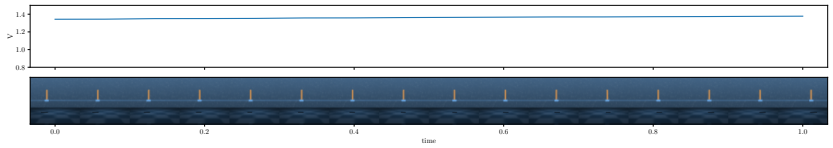


Figure 2.15 Potential energy of the trained KeyCLD model of the cartpole environment. The pole is fixed and the cart moves from left to right. As expected, the change in potential energy in this sequence is very low (compare to Fig. 2.14 with the same axis). A horizontal movement has no impact on the gravity potential.

2.5 Conclusion and Future Work

We introduce the use of keypoints to learn Lagrangian dynamics from images. Learned keypoint representations derived from images are directly used as positional state vector for learning constrained Lagrangian dynamics. The pendulum, cartpole and acrobot systems of dm_control are adapted as benchmarks. Previous works in literature on learning Lagrangian or Hamiltonian dynamics from images were benchmarked on very simple renderings of flat sprites on blank backgrounds, whereas dm_control is rendered with lighting effects, shadows, reflections and backgrounds. Also the recently proposed benchmarks by Botev et al. (2021) have minimalistic visuals and do not cover a control aspect or external forces. We believe that working towards more realistic datasets is crucial for applying Lagrangian models in the real world.

The challenge of learning Lagrangian dynamics from more complex images should not be underestimated. Despite our

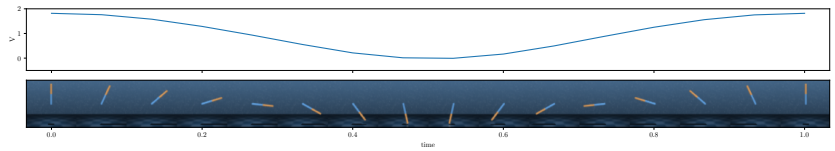


Figure 2.16 Potential energy of the trained KeyCLD model of the acrobot environment. The first link makes a full rotation, the second link is fixed relative to the first link. As expected, the potential energy follows a smooth sinusoidal path throughout this sequence.

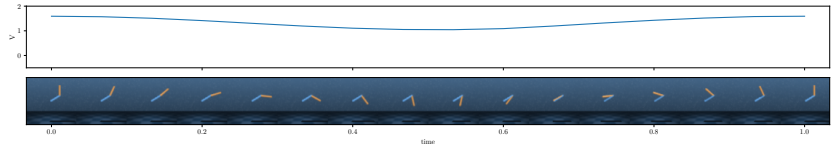


Figure 2.17 Potential energy of the trained KeyCLD model of the acrobot environment. The first link is fixed and the second link makes a full rotation. Again, the potential energy follows a smooth sinusoidal path throughout this sequence. Please compare with Fig. 2.16, where both links are moving. Here the potential energy changes less, because the first link is not moving.

best efforts in implementing and training Lag-caVAE, Lag-VAE (Zhong and Leonard (2020)) and HGN (Toth et al. (2020)), they perform very poorly on our dm_control quantitative benchmark (see Tab. 2.2). KeyCLD is capable of making long-term predictions and learning accurate energy models, suitable for simple energy shaping control. KeyCLD is compared to a Lagrangian dynamics model without constraints (KeyLD) and a general second order neural ODE (KeyODE2). Both yield worse results on the benchmark, with comparable results (see Tab. 2.2). This suggests that when no constraint prior is available, the benefit of a Lagrangian prior is limited. When the constraint function is known, the constrained Lagrangian formulation is very beneficial for long-term predictions.

The main limitations of our work are that we only consider 2D systems, where the plane of the system is parallel with the camera plane. Secondly, we do not model energy dissipation, all systems

have perfect energy conservation. Finally, the constraint function is given and we benchmark on relatively simple dynamical systems with few degrees of freedom.

Elevation to 3D, *e.g.* setups with multiple cameras, is an interesting future direction. Modelling contacts by using inequality constraints could also be a useful addition. And modelling energy dissipation is necessary for real-world applications. Several recent papers have proposed methods to incorporate energy dissipation in the Lagrangian dynamics models (Zhong et al. (2020) and Greydanus and Sosanya (2022)). However, Gruver et al. (2021) argue that modelling the acceleration directly with a second order differential equation and expressing the system in Cartesian coordinates, is a better approach. Further research into both approaches would clarify the benefit of Lagrangian and Hamiltonian priors on real-world applications. Applications on more complex scenarios with more degrees of freedom are also interesting for future work.

2.6 Broader impact

A tenacious divide exists between control engineering researchers and computer science researchers working on control. Where the first would use known equations of motion for a specific class of systems and investigate system identification, the latter would strive for the most general method with no prior knowledge. We believe this is a spectrum worth exploring, and as such use strong physics priors as Lagrangian mechanics, but still model *e.g.* the input matrix and the potential energy with arbitrary neural networks. The broad field of model-based reinforcement learning could benefit from decades of theory and practice in classic control theory and system identification. We hope this work could help bridge both worlds.

Using images as input is, in a broad sense, very powerful. Since camera sensors are consistently becoming cheaper and more powerful due to advancements in technology and scaling opportunities, we can leverage these rich information sources for a

deeper understanding of the world our intelligent agents are acting in. Image sensors can replace and enhance multiple other sensor modalities, at a lower cost.

To conclude, this work demonstrates the ability to efficiently model and control dynamical systems that are captured by cameras, with no supervision and minimal prior knowledge. We want to stress that we have shown it is possible to learn both the Lagrangian dynamics and state estimator model from images in one end-to-end process. The complex interplay between both, often makes them the most labour intensive parts in system identification. We believe this is a gateway step in achieving reliable end-to-end learned control from pixels.

2.7 Appendix

2.7.1 Implementation of constrained Euler-Lagrange equations in JAX

It could seem a daunting task to implement the derivation of the constrained Euler-Lagrange equations (2.14) in an autograd library. As an example, we provide an implementation in JAX (Bradbury et al. (2018)).

```
import jax
import jax.numpy as jnp


def constraint_fn(x):
    # function that returns a vector with constraint values
    c = jnp.array([
      ...,
    ])
    return c


def mass_matrix(params, x):
    # function that returns the mass matrix
    ...
    return m


def potential_energy(params, x):
    # function that returns the potential energy
    ...
    return V
```

```
23
24
25 def input_matrix(params, x):
26     # function that returns the input matrix
27     ...
28     return g
29
30
31 def euler_lagrange(params, x, x_t, action):
32     m_inv = jnp.linalg.pinv(mass_matrix(params, x))
33     f = - jax.grad(potential_energy, 1)(params, x) + input_matrix(params, x)
           ↪ @ action
34
35     Dphi = jax.jacobian(constraint_fn)(x)
36     DDphi = jax.jacobian(jax.jacobian(constraint_fn))(x)
37
38     # Lagrange multiplicators:
39     l = jnp.linalg.pinv(Dphi @ m_inv @ Dphi.T) @ (Dphi @ m_inv @ f +
           ↪ DDphi @ x_t @ x_t)
40     x_tt = m_inv @ (f - Dphi.T @ l)
41
42     return x_tt
```

2.7.2 Rigid bodies as sets of point masses

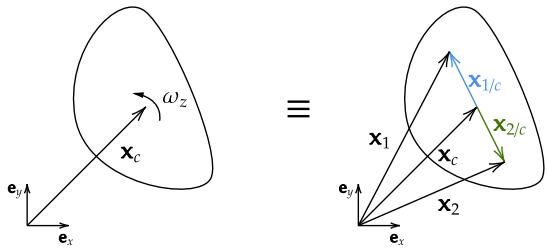


Figure 2.22 Any 2D rigid body with mass m and rotational inertia I is equivalent to a set of two point masses $\mathbf{x}_1$ and $\mathbf{x}_2$ with masses m_1 and m_2. The kinetic energy of the rigid body, expressed in a translational part and a rotational part, is equal to the sum of the kinetic energies of the point masses.

The position of a rigid body in 2D is fully described by the position of its center of mass $\mathbf{x}_c$ and orientation θ. Potential energy only

depends on the position, thus if we want to describe the potential energy with an equivalent rigid set of point masses, two points are sufficient to fully determine $\mathbf{x}_c$ and θ. For the kinetic energy, we provide the following Theorem and proof:

Theorem 2. *For any 2D rigid body, described by its center of mass* $\mathbf{c}$*, mass* m *and rotational inertia* I*, there exists an equivalent rigid set of two point masses* $\mathbf{x}_1$ *and* $\mathbf{x}_2$ *with masses* m_1 *and* m_2*.*

Proof. To find conditions such that the kinetic energy expressed in two point masses should be equal to the rigid body representation, we start by expressing general 3D-movement:

$$\mathbf{x}_i = \mathbf{x}_c + \mathbf{x}_{i/c}, \qquad i \in \{1, 2\}\,. \tag{2.25}$$

Where the vector $\mathbf{x}_c$ are the coordinates of the center of mass and the vector $\mathbf{x}_{i/c}$ is the position of the point mass relative to the center of mass. Since this relative position $\mathbf{x}_{i/c}$ has fixed length, only a rotation is possible and hence the equation of the velocity is:

$$\dot{\mathbf{x}}_i = \dot{\mathbf{x}}_c + \boldsymbol{\omega} \times \mathbf{x}_{i/c}, \qquad i \in \{1, 2\}\,, \tag{2.26}$$

where $\boldsymbol{\omega}$ is the rotational velocity of the body. Substituting this in the kinetic energy of the point masses, we get:

$$T = \frac{1}{2}\sum_{i=1}^{2} m_i \|\dot{\mathbf{x}}_c + \boldsymbol{\omega} \times \mathbf{x}_{i/c}\|^2 \tag{2.27}$$

$$= \frac{1}{2}\sum_{i=1}^{2} m_i \Big(\|\dot{\mathbf{x}}_c\|^2 + \|\boldsymbol{\omega} \times \mathbf{x}_{i/c}\|^2 + 2\mathbf{x}_{i/c} \cdot (\dot{\mathbf{x}}_c \times \boldsymbol{\omega}) \Big)\,. \tag{2.28}$$

Where we calculated the square and used the circular shift property of the triple product on the last term.

For movement in the 2D-plane (*i.e.*, $\boldsymbol{\omega} = \vec{\mathbf{e}}_z \omega_z$ and $\mathbf{x}_i = \vec{\mathbf{e}}_x \mathbf{x}_{i,x} + \vec{\mathbf{e}}_y \mathbf{x}_{i,y}$), this becomes:

$$T = \frac{1}{2}\sum_{i=1}^{2} m_i \Big(\|\dot{\mathbf{x}}_c\|^2 + \|\mathbf{x}_{i/c}\|^2 \omega_z^2 + 2\mathbf{x}_{i/c} \cdot (\dot{\mathbf{x}}_c \times \boldsymbol{\omega}) \Big) \tag{2.29}$$

$$= \frac{1}{2}(m_1 + m_2)\|\dot{\mathbf{x}}_c\|^2 + \frac{1}{2}(m_1 \|\mathbf{x}_{1/c}\|^2 + m_2 \|\mathbf{x}_{2/c}\|^2)\omega_z^2 + (m_1 \mathbf{x}_{1/c} + m_2 \mathbf{x}_{2/c}) \cdot (\dot{\mathbf{x}}_c \times \boldsymbol{\omega})\,. \tag{2.30}$$

Matching the kinetic energy of the 2 point masses (equation (2.29)) with that of the rigid body representation (left hand side of Fig. 2.22), we get following conditions:

$$\begin{cases} m = m_1 + m_2 \,, \\ I = m_1 \|\mathbf{x}_{1/c}\|^2 + m_2 \|\mathbf{x}_{2/c}\|^2 \,, \\ \mathbf{0} = m_1 \mathbf{x}_{1/c} + m_2 \mathbf{x}_{2/c} \,. \end{cases} \tag{2.31}$$

Since the last equation is a vector equation, this gives us four equations in six unknowns (m_1,m_2,$\mathbf{x}_{1,x}$,$\mathbf{x}_{1,y}$,$\mathbf{x}_{2,x}$,$\mathbf{x}_{2,y}$), which leaves us the freedom to choose two. □

It follows from the third condition of (2.31) that points $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_c$ should be collinear. To conclude, we can freely choose the positions of the point masses (as long as $\mathbf{x}_c$ is on the line between them), and will be able to model the rigid body as a set of two point masses. In practice, KeyCLD will freely choose the keypoint positions to be able to model the dynamics. Depending on the constraints in the system, it is possible to further reduce the number of necessary keypoints. See Appendix 2.7.4 for examples.

2.7.3 Energy shaping control

Recent works of Zhong et al. (2020) and Zhong and Leonard (2020) use energy shaping in generalized coordinates. In Cartesian coordinates, energy shaping can still be used. This is easily seen from the fact that for the holonomic constraints $\mathbf{\Phi}(\mathbf{x}) \equiv 0$, we have the derivative $D\mathbf{\Phi}(\mathbf{x})\dot{\mathbf{x}} = \mathbf{0}$, which means that the constraint forces in equation (2.14) are perpendicular to the path and hence do no work nor influence the energy (Lanczos (2020)).

Energy shaping control makes sure that the controlled system behaves according to a potential energy $V_d(\mathbf{x})$ instead of $V(\mathbf{x})$:

$$\mathbf{u} = (\mathbf{g}^\top \mathbf{g})^{-1} \mathbf{g}^\top \left(\nabla_{\mathbf{x}} V - \nabla_{\mathbf{x}} V_d\right) - y_{\text{passive}} \,, \tag{2.32}$$

where y_{passive} can be any passive output, the easiest choice being $y_{\text{passive}} = k_d \mathbf{g}^\top \dot{\mathbf{x}}$, where k_d is a tuneable control parameter. The

proposed potential energy V_d should be such that:

$$\mathbf{x}^* = \text{argmin} V_d(\mathbf{x}) \,, \tag{2.33}$$

$$\mathbf{0} = \mathbf{g}^{\perp} \left(\nabla_{\mathbf{x}} V - \nabla_{\mathbf{x}} V_d \right) \,, \tag{2.34}$$

Where $\mathbf{g}^{\perp}$ is the left-annihilator of $\mathbf{g}$, meaning that $\mathbf{g}^{\perp}\mathbf{g} = \mathbf{0}$. For fully actuated systems, the first condition of equation (2.33) is always met and the easiest choice is:

$$V_d(\mathbf{x}) = (\mathbf{x} - \mathbf{x}^*)^{\top} k_p (\mathbf{x} - \mathbf{x}^*) \,, \tag{2.35}$$

where k_p is a tuneable control parameter. The desired equilibrium position $\mathbf{x}^*$ is obtained by processing an image of the desired position with the keypoint estimator model. Finally, the passivity-based controller that is used is:

$$\mathbf{u} = (\mathbf{g}^{\top}\mathbf{g})^{-1}\mathbf{g}^{\top} \left[\nabla_{\mathbf{x}} V - k_p (\mathbf{x} - \mathbf{x}^*) \right] - k_d \mathbf{g}^{\top} \dot{\mathbf{x}} \,. \tag{2.36}$$

Changing the behavior of the kinetic energy is also possible (Gomez-Estern et al. (2001)), but is left for future work. Many model-based reinforcement learning algorithms require the learning of a full neural network as controller. Whilst in this work, due to knowledge of the potential energy, we only need to tune two parameters k_p and k_d.

2.7.4 Details about the dm_control environments and data generation

We adapted the pendulum, cartpole and acrobot environments from dm_control (Tunyasuvunakool et al. (2020)) implemented in MuJoCo (Todorov et al. (2012)). Both are released under the Apache-2.0 license. Following changes were made to the environments to adapt them to our use-case:

Pendulum

The camera was repositioned so that it is in a parallel plane to the system. Friction was removed. Torque limits of the motor are increased.

Cartpole

The camera was moved further away from the system to enable a wider view, the two rails are made longer and the floor lowered so that they are not cut-off with the wider view. All friction is removed. The pole is made twice as thick, the color of the cart is changed. Torque limits are increased and actuation is added to the cart to make full actuation possible.

Acrobot

The camera and system are moved a little bit upwards. The two poles are made twice as thick, and one is changed in color. Torque limits are increased and actuation is added to the upper part to make full actuation possible.

Data generation

For every environment, 500 runs of 50 timesteps are generated with a 10% validation split. The initial state for every sequence is at a random position with small random velocity. The control inputs $\mathbf{u}$ are constant throughout a sequence, and uniform randomly chosen between the force and torque limits of the input. We set $\mathbf{u} = \mathbf{0}$ for 20% of the sequences. We found this helps the model to learn the dynamics better, discouraging confusion of the energy models with external actions.

The constraint function for each of the environments are given in Fig. 2.23. As explained in 2.7.2, every rigid body needs to be represented by two keypoints. But due to the constraints it is possible to omit certain keypoints, because they do not move or coincide with other keypoints. As experimentally validated, we can thus model all three systems with a lower number of keypoints, where the number of keypoints equals the number of bodies.

Pendulum

One keypoint is used to model the pendulum. The second keypoint of this rigid body can be omitted because it can be assumed to be at the origin. Due to the constraint function, this point will provide no kinetic energy since it will not move. Since the other keypoints position and mass is freely chosen, any pendulum can be modelled. The constraint function expresses that the distance l_1 from the origin to $\mathbf{x_1}$ is fixed. The value of l_1 in the implementation is irrelevant because it vanishes when taking the Jacobian.

Cartpole

Two keypoints are used to model the cartpole. The constraint function expresses that $\mathbf{x}_1$ does not move in the vertical direction and the distance l_1 between $\mathbf{x}_1$ and $\mathbf{x}_2$ is constant. Again, the values of l_1 and l_2 in the implementation are irrelevant.

Acrobot

Two keypoints are used to model the acrobot. The constraint function expresses that lengths l_1 and l_2 are constant through time. Again, the values are irrelevant in the implementation.

2.7.5 Training hyperparameters and details

All models were trained on one NVIDIA RTX 2080 Ti GPU.

KeyCLD, KeyLD and KeyODE2

We use the Adam optimizer (Kingma and Ba (2015)), implemented in Optax (Hessel et al. (2020)) with a learning rate of 3×10^{-4}. We use the exact same hyperparameters for all the environments and did not tune them individually. Dynamics loss weight $\lambda = 1$, $\sigma = 0.1$ for the Gaussian blobs in $\mathbf{s}'$. The hidden layers in the keypoint estimator and renderer model have at the first block

Table 2.3 Number of parameters of the potential energy model.

	Pendulum	Cartpole	Acrobot
Dense_0	96	160	160
Dense_1	1056	1056	1056
Dense_2	33	33	33
Total	1185	1249	1249

Table 2.4 Number of parameters of the input matrix model.

	Pendulum	Cartpole	Acrobot
Dense_0	96	160	160
Dense_1	1056	1056	1056
Dense_2	66	264	264
Total	1218	1480	1480

32 features, this increases to respectively 64 and 128 after every maxpool operation. All convolutions have kernel size 3×3, and maxpool operations scale down with factor 2 with a kernel size of 2×2. See Tabs. 2.5 and 2.6 for the number of parameters.

The potential energy is modelled with an MLP with two hidden layers with 32 neurons and celu activation functions (Barron (2017)). The weights are initialized with a normal distribution with standard deviation 0.01. See Tab. 2.3 for the number of parameters. Likewise, the input matrix is modelled with an MLP similar to the potential energy. The outputs of this MLP are reshaped in the shape of the input matrix. See Tab. 2.4 for the number of parameters.

The KeyODE2 dynamics model is an MLP with three hidden layers with each 64 neurons. We chose a higher number of layers and neurons, to allow this model more expressivity compared to the potential energy and input matrix models of KeyCLD.

Table 2.5 Number of parameters of the keypoint encoder model.

	Pendulum	Cartpole	Acrobot
Block_0/Conv_0	896	896	896
Block_0/GroupNorm_0	64	64	64
Block_1/Conv_0	18496	18496	18496
Block_1/GroupNorm_0	128	128	128
Block_2/Conv_0	73856	73856	73856
Block_2/GroupNorm_0	256	256	256
Block_3/Conv_0	110656	110656	110656
Block_3/GroupNorm_0	128	128	128
Block_4/Conv_0	27680	27680	27680
Block_4/GroupNorm_0	64	64	64
Conv_0	289	578	578
Total	232513	232802	232802

Table 2.6 Number of parameters of the renderer model.

	Pendulum	Cartpole	Acrobot
Seed Tensor	126976	126976	122880
Block_0/Conv_0	9248	9248	9248
Block_0/GroupNorm_0	64	64	64
Block_1/Conv_0	18496	18496	18496
Block_1/GroupNorm_0	128	128	128
Block_2/Conv_0	73856	73856	73856
Block_2/GroupNorm_0	256	256	256
Block_3/Conv_0	110656	110656	110656
Block_3/GroupNorm_0	128	128	128
Block_4/Conv_0	27680	27680	27680
Block_4/GroupNorm_0	64	64	64
Conv_0	867	867	867
Total	368419	364323	364323

Lag-caVAE, Lag-VAE and HGN

For the Lag-caVAE and Lag-VAE baselines, the official public codebase was used (Zhong and Leonard (2020)). We adapted the implementation to work with the higher input resolution of 64 by 64 (instead of 32 by 32), and 3 color channels (instead of 1).

For the HGN baseline, we used the implementation that was also released by Zhong and Leonard (2020). The architecture was adapted to work with the higher input resolution of 64 by 64 (instead of 32 by 32) by adding an extra upscale layer in the decoder, and a maxpool layer and one extra convolutional layer in the encoder.

2.7.6 Failure cases

A possible failure case is that the model learns a faulty keypoint representation that does not correspond to the given constraint function. This results in a failed model, and the training is stuck in this local minima. The encoder model will keep focussing on this erronous representation and is unable to switch to the correct keypoints. Figure 2.24 shows an example of this failure case. We observed this failure in a minority of the experiments. This can be mitigated by retraining the model.

2.7.7 Ablating L_e

A binary cross-entropy loss L_e is formulated over $\mathbf{s}$ and $\mathbf{s}'$ to encourage the Gaussian prior. When L_e is omitted, the model can get stuck in a local minima where the encoder does not learn to predict keypoints, but rather larger regions or static values. Image 2.25 shows an example of this failure case.

Figure 2.18 Visualization of the input matrix of the trained KeyCLD model of the pendulum environment. The input of this environment is a torque acting on the pendulum. In the KeyCLD framework this is correctly modelled with a force acting perpendicular on the pendulum.

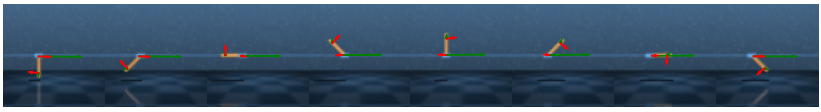


Figure 2.19 Visualization of the input matrix of the trained KeyCLD model of the cartpole environment. This environment has two inputs, a horizontal force acting on the cart, and a torque acting on the pole. The horizontal force corresponds to the green vectors. The first vector acting on the cart keypoint stays constant, and the second vector is negligibly small, since the horizontal force does not act on the pole. The torque corresponds to the red vectors, it is modelled with forces acting on the pole in opposite directions, such that the residual force can be zero.

Figure 2.20 Visualization of the input matrix of the trained KeyCLD model of the acrobot environment. This environment has two inputs, two torques acting on each pole. The torques are modelled with opposite forces on each end of the poles, such that the residual force can be zero.

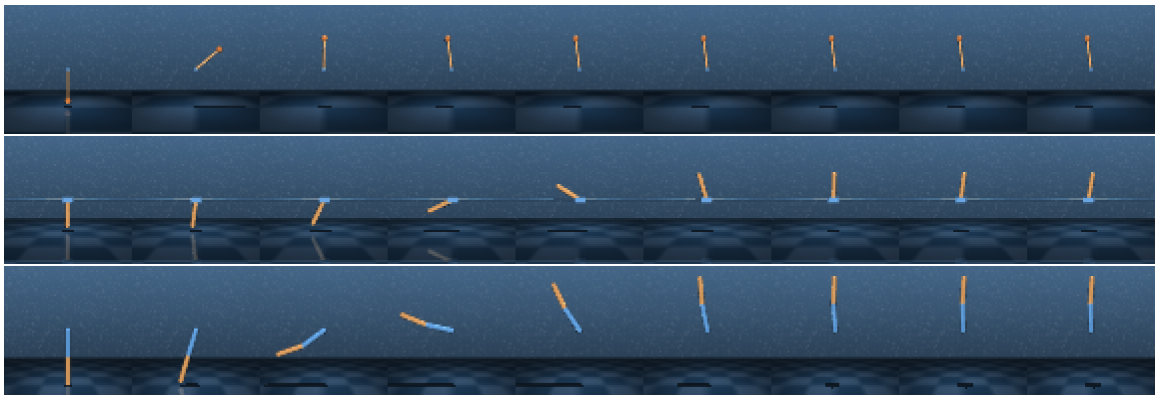


Figure 2.21 KeyCLD allows using energy shaping control because the learned potential energy model is available. Based on a swing-up target image $\mathbf{z}^*$, the target state $\mathbf{x}^*$ is determined by the keypoint detector model. The sequences show that all three systems can achieve the target state. The control parameters $k_p = 5.0$ and $k_d = 2.0$ are the same for all systems, demonstrating the generality of the control method.

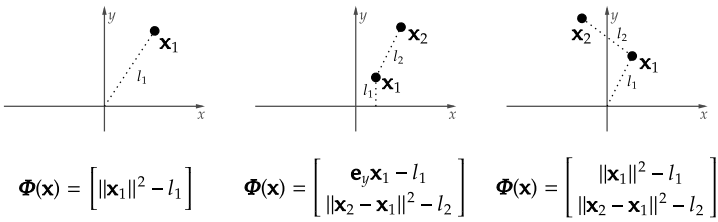


Figure 2.23 From left to right the pendulum, cartpole and acrobot dm_control environments. The respective constraint functions are given below each schematic.

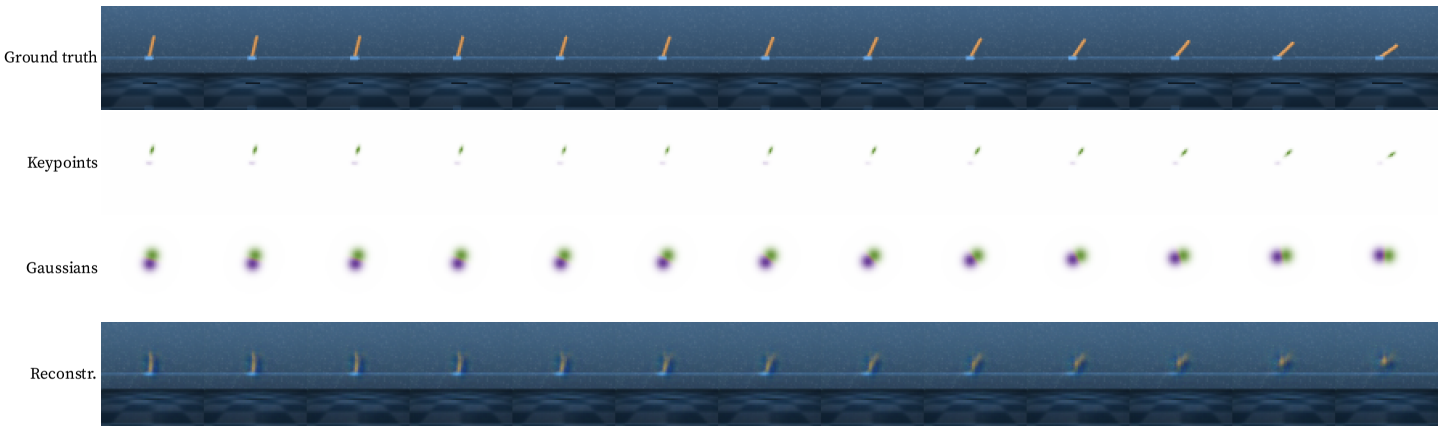


Figure 2.24 Failure case of KeyCLD on the cartpole environment. Keypoints are indicated in green and purple. The model erronously assigned the green keypoint to the pole. Since the given constraint function dictates that the green keypoint can only move horizontally, this results in faulty model.

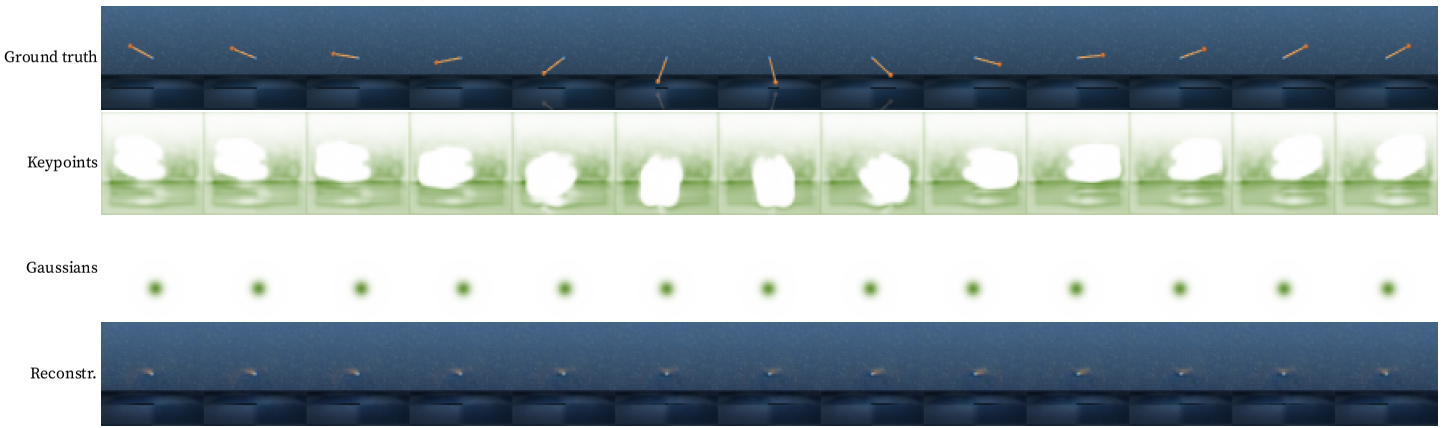


Figure 2.25 Omitting L_e can result in poor learning of keypoints. The encoder model does not predict distinct keypoints, but other shapes. This is effectly a local minima in the learning process, since the model is uncapable of switching to a correct representation.

3

3

Variational Inference of SDEs Driven by Fractional Noise

3

Variational Inference of SDEs Driven by Fractional Noise

The preceding chapter considers physics priors in a deterministic manner. This chapter takes a step further towards learning *stochastic* dynamical models from video. Concretely, we use variational inference (VI) to learn the parameters of continuous-time models, *i.e.*, stochastic differential equations (SDEs). Basic notions on VI, stochastic dynamics and stochastic calculus are provided in Secs. 1.4 to 1.6. Additionally, this chapter makes use of incomplete and complete Gamma functions, which are introduced in more detail in Sec. 1.7. This chapter addresses the second goal of this dissertation, as outlined in Sec. 1.8 and is partially based on Daems, Opper, et al. (2024).

We present a novel variational framework for performing inference in (neural) stochastic differential equations (SDEs) driven by Markov-approximate fractional Brownian motion (fBM). SDEs offer a versatile tool for modeling real-world continuous-time dynamic systems with inherent noise and randomness. Combining SDEs with the powerful inference capabilities of variational methods, enables the learning of representative distributions through stochastic gradient descent. However, conventional SDEs typically assume the underlying noise to follow a Brownian motion (BM), which hinders their ability to capture long-term dependencies. In contrast, fractional Brownian motion (fBM) extends BM to encompass non-Markovian dynamics, but existing methods for inferring fBM parameters are either computationally demanding or statistically inefficient. By building upon the Markov approximation of fBM, we derive the evidence lower bound

essential for efficient variational inference of posterior path measures, drawing from the well-established field of stochastic analysis. Additionally, we provide a closed-form expression for optimal approximation coefficients and propose to use neural networks to learn the drift, diffusion and control terms within our variational posterior, leading to the variational training of neural-SDEs. In this framework, we also optimize the Hurst index, governing the nature of our fractional noise. Beyond validation on synthetic data, we contribute a novel architecture for variational latent video prediction,—an approach that, to the best of our knowledge, enables the first variational neural-SDE application to video perception.

3.1 Introduction

Our surroundings constantly evolve over time, influenced by several dynamic factors, manifesting in various forms, from the weather patterns and the ebb & flow of financial markets to the movements of objects (Yu et al. (2023) and Rempe et al. (2021)) & observers, and the subtle deformations that reshape our environments (Gojcic et al. (2021)). Stochastic differential equations (SDEs) provide a natural way to capture the randomness and continuous-time dynamics inherent in these real-world processes. To extract meaningful information about the underlying system, *i.e.*, to infer the model parameters and to accurately predict the unobserved paths, variational inference (VI) (Bishop and Nasrabadi (2006)) is used as an efficient means, computing the posterior probability measure over paths (Opper (2019), Li et al. (2020), and Ryder et al. (2018))[1].

The traditional application of SDEs assumes that the underlying noise processes are generated by standard Brownian motion (BM) with independent increments. Unfortunately, for many practical scenarios, BM falls short of capturing the full complexity and richness of the observed real data, which often

1. KL divergence between two SDEs over a finite time horizon has been well-explored in the control literature (Theodorou (2015) and Kappen and Ruiz (2016)).

contains long-range dependencies, rare events, and intricate temporal structures that cannot be faithfully represented by a Markovian process. The non-Markovian fractional Brownian motion (fBM) (Mandelbrot and Van Ness (1968)) extends BM to stationary increments with a more complex dependence structure, *i.e.*, long-range dependence vs. roughness/regularity controlled by its *Hurst index* (Gatheral et al. (2018)). Yet, despite its desirable properties, the computational challenges and intractability of analytically working with fBMs pose significant challenges for inference.

In this chapter, we begin by providing a tractable variational inference framework for SDEs driven by fractional Brownian motion (Types I & II). To this end, we benefit from the relatively under-explored *Markov representation* of fBM and path-wise approximate fBM through a linear combination of a finite number of Ornstein-Uhlenbeck (OU) processes driven by a common noise (Carmona and Coutin (1998a, 1998b) and Harms and Stefanovits (2019)). We further introduce a differentiable method to optimise for the associated coefficients and conjecture (as well as empirically validate) that this *strong* approximation enjoys super-polynomial convergence rates, allowing us to use a handful of processes even in complex problems.

Such *Markov-isation* also allows us to inherit the well-established tools of traditional SDEs including Girsanov's change of measure theorem (Øksendal (2003)), which we use to derive and maximise the corresponding *evidence lower bound* (ELBO) to yield posterior path measures as well as maximum likelihood estimates as illustrated in Fig. 3.1. We then use our framework in conjunction with neural networks to devise VI for neural-SDEs (Liu et al. (2019) and Li et al. (2020)) driven by the said fractional diffusion. We deploy this model along with a novel neural architecture for the task of enhanced video prediction. To the best of our knowledge, this is the first time either fractional or variational neural-SDEs are used to model videos. Our contributions are:

- We make accessible the relatively uncharted Markovian embedding of the fBM and its strong approximation, to the machine learning community. This allows us to employ the tra-

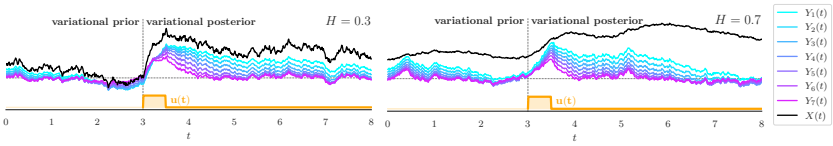


Figure 3.1 We leverage the Markov approximation, where the non-Markovian fractional Brownian motion with Hurst index H is approximated by a linear combination of a finite number of Markov processes $(Y_1(t), \ldots, Y_K(t))$, and propose a variational inference framework in which the posterior is steered by a control term $u(t)$. Note the long-term memory behaviour of the processes, where individual $Y_k(t)$s have varying transient effects, from $Y_1(t)$ having the longest memory to $Y_7(t)$ the shortest, and tend to forget the action of $u(t)$ after a certain time frame.

ditional machinery of SDEs in working with non-Markovian systems.

- We show how to balance the contribution of Markov processes by optimising for the combination coefficients in closed form. We further estimate the (time-dependent) Hurst index from data.
- We derive the evidence lower bound for SDEs driven by approximate fBM of both Types I and II.
- We model the drift, diffusion and control terms in our framework by neural networks, and propose a novel architecture for video prediction.

3.2 Related Work

Fractional noises and neural-SDEs. fBM (Mandelbrot and Van Ness (1968)) was originally used for the simulation of rough volatility in finance (Gatheral et al. (2018)). Using the Lemarié-Meyer wavelet representation, Allouche et al. (2022) provided a large probability bound on the deep-feedforward RELU network approximation of fBM, where up to log terms, a uniform error of $O(N^{-H})$ is achievable with $\log(N)$ hidden layers and $O(N)$ parameters. Tong et al. (2022) approximated the fBM (only Type II) with sparse Gaussian processes. Unfortunately, they are

limited to Euler-integration and to the case of $H > 1/3$. Their model was also not applied to videos. Recently, Luxuan Yang et al. (2023) applied Levy driven neural-SDEs to times series prediction and Hayashi and Nakagawa (2022) considered neural-SDEs driven by fractional noise. Neither of those introduce a variational framework. Both Liao et al. (2019) and Morrill et al. (2021) worked with *rough path theory* to model long time series via rough neural-SDEs. To the best of our knowledge, we are the firsts to devise a VI framework for neural-SDEs driven by a path-wise (strong) approximation of fBM.

SDEs and visual understanding. Apart from the recent video diffusion models (Luo et al. (2023), R. Yang et al. (2022), and Ho et al. (2022)), SDEs for spatiotemporal visual generation is relatively unexplored. S. Park et al. (2021) and Ali et al. (2023) used neural-ODEs to generate and manipulate videos while Rempe et al. (2020) used neural-ODEs for temporal 3D point cloud modeling. SDENet (Kong et al. (2020)) and MDSDE-Net (Zhang et al. (2023)) learned drift and diffusion networks for uncertainty estimation of images using out-of-distribution data. Tong et al. (2022) used approximate-fBMs in score-based diffusion modeling for image generation. Gordon and Parde (2021) briefly evaluated different neural temporal models for video generation. While Babaeizadeh et al. (2018) used VI for video prediction, they did not employ SDEs. To the best of our knowledge, we are the firsts to use neural-SDEs in a variational framework for video understanding.

We first tailor and make accessible the fractional Brownian motion (fBM) and its relatively less explored Markov approximations for the learning community. We then describe the SDEs driven by fBM and its approximation before delving into the inference.

3.2.1 Fractional Brownian motion (fBM) & Its Markov Approximation

Definition 1 (Fractional Brownian motion (Type I))**.** *fBM (Type I) is a self-similar, non-Markovian, non-martingale, zero-mean Gaus-*

sian process $\left(B_H^{(I)}(t)\right)_{t\in[0,T]}$ *for* $T>0$ *with a covariance:*

$$\begin{aligned}&\operatorname{Cov}\left(B_H^{(I)}(t_1),B_H^{(I)}(t_2)\right)\\&\qquad=\frac{V_H^{(I)}}{2}\left(|t_1|^{2H}+|t_2|^{2H}-|t_1-t_2|^{2H}\right),\end{aligned}\tag{3.1}$$

with scaling constant (Lim and Sithi (1995)):

$$V_H^{(I)}=\operatorname{Var}\left(B_H^{(I)}(1)\right)=\frac{-\Gamma(2-2H)\cos(\pi H)}{(2H-1)\pi H}\tag{3.2}$$

$$=\frac{\Gamma(2-2H)\operatorname{sinc}(H-1/2)}{2H},\tag{3.3}$$

where $0<H<1$ *is the* Hurst index*,* Γ *is the Gamma function and* $\operatorname{sinc}(x)=\sin(\pi x)/(\pi x)$ *is the normalized sine cardinal.*

Definition 2 (Fractional Brownian motion (Type II))**.** *fBM (Type II) is a self-similar, non-Markovian, non-martingale, zero-mean Gaussian process* $\left(B_H^{(II)}(t)\right)_{t\in[0,T]}$ *for* $T>0$ *with a covariance:*

$$\begin{aligned}&\operatorname{Cov}\left(B_H^{(II)}(t_1),B_H^{(II)}(t_2)\right)\\&=V_H^{(II)}2H\int_0^{\min(t_1,t_2)}((t_1-u)(t_2-u))^{H-1/2}\,\mathrm{d}u\end{aligned}\tag{3.4}$$

$$\begin{aligned}&=V_H^{(II)}\frac{2H(t_1t_2)^{H-1/2}\min(t_1,t_2)}{H+1/2}\\&\qquad\cdot{}_2F_1\left(1/2-H,1,H+3/2,\frac{\min(t_1,t_2)}{\max(t_1,t_2)}\right),\end{aligned}\tag{3.5}$$

with scaling constant:

$$V_H^{(II)}=\operatorname{Var}\left(B_H^{(II)}(1)\right)=\frac{1}{2H\Gamma^2(H+1/2)},\tag{3.6}$$

where $0<H<1$ *is the* Hurst index*,* Γ *is the Gamma function and* ${}_2F_1$ *is the hypergeometric function.*

fBM (Type I & II) recover *Brownian motion (BM)* for $H=1/2$ (regular diffusion) and generalize it for other choices. The increments are (i) positively correlated for $H>1/2$ (super-diffusion)

where the tail behaviour is infinitely heavier than that of BM, and (ii) negatively correlated for $H < 1/2$ (sub-diffusion). The increments of Type I are stationary, however, the Type II model implies nonstationary increments of which the marginal distributions are dependent on the time relative to the start of the observed sample, *i.e.*, all realizations would have to be found very close to the unconditional mean (*i.e.*, the origin) (Lim and Sithi (1995) and Davidson and Hashimzade (2009)).

Definition 3 (Integral representations of fBM)**.** *$B_H^{(I,II)}$ admit the following integral forms due to the Mandelbrot van-Ness and Weyl representations, respectively (Mandelbrot and Van Ness (1968)):*

$$B_H^{(I)}(t) = \frac{1}{\Gamma(H+\frac{1}{2})}\left(\int_{-\infty}^{t}(t-s)^{H-\frac{1}{2}}\,\mathrm{d}W(s) - \int_{-\infty}^{0}(-s)^{H-\frac{1}{2}}\,\mathrm{d}W(s)\right), \tag{3.7}$$

$$B_H^{(II)}(t) = \frac{1}{\Gamma(H+\frac{1}{2})}\int_{0}^{t}(t-s)^{H-\frac{1}{2}}\,\mathrm{d}W(s)\,. \tag{3.8}$$

One can perceive Type I as a generalization of Type II, in the sense that the integral starts at $-\infty$ instead of at 0, and where the addition of the second integral ensures $B_H^{(I)}(0) = 0$. The stationarity of the increments of Type I is reflected in the fact that the integral starts at $-\infty$. The increments of Type II are nonstationary, as the integral starts at 0, *i.e.*, the integrands are dependent on the time relative to the origin.

Remark 1 (Stationarity of the increments of fBM)**.** *The increments of the fBM processes are stationary for Type I and nonstationary for Type II:*

$$B_H^{(I)}(t+\Delta t) - B_H^{(I)}(t) \overset{d}{=} B_H^{(I)}(\Delta t)\,, \tag{3.9}$$

$$B_H^{(II)}(t+\Delta t) - B_H^{(II)}(t) \overset{d}{\neq} B_H^{(II)}(\Delta t)\,, \tag{3.10}$$

where $\overset{d}{=}$ and $\overset{d}{\neq}$ denote (in)equality in distribution. This is the main difference between Type I and Type II.

Remark 2 (Self-similarity of fBM)**.** *One can show, for both Type I & II, that the process $B_H^{(I,II)}$ is self-similar:*

$$B_H^{(I,II)}(at) \overset{d}{=} a^H B_H^{(I,II)}(t)\,, \quad a > 0, \tag{3.11}$$

where $\overset{d}{=}$ denotes equality in distribution. This means that the process has the same statistical properties at all time scales. Likewise, the covariance function is scaled by a^{2H}:

$$\begin{aligned}\operatorname{Cov}&\left(B_H^{(I,II)}(at_1), B_H^{(I,II)}(at_2)\right)\\ &= a^{2H}\operatorname{Cov}\left(B_H^{(I,II)}(t_1), B_H^{(I,II)}(t_2)\right)\,.\end{aligned} \tag{3.12}$$

Remark 3. *The scaling constants $V_H^{(I,II)}$ above are defined as the variance at time $t = 1$. Note that this variance changes with the choice of H. An alternative approach is to define the covariance at time $t = 1$ fixed as 1 for all Hurst values H. This is done by scaling the process appropriately:*

$$\begin{aligned}\operatorname{Cov}&\left(B_H^{(I,II)}(t_1), B_H^{(I,II)}(t_2)\right)\\ &\to \frac{\operatorname{Cov}\left(B_H^{(I,II)}(t_1), B_H^{(I,II)}(t_2)\right)}{V_H^{(I,II)}}\,,\end{aligned} \tag{3.13}$$

$$B_H^{(I,II)}(t) \to \frac{B_H^{(I,II)}(t)}{\sqrt{V_H^{(I,II)}}}\,. \tag{3.14}$$

All results in this work can thus be adapted to this alternative definition.

Proposition 1 (Markov representation of fBM (Harms and Stefanovits (2019)))**.** *The long memory processes $B_H^{(I,II)}(t)$ can be represented by an infinite linear combination of Markov processes $Y_\gamma(t)$ and $U_\gamma(t)$, all driven by the same Wiener noise, but with different time scales, defined by speed of mean reversion γ. For both types we have representations of the form:*

$$B_H(t) = \begin{cases} \displaystyle\int_0^\infty (Y_\gamma(t) - Y_\gamma(0))\mu(\gamma)\,\mathrm{d}\gamma, & H < \frac{1}{2},\\ \displaystyle\int_0^\infty (U_\gamma(t) - U_\gamma(0))\nu(\gamma)\,\mathrm{d}\gamma, & H > \frac{1}{2}, \end{cases} \tag{3.15}$$

where

$$\mu(\gamma) = \frac{\gamma^{-(H+\frac{1}{2})}}{\Gamma(H+\frac{1}{2})\Gamma(\frac{1}{2}-H)}\,, \tag{3.16}$$

$$\nu(\gamma) = \frac{\gamma^{-(H-\frac{1}{2})}}{\Gamma(H+\frac{1}{2})\Gamma(\frac{3}{2}-H)}\,. \tag{3.17}$$

Note, these non–negative densities are not normalisable. To simplify notation, we will drop explicit dependency on the types (I, II) in what follows. For each $\gamma \geq 0$, and for both types I and II, the processes $Y_\gamma(t)$ and $U_\gamma(t)$ are defined as the system of stochastic differential equations (SDEs):

$$\mathrm{d}Y_\gamma(t) = -\gamma Y_\gamma(t)\,\mathrm{d}t + \mathrm{d}W(t)\,, \tag{3.18}$$

$$\mathrm{d}U_\gamma(t) = (-\gamma U_\gamma(t) + Y_\gamma(t))\,\mathrm{d}t\,. \tag{3.19}$$

Type I and Type II differ in the initial conditions for $Y_\gamma(0)$ and $U_\gamma(0)$:

$$Y_\gamma^{(I)}(0) = \int_{-\infty}^{0} e^{\gamma s}\,\mathrm{d}W(s) \qquad \text{and} \quad Y_\gamma^{(II)}(0) = 0\,,$$

$$U_\gamma^{(I)}(0) = -\int_{-\infty}^{0} s e^{\gamma s}\,\mathrm{d}W(s) \quad \text{and} \quad U_\gamma^{(II)}(0) = 0\,.$$

Sketch of the proof. The Markov processes $(Y_\gamma(t), U_\gamma(t))$ are solved by:

$$Y_\gamma(t) = Y_\gamma(0)e^{-\gamma t} + \int_0^t e^{-\gamma(t-s)}\,\mathrm{d}W(s)\,, \tag{3.20}$$

$$U_\gamma(t) = U_\gamma(0)e^{-\gamma t} + Y_\gamma(0)te^{-\gamma t} + \int_0^t (t-s)e^{-\gamma(t-s)}\,\mathrm{d}W(s)\,. \tag{3.21}$$

For $H < 1/2$, the term $\tau^{H-1/2}/\Gamma(H+1/2)$ in Eqs. (3.7) and (3.8) is the Laplace transform of μ:

$$\mathcal{L}(\mu)(\tau) = \int_0^\infty e^{-\gamma\tau}\mu(\gamma)\,\mathrm{d}\gamma = \frac{\tau^{H-1/2}}{\Gamma(H+1/2)}\,. \tag{3.22}$$

Thus we can write for Type II:

$$B_H^{(II)}(t) = \int_0^t \int_0^\infty e^{-\gamma(t-s)} \mu(\gamma)\, \mathrm{d}\gamma\, \mathrm{d}W(s) \tag{3.23}$$

$$= \int_0^\infty \int_0^t e^{-\gamma(t-s)}\, \mathrm{d}W(s) \mu(\gamma)\, \mathrm{d}\gamma \tag{3.24}$$

$$= \int_0^\infty \left(Y_\gamma(t) - Y_\gamma(0)\right) \mu(\gamma)\, \mathrm{d}\gamma,\ Y_\gamma(0) = 0\,. \tag{3.25}$$

Alternatively, for $H > 1/2$, $\tau^{H-3/2}/\Gamma(H+1/2)$ is the Laplace transform of ν:

$$\tau \mathcal{L}(\nu)(\tau) = \tau \int_0^\infty e^{-\gamma\tau} \nu(\gamma)\, \mathrm{d}\gamma = \frac{\tau^{H-1/2}}{\Gamma(H+1/2)}\,. \tag{3.26}$$

Again for Type II, this leads to:

$$B_H^{(II)}(t) = \int_0^t (t-s) \int_0^\infty e^{-\gamma(t-s)} \nu(\gamma)\, \mathrm{d}\gamma\, \mathrm{d}W(s) \tag{3.27}$$

$$= \int_0^\infty \int_0^t (t-s) e^{-\gamma(t-s)}\, \mathrm{d}W(s) \nu(\gamma)\, \mathrm{d}\gamma \tag{3.28}$$

$$= \int_0^\infty \left(U_\gamma(t) - U_\gamma(0)\right) \nu(\gamma)\, \mathrm{d}\gamma,\ U_\gamma(0) = 0\,. \tag{3.29}$$

One can show similar derivations for Type I, only differing in the aforementioned initial conditions. See Harms and Stefanovits (2019) for the full proof. □

Proposition 2 (Markov representation of fBM using only the Ornstein–Uhlenbeck processes $Y_\gamma(t)$)**.** *The relationship (Harms and Stefanovits (2019)):*

$$U_\gamma(t) = -\partial_\gamma Y_\gamma(t) + \left(\partial_\gamma Y_\gamma(0) + U_\gamma(0)\right) e^{-\gamma t}\,, \tag{3.30}$$

allows writing the Markov representation only in function of the Ornstein–Uhlenbeck processes $Y_\gamma(t)$:

$$B_H(t) = \begin{cases} \displaystyle\int_0^\infty \left(Y_\gamma(t) - Y_\gamma(0)\right) \mu(\gamma)\, \mathrm{d}\gamma, H < \frac{1}{2}, \\ \displaystyle -\int_0^\infty \partial_\gamma \left(Y_\gamma(t) - Y_\gamma(0)\right) \nu(\gamma)\, \mathrm{d}\gamma, H > \frac{1}{2}. \end{cases} \tag{3.31}$$

Definition 4 (Markov approximation of fBM (MA-fBM))**.** *Eq. (3.31) suggests that $B_H(t)$ could be well approximated by a Markov process $\hat{B}_H(t)$ by (i) truncating the integrals at finite γ values $(\gamma_1 ... \gamma_K)$ and (ii) approximating the integral by a numerical quadrature as a finite linear combination involving quadrature points and weights $\{\omega_k\}$. Changing the notation $Y_{\gamma_k}(t) \to Y_k(t)$:*

$$B_H(t) \approx \hat{B}_H(t) \equiv \sum_{k=1}^{K} \omega_k \left(Y_k(t) - Y_k(0) \right), \tag{3.32}$$

where for fixed γ_k the choice of ω_k depends on H and the choice of "Type I" or "Type II". For "Type II", we set $Y_k(0) = 0$. Since $Y_k(t)$ is normally distributed Thm. 2.16 and can be assumed stationary for "Type I", we can simply sample $\left(Y_1^{(I)}(0), \ldots, Y_K^{(I)}(0) \right)$ from a normal distribution with mean $\mathbf{0}$ and covariance $\boldsymbol{C}_{k,l} = 1/(\gamma_k + \gamma_l)$ (see Eq. (3.98)).

This strong approximation provably bounds the sample paths:

Theorem 3 (Alfonsi and Kebaier (2021))**.** *For rough kernels ($H < 1/2$) and $\{\omega_k\}$ following a Gaussian quadrature rule, there exists a constant c per every $t \in (0, T)$ such that:*

$$\mathbb{E}|B_H^{(II)}(t) - \hat{B}_H^{(II)}(t)| \leq O(K^{-cH}), \text{ where } 1 < c \leq 2, \tag{3.33}$$

as $K \to \infty$. Note that, in our setting, $B_H^{(II)}(0) = \hat{B}_H^{(II)}(0) = 0$.

Remark 4 (Stationarity of the increments of MA-fBM)**.** *The increments of MA-fBM are stationary for Type I and nonstationary for Type II:*

$$\hat{B}_H^{(I)}(t + \Delta t) - \hat{B}_H^{(I)}(t) \overset{d}{=} \hat{B}_H^{(I)}(\Delta t), \tag{3.34}$$

$$\hat{B}_H^{(II)}(t + \Delta t) - \hat{B}_H^{(II)}(t) \overset{d}{\neq} \hat{B}_H^{(II)}(\Delta t). \tag{3.35}$$

This follows naturally from the fact that the underlying Ornstein–Uhlenbeck processes $Y_k(t)$ are stationary for Type I and nonstationary for Type II, because of their different initial conditions $Y_k(0)$. Thus, MA-fBM has the same similarity properties as fBM (cf. Remark 1).

3.2.2 SDEs driven by (fractional) BM

Definition 5 (SDEs driven by BM (BMSDE))**.** *A common generative model for stochastic dynamical systems considers a set of observational data* $\mathcal{D} = \{O_1, \ldots, O_N\}$*, where the* O_i *are generated (conditionally) independent at random at discrete times* t_i *with a likelihood* $p_\theta(O_i \mid X(t_i))$*. The prior information about the unobserved path* $\{X(t); t \in [0,\ T]\}$ *of the latent process* $X(t) \in \mathbb{R}^M$ *is given by the assumption that* $X(t)$ *fulfils the SDE:*

$$\mathrm{d}X(t) = b_\theta(X(t), t)\,\mathrm{d}t + \sigma_\theta(X(t), t)\,\mathrm{d}W(t) \tag{3.36}$$

The drift function $b_\theta(X, t) \in \mathbb{R}^D$ *models the deterministic part of the change* $\mathrm{d}X(t)$ *of the state variable* $X(t)$ *during the infinitesimal time interval* $\mathrm{d}t$*, whereas the diffusion matrix* $\sigma_\theta(X(t), t) \in \mathbb{R}^{D\times D}$ *(assumed to be symmetric and non–singular, for simplicity) encodes the strength of the added Gaussian white noise process, where* $\mathrm{d}W(t) \in \mathbb{R}^D$ *is the infinitesimal increment of a vector of independent Wiener processes during* $\mathrm{d}t$*.*

Definition 6 (SDE driven by fBM (fBMSDE))**.** *Dfn. 8 can be formally extended to the case of fractional Brownian motion replacing* $\mathrm{d}W(t)$ *by* $\mathrm{d}B_H(t)$ *(Guerra and Nualart (2008)):*

$$\mathrm{d}X(t) = b_\theta(X(t), t)\,\mathrm{d}t + \sigma_\theta(X(t), t)\,\mathrm{d}B_H(t). \tag{3.37}$$

Remark 5. *Care must be taken in a proper definition of the diffusion part in the fBMSDE Eq. (3.37) and in developing appropriate numerical integrators for simulations, when the diffusion* $\sigma_\theta(X(t), t)$ *explicitly depends on the state* $X(t)$*. Corresponding stochastic integrals of the Itô type cannot be applied when* $H < 1/2$ *and other approaches (which are generalisations of the Stratonovich SDE for* $H = \frac{1}{2}$*) are necessary (Lysy and Pillai (2013)).*

3.3 Methods

Our goal is to extend variational inference (VI) (Bishop and Nasrabadi (2006)) to the case where the Wiener process

in Eq. (PRIOR-SDE) is replaced by an fBM as in Dfn. 6. Unfortunately, the processes defined by Eq. (3.37) are not Markovian preventing us from resorting to the standard Girsanov change of measure approach known for "ordinary" SDE to compute KL–divergences and ELBO functionals needed for VI (Opper (2019)). While Tong et al. (2022) leverage sparse approximations for Gaussian processes, this makes B_H *conditioned* on a finite but larger number of so–called *inducing variables*. We take a completely different and conceptually simple approach to VI for fBMSDE based on the exact representation of $B_H(t)$ given in Prop. 1. To this end, we first show how the *strong* Markov-approximation in Dfn. 4 can be used to approximate an SDE driven by fBM, before delving into the VI for the *Markov-Approximate fBMSDE*.

Definition 7 (Markov-Approximate fBMSDE (MA-fBMSDE))**.** *Substituting the fBM, $B_H(t)$, in Dfn. 6 by the finite linear combination of OU-processes $\hat{B}_H(t)$, we define MA-fBMSDE as:*

$$\mathrm{d}X(t) = b_\theta\left(X(t), t\right)\mathrm{d}t + \sigma_\theta\left(X(t), t\right)\mathrm{d}\hat{B}_H(t), \tag{3.38}$$

where $\mathrm{d}\hat{B}_H(t) = \sum_{k=1}^{K}\omega_k\,\mathrm{d}Y_k(t)$ with $\mathrm{d}Y_k(t) = -\gamma_k Y_k(t)\,\mathrm{d}t + \mathrm{d}W(t)$ (cf. Dfn. 4).

Proposition 3. *$X(t)$ can be augmented by the finite number of Markov processes $Y_k(t)$ (approximating $B_H(t)$) to a higher dimensional state variable of the form $Z(t) \doteq (X(t), Y_1(t), \ldots Y_K(t)) \in \mathbb{R}^{(K+1)\times D}$, such that the joint process of the augmented system becomes Markovian and can be described by an 'ordinary' SDE:*

$$\mathrm{d}Z(t) = h_\theta\left(Z(t), t\right)\mathrm{d}t + \Sigma_\theta\left(Z(t), t\right)\mathrm{d}W(t), \tag{3.39}$$

where the augmented drift vector $h_\theta \in \mathbb{R}^{(K+1)\times D}$ and the augmented

diffusion matrix $\Sigma_\theta(Z,t) \in \mathbb{R}^{(K+1)\times D\times D}$ *are given by*

$$h_\theta(Z,t) = \begin{pmatrix} b_\theta(X,t) - \sigma_\theta(X,t)\sum_k \omega_k\gamma_k Y_k \\ -\gamma_1 Y_1 \\ \cdots \\ -\gamma_K Y_K \end{pmatrix}, \tag{3.40}$$

$$\Sigma_\theta(Z,t) = \begin{pmatrix} \bar{\omega}\sigma_\theta(X,t) \\ I_D \\ \vdots \\ I_D \end{pmatrix}, \tag{3.41}$$

where $I_D \in \mathbb{R}^{D\times D}$ *is the identity matrix . We will refer to Eq. (3.39) as the* variational prior.

Proof. Each of the D components of the vectors Y_k uses the same scalar weights $\omega_k \in \mathbb{R}$. Also, note that each Y_k is driven by the *same* vector of Wiener processes. Hence, we obtain the system of SDEs given by

$$\begin{aligned} \mathrm{d}X(t) &= b_\theta(X(t),t)\,\mathrm{d}t - \sigma_\theta(X(t),t)\sum_k \omega_k\gamma_k Y_k(t)\,\mathrm{d}t \\ &\quad + \bar{\omega}\sigma_\theta(X(t),t)\,\mathrm{d}W(t) \qquad (3.42) \\ \mathrm{d}Y_k(t) &= -\gamma_k Y_k(t)\,\mathrm{d}t + \mathrm{d}W(t) \quad \text{for} \quad k = 1,\ldots,K \qquad (3.43) \end{aligned}$$

where $\bar{\omega} \doteq \sum_k \omega_k$. This system of equations can be collectively represented in terms of the augmented variable $Z(t) := (X(t), Y_1(t), \ldots Y_K(t)) \in \mathbb{R}^{(K+1)\times D}$ leading to a single SDE specified by Eqs. (3.39) to (3.41). □

Eq. (3.39) represents a standard SDE driven by Wiener noise allowing us to utilise the standard tools of stochastic analysis, such as the Girsanov change of measure theorem and derive the *evidence lower bounds* (ELBO) required for VI. This is what we will exactly do in the sequel.

Proposition 4 (Controlled MA-fBMSDE)**.** *The paths of Eq. (3.39) can be steered by adding a control term* $u(X, Y_1, \ldots, Y_K, t) \in \mathbb{R}^D$ *that*

depends on all variables to be optimised, to the drift h_θ resulting in the transformed SDE, a.k.a. the variational posterior*:*

$$\begin{aligned} \mathrm{d}\tilde{Z}(t) = &\left(h_\theta\left(\tilde{Z}(t), t\right) + \Sigma_\theta(\tilde{Z}(t), t) u(\tilde{Z}(t), t)\right) \mathrm{d}t \\ &+ \Sigma_\theta\left(\tilde{Z}(t), t\right) \mathrm{d}W(t) \end{aligned} \tag{3.44}$$

Sketch of the proof. Using the fact that the posterior probability measure over paths $\tilde{Z}(t)$ $\{\tilde{Z}(t); t \in [0, T]\}$ is absolutely continuous w.r.t. the prior process, we apply the Girsanov theorem (*cf.* Sec. 3.7.2) on Eq. (3.39) to write the new drift, from which the posterior SDE in Eq. (3.44) is obtained. □

We will refer to Eq. (3.44) as the *variational posterior*. In what follows, we will assume a parametric form for the control function $u(\tilde{Z}(t), t) \equiv u_\phi(\tilde{Z}(t), t)$ (as *e.g.* given by a neural network) and will devise a scheme for inferring the *variational parameters* (θ, ϕ)*, i.e.,* variational inference.

Proposition 5 (Variational Inference for MA-fBMSDE)**.** *The variational parameters ϕ are optimised by minimising the KL–divergence between the posterior and the prior, where the corresponding evidence lower bound (ELBO) to be maximised is:*

$$\begin{aligned} &\log p\left(O_1, O_2, \ldots, O_N \mid \theta\right) \geq \\ &\quad \mathbb{E}_{\tilde{Z}_u}\Bigg[\sum_{i=1}^{N} \log p_\theta\left(O_i \mid \tilde{Z}(t_i)\right) - \int_0^T \frac{1}{2}\left\|u_\phi\left(\tilde{Z}(t), t\right)\right\|^2 \mathrm{d}t \\ &\qquad - D_{\mathrm{KL}}\left(p_\theta\left(\tilde{Z}(0)\right) || q_\phi\left(\tilde{Z}(0)\right)\right)\Bigg], \end{aligned} \tag{3.45}$$

where the observations $\{O_i\}$ are included by likelihoods $p_\theta\left(O_i \mid \tilde{Z}(t_i)\right)$ and the expectation is taken over random paths of the approximate posterior process defined by (Eq. (3.44)).

Sketch of the proof. Since we can use Girsanov's theorem II (Øksendal (2003)), the variational bound derived in Li et al. (2020) (App. 9.6.1) directly applies. □

Remark 6. *It is noteworthy that the measurements with their likelihoods* $p_\theta\left(O_i \mid \tilde{X}(t_i)\right)$ *depend only on the component* $\tilde{X}(t)$ *of the augmented state* $\tilde{Z}(t)$*. The additional variables* $\tilde{Y}_k(t)$ *which are used to model the noise in the SDE are not directly observed. However, computation of the ELBO requires initial values for all state variables* $\tilde{Z}(0)$ *(or their distribution), where the difference between Type I and Type II MA-fBM is significant. For Type II,* $\tilde{Y}_k(0) = 0$*, for both prior and posterior, and only* $\tilde{X}(0)$ *needs to be modeled. For Type I, the full* $p_\theta\left(\tilde{Z}(0)\right)$ *and* $q_\phi\left(\tilde{Z}(0)\right)$ *should be incorporated, wherein* $p_\theta\left(\tilde{Y}_k(0)\right)$ *is given by Dfn. 4.*

Remark 7. *For MA-fBM Type I, in case we want to optimize or learn* $p_\theta\left(\tilde{Z}(0)\right)$*, we need to adhere to Dfn. 4, i.e.,* $p_\theta\left(\tilde{Y}_k(0), \tilde{Y}_l(0)\right) \sim \mathcal{N}\left(0, \frac{1}{\gamma_k+\gamma_l}\right)$*. Suppose for simplicity the scalar case* $D = 1$ *and* $p_\theta\left(\tilde{Z}(0)\right)$ *is modelled with a multivariate Gaussian, we propose a parameterization*

$$p_\theta\left(\tilde{Z}(0)\right) = \mathcal{N}\left([m_\theta, 0, \ldots, 0]^\top, \begin{bmatrix} c_\theta & \boldsymbol{w}_\theta^\top \\ \boldsymbol{w}_\theta & \boldsymbol{C} \end{bmatrix}\right), \tag{3.46}$$

$$\boldsymbol{C}_{(k,l)} = \frac{1}{\gamma_k + \gamma_l}, \tag{3.47}$$

where $m_\theta \in \mathbb{R}$*,* $\alpha_\theta \in \mathbb{R}$*,* $\boldsymbol{w}_\theta \in \mathbb{R}^K$*, and* $c_\theta = e^{\alpha_\theta} + \boldsymbol{w}_\theta \boldsymbol{C}^{-1} \boldsymbol{w}_\theta^\top$*. This specific construction guarantees a valid covariance matrix (postive semi-definite) for any choice of* α_θ *and* $\boldsymbol{w}_\theta$*, due to properties of its Schur complement (Gallier et al. (2010)). This can be trivially extended to the multi-dimensional case, where the* D *components of each* $Y_k(0)$ *are independent.*

3.3.1 Optimizing the approximation

In the literature, different choices of γ_k and ω_k have been proposed (Harms and Stefanovits (2019), Carmona and Coutin (1998a), and Carmona et al. (2000)) and for certain choices, it is possible to obtain a superpolynomial rate, as shown by Bayer

and Breneis (2023a) for the Type II case. In this work we define our choice of γ_k as a geometric series, defined by its endpoints

$$(\gamma_1, \ldots, \gamma_K) \equiv (\gamma_{\min}, \ldots, \gamma_{\max}), \tag{3.48}$$

or, more verbose,

$$\gamma_k \equiv \gamma_{\min}^{1-z} \gamma_{\max}^{z}, \quad z = \frac{k-1}{K-1}, \quad k = 1, \ldots, K. \tag{3.49}$$

Thus in what follows, the choice of γ_k is entirely defined by K, the number of OU processes, and $\gamma_{\min}$ and $\gamma_{\max}$. Rather than relying on methods of numerical quadrature, we consider a simple measure for the quality of the approximation over a fixed time interval $[0,\ T]$ which can be *optimised analytically* for both types I and II. To optimize ω_k values, we first provide a closed form expression for the approximation error and then show how we can solve for the ω_k that minimizes this error.

$$\mathcal{E}^{(I,II)}(\boldsymbol{\omega}) = \frac{\int_0^T \mathbb{E}\left[\left(\hat{B}_H^{(I,II)}(t) - B_H^{(I,II)}(t)\right)^2\right] \mathrm{d}t}{\int_0^T \mathbb{E}\left[B_H^{(I,II)}(t)^2\right] \mathrm{d}t} \tag{3.50}$$

$$= \frac{\int_0^T \mathbb{E}\left[\hat{B}_H^{(I,II)}(t)^2\right] \mathrm{d}t}{\int_0^T \mathbb{E}\left[B_H^{(I,II)}(t)^2\right] \mathrm{d}t} - 2\frac{\int_0^T \mathbb{E}\left[\hat{B}_H^{(I,II)}(t) B_H^{(I,II)}(t)\right] \mathrm{d}t}{\int_0^T \mathbb{E}\left[B_H^{(I,II)}(t)^2\right] \mathrm{d}t} + 1 \tag{3.51}$$

$$= \sum_{i,j} \omega_i \omega_j \frac{\boldsymbol{A}_{i,j}^{(I,II)}}{c^{(I,II)}} - 2\sum_k \omega_k \frac{\boldsymbol{b}_k^{(I,II)}}{c^{(I,II)}} + 1 \tag{3.52}$$

$$= \boldsymbol{\omega}^T \frac{\boldsymbol{A}^{(I,II)}}{c^{(I,II)}} \boldsymbol{\omega} - 2\frac{\boldsymbol{b}^{(I,II)\,T}}{c^{(I,II)}}\ \boldsymbol{\omega} + 1. \tag{3.53}$$

Where for Type I, using Eqs. (3.1), (3.107) and (3.120),

$$\boldsymbol{A}_{i,j}^{(I)} = \int_0^T \frac{2 - e^{-\gamma_i t} - e^{-\gamma_j t}}{\gamma_i + \gamma_j} \mathrm{d}t \tag{3.54}$$

$$= \frac{2T + \frac{e^{-\gamma_i T} - 1}{\gamma_i} + \frac{e^{-\gamma_j T} - 1}{\gamma_j}}{\gamma_i + \gamma_j}, \tag{3.55}$$

$$\begin{aligned}\boldsymbol{b}_k^{(I)} &= \int_0^T \frac{2 - e^{-\gamma_k t} - Q\left(H + 1/2, \gamma_k t\right) e^{\gamma_k t}}{\gamma_k^{H+1/2}} \,\mathrm{d}t \\ &= \frac{2T}{\gamma_k^{H+1/2}} - \frac{T^{H+1/2}}{\gamma_k \Gamma(H + 3/2)} \\ &\quad + \frac{e^{-\gamma_k T} - Q\left(H + 1/2, \gamma_k T\right) e^{\gamma_k T}}{\gamma_k^{H+3/2}} \,, \end{aligned} \tag{3.56}$$

$$c^{(I)} = \int_0^T V_H^{(I)} t^{2H} \,\mathrm{d}t = V_H^{(I)} \frac{T^{2H+1}}{2H+1} \,, \tag{3.57}$$

where $Q(z,x) = \frac{1}{\Gamma(z)} \int_x^\infty t^{z-1} e^{-t} \,\mathrm{d}t$ is the regularized upper incomplete gamma function. Similarly for Type II, using Eqs. (3.4), (3.112) and (3.124),

$$\boldsymbol{A}_{i,j}^{(II)} = \int_0^T \frac{1 - e^{-(\gamma_i + \gamma_j)t}}{\gamma_i + \gamma_j} \,\mathrm{d}t = \frac{T + \frac{e^{-(\gamma_i+\gamma_j)T} - 1}{\gamma_i + \gamma_j}}{\gamma_i + \gamma_j} \,, \tag{3.58}$$

$$\begin{aligned}\boldsymbol{b}_k^{(II)} &= \int_0^T \frac{P\left(H + 1/2, \gamma_k t\right)}{\gamma_k^{H+1/2}} \,\mathrm{d}t \\ &= \frac{T}{\gamma_k^{H+1/2}} P\left(H + 1/2, \gamma_k T\right) \\ &\quad - \frac{H + 1/2}{\gamma_k^{H+3/2}} P\left(H + 3/2, \gamma_k T\right) \,, \end{aligned} \tag{3.59}$$

$$c^{(II)} = \int_0^T V_H^{(II)} t^{2H} \,\mathrm{d}t = V_H^{(II)} \frac{T^{2H+1}}{2H+1} \,, \tag{3.60}$$

where $P(z,x) = \frac{1}{\Gamma(z)} \int_0^x t^{z-1} e^{-t} \,\mathrm{d}t$ is the regularized lower incomplete gamma function. The quadratic form (Eq. (3.53)) is minimal for the solution of

$$\boldsymbol{A}^{(I,II)} \boldsymbol{\omega}^* = \boldsymbol{b}^{(I,II)} \,, \tag{3.61}$$

and the optimal $\boldsymbol{\omega}^*$ results in an approximation error

$$\mathcal{E}^{(I,II)}(\boldsymbol{\omega}^*) = 1 - \frac{\boldsymbol{b}^{(I,II)\,T}}{c^{(I,II)}} \boldsymbol{\omega}^* \,. \tag{3.62}$$

Proposition 6 (Exactly one optimal solution for MA-fBMSDE weights)**.** *There is exactly one solution for the optimal weights $\boldsymbol{\omega}^*$ in Eq. (3.61) for any choice of γ_k where $\gamma_k \neq \gamma_l$ for $k \neq l$ and $0 \leq \gamma_k < \infty$ for all γ_k.*

Proof. Eq. (3.61) has exactly one solution if $\boldsymbol{A}^{(I,II)}$ is positive definite, which is defined as

$$\boldsymbol{\omega}^T \boldsymbol{A}^{(I,II)} \boldsymbol{\omega} > 0 \text{ for all } \boldsymbol{\omega} \in \mathbb{R}^K \setminus \{\mathbf{0}\}\,. \tag{3.63}$$

Since (see Eqs. (3.51) and (3.53))

$$\boldsymbol{\omega}^T \boldsymbol{A}^{(I,II)} \boldsymbol{\omega} = \int_0^T \mathbb{E}\left[\hat{B}_H^{(I,II)}(t)^2\right] \mathrm{d}t\,, \tag{3.64}$$

$\boldsymbol{A}$ is positive definite if

$$\int_0^T \mathbb{E}\left[\hat{B}_H^{(I,II)}(t)^2\right] \mathrm{d}t > 0 \text{ for all } \boldsymbol{\omega} \in \mathbb{R}^K \setminus \{\mathbf{0}\}\,. \tag{3.65}$$

This inequality holds unless $\hat{B}_H^{(I,II)}(t) = 0$. Recall that $\hat{B}_H^{(I,II)}(t)$ is a linear combination of K Ornstein–Uhlenbeck processes with speed of mean reversion γ_k driven by the same Wiener process. Under the trivial conditions that $\gamma_i \neq \gamma_j$ (so they can not be cancelled out) and $0 \leq \gamma_k < \infty$, $\hat{B}_H^{(I,II)}(t) = 0 \iff \boldsymbol{\omega} = \mathbf{0}$. □

3.4 Applications & Evaluations

3.4.1 Fractional Ornstein-Uhlenbeck process

Applying our method on linear problems, allows comparing empirical results to analytical formulations derived *e.g.* using Gaussian process methodology. We assess the reconstruction capability of our method on a fractional Ornstein-Uhlenbeck (fOU) process, that is an OU–process driven by fBM: $\mathrm{d}X(t) = -\theta X(t)\,\mathrm{d}t + \mathrm{d}B_H^{(I)}$, where the speed of mean reversion $\theta > 0$. This is a stationary Gaussian process. For $H > 1/2$, we have an analytical solution for the covariance (see Sec. 3.7.3 for more detail):

$$\begin{aligned}\operatorname{Cov}&\left(X(t_1), X(t_2)\right) \\ &= \frac{H(2H-1)}{2\theta}\left(\Gamma(2H-1)\frac{e^{-\theta t} + e^{\theta t} Q(2H-1, \theta t)}{\theta^{2H-1}}\right. \\ &\qquad \left. + \int_0^t e^{-\theta(t-u)} u^{2H-2}\,\mathrm{d}u\right), \quad t = |t_1 - t_2|\,.\end{aligned} \tag{3.66}$$

Additionally, for the fOU process driven by MA-fBM (Type I): $\mathrm{d}X(t) = -\theta X(t)\,\mathrm{d}t + \mathrm{d}\hat{B}_H^{(I)}$, the covariance is given by Eq. (3.155) (see Sec. 3.7.3 for more detail). Since this is a stationary process, it does not start at 0 and we need the distribution of the initial values $(X(0), Y_1(0), \ldots, Y_K(0))$ (Sec. 3.7.3):

$$\mathrm{Var}(X(0)) = \sum_{k,l} \frac{\omega_k \omega_l}{2\theta} \left(1 - \frac{2\gamma_l^2}{(\theta+\gamma_l)(\gamma_k+\gamma_l)}\right), \tag{3.67}$$

$$\mathrm{Cov}\left(X(0), Y_l(0)\right) = \sum_k \frac{\omega_k \gamma_l}{(\theta+\gamma_l)(\gamma_k+\gamma_l)}, \tag{3.68}$$

$$\mathrm{Cov}\left(Y_k(0), Y_l(0)\right) = \frac{1}{\gamma_k+\gamma_l}. \tag{3.69}$$

Lastly, we define a model that is optimized using our VI approach. The control function is a dense neural network with two hidden layers of 1000 neurons each. Time is encoded using a series of sine and cosine functions and fed as input to the network, together with the state of the process

$$[X(t), Y_1(t), \ldots, Y_K(t), s_1(t), c_1(t), \ldots, s_8(t), c_8(t)], \tag{3.70}$$

$$s_n(t) = \sin\left(\frac{n\pi t}{t_{\max}}\right), \quad c_n(t) = \cos\left(\frac{n\pi t}{t_{\max}}\right), \tag{3.71}$$

where $t_{\max}$ is the length of the data sequence. The posterior of the initial conditions $q_\theta(X(0), Y_1(0), \ldots, Y_K(0))$ is defined by $\mathcal{N}(\boldsymbol{m}_\theta, \boldsymbol{C}_\theta)$ where $\boldsymbol{C}_\theta$ is parameterized by its Cholesky decomposition, such that it is guaranteed to be positive semi-definite. Concretely, the parameters to be optimized are thus the neural network weights in u_θ, $\boldsymbol{m}_\theta$ and the Cholesky decomposition of $\boldsymbol{C}_\theta$.

For easier notation, we define for random variables $\boldsymbol{X} \in \mathbb{R}^{N_X}$ and $\boldsymbol{Y} \in \mathbb{R}^{N_Y}$ the matrix $K(\boldsymbol{X}, \boldsymbol{Y}) \in \mathbb{R}^{N_X \times N_Y}$ as:

$$K(\boldsymbol{X}, \boldsymbol{Y})_{i,j} = \mathbb{E}\left[\boldsymbol{X}_i \boldsymbol{Y}_j\right]. \tag{3.72}$$

We generate one data sample $\boldsymbol{x}$ at time points $\boldsymbol{t}$ using the true covariance kernel and measurement noise σ:

$$\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{0}, K(X(\boldsymbol{t}), X(\boldsymbol{t})) + \sigma^2 \boldsymbol{I}). \tag{3.73}$$

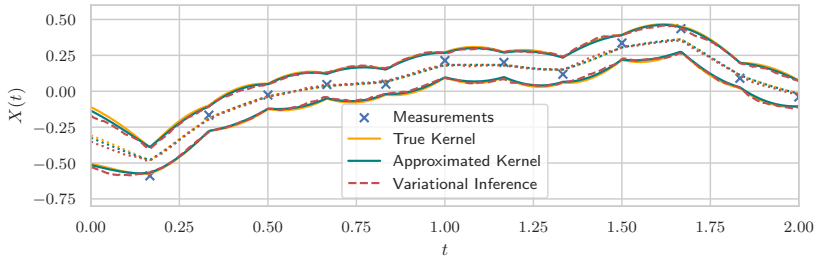


Figure 3.2 Visualization of the fractional Ornstein–Uhlenbeck (fOU) process with $H = 0.7$, $\theta = 5$ and measurement noise $\sigma = 0.1$, for 12 equally spaced measurements between $t = 0$ and $t = 2$. Posterior distributions are shown as the mean ± 1 standard deviation. The approximated kernel is close to the true kernel, showing the good approximation quality of MA-fBM. Similarly the learned posterior model using VI is close to the true kernel, showing the validity of our VI approach.

Using Gaussian process methodology (Rasmussen, Williams, et al. (2006)), we can derive the exact posterior at test points $\boldsymbol{t}_*$ using the true kernel:

$$\mathbb{E}\left[\tilde{X}(\boldsymbol{t}_*)\right] = K(\boldsymbol{t}_*, \boldsymbol{t})\left(K(\boldsymbol{t}, \boldsymbol{t}) + \sigma^2 \boldsymbol{I}\right)^{-1} \boldsymbol{x}\,, \tag{3.74}$$

$$\begin{aligned}\mathbb{E}\left[\tilde{X}(\boldsymbol{t}_*)^2\right] &= K(\boldsymbol{t}_*, \boldsymbol{t}_*) \\ &\quad - K(\boldsymbol{t}_*, \boldsymbol{t})\left(K(\boldsymbol{t}, \boldsymbol{t}) + \sigma^2 \boldsymbol{I}\right)^{-1} K(\boldsymbol{t}, \boldsymbol{t}_*)\,,\end{aligned} \tag{3.75}$$

and similarly for the approximated covariance for $\hat{X}(\boldsymbol{t}_*)$. We compare the analytical solution using the true and the approximated kernels, with the VI approach in Fig. 3.2 shows a visual comparison of the analytical solution (true and approximated), and the VI approach. The VI approach is visualized by sampling 256 posterior paths of the trained model, and showing the mean and ± 1 standard deviation. The approximated kernel is close to the true kernel, showing the good approximation quality of MA-fBM. And the learned model using VI is close to the true kernel, showing the validity of our VI approach.

Our approach additionally allows us to estimate the Hurst index H and the speed of mean reversion θ. We can simply add the

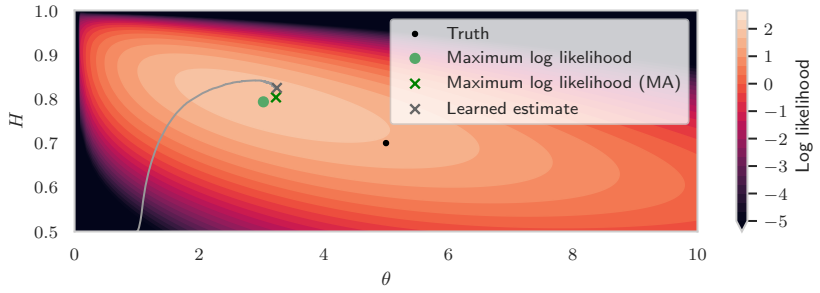


Figure 3.3 Joint learning of the posterior model, and H and θ using gradient descent. The log evidence using the true kernel is shown as reference. The gray path shows the optimization path of H and θ, initialized at respectively 0.5 and 1. The black dot indicates the true H and θ, and the green dot and green cross show the theoretical maximum log evidence using the true kernel and the approximated kernel, respectively. There is some distance between the true values and the maximum log evidence, which is expected due to the limited number of measurements and the measurement noise. The estimation of H and θ is reasonably close to the theoretical maximum log evidence using the MA kernel. This shows that our VI approach is valid and suitable for estimating the parameters of the fOU process.

parameters to the model, and optimize the ELBO with respect to these parameters using gradient descent, jointly with the control function and posterior of the initial values. This leads to the results for an estimation of H and θ presented in Fig. 3.3, on the same data sequence of Fig. 3.2.

3.4.2 Estimating time-dependent Hurst index

Since our method of optimizing ω_k is tractable and differentiable, we can directly optimize a parameterized H by maximizing the ELBO. Also a time–dependent Hurst index $H(t)$ can be modelled, leading to multifractional Brownian Motion (Peltier and Véhel (1995)). We directly compare with a toy problem presented in Tong et al. 2022, Sec. 5.2. We use the same model for $H(t)$, a

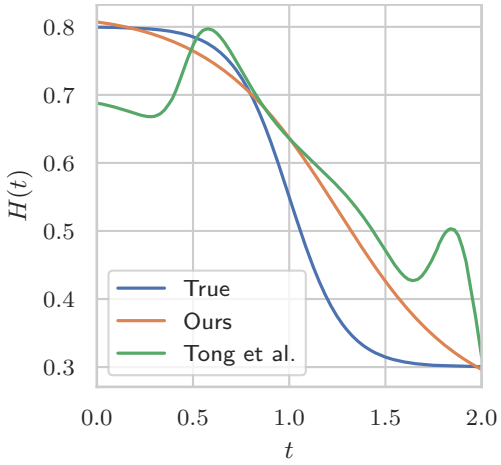


Figure 3.4 Estimating time–dependent $H(t)$ from data.

neural network with one hidden layer of 10 neurons and activation function tanh, and a final sigmoid activation, and the same input $[\sin(t), \cos(t), t]$. We use $\hat{B}_H^{(II)}$ since their method is Type II. Fig. 3.4 shows a reasonable estimation of $H(t)$, which is more accurate than the result from Tong et al. (2022), *cf.* Sec. 3.7.5 for more details.

3.4.3 Financial data

The fractional Cox-Ingersol-Ross (fCIR) process (Lysy and Pillai (2013)) is defined as

$$\mathrm{d}X(t) = -\theta_t(X(t) - \mu_t)\,\mathrm{d}t + \sigma_t X(t)^{1/2}\,\mathrm{d}B_{H_t}(t)\,. \tag{3.76}$$

Within our framework, we define the fCIR process as the prior, and train a posterior SDE on 3-Month US Treasury Bills data from 1954 to 2013 (Fig. 3.5). Note that we defined the fCIR process with time-dependent parameters $(\theta_t, \mu_t, \sigma_t, H_t)$. This allows us to use it to model the full range of data, where the parameters would change over time, at a relatively slow rate. We parameterize the parameters using a cubic spline, with 50 points (meaning around one point per year). The control function for the posterior is parameterized as a fully connected neural network with one

Figure 3.5 3-Month US Treasury Bills.

hidden layer of 1000 neurons and the $\tanh$ activation function. Its input is the state of the posterior $(\tilde{X}(t), \tilde{Y}_1(t), \dots, \tilde{Y}_K(t))$ and a learned time-encoding $f_t \in \mathbb{R}^8$. f_t is a learned feature vector that is defined seperately for each day in the dataset, and linearly interpolated inbetween. We use Type I MA-fBM with time horizon T set at 14863, the number of days in the dataset. We set the integration timestep Δt at 0.5, and choose $\gamma_{\max}$ at 0.9, so the condition is not violated. We then chose $\gamma_{\min} = 10^{-5}$ and $K = 25$ for the lowest approximation error, based on plots we generated for these conditions as in Figs. 3.13 to 3.15. We also compare to the fractional Ornstein-Uhlenbeck (fOU) process

$$\mathrm{d}X(t) = -\theta_t(X(t) - \mu_t)\,\mathrm{d}t + \sigma_t\,\mathrm{d}B_{H_t}(t)\,. \tag{3.77}$$

For training, we take subsets of the data that are 1260 days long. For the initial values, we take $\tilde{X}(t_0)$ as the given first datapoint x_0. Fig. 3.6 shows the estimated time-dependent Hurst index H_t for both the fCIR and the fOU process.

3.4.4 Latent video models

To assess the video modelling capabilities of our framework, we train models on stochastic video datasets. The prior drift h_θ, diffusion σ_θ and control term u are parameterized by neural networks. The prior model is used as a stochastic video predictor, where we condition on the first N frames to predict the next frames in the sequence. More intuitively, the posterior model reconstructs the given sequence of frames, while minimizing the

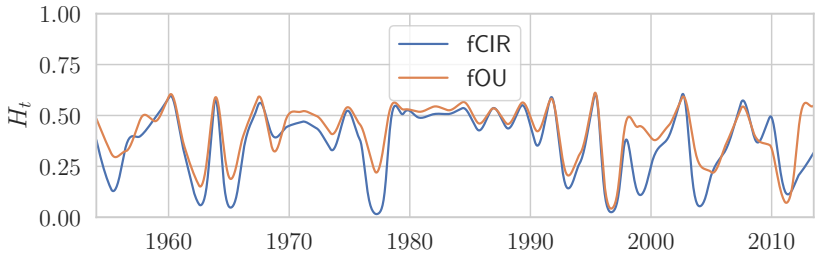


Figure 3.6 Estimate of time dependent Hurst H_t on the 3-Month US Treasury Bills dataset.

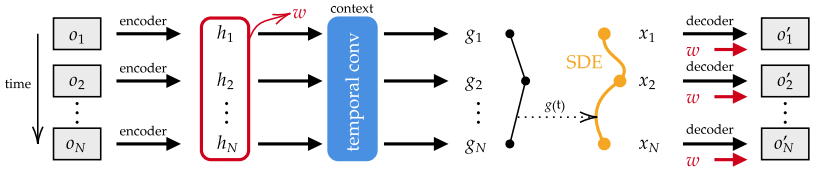


Figure 3.7 Schematic of the latent SDE video model. Video frames $\{o_i\}_i$ are encoded to vectors $\{h_i\}_i$. The static content vector w, that is free of the dynamic information, is inferred from $\{h_i\}_i$. The context model processes the information with temporal convolution layers, so that its outputs $\{g_i\}_i$ contain information from neighbouring frames. A linear interpolation on $\{g_i\}_i$ allows the posterior SDE model to receive time–appropriate information $g(t)$, at (intermediate) time–steps chosen by the SDE solver. Finally, the states $\{x_i\}_i$ and static w are decoded to reconstruct frames $\{o'_i\}_i$.

Table 3.1 Stochastic Moving MNIST results.

Model	ELBO	PSNR
SVG	N/A	14.50
SLRVP	N/A	16.93
BM	−913.60	14.90
MA-fBM	−608.00	15.30

Table 3.2 Double pendulum results.

Model	ELBO	PSNR
BM	−545.13	26.11
MA-fBM	−636.61	27.09

control actions of u. This leads to a prior that will model the dataset, so that the posterior will be able to model the *specific* data sequence during training with minimal u input. It is paramount that the control function u receives relevant information during the SDE integration, so that it can steer the SDE in the right direction. See Fig. 3.7 for a schematic explanation of our model and Sec. 3.7.5 for a detailed explanation of submodel architectures and hyperparameters.

We evaluate the stochastic video predictions by sampling 100 predictions and reporting the Peak Signal-to-Noise Ratio (PSNR) of the best sample, calculated frame-wise and averaged over time. This is the same approach as Franceschi et al. (2020) which allows a direct comparison. Furthermore, we report the ELBO on the test set, indicating how well the model has captured the data.

We train models on Stochastic Moving MNIST (SM-MNIST) (Denton and Fergus (2018)), a video dataset where two MNIST numbers move on a canvas and bounce off the edge with random velocity in a random direction. Our MA-fBM driven model is on par with closely related discrete-time methods such as SVG (Denton and Fergus (2018)) or SLRVP (Franceschi et al. (2020)), in terms of PSNR, and is better than the BM baseline in terms of PSNR and ELBO (Tab. 3.1). The Hurst index was optimized during training, and reached $H = 0.90$ at convergence (long-term memory), indicating that MA-fBM is better suited to the data than BM.

We also report results on a real-world video dataset of a double pendulum (Asseman et al. (2018)), where we investigate whether the chaotic behaviour can be modelled by an SDE driven by fBM. Our MA-fBM driven model is better than the BM baseline, both for the test set ELBO as for the PSNR metric (Tab. 3.2). The Hurst index reached a value of $H = 0.93$ at convergence. See Fig. 3.8 for stochastic video predictions and Sec. 3.7.6 for additional qualitative results.

3.5 Further Studies

We implemented our method in JAX (Bradbury et al. (2018)), using Diffrax (Kidger (2021)) for SDE solvers, Optax (Babuschkin et al. (2020)) for optimization, Diffrax (Babuschkin et al. (2020)) for distributions and Flax (Heek et al. (2023)) for neural networks. Unlike Tong et al. (2022) our approach is agnostic to discretization and the choice of the solver. Hence, in all experiments we can use the *Stratonovich–Milstein* solver, *cf.* Sec. 3.7.5 for more details.

3.5.1 Sampling trajectories for qualitative evaluation of the Markov approximation

Since fBM and MA-fBM processes are Gaussian, we can sample trajectories on predefined time points by using their respective covariance kernels. By choosing a list of N time points $(t_1, \dots, t_N)$, we calculate covariance matrices $\boldsymbol{K}$, $\boldsymbol{C}$ and $\hat{\boldsymbol{K}}$ defined as

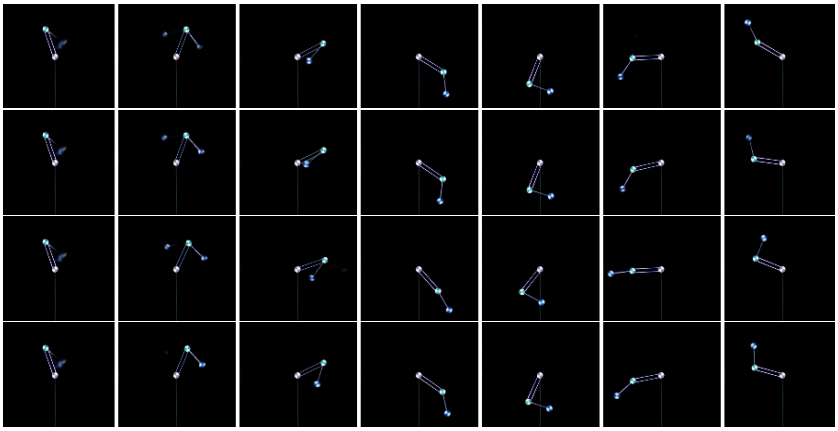
$$\boldsymbol{K}_{i,j} = \mathbb{E}\left[B_H(t_i) B_H(t_j)\right] , \tag{3.78}$$

$$\boldsymbol{C}_{i,j} = \mathbb{E}\left[B_H(t_i) \hat{B}_H(t_j)\right] , \tag{3.79}$$

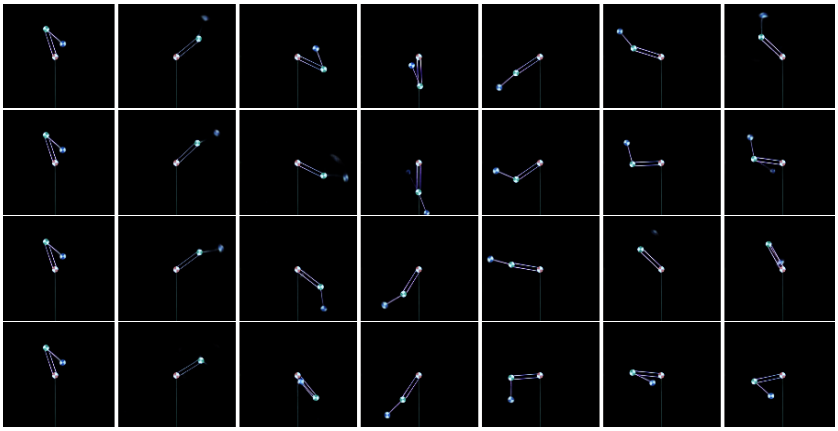
$$\hat{\boldsymbol{K}}_{i,j} = \mathbb{E}\left[\hat{B}_H(t_i) \hat{B}_H(t_j)\right] , \tag{3.80}$$

where we have respectively Eqs. (3.1), (3.107) and (3.119) for Type I and Eqs. (3.4), (3.112) and (3.123) for Type II. We start by sampling a trajectory $\boldsymbol{x}$ of the true fBM process $B_H(t)$:

$$\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{K}) . \tag{3.81}$$



(a) BM



(b) MA-fBM

Figure 3.8 Stochastic video predictions using the trained prior of a model driven by BM (a) and a model driven by MA-fBM (b) trained on the double pendulum dataset. The initial state is conditioned on the same data for all samples. Four samples are shown for each model, and 7 evenly spaced frames from the total of 20 frames in the sequence are shown. The MA-fBM samples show a more diverse, chaotic behaviour, thus better capturing the dynamics in the data.

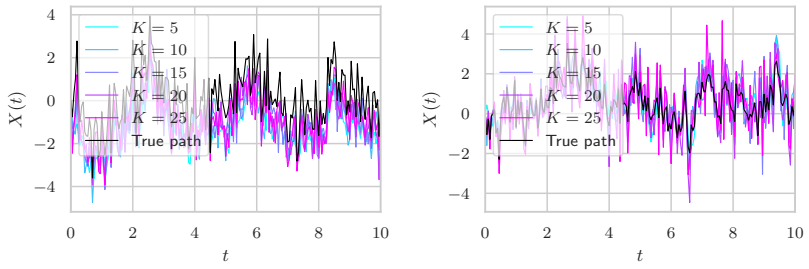
Next, we sample a trajectory $\hat{\boldsymbol{x}}$ of the MA-fBM process $\hat{B}_H(t)$ by using the conditional distribution on $\boldsymbol{x}$:

$$\hat{\boldsymbol{x}} \sim \mathcal{N}(\boldsymbol{\mu}|\boldsymbol{x}, \boldsymbol{\Sigma}|\boldsymbol{x})\,, \tag{3.82}$$

$$\boldsymbol{\mu}|\boldsymbol{x} = \boldsymbol{C}^\top \boldsymbol{K}^{-1}\boldsymbol{x}\,, \tag{3.83}$$

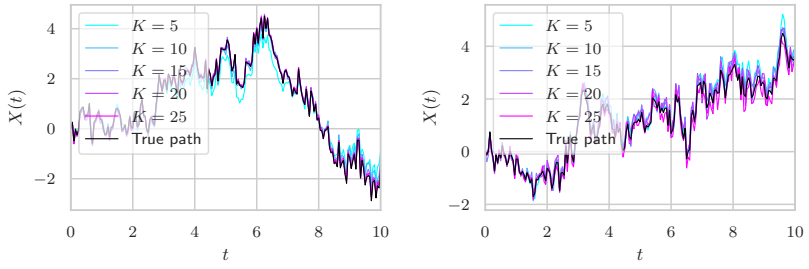
$$\boldsymbol{\Sigma}|\boldsymbol{x} = \hat{\boldsymbol{K}} - \boldsymbol{C}^\top \boldsymbol{K}^{-1}\boldsymbol{C}\,. \tag{3.84}$$

This approach allows us to qualitatively evaluate the Markov approximation by visually comparing sampled trajectories of the true fBM process and the MA-fBM process. In Figs. 3.9 to 3.12, we show the sampled trajectories for both Type I and Type II for various Hurst indices H and number of OU–processes K.



(a) H=0.1 (Type I) **(b)** H=0.1 (Type II)

Figure 3.9 Generated trajectories of (**a**) Type I and (**b**) Type II MA-fBM compared to true fBM for varying K and $H = 0.1$.

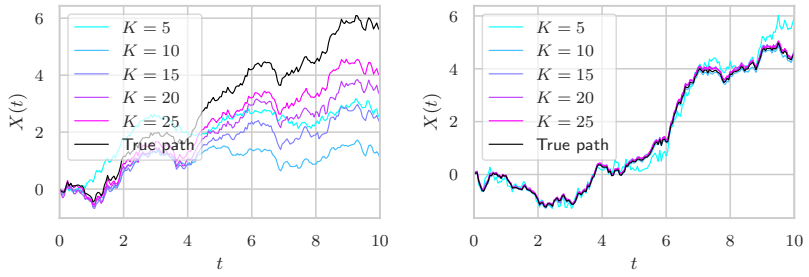


(a) H=0.3 (Type I) **(b)** H=0.3 (Type II)

Figure 3.10 Generated trajectories of (**a**) Type I and (**b**) Type II MA-fBM compared to true fBM for varying K and $H = 0.3$.

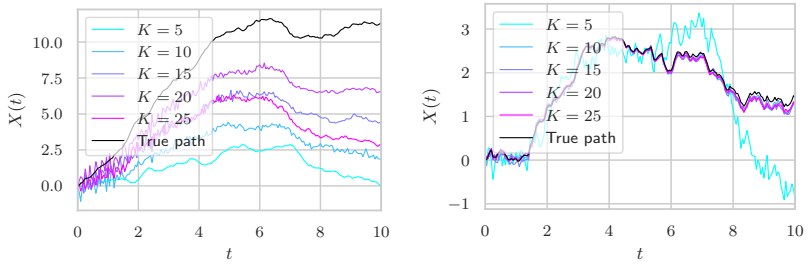
3.5.2 Numerical analysis of the approximation accuracy

Recent works on the Markov approximation of fBM have discussions and proofs on optimal rates of convergence with the number of Markov processes used in the approximation (Harms (2020) and Bayer and Breneis (2023a, 2023b)). The closed-form optimal solution presented in Sec. 3.3.1 does not allow a rate



(a) H=0.7 (Type I) **(b)** H=0.7 (Type II)

Figure 3.11 Generated trajectories of (**a**) Type I and (**b**) Type II MA-fBM compared to true fBM for varying K and $H = 0.7$.

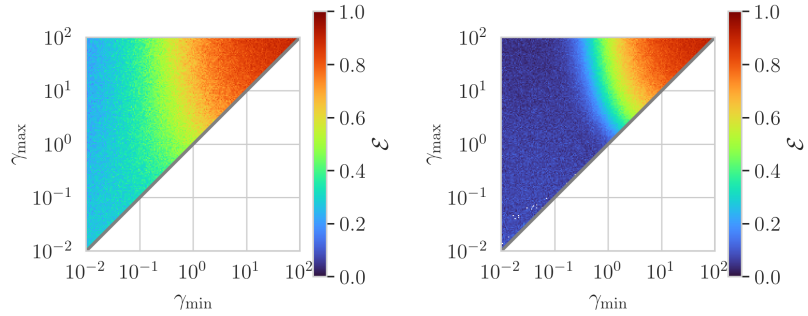


(a) H=0.9 (Type I) **(b)** H=0.9 (Type II)

Figure 3.12 Generated trajectories of (**a**) Type I and (**b**) Type II MA-fBM compared to true fBM for varying K and $H = 0.9$.

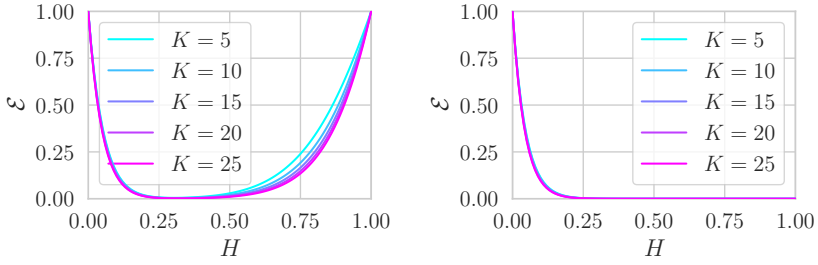
of convergence analysis, since its from is free of any structure. In this section we provide numerical analysis of the errors for varying H, number of Markov processes K and choices of γ_k.

To find optimal $(\gamma_{\min}, \gamma_{\max})$ values, one can perform grid searches, for varying Hurst H, number of Markov processes K and time horizon T values, for both types I and II. We use the approximation error defined in Eq. (3.62) as a metric and show an example of this in Fig. 3.13. Figs. 3.14 and 3.15 shows the approximation error for varying H and K. Note that Carmona and Coutin (1998b) show that $\gamma_k \Delta t > 1/2$ leads to unstable integration of the OU–processes, where Δt is the integration step. Care should be taken that $\gamma_{\max} \Delta t < 1/2$,



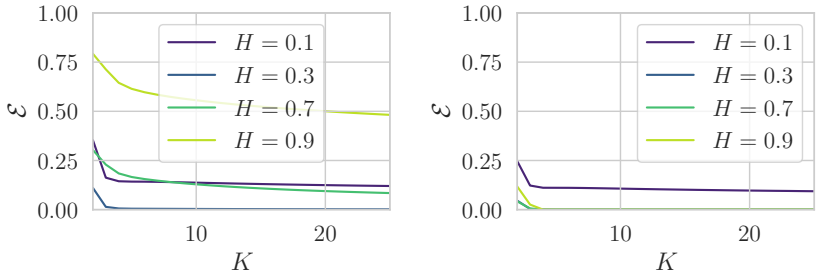
(a) Type I **(b)** Type II

Figure 3.13 Approximation error for Type I and Type II with time horizon $T = 10$, Hurst index $0 < H < 1$ and number of OU–processes $K = 5$. For Type II, decreasing $\gamma_{\min}$ below $1/T$ appears to be unneccesary, as the OU–process governed by $\gamma_{\min}$ (with the longest time-dependency) reaches equilibrium after $1/\gamma_{\min}$. For Type I, a longer time-dependency than the time horizon T is still relevant, since this is taken into account by the definition of Type I fBM. The part of this longer time-dependency that precedes the start time of the model ($t = 0$) is modelled by the initial values $(Y_1(0), \ldots, Y_K(0))$.



(a) Type I. **(b)** Type II.

Figure 3.14 MA-fBM Approximation error for varying K in function of H and with a time horizon $T = 10$, $\gamma_{\min} = 10^{-2}$ and $\gamma_{\max} = 10^2$.



(a) Type I. **(b)** Type II.

Figure 3.15 MA-fBM Approximation error for varying Hurst index and a time horizon $T = 10$, in function of K and with $\gamma_{\min} = 10^{-2}$ and $\gamma_{\max} = 10^{2}$.

either by decreasing $\gamma_{\max}$ or decreasing the integration step Δt. Lastly, since an OU–process reaches equilibrium after time $1/\gamma$, a practical lower bound in the case of Type II for $\gamma_{\min}$ is T, the length of the modelled sequences. This ensures that the memory of the MA-fBM process is modelled for at least the length of the sequence. For Type I, this is not valid since an infinite history is approximated.

3.5.3 Impact of K and the #parameters on inference time

We investigate the factors that influence the training time in Fig. 3.16, where K OU–processes are gradually included to systems with increasing number of network parameters. Note that, since our approximation is driven by 1 *Wiener process*, and the control function $u(\tilde{Z}(t), t)$ is scalar, the impact on computational load of including more processes is limited and the run-time is still dominated by the size of the neural networks. This is good news as different applications might demand different number of OU–processes.

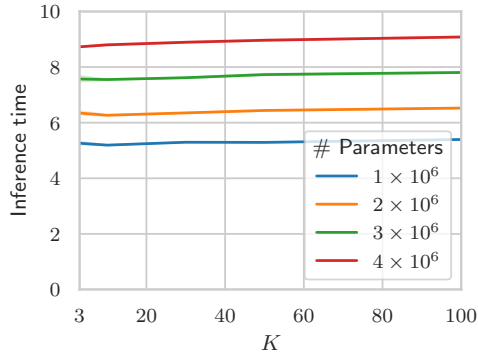


Figure 3.16 Impact of K on the inference time of a typical model with varying capacity. The scale of the neural networks has a higher impact on the speed than K.

3.6 Conclusion

In this chapter, we have proposed a new approach for performing variational inference on stochastic differential equations driven by *fractional* Brownian motion (fBM). We began by uncovering the relatively unexplored Markov representation of fBM, allowing us to approximate non-Markovian paths using a linear combination of Wiener processes. This approximation enabled us to derive evidence lower bounds through Girsanov's change of measure, yielding posterior path measures as well as likelihood estimates. We also solved for optimal coefficients for combining these processes, in closed form. Our diverse experimental study, spanning fOU bridges and Hurst index estimation, have consistently validated the effectiveness of our approach. Moreover, our novel, continuous-time architecture, powered by Markov-approximate fBM driven neural-SDEs, has demonstrated improvements in video prediction, particularly when inferring the Hurst parameter during inference.

Limitations and future work. In our experiments, we observed increased computational overhead for larger time horizons due to SDE integration, although the expansion of the number of processes incurred minimal runtime costs. We have also observed super-polynomial convergence empirically and recalled weaker

polynomial rates in the literature. Our Markov approximation still lacks a tight convergence bound. Our future work will also extend our framework to (fractional) Levy processes, which offer enhanced capabilities for modeling *heavy-tailed* noise/data distributions.

3.7 Appendix

3.7.1 State dependent diffusions

For the case, where the diffusion $\sigma(X,t)$ explicitly depends on the state variable X, our Markovian approximation results in a 'standard' white noise SDE for the augmented system. As such, it does not suffer from problems with proper definitions of stochastic integrals as compared to the original SDE driven by fBM for such cases. Hence, a straightforward Itô–interpretation of our augmented SDE is, in principle, possible. This might indicate, at first glance, that simple numerical solvers such as Euler's method could be sufficient for simulating the augmented SDE required for computing posterior expectations for the ELBO. While this point needs further theoretical investigation, preliminary simulations for for simple models with state dependent diffusions indicate that an Euler approximation (in accordance with known results for direct simulations of SDE driven by fBM (Lysy and Pillai (2013))) quickly lead to deviations from known analytical results. Hence, for state dependent diffusions, we resort to the Stratonovich interpretation of the augmented system and use corresponding higher order solvers (Kidger (2021))[2]. This approach yields excellent (pathwise) agreements with exact analytical results as we show in Sec. 3.4.1. Although the ELBO for SDE is derived from Girsanov's change of measure theorem for Itô–SDE, by the known correspondence (resulting in a change of drift functions, when diffusions are state dependent) (Gardiner et al. (1985)) between Itô and Stratonovich SDE we conclude that within this approach, optimisation of the ELBO with respect to

2. see *e.g.*, https://docs.kidger.site/diffrax/usage/how-to-choose-a-solver/#stochastic-differential-equations

model parameters will also yield the corresponding estimates for the Stratonovich interpretation.

3.7.2 The Girsanov theorem II and the KL divergence of measures

We now state the variation II of the Girsanov theorem (Øksendal (2003)) in our notation. Let $X(t) \in \mathbb{R}^n$ be an Itô process w.r.t. measure P of the form:

$$\mathrm{d}X(t) = b_\theta\left(X(t), t\right)\mathrm{d}t + \sigma_\theta\left(X(t), t\right)\mathrm{d}W(t), \tag{3.85}$$

where $0 \leq t \leq T$, $W(t) \in \mathbb{R}^m$, $b_\theta\left(X(t), t\right) \in \mathbb{R}^n$ and $\sigma_\theta\left(X(t), t\right) \in \mathbb{R}^{n\times m}$. Define a measure Q via:

$$\frac{\mathrm{d}Q}{\mathrm{d}P} = M_T := \exp\Bigg[-\int_0^T u(X(t), t)\,\mathrm{d}W(t) - \frac{1}{2}\int_0^T u^2(X(t), t)\,\mathrm{d}t\Bigg]. \tag{3.86}$$

Then

$$W'(t) := \int_0^T u(X(t), t)\,\mathrm{d}t + W(T)\,, \tag{3.87}$$

is a Brownian motion w.r.t. Q and the process $X(t)$ has the following representation in terms of $W'(t)$:

$$\mathrm{d}X(t) = \alpha_\theta\left(X(t), t\right)\mathrm{d}t + \sigma_\theta\left(X(t), t\right)\mathrm{d}W'(t), \tag{3.88}$$

where the new drift is:

$$\alpha_\theta\left(X(t), t\right) = b_\theta\left(X(t), t\right) - \sigma_\theta\left(X(t), t\right)u\left(X(t), t\right). \tag{3.89}$$

We can also rewrite the Radon–Nykodim derivative in Eq. (3.86) as

$$\frac{\mathrm{d}Q}{\mathrm{d}P} = \exp\left[\int_0^T u\left(X(t), t\right)\mathrm{d}W(t) - \frac{1}{2}\int_0^T u^2\left(X(t), t\right)\mathrm{d}t\right] \tag{3.90}$$

$$= \exp\left[\int_0^T u\left(X(t), t\right)\left(\mathrm{d}W'(t) + u\left(X(t), t\right)\mathrm{d}t\right) - \frac{1}{2}\int_0^T u\left(X(t), t\right)\mathrm{d}t\right] \tag{3.91}$$

$$= \exp\left[\int_0^T u\left(X(t), t\right)\mathrm{d}W'(t) + \frac{1}{2}\int_0^T u\left(X(t), t\right)\mathrm{d}t\right]. \tag{3.92}$$

Thus, similar to Li et al. (2020), we get the KL divergence

$$\mathbb{E}_Q\left[\ln\frac{\mathrm{d}Q}{\mathrm{d}P}\right] = \frac{1}{2}\int_0^T E_Q[u^2\left(X(t),t\right)]\,\mathrm{d}t. \tag{3.93}$$

3.7.3 Covariances

Ornstein-Uhlenbeck processes. Observe two Ornstein-Uhlenbeck (OU) processes driven by the same Wiener process:

$$\begin{cases}\mathrm{d}Y_k(t) = -\gamma_k Y_k(t)\,\mathrm{d}t + \mathrm{d}W(t)\,,\\ \mathrm{d}Y_l(t) = -\gamma_l Y_l(t)\,\mathrm{d}t + \mathrm{d}W(t)\,.\end{cases} \tag{3.94}$$

When the processes start at $-\infty$, they are stationary and in equilibrium at $t = 0$. Using Itô isometry (see Sec. 1.6.3), the covariance between the two processes at respectively time t_1 and t_2 is

$$\begin{aligned}&\mathrm{Cov}\left(Y_k(t_1), Y_l(t_2)\right)\\ &\quad= \mathbb{E}\left[\int_{-\infty}^{t_1} e^{-\gamma_k(t_1-s)}\,\mathrm{d}W(s)\int_{-\infty}^{t_2} e^{-\gamma_l(t_2-s)}\,\mathrm{d}W(s)\right] &(3.95)\\ &\quad= \int_{-\infty}^{\min(t_1,t_2)} e^{-\gamma_k(t_1-s)}e^{-\gamma_l(t_2-s)}\,\mathrm{d}s &(3.96)\\ &\quad= \frac{e^{-\gamma_t|t_1-t_2|}}{\gamma_k+\gamma_l}\,,\quad \gamma_t = \begin{cases}\gamma_k, & t_1 > t_2\,,\\ \gamma_l, & t_1 < t_2\,.\end{cases} &(3.97)\end{aligned}$$

This means that the covariance at equilibrium ($t = 0$) is

$$\mathrm{Cov}\left(Y_k(0), Y_l(0)\right) = \frac{1}{\gamma_k+\gamma_l}\,. \tag{3.98}$$

Alternatively, when the processes start at $t = 0$ with initial values $Y_k(0) = Y_l(0) = 0$, the covariance is

$$\begin{aligned}&\mathrm{Cov}\left(Y_k(t_1), Y_l(t_2)\right)\\ &\quad= \mathbb{E}\left[\int_0^{t_1} e^{-\gamma_k(t_1-s)}\,\mathrm{d}W(s)\int_0^{t_2} e^{-\gamma_l(t_2-s)}\,\mathrm{d}W(s)\right] &(3.99)\\ &\quad= \int_0^{\min(t_1,t_2)} e^{-\gamma_k(t_1-s)}e^{-\gamma_l(t_2-s)}\,\mathrm{d}s &(3.100)\\ &\quad= \frac{e^{-\gamma_t|t_1-t_2|} - e^{-\gamma_k t_1-\gamma_l t_2}}{\gamma_k+\gamma_l}\qquad \gamma_t = \begin{cases}\gamma_k, & t_1 > t_2\,,\\ \gamma_l, & t_1 < t_2\,.\end{cases} &(3.101)\end{aligned}$$

MA-fBM (Type I). Recall that (Dfn. 4)

$$\hat{B}_H^{(I)}(t) = \sum_k \omega_k (Y_k(t) - Y_k(0)) \,, \tag{3.102}$$

where $Y_k(t)$ starts at $-\infty$ and is stationary. Thus

$$\begin{aligned}
&\mathrm{Cov}\left(\hat{B}_H^{(I)}(t_1), \hat{B}_H^{(I)}(t_2)\right) \\
&\quad = \mathbb{E}\Bigg[\left(\sum_k \omega_k \left(Y_k(t_1) - Y_k(0)\right)\right) \\
&\qquad\qquad \cdot \left(\sum_k \omega_k \left(Y_k(t_2) - Y_k(0)\right)\right)\Bigg] && (3.103) \\
&\quad = \sum_{k,l} \omega_k \omega_l \mathbb{E}[\left(Y_k(t_1) - Y_k(0)\right)\left(Y_l(t_2) - Y_l(0)\right)] && (3.104) \\
&\quad = \sum_{k,l} \omega_k \omega_l \Big(\mathbb{E}\left[Y_k(t_1) Y_l(t_2)\right] - \mathbb{E}\left[Y_k(t_1) Y_l(0)\right] \\
&\qquad\qquad - \mathbb{E}\left[Y_k(0) Y_l(t_2)\right] + \mathbb{E}\left[Y_k(0) Y_l(0)\right]\Big) && (3.105) \\
&\quad = \sum_{k,l} \omega_k \omega_l \frac{1 - e^{-\gamma_k t_1} - e^{-\gamma_l t_2} + e^{-\gamma_t |t_1 - t_2|}}{\gamma_k + \gamma_l} \,, \\
&\qquad\qquad \gamma_t = \begin{cases} \gamma_k, & t_1 > t_2 \,, \\ \gamma_l, & t_1 < t_2 \,, \end{cases} && (3.106) \\
&\quad = \sum_{k,l} \omega_k \omega_l \frac{1 - e^{-\gamma_k t_1} - e^{-\gamma_l t_2} + e^{-\gamma_k |t_1 - t_2|}}{\gamma_k + \gamma_l} && (3.107)
\end{aligned}$$

using Eq. (3.97) and where the last simplifying step is possible due to the symmetry of the double summation.

MA-fBM (Type II). Recall that (Dfn. 4)

$$\hat{B}_H^{(II)}(t) = \sum_k \omega_k Y_k(t), \quad Y_k(0) = 0, \quad k = 1, \ldots, K \,. \tag{3.108}$$

Thus

$$\begin{aligned}
&\operatorname{Cov}\left(\hat{B}_H^{(II)}(t_1), \hat{B}_H^{(II)}(t_2)\right) \\
&\quad = \mathbb{E}\left[\left(\sum_k \omega_k Y_k(t_1)\right)\left(\sum_l \omega_l Y_l(t_2)\right)\right] && (3.109)\\
&\quad = \sum_{k,l} \omega_k \omega_l \mathbb{E}\left[Y_k(t_1) Y_l(t_2)\right] && (3.110)\\
&\quad = \sum_{k,l} \omega_k \omega_l \frac{e^{-\gamma_t |t_1 - t_2|} - e^{-\gamma_k t_1 - \gamma_l t_2}}{\gamma_k + \gamma_l}\,, \\
&\qquad\qquad \gamma_t = \begin{cases} \gamma_k, & t_1 > t_2\,, \\ \gamma_l, & t_1 < t_2\,, \end{cases} && (3.111)\\
&\quad = \sum_{k,l} \omega_k \omega_l \frac{e^{-\gamma_k |t_1 - t_2|} - e^{-\gamma_k t_1 - \gamma_l t_2}}{\gamma_k + \gamma_l}\,, && (3.112)
\end{aligned}$$

using Eq. (3.101) and where the last simplifying step is possible due to the symmetry of the double summation.

fBM and MA-fBM (Type I). Since (Dfn. 4)

$$\hat{B}_H^{(I)}(t) = \sum_k \omega_k \left(Y_k(t) - Y_k(0)\right), \tag{3.113}$$

where (Eq. (3.20))

$$Y_k(t) - Y_k(0) = Y_k(0)(e^{-\gamma_k t} - 1) + \int_0^t e^{-\gamma_k (t-s)}\, \mathrm{d}W(s)\,, \tag{3.114}$$

and $Y_k(t)$ starts at $-\infty$:

$$Y_k(0) = \int_{-\infty}^0 e^{\gamma_k s}\, \mathrm{d}W(s)\,, \tag{3.115}$$

we can write

$$\begin{aligned}
\hat{B}_H^{(I)}(t) = \sum_k \omega_k \Bigg(&(e^{-\gamma_k t} - 1) \int_{-\infty}^0 e^{\gamma_k s}\, \mathrm{d}W(s) \\
&+ \int_0^t e^{-\gamma_k (t-s)}\, \mathrm{d}W(s) \Bigg)\,. \qquad (3.116)
\end{aligned}$$

Using Eq. (3.7) and Itô isometry (see Sec. 1.6.3) we can express the covariance as

$$\begin{aligned}
&\operatorname{Cov}\left(B_H^{(I)}(t_1), \hat{B}_H^{(I)}(t_2)\right) \\
&= \frac{1}{\Gamma(H+1/2)} \sum_k \omega_k \mathbb{E}\Bigg[\\
&\qquad \left(\int_{-\infty}^{0} \left((t_1 - s)^{H-1/2} - (-s)^{H-1/2} \right) \mathrm{d}W(s) \right. \\
&\qquad\qquad \left. + \int_0^{t_1} (t_1 - s)^{H-1/2} \, \mathrm{d}W(s) \right) \\
&\qquad \cdot \left((e^{-\gamma_k t_2} - 1) \int_{-\infty}^{0} e^{\gamma_k s} \, \mathrm{d}W(s) \right. \\
&\qquad\qquad \left. + \int_0^{t_2} e^{-\gamma_k (t_2 - s)} \, \mathrm{d}W(s) \right) \Bigg] \qquad (3.117) \\
&= \frac{1}{\Gamma(H+1/2)} \sum_k \omega_k \Bigg(\\
&\qquad (e^{-\gamma_k t_2} - 1) \int_{-\infty}^{0} \left((t_1 - s)^{H-1/2} - (-s)^{H-1/2} \right) e^{\gamma_k s} \, \mathrm{d}s \\
&\qquad + \int_0^{\min(t_1, t_2)} (t_1 - s)^{H-1/2} e^{-\gamma_k (t_2 - s)} \, \mathrm{d}s \Bigg) \qquad (3.118) \\
&= \sum_k \omega_k \frac{1}{\gamma_k^{H+1/2}} \Big(1 - e^{-\gamma_k t_2} - e^{\gamma_k t_1} Q(H+1/2, \gamma_k t_1) \\
&\qquad + e^{-\gamma_k (t_2 - t_1)} Q\left(H+1/2, \gamma_k \max(t_1 - t_2, 0)\right) \Big) \qquad (3.119)
\end{aligned}$$

where $Q(z,x) = \frac{1}{\Gamma(z)} \int_x^\infty t^{z-1} e^{-t} \, \mathrm{d}t$ is the regularized upper incomplete gamma function. For the special case $t = t_1 = t_2$ we arrive at

$$\begin{aligned}
&\operatorname{Cov}\left(B_H^{(I)}(t), \hat{B}_H^{(I)}(t)\right) \\
&\qquad = \sum_k \omega_k \frac{2 - e^{-\gamma_k t} - e^{\gamma_k t} Q(H+1/2, \gamma_k t)}{\gamma_k^{H+1/2}} . \qquad (3.120)
\end{aligned}$$

fBM and MA-fBM (Type II). Using Dfn. 4 and Eq. (3.8) we can

write the covariance as

$$\begin{aligned}
&\operatorname{Cov}\left(B_H^{(II)}(t_1), \hat{B}_H^{(II)}(t_2)\right) \\
&= \frac{1}{\Gamma(H+1/2)} \sum_k \omega_k \mathbb{E}\Bigg[\int_0^{t_1} (t_1 - s)^{H-1/2} \,\mathrm{d}W(s) \\
&\qquad\qquad \cdot \int_0^{t_2} e^{-\gamma_k(t_2-s)} \,\mathrm{d}W(s)\Bigg] \qquad (3.121) \\
&= \frac{1}{\Gamma(H+1/2)} \sum_k \omega_k \\
&\qquad\qquad \cdot \int_0^{\min(t_1,t_2)} (t_1 - s)^{H-1/2} e^{-\gamma_k(t_2-s)} \,\mathrm{d}s \qquad (3.122) \\
&= \sum_k \omega_k \frac{e^{-\gamma_k(t_2-t_1)}}{\gamma_k^{H+1/2}} \Big(Q(H+1/2, \gamma_k \max(t_1 - t_2, 0)) \\
&\qquad\qquad - Q(H+1/2, \gamma_k t_1)\Big). \qquad (3.123)
\end{aligned}$$

For the special case $t = t_1 = t_2$ we arrive at

$$\operatorname{Cov}\left(B_H^{(II)}(t), \hat{B}_H^{(II)}(t)\right) = \sum_k \omega_k \frac{P(H+1/2, \gamma_k t)}{\gamma_k^{H+1/2}} \qquad (3.124)$$

where $P(z,x) = \frac{1}{\Gamma(z)} \int_0^x t^{z-1} e^{-t} \,\mathrm{d}t$ is the regularized lower incomplete gamma function.

Stationary fractional Ornstein–Uhlenbeck process. For the stationary fractional Ornstein-Uhlenbeck (fOU) process driven by true fBM (Type I), the covariance is given by Lysy and Pillai

(2013), Appendix A[3], where $t = |t_1 - t_2|$:

$$\operatorname{Cov}\left(X(t_1), X(t_2)\right) = \frac{H(2H-1)}{2\theta} \int_{-\infty}^{\infty} e^{-\theta|t-u|} |u|^{2H-2} \mathrm{d}u \tag{3.125}$$

$$= \frac{H(2H-1)}{2\theta} \left(\frac{e^{-\theta t}\Gamma(2H-1) + e^{\theta t}\Gamma(2H-1, \theta t)}{\theta^{2H-1}} + e^{-\theta t} \int_0^t e^{\theta u} u^{2H-2} \, \mathrm{d}u \right) \tag{3.126}$$

$$= \frac{H(2H-1)}{2\theta} \left(\Gamma(2H-1) \frac{e^{-\theta t} + e^{\theta t} Q(2H-1, \theta t)}{\theta^{2H-1}} + \int_0^t e^{-\theta(t-u)} u^{2H-2} \, \mathrm{d}u \right) \tag{3.127}$$

Studying the stationary fOU process is thus possible for $H > 1/2$, allowing us to further research the validity of our approach. Here, we will derive the necessary covariances for the stationary fOU process driven by MA-fBM (Type I):

$$\begin{cases} \mathrm{d}X(t) = -\theta X(t) \, \mathrm{d}t + \sigma \sum_k \omega_k \, \mathrm{d}Y_k(t) \\ \mathrm{d}Y_k(t) = -\gamma_k Y_k(t) \, \mathrm{d}t + \mathrm{d}W(t) \end{cases} \tag{3.128}$$

The stationary fOU process starts at equilibrium, thus we need to calculate covariances at $t = 0$ to be able to sample the necessary initial values of the process. At equilibrium (Eq. (3.98)):

$$\operatorname{Cov}\left(Y_k(0), Y_l(0)\right) = \frac{1}{\gamma_k + \gamma_l} \tag{3.129}$$

and we start by using Itô calculus (see Sec. 1.6.2) to derive other

3. Please note there is a small mistake in Lysy and Pillai (2013), Appendix A, in the working out of the indefinite integral. We start from the indefinite integral, which is correct.

useful covariances:

$$\mathbb{E}\left[\mathrm{d}Y_k(0)\,\mathrm{d}Y_l(0)\right] = \mathrm{d}t \tag{3.130}$$

$$\mathbb{E}\left[\mathrm{d}Y_k(0)Y_l(0)\right] = -\gamma_k\mathbb{E}\left[Y_k(0)Y_l(0)\right]\mathrm{d}t \tag{3.131}$$

$$\begin{aligned}\mathbb{E}\left[\mathrm{d}X(0)Y_l(0)\right] &= -\theta\mathbb{E}\left[X(0)Y_l(0)\right]\mathrm{d}t + \sigma\sum_k \omega_k\mathbb{E}\left[\mathrm{d}Y_k(0)Y_l(0)\right] && (3.132)\\ &= -\theta\mathbb{E}\left[X(0)Y_l(0)\right]\mathrm{d}t - \sigma\sum_k \omega_k\gamma_k\mathbb{E}\left[Y_k(0)Y_l(0)\right]\mathrm{d}t && (3.133)\\ &= -\theta\mathbb{E}\left[X(0)Y_l(0)\right]\mathrm{d}t - \sigma\sum_k \omega_k\frac{\gamma_k}{\gamma_k+\gamma_l}\mathrm{d}t && (3.134)\end{aligned}$$

$$\mathbb{E}\left[X(0)\,\mathrm{d}Y_l(0)\right] = -\gamma_l\mathbb{E}\left[X(0)Y_l(0)\right]\mathrm{d}t \tag{3.135}$$

$$\begin{aligned}\mathbb{E}\left[\mathrm{d}X(0)\,\mathrm{d}Y_l(0)\right] &= \sigma\sum_k \omega_k\mathbb{E}\left[\mathrm{d}Y_k(0)\,\mathrm{d}Y_l(0)\right] && (3.136)\\ &= \sigma\sum_k \omega_k\,\mathrm{d}t && (3.137)\end{aligned}$$

$$\begin{aligned}\mathbb{E}\left[X(0)\,\mathrm{d}X(0)\right] &= -\theta\mathbb{E}\left[X^2\right]\mathrm{d}t + \sigma\sum_l \omega_l\mathbb{E}\left[X(0)\,\mathrm{d}Y_l(0)\right] && (3.138)\\ &= -\theta\mathbb{E}\left[X^2\right]\mathrm{d}t - \sigma\sum_l \omega_l\gamma_l\mathbb{E}\left[X(0)Y_l(0)\right]\mathrm{d}t && (3.139)\end{aligned}$$

$$\begin{aligned}\mathbb{E}\left[\mathrm{d}X(0)\,\mathrm{d}X(0)\right] &= \sigma^2\sum_{k,l} \omega_k\omega_l\mathbb{E}\left[\mathrm{d}Y_k(0)\,\mathrm{d}Y_l(0)\right] && (3.140)\\ &= \sigma^2\sum_{k,l} \omega_k\omega_l\,\mathrm{d}t && (3.141)\end{aligned}$$

Itô yields $\mathrm{d}(X(0)Y_l(0)) = \mathrm{d}X(0)Y_l(0) + X(0)\,\mathrm{d}Y_l(0) + \mathrm{d}X(0)\,\mathrm{d}Y_l(0)$ and at equilibrium $\mathbb{E}\left[\mathrm{d}(X(0)Y_l(0))\right] = 0$ thus

$$\begin{aligned}0 &= -\theta\mathbb{E}\left[X(0)Y_l(0)\right]\mathrm{d}t - \sigma\sum_k \omega_k\frac{\gamma_k}{\gamma_k+\gamma_l}\mathrm{d}t\\ &\quad - \gamma_l\mathbb{E}\left[X(0)Y_l(0)\right]\mathrm{d}t + \sigma\sum_k \omega_k\,\mathrm{d}t && (3.142)\\ &= -(\theta+\gamma_l)\mathbb{E}\left[X(0)Y_l(0)\right] + \sigma\sum_k \omega_k\left(1-\frac{\gamma_k}{\gamma_k+\gamma_l}\right) && (3.143)\\ &= -(\theta+\gamma_l)\mathbb{E}\left[X(0)Y_l(0)\right] + \sigma\sum_k \omega_k\frac{\gamma_l}{\gamma_k+\gamma_l}\,, && (3.144)\end{aligned}$$

thus

$$\operatorname{Cov}\left(X(0)Y_l(0)\right) = \sigma \sum_k \frac{\omega_k \gamma_l}{(\theta+\gamma_l)(\gamma_k+\gamma_l)}\,. \tag{3.145}$$

Likewise, Itô yields $\mathrm{d}(X(0)^2) = 2X(0)\,\mathrm{d}X(0) + \mathrm{d}X(0)\,\mathrm{d}X(0)$ and at equilibrium $\mathbb{E}\left[\mathrm{d}\left(Y_k(0)Y_l(0)\right)\right] = 0$ thus

$$\begin{aligned} 0 &= 2\left(-\theta\mathbb{E}\left[X(0)^2\right]\mathrm{d}t - \sigma\sum_l \omega_l\gamma_l\mathbb{E}\left[X(0)Y_l(0)\right]\mathrm{d}t\right) \\ &\quad + \sigma^2\sum_{k,l}\omega_k\omega_l\,\mathrm{d}t \end{aligned} \tag{3.146}$$

$$\begin{aligned} &= -2\theta\mathbb{E}\left[X(0)^2\right] - 2\sigma^2\sum_{k,l}\omega_k\omega_l\frac{\gamma_l^2}{(\theta+\gamma_l)(\gamma_k+\gamma_l)} \\ &\quad + \sigma^2\sum_{k,l}\omega_k\omega_l\,, \end{aligned} \tag{3.147}$$

$$\operatorname{Var}\left(X(0)\right) = \sigma^2\sum_{k,l}\frac{\omega_k\omega_l}{2\theta}\left(1 - \frac{2\gamma_l^2}{(\theta+\gamma_l)(\gamma_k+\gamma_l)}\right)\,. \tag{3.148}$$

We can now derive the covariance of the process itself.

$$X(t) = X(0)e^{-\theta t} + \underbrace{\int_0^t e^{-\theta(t-s)}\,\mathrm{d}\hat{B}_H^{(I)}(s)}_{\mathrm{I}}\,, \tag{3.149}$$

$$\mathrm{I} = \sum_k \omega_k \int_0^t e^{-\theta(t-s)}\left(-\gamma_k Y_k(s)\,\mathrm{d}s + \mathrm{d}W(s)\right) \tag{3.150}$$

$$\begin{aligned} &= \sum_k \omega_k \int_0^t e^{-\theta(t-s)}\left(-\gamma_k\left(Y_k(0)e^{-\gamma_k s}\right.\right. \\ &\qquad\qquad \left.\left. + \int_0^s e^{-\gamma_k(s-r)}\,\mathrm{d}W(r)\right)\mathrm{d}s + \mathrm{d}W(s)\right) \end{aligned} \tag{3.151}$$

$$\begin{aligned} &= \sum_k \omega_k\left(-\gamma_k Y_k(0)\int_0^t e^{-\theta(t-s)-\gamma_k s}\,\mathrm{d}s\right. \\ &\qquad + \int_0^t -\gamma_k e^{-\theta(t-s)}\int_0^s e^{-\gamma_k(s-r)}\,\mathrm{d}W(r)\,\mathrm{d}s \\ &\qquad \left. + \int_0^t e^{-\theta(t-s)}\,\mathrm{d}W(s)\right)\,, \end{aligned} \tag{3.152}$$

Since the stochastic integrals in the last two terms start at 0, they are uncorrelated with $X(0)$ and we can write:

$$\mathbb{E}\left[X(0)X(t)\right] = \mathbb{E}\left[X(0)^2\right] e^{-\theta t} - \sum_k \omega_k \gamma_k \mathbb{E}\left[X(0)Y_k(0)\right] \int_0^t e^{-\theta(t-s)-\gamma_k s}\,\mathrm{d}t\,, \quad (3.153)$$

in which the covariances on the right hand side are given by Eqs. (3.145) and (3.148) and where

$$\int_0^t e^{-\theta(t-s)-\gamma_k s}\,\mathrm{d}t = \begin{cases} \dfrac{e^{-\theta t} - e^{-\gamma_k t}}{\gamma_k - \theta}, & \theta \neq \gamma_k, \\ te^{-\theta t}, & \theta = \gamma_k\,. \end{cases} \quad (3.154)$$

Finally, from the stationarity of the process follows:

$$\mathrm{Cov}\left(X(t_1), X(t_2)\right) = \mathbb{E}\left[X(0)X(t = |t_2 - t_1|)\right]\,. \quad (3.155)$$

3.7.4 An alternative approach for Markov approximation of fractional Brownian motion

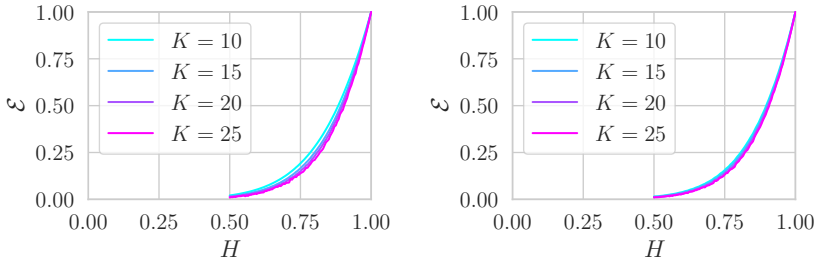
For $H > 1/2$, we can define the Markov representation in terms of U_k processes instead of Y_k processes (see Prop. 1), such that (compare to Eq. (3.31)):

$$B_H(t) = \int_0^\infty \left(U_\gamma(t) - U_\gamma(0)\right)\nu(\gamma)\,\mathrm{d}\gamma, \quad H > 1/2\,, \quad (3.156)$$

where

$$\mathrm{d}U_\gamma(t) = \left(-\gamma U_\gamma(t) + Y_\gamma(t)\right)\mathrm{d}t\,. \quad (3.157)$$

We chose to work only with Y_k processes in Chapter 3 for simplicity. Otherwise, one would need to switch between Y_k and U_k when crossing $H = 0.5$, and augmenting the state would add $2K$ extra processes instead of K. One could say we have approximated U_k by finite differencing neighboring Y_k processes. Thus, our optimal ω weights approach *implicitly* handles this finite differencing. This begs the question, would the MA-fBM approximation improve by using the Z processes directly? We investigate this, specifically for Type I.



(a) Type I. **(b)** Type I, using the U_k processes.

Figure 3.17 MA-fBM Approximation error for varying K in function of H and with a time horizon $T = 10$, $\gamma_{\min} = 10^{-2}$ and $\gamma_{\max} = 10^2$.

Starting from the $\boldsymbol{A}^{(I)}$ and $\boldsymbol{b}^{(I)}$ from Chapter 3, we can construct analogously $\boldsymbol{A}^{(Z)}$ and $\boldsymbol{b}^{(Z)}$:

$$\boldsymbol{A}^{(Z)}_{(k,l)} = \frac{\partial^2}{\gamma_k \gamma_l} \boldsymbol{A}^{(I)}_{(k,l)}, \tag{3.158}$$

$$\boldsymbol{b}^{(Z)}_{(k)} = \frac{\partial}{\gamma_k} \boldsymbol{b}^{(I)}_{(k)}. \tag{3.159}$$

This leads to the following error rates in function of H presented in Fig. 3.17. The error slightly improves for lower values of K, which is to be expected because the original approach *implicitly* approximates the U_k processes by finite differencing neighboring Y_k processes, thus benefits from a higher number of processes K. The original MA-fBM approximates $\partial_\gamma Y_\gamma(t)$, thus it is not surprising that the error is lower using the U processes. However the difference is only small, and the error still goes up to 1 towards $H = 1$. The practical benefit of working out a Markov approximation using both Y_k and U_k processes thus seems limited.

3.7.5 Details on Model Architectures & Hyperparameters

Time dependent Hurst index. We directly compare our method with the data and estimate found in the published

codebase of Tong et al. (2022)[4]. We choose $K = 5$ and $\gamma_k = (\frac{1}{20}, \dots, 20)$ and use "Type II" (to match the data and noise type in Tong et al. (2022)). The optimal definitions for ω_k, with time horizon $T = 2$ are used. The control function is a neural network with two hidden layers of each 1000 neurons, with $\tanh$ activation function. Its input is represented as $[\sin t, \cos t, \sin 2t, \cos 2t, \dots, \sin 5t, \cos 5t, X(t), Y_1(t), \dots, Y_K(t)]$. The model is trained for 1000 training steps with a batch size of 4. We use the Adam (Kingma and Ba (2015)) optimizer with a learning rate 3×10^{-3}, scheduled with cosine decay to 3×10^{-4} by the end of training. We use the *Stratonovich-Milstein* SDE solver (Kidger (2021)). The integration step is 0.005 and observation noise $\sigma = 0.025$ (both identical to Tong et al. (2022)).

Latent video model.**Stochastic moving MNIST**. For the MA-fBM model, $K = 5$ and $\gamma_k = (\frac{1}{20}, \dots, 20)$. We use "Type I" and the corresponding definitions for ω_k, with a time horizon $T = 2.4$. For the BM model, $K = 1$, $\gamma_1 = 0$ and $\omega = 1$, which naturally corresponds to white Brownian motion. The number of latent dimensions $D = 6$.

The encoder model consists of four blocks, containing a convolution layer, maxpool, groupnorm and SiLU activation. Each block reduces spatial dimension by 2, and the number of features in each block is $(64, 128, 256, 256)$. The last output is flattened and is the input of a dense layer, with h as output with 64 features.

The median over the time axis of h is fed into a two layers neural network to produce the static content vector w. Since the median is permutation invariant, w contains no dynamic information, only static information. w also has 64 features.

The context model consists of two subsequent $1-D$ convolutions in the temporal dimension. Thus, information is shared over different frames, which is necessary for inference. The output of this model is g.

4. https://github.com/anh-tong/fractional_neural_sde/blob/7565a2/fractional_neural_sde/example.ipynb

To start the SDE integration, we need an initial state that is conditioned on the data. We define a three layer neural network model that receives (g_1, h_1, h_2, h_3) and outputs the parameters of the posterior distribution q_{x_1} of the initial state of the SDE. x_1 is sampled from q_{x_1}, which we model as a diagonal Normal distribution. The parameters of a prior model p_{x_1} are also optimized, and the Kullback-Leibler divergence $D_{\mathrm{KL}}(p_{x_1}, q_{x_1})$ is added to the loss function. This approach for training neural SDEs is similar to others in literature (Li et al. (2020)).

The prior drift $b_\theta(X, t)$ and the control function $u(Z(t), t)$ have the same architecture, a neural network with two hidden layers of each 200 neurons, with $\tanh$ activation functions. The shared diffusion $\sigma_\theta(X, t)$ is implemented so that the noise is commutative to allow Milstein solvers (Li et al. (2020) and Kidger et al. (2021b)), *i.e.*, $\sigma_\theta(X, t)$ is diagonal and the i-th component on the diagonal only receives $X_i(t)$ as input, where we have defined D separate neural networks for each component. Each neural network has two layers with 200 neurons and $\tanh$ activations.

b_θ and σ_θ receive $X(t)$ as input. The control function a concatenated vector of $(X(t), Y_1(t), \ldots, Y_K(t), g(t))$. $g(t)$ is a linear interpolation of g at time t. This enables the control function to use appropriate information to be able to steer the process correctly.

The resulting states x after integration of the SDE are fed, together with the static content vector w in the decoder model. The decoder model has first a dense layer. The outputs of this first layer are shaped in a 4×4 spatial grad. Subsequently, four blocks with a convolution layer, groupnorm, a spatial nearest neighbour upsampling layer and a SiLU activation. Thus, the model reaches the correct resolution of 64×64. Two additional convolution layers with SiLU activation and a final sigmoid activation complete the decoder model.

We train on sequences of 25 frames, with a time length of 2.4 (0.1 per frame). The frames have resolution 64×64 and 1 color channel. Each model was trained for 187500 training steps with a batch size of 32. We use the Adam (Kingma and Ba (2015)) optimizer with fixed learning rate 3×10^{-4}. We use the *Stratonovich–*

Milstein SDE solver (Kidger (2021)) with an integration step of 0.033 (3 integration steps per data frame). Models were trained on a single NVIDIA GeForce RTX 4090, which takes around 39 hours for one model.

Double pendulum. We use the train-test split from the original dataset (Asseman et al. (2018)). The videos are recorded with a high speed camera, we used every 10th frame to increase the challenge of the dataset. We resized the frames to a resolution of 128×128 resolution. Therefore, we added one block to the encoder and decoder model to achieve this resolution, compared to the model for Stochastic Moving MNIST (SM-MNIST). We did not use the static content vector w, since there is minimal static information in this dataset, and used $D = 8$ latent dimensions. The models were trained for 124916 training steps, and we trained around 32 hours for one model. Beyond these outlined differences, all other details are equal to the SM-MNIST model.

3.7.6 Video Models

We conducted experiments on two video datasets: Stochastic Moving MNIST (SM-MNIST) (Denton and Fergus (2018)) and the real video dataset of a chaotic double pendulum (Asseman et al. (2018)).

SM-MNIST and the double pendulum dataset contain different forms of nuisances and present different challenges to our stochastic model. First, SM-MNIST digits move with a constant velocity along a trajectory until they hit at wall at which point they bounce off with a random speed and direction. This sudden event intersperses the deterministic motion with moments of uncertainty, *i.e.*, each time a digit hits a wall. This is the reason why a stochastic model fits better than an ODE and unlike BM, our noise can model the smooth and correlated trajectory simply by raising the Hurst index.

On the other hand, the double pendulum dataset is actually governed by a set of coupled ordinary differential equations. However, despite being a simple physical system, it exhibits a rich dynamic behavior with a strong sensitivity to initial conditions

and noises in the environment (motion of the air in the room, sound vibrations, vibration of the table due to coupling with the pendulum etc.). Combined with the chaotic nature of the system, this creates a major challenge for any model based upon smooth ODEs. Our model on the other hand heavy lifts this difficulty onto the (fractional) stochastic noise, leading to a more appropriate model. As shown in Tab. 3.2, our model outperforms the BM baseline also in this dataset.

Figs. 3.18 and 3.19 show the posterior reconstructions of models trained on the SM-MNIST and the double pendulum dataset respectively. Fig. 3.20 shows stochastic video prediction samples of SM-MNIST.

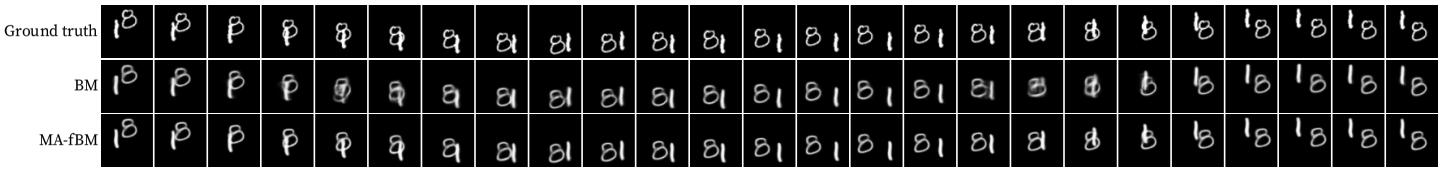


Figure 3.18 Posterior reconstructions of a model driven by BM and a model driven by MA-fBM, conditioned on the same data ('Ground truth').

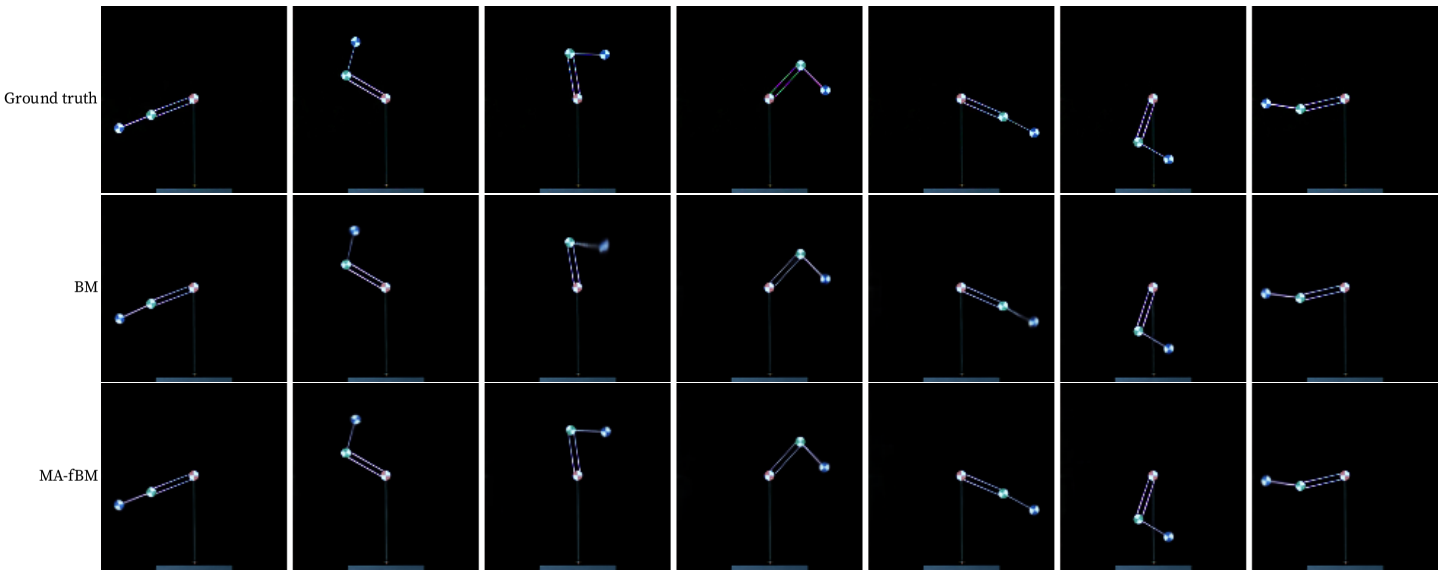


Figure 3.19 Posterior reconstructions of a model driven by BM and a model driven by MA-fBM, trained on the double pendulum dataset. Both are conditioned on the same data ('Ground truth'). We show 7 evenly spaced frames of the total 20 frames.

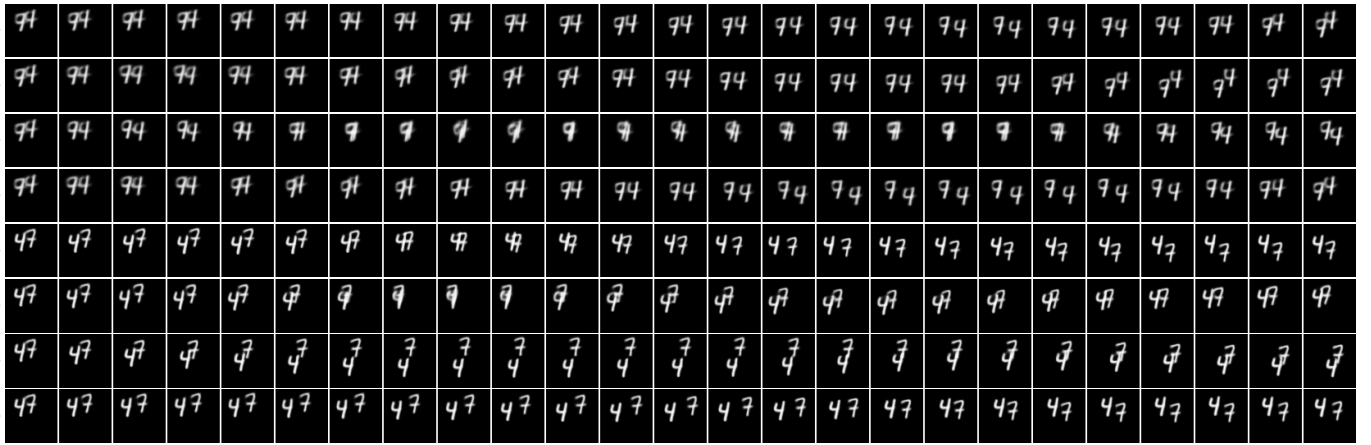


Figure 3.20 Stochastic predictions using the trained prior of a model driven by BM and a model driven by MA-fBM, where the initial state is conditioned on the same data. Four samples are shown for each model. The MA-fBM samples show more diverse movements, thus better capturing the dynamics in the data. The BM samples are more similar, indicating a less powerful prior was learned.

4

4

Efficient Training of Neural SDEs Using Stochastic Optimal Control

4

Efficient Training of Neural SDEs Using Stochastic Optimal Control

As is clear from the previous chapter, learning SDEs from data is computationally challenging. Due to the iterative nature of maximizing the evidence lower bound (ELBO), which requires solving the SDE for each iteration, the training of neural SDEs is generally slow and unstable. This chapter is a step towards the final goals of this thesis (see Sec. 1.8), how do we more efficiently learn SDEs from data?

We present a hierarchical, control theory inspired method for variational inference (VI) for neural stochastic differential equations (SDEs). In this chapter, we propose to decompose the control term into linear and residual non-linear components and derive an optimal control term for linear SDEs, using stochastic optimal control. Modeling the non-linear component by a neural network, we show how to efficiently train neural SDEs without sacrificing their expressive power. Since the linear part of the control term is optimal and does not need to be learned, the training is initialized at a lower cost and we observe faster convergence. The work in this chapter is based on Daems et al. (2025).

4.1 Introduction

Continuous-time models of dynamical systems provide a powerful framework for capturing the intricate variations in real-world phenomena. Among these, stochastic differential equations (SDEs) extend the capabilities of deterministic models by abstracting away unaccounted factors into instantaneous noise.

SDEs naturally model various processes, including the motion of small particles (*e.g.*, molecules) and financial market dynamics. When combined with neural networks (Tzen and Raginsky (2019) and Li et al. (2020)), they become expressive tools for learning from irregular time-series observations.

Despite their promise, path-wise inference for neural SDEs remains a notorious challenge due to the complexity in fitting the non-Gaussian posterior distributions. Variational inference (VI) has become a prevalent tool with significant success in scaling inference methods (Daems, Opper, et al. (2024)). Yet, computational challenges persist.

Existing works attempt to address these issues in VI for neural SDEs in various ways. S. W. Park et al. (2021) introduced finite-dimensional matching for efficient path comparison to train neural SDEs. Kidger et al. (2021b) adopted a generative-adversarial approach to train these models. Course and Nair (2024) proposed an amortized method for fast VI in latent neural SDEs, scaling efficiently with data size using a linear posterior. However, resorting to linear posteriors is a severe limitation in practice.

Inspired by *optimal control theory*, we propose a novel approach to efficiently perform VI in neural SDEs. Our key idea is to represent the prior as the combination of a linear model and a residual non-linear model. We leverage this decomposition to split the control function—used to compute the variational posterior—into two components. The first linear component is tractable and admits a closed-form solution, making it computationally efficient but less expressive. The residual non-linear component, modeled by a neural network, captures higher-order effects at the cost of iterative optimization. We combine the strengths of these two approaches. First, we compute the linear part in closed form, which serves as an efficient initialization for the neural network modeling the non-linear residual. This hierarchical design allows us to achieve faster and more stable inference compared to existing approaches that directly model the full control term (Li et al. (2020) and Daems, Opper, et al. (2024)).

In summary, our contributions are:

1. We derive the optimal control function solution for VI of a linear SDE driven by Brownian motion (BM), or by Markov–approximated fractional BM.

2. We propose a neural SDE model with a linear and a residual non-linear (neural network) part, both for the prior SDE and the control terms, for which the linear part is optimal and does not need to be optimized or learned.

3. We show that our proposed model trains faster and more stable than a standard non-linear network model on a financial data.

4.2 Variational Inference of Stochastic Differential Equations

Definition 8 (SDE driven by BM (BMSDE))**.** *A common generative model for stochastic dynamical systems considers a set of observational data* $\mathcal{D} = \{O_1, \ldots, O_M\}$*, where the* O_i *are generated (conditionally) independent at random at discrete times* t_i *with a likelihood* $p_\theta(O_i \mid X(t_i))$*. The prior information about the unobserved path* $\{X(t); t \in [0,\ T]\}$ *of the latent process* $X(t) \in \mathbb{R}^D$ *is given by the assumption that* $X(t)$ *fulfils the SDE:*

$$\mathrm{d}X(t) = b_\theta(X(t), t)\,\mathrm{d}t + \sigma_\theta(X(t), t)\,\mathrm{d}W(t)\,. \qquad \text{(PRIOR-SDE)}$$

The drift function $b_\theta(X(t), t) \in \mathbb{R}^D$ *models the deterministic part of the change* $\mathrm{d}X(t)$ *of the state variable* $X(t)$ *during the infinitesimal time interval* $\mathrm{d}t$*, whereas the diffusion matrix* $\sigma_\theta(X(t), t) \in \mathbb{R}^{D\times B}$ *encodes the strength of the added Gaussian white noise process, where* $\mathrm{d}W(t) \sim \mathcal{N}(0, \mathrm{d}t) \in \mathbb{R}^B$ *is the infinitesimal increment of a vector of independent Wiener processes during* $\mathrm{d}t$*.*

Definition 9 (Posterior SDE)**.** *The paths of the* PRIOR-*SDE can be steered by adding a control term* $u(X(t), t)$ *that depends on all variables to be optimised and the observations, to the drift resulting in the*

variational posterior Opper 2019; Li et al. 2020:

$$\begin{aligned} \mathrm{d}\tilde{X}(t) = b_\theta\left(\tilde{X}(t), t\right)\mathrm{d}t + \sigma_\theta\left(\tilde{X}(t), t\right) u\left(\tilde{X}(t), t\right)\mathrm{d}t \\ + \sigma_\theta\left(\tilde{X}(t), t\right)\mathrm{d}W(t)\,. \end{aligned} \tag{4.1}$$

In what follows, we will assume a parametric form for the control function $u(\tilde{X}(t), t) \equiv u_\phi(\tilde{X}(t), t)$ and will recall a scheme for inferring the *variational parameters* (θ, ϕ), *i.e.*, variational inference.

Proposition 7 (Variational Inference for BMSDE (Opper (2019) and Li et al. (2020))). *The variational parameters ϕ are optimised by minimising the KL–divergence between the posterior and the prior, where the corresponding evidence lower bound (ELBO) is maximized to find the most likely parameters θ:*

$$\begin{aligned} &\sum_{i=1}^{M} \log p\left(O_i \mid \theta\right) \geq \\ &\quad \mathbb{E}_{\tilde{X}}\left[\sum_{i=1}^{M} \log p_\theta\left(O_i \mid \tilde{X}(t_i)\right) - \int_0^T \frac{1}{2}\left\|u_\phi\left(\tilde{X}(t), t\right)\right\|^2 \mathrm{d}t\right], \end{aligned} \tag{4.2}$$

where the observations $\{O_i\}$ are included by likelihoods $p_\theta\left(O_i \mid \tilde{X}(t_i)\right)$ and the expectation is taken over random paths of the approximate posterior process defined by (Eq. (4.1)).

4.3 Optimal Control for Variational Inference for SDEs

Our approach uses optimal control to decouple the possible linear and non-linear effects in the drift. While the linear part is easier to solve in closed-form, the non-linear terms will account for the complex variations in real data. In the sequel, we describe these two parts, respectively, finally leveraging the strengths of both.

4.3.1 Optimal posterior control term for a linear prior SDE

The control term $u(x,t) := u_\phi(x,t)$ can be obtained explicitly from the solution of the transformed *Hamilton–Jacobi–Bellman* equation (HJBE) (Kappen (2005), Archambeau and Opper (2011), and Maoutsa and Opper (2022)):

$$u(x,t) = \sigma_\theta(x,t)^\top \nabla_x \log \mathbb{E}_{\text{prior}}\left[\prod_{i:t_i>t} p_\theta\left(O_i \mid X(t_i)\right) | X(t) = x\right]. \quad (4.3)$$

In general, such expectations over the paths of the PRIOR-SDE involve solving second order partial differential equations in the $D+1$ variables and are intractable in closed form. However, in what follows, we will show how to compute it exactly when both the prior process $X(t)$ and the observation likelihood are Gaussian. This requires the drift $b_\theta(x,t)$ to be a linear function in x, and the diffusion $\sigma_\theta(t)$ independent of x.

Proposition 8. *For a process $X(t)$ with linear drift and state-independant diffusion $\sigma(t)$ where we have M observations $\boldsymbol{O} = [O(T_1), \ldots, O(T_M)]$ after time t, the optimal control term takes the form:*

$$u(x,t) = \sigma(t)^\top \nabla_x \log \mathcal{N}(\boldsymbol{O}; \boldsymbol{m}_x, \boldsymbol{C} + \boldsymbol{\Sigma}_0) \quad (4.4)$$

$$= \sigma(t)^\top \left(\nabla_x \boldsymbol{m}_x\right)^\top \left(\boldsymbol{C} + \boldsymbol{\Sigma}_0\right)^{-1} \left(\boldsymbol{O} - \boldsymbol{m}_x\right), \quad (4.5)$$

where $p(\boldsymbol{X}(\boldsymbol{T})|x) = \mathcal{N}(\boldsymbol{m}_x, \boldsymbol{C})$ is the joint Gaussian distribution of the solutions of the prior SDE $\boldsymbol{X}(\boldsymbol{T}) = [X(T_1), \ldots, X(T_M)]$ conditioned on $X(t) = x$ having mean vector by $\boldsymbol{m}_x$ and covariance matrix by $\boldsymbol{C}$. The observation likelihood is assumed to be of the form $\mathcal{N}(\boldsymbol{O}; 0, \boldsymbol{\Sigma}_0)$.

Sketch of the proof. Under these assumptions, the expectation in Eq. (4.3) for $X(t)$ becomes an M dimensional Gaussian integral of the form:

$$\mathbb{E}_{\text{prior}}[\ldots] = \int p(\boldsymbol{X}(\boldsymbol{T})|x) p(\boldsymbol{O}|\boldsymbol{X}(\boldsymbol{T}))\, \mathrm{d}\boldsymbol{X}(\boldsymbol{T}) \quad (4.6)$$

$$= \mathcal{N}(\boldsymbol{O}; \boldsymbol{m}_x, \boldsymbol{C} + \boldsymbol{\Sigma}_0). \quad (4.7)$$

□

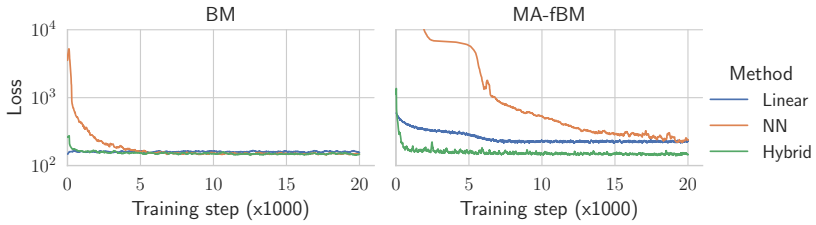


Figure 4.1 We show the loss (negative ELBO) curves of the models driven by BM (left) and MA-fBM (right). For both experiments, our proposed hybrid model (green) starts training with a loss that is multiple orders of magnitude smaller and converges much faster than a standard non-linear neural network model (blue). Our hybrid model (green) also performs better than the strictly linear model (orange), especially for the MA-fBM experiment.

Specifically, for a one–dimensional process $X(t) \in \mathbb{R}$ parameterized by $\lambda \in \mathbb{R}_+$, $\eta \in \mathbb{R}$ and constant diffusion $\varsigma \in \mathbb{R}_+$:

$$\mathrm{d}X(t) = (-\lambda X(t) + \eta)\,\mathrm{d}t + \varsigma\,\mathrm{d}B(t)\,, \tag{4.8}$$

we can write the solution at some later time T conditioned on the state x at current time t as (Särkkä and Solin (2019)):

$$X(T) = xe^{-\lambda(T-t)} + \int_t^T e^{-\lambda(T-s)}\eta\,\mathrm{d}s + \int_t^T e^{-\lambda(T-s)}\varsigma\,\mathrm{d}B(s) \tag{4.9}$$

which leads to the mean and covariance:

$$\begin{aligned}
\boldsymbol{m}_{x(i)} &= \mathbb{E}\left[X(T_i)|X(t) = x\right] && (4.10)\\
&= xe^{-\lambda(T_i-t)} + \frac{\eta}{\lambda}\left(1 - e^{-\lambda(T_i-t)}\right)\,, && (4.11)\\
\boldsymbol{C}_{(i,j)} &= \mathrm{Cov}\left(X(T_i), X(T_j)\right) && (4.12)\\
&= \varsigma^2 \int_t^{\min(T_i,T_j)} e^{-\lambda(T_i-s)}e^{-\lambda(T_j-s)}\,\mathrm{d}s && (4.13)\\
&= \varsigma^2 \frac{e^{-\lambda|T_i-T_j|} - e^{-\lambda(T_i+T_j-2t)}}{2\lambda}\,. && (4.14)
\end{aligned}$$

4.3.2 Incorporating non-linear residual terms

We propose to define a prior SDE composed of linear and non-linear drifts as

$$\begin{aligned} \mathrm{d}X(t) = &\left(-\lambda_\theta X(t) + \eta_\theta + b_\theta\left(X(t)\right)\right)\mathrm{d}t \\ &+ \left(\varsigma_\theta + \sigma_\theta(X(t))\right)\mathrm{d}W(t)\,, \end{aligned} \tag{4.15}$$

where $b_\theta(\cdot)$ and $\sigma_\theta(\cdot)$ are non-linear functions (e.g. neural networks) and θ indicates learnable parameters. Equivalently, the control term is defined as

$$u(\tilde{X}(t), t) \equiv u_c(\tilde{X}(t), t) + u_\phi(\tilde{X}(t), t) \tag{4.16}$$

where $u_c(\cdot)$ is the analytical optimal control solution (Eq. (4.5)) that depends on λ_θ, η_θ and ς_θ (Eqs. (4.10) and (4.14)) and u_ϕ is a residual non-linear control term, modeled e.g. by a neural network. For a purely linear model, without the non-linear components, the ELBO would be optimal by definition. However, such a model would not be expressive, *i.e.*, not be able to capture realistic, non-linear data. The core idea of our work is to combine the linear terms with the residual non-linear terms $b_\theta(\cdot)$, $\sigma_\theta(\cdot)$ and $u_\phi(\cdot)$ such that the training of the model is more robust and fast, benefiting from the best of both worlds.

Furthermore, a crucial advantage of the linear model is the use of the tractable log-likelihood function $\log \mathcal{N}(\boldsymbol{O}; \boldsymbol{m}_x, \boldsymbol{C} + \boldsymbol{\Sigma}_0)$ to directly find λ_θ, η_θ and ς_θ, without having to solve computationally costly SDEs. This allows initialization of training where the linear component is already optimal.

4.3.3 Extension to fractional Brownian motion

We can extend this method to variational inference for SDEs driven by fBM, as presented in Chapter 3. Remember that Markov approximated fBM (MA-fBM) is defined as an augmentation of the state with K Ornstein-Uhlenbeck processes $Y_k(t)$ (see Prop. 3). Since this augmented process is driven by

one single Wiener process, the control term can be written as (compare to Eq. (4.5)):

$$u(x,y,t) = \left(\sigma(t)\bar{\omega}\nabla_x \boldsymbol{m}_{x,y} + [1,\dots,1]^\top \nabla_y \boldsymbol{m}_{x,y}\right)^\top \cdot (\boldsymbol{C} + \boldsymbol{\Sigma}_0)^{-1} (\boldsymbol{O} - \boldsymbol{m}_{x,y}) \,. \quad (4.17)$$

Note that this result is limited to the one-dimensional case, *i.e.*, $X(t) \in \mathbb{R}$.

Note that, because everything is driven by the same Wiener process, we only need the prior means $\boldsymbol{m}_{x,y}$ and covariances $\boldsymbol{C}$ at the measurement times $\boldsymbol{T}$ for $X(t)$, not for the $Y_k(t)$ processes. We find them by writing the solution at a later time T conditioned on the state $(x, y_1, \dots, y_K)$ at current time t. Note that the solutions are valid for both Type I and Type II of MA-fBM. Since they are conditioned on the current state $(x, y_1, \dots, y_K)$ at time t, the initial conditions at $t = 0$, which are the defining difference between Type I and Type II, are irrelevant. For ease of notation we define the augmented state $z = (x, y_1, \dots, y_K)$, denoting as usual $z = z(t)$, $x = x(t)$ and $y_k = y_k(t)$. Again, for a linear prior SDE given by

$$\mathrm{d}X(t) = (-\lambda X(t) + \eta)\,\mathrm{d}t + \varsigma\,\mathrm{d}\hat{B}_H(t)\,, \quad (4.18)$$

we write the solution at some later time T conditioned on the current state:

$$X(T) = x e^{-\lambda(T-t)} + \underbrace{\int_t^T e^{-\lambda(T-s)}\eta\,\mathrm{d}s}_{\mathrm{I}} + \underbrace{\varsigma\int_t^T e^{-\lambda(T-s)}\,\mathrm{d}\hat{B}_H(s)}_{\mathrm{II}}\,, \quad (4.19)$$

$$\mathrm{I} = \frac{\eta}{\lambda}\left(1 - e^{-\lambda(T-t)}\right), \quad (4.20)$$

$$\mathrm{II} = \varsigma\sum_k \omega_k \int_t^T e^{-\lambda(T-s)}\left(-\gamma_k Y_k(s)\,\mathrm{d}s + \mathrm{d}W(s)\right) \quad (4.21)$$

$$= \varsigma\sum_k \omega_k \int_t^T e^{-\lambda(T-s)}\Big(-\gamma_k(y_k e^{-\gamma_k(s-t)}$$

$$+ \int_t^s e^{-\gamma_k(s-r)} \, \mathrm{d}W(r)) \, \mathrm{d}s + \mathrm{d}W(s) \Bigg) \tag{4.22}$$

$$\begin{aligned} = \varsigma \sum_k \omega_k \Bigg(& - \gamma_k y_k \int_t^T e^{-\lambda(T-s)-\gamma_k(s-t)} \, \mathrm{d}s \\ & + \int_t^T -\gamma_k e^{-\lambda(T-s)} \int_t^s e^{-\gamma_k(s-r)} \, \mathrm{d}W(r) \, \mathrm{d}s \\ & + \int_t^T e^{-\lambda(T-s)} \, \mathrm{d}W(s) \Bigg) \end{aligned} \tag{4.23}$$

$$\begin{aligned} = \varsigma \sum_k \omega_k \Bigg(& \gamma_k y_k \frac{e^{-\lambda(T-t)} - e^{-\gamma_k(T-t)}}{\lambda - \gamma_k} \\ & + \int_t^T \int_r^T -\gamma_k e^{-\lambda(T-s)-\gamma_k(s-r)} \, \mathrm{d}s \, \mathrm{d}W(r) \\ & + \int_t^T e^{-\lambda(T-s)} \, \mathrm{d}W(s) \Bigg) \end{aligned} \tag{4.24}$$

$$\begin{aligned} = \varsigma \sum_k \omega_k \Bigg(& \gamma_k y_k \frac{e^{-\lambda(T-t)} - e^{-\gamma_k(T-t)}}{\lambda - \gamma_k} \\ & + \int_t^T \int_s^T -\gamma_k e^{-\lambda(T-r)-\gamma_k(r-s)} \, \mathrm{d}r + e^{-\lambda(T-s)} \, \mathrm{d}W(s) \Bigg) \end{aligned} \tag{4.25}$$

$$\begin{aligned} = \varsigma \sum_k \omega_k \Bigg(& \gamma_k y_k \frac{e^{-\lambda(T-t)} - e^{-\gamma_k(T-t)}}{\lambda - \gamma_k} \\ & + \int_t^T \gamma_k \frac{e^{-\lambda(T-s)} - e^{-\gamma_k(T-s)}}{\lambda - \gamma_k} + e^{-\lambda(T-s)} \, \mathrm{d}W(s) \Bigg) . \end{aligned} \tag{4.26}$$

Putting everything together leads to:

$$\begin{aligned} X(T) = & x e^{-\lambda(T-t)} + \frac{\eta}{\lambda} \left(1 - e^{-\lambda(T-t)} \right) \\ & + \varsigma \sum_k \omega_k \Bigg(y_k \gamma_k \frac{e^{-\lambda(T-t)} - e^{-\gamma_k(T-t)}}{\lambda - \gamma_k} \\ & + \int_t^T \frac{\lambda e^{-\lambda(T-s)} - \gamma_k e^{-\gamma_k(T-s)}}{\lambda - \gamma_k} \, \mathrm{d}W(s) \Bigg) . \end{aligned} \tag{4.27}$$

Thus the distribution of future solutions $\boldsymbol{X}(\boldsymbol{T})$ is Gaussian with

mean and covariance:

$$\boldsymbol{m}_{x,y_{(i)}} = xe^{-\lambda(T_i-t)} + \frac{\eta}{\lambda}\left(1 - e^{-\lambda(T_i-t)}\right) + \varsigma \sum_k \omega_k y_k \gamma_k \frac{e^{-\lambda(T_i-t)} - e^{-\gamma_k(T_i-t)}}{\lambda - \gamma_k}, \tag{4.28}$$

$$\boldsymbol{C}_{(i,j)} = \varsigma^2 \sum_{k,l} \omega_k \omega_l \frac{\zeta(\gamma_k, \gamma_l) - \zeta(\gamma_k, \lambda) - \zeta(\lambda, \gamma_l) + \zeta(\lambda, \lambda)}{(\lambda - \gamma_k)(\lambda - \gamma_l)}, \tag{4.29}$$

where

$$\zeta(\alpha, \beta) = \frac{\alpha\beta}{\alpha + \beta}(e^{-\alpha(T_i - \min(T_i,T_j)) - \beta(T_j - \min(T_i,T_j))} - e^{-\alpha(T_i-t)-\beta(T_j-t)}). \tag{4.30}$$

In the derivations above, we have silently assumed $\lambda \neq \gamma_k$ for any γ_k. If $\lambda = \gamma_k$, one should take the limit of $\lambda \to \gamma_k$. In our implementation, we linearize the $u(x, y, t)$ function around the γ_k values.

4.4 Experiments

We apply our method on the first 500 days of the 3–Month US Treasury Bills[1]. We compare the training of our proposed hybrid model with the non-linear residual part to the training of a standard non-linear model and a strictly linear model. We also apply our method to the SDEs driven by MA-fBM, presented in Sec. 4.3.2. The non-linear prior drift $b_\theta(\cdot)$, diffusion $\sigma_\theta(\cdot)$ and control term $u_\phi(\cdot)$ are neural networks. The observations are encoded by an additional neural network into $u_\phi(\cdot)$, as is typically done in VI for SDEs (Li et al. (2020) and Daems, Opper, et al. (2024)). All neural networks have three layers, 128 hidden neurons and the $\tanh$ activation function. The observations noise $\boldsymbol{\Sigma}_0 = 0.1^2\mathbf{I}$. For the MA-fBM experiment we set a Hurst index of 0.65 which is a reasonable choice for this data (Lysy and Pillai (2013)). Fig. 4.1 shows the loss (negative ELBO) curves of the three models, both for the models driven by BM and MA-fBM.

1. https://fred.stlouisfed.org/series/DTB3

4.5 Conclusion

We present an optimal control inspired method for efficient variational inference for (neural) SDEs. Under practically reasonable assumptions, we explicitly formulate the control term with linear and residual non-linear components and derive a closed-form control term for the linear part using stochastic optimal control. This model is shown to converge faster than a standard non-linear SDE, both for SDEs driven by BM and Markov-approximate fBM.

Future work and limitations. Our work applies only to 1-d SDEs, future work will involve a multi-dimensional formulation. We also plan to cover latent SDEs (Course and Nair (2024)).

5

5

Conclusion and Future Work

5

Conclusion and Future Work

This dissertation made several contributions to the field of deterministic and stochastic dynamical models learned from video. We conclude our work by discussing and summarizing the main contributions and listing future research directions.

5.1 Conclusion

We introduced a novel approach in Chapter 2 using keypoints to learn Lagrangian dynamics directly from image data. By employing learned keypoint representations as positional state vectors, we established a framework for learning constrained Lagrangian dynamics from visual inputs. Our work advances beyond previous approaches that were limited to simplistic visual scenarios, as we deliberately utilized more complex renderings from dm_control that incorporate lighting effects, shadows, reflections, and backgrounds. Our proposed KeyCLD model demonstrated capability in making long-term predictions and learning accurate energy models suitable for simple energy shaping control. Comparative analysis with unconstrained Lagrangian dynamics models (KeyLD) and general second-order neural ODEs (KeyODE2) confirmed that when constraint functions are known, the constrained Lagrangian formulation offers significant benefits for long-term prediction accuracy. This research helps bridge the divide between traditional control engineering approaches and computer science methodologies. Rather than choosing between known equations of motion or completely general methods with no prior knowledge, we explored the valuable middle ground by

incorporating strong physics priors such as Lagrangian mechanics while modeling components like input matrices and potential energy with flexible neural networks.

We developed a new approach in Chapter 3 for variational inference (VI) on stochastic differential equations (SDEs) driven by fractional Brownian motion (fBM). By leveraging the relatively unexplored Markov representation of fBM, we approximated non-Markovian paths using linear combinations of Wiener processes. This approximation enabled the derivation of evidence lower bounds through Girsanov's change of measure, yielding posterior path measures and likelihood estimates with optimal coefficients in closed form. Our experimental validation across fractional Ornstein-Uhlenbeck bridges and Hurst index estimation confirmed the effectiveness of this approach. Furthermore, our continuous-time architecture utilizing Markov-approximate fBM-driven neural-SDEs demonstrated improvements in video prediction, particularly when inferring the Hurst parameter during inference.

We build upon the approaches presented in Opper (2019) and Li et al. (2020) for variational inference (VI) for stochastic differential equations (SDEs). It entails solving SDEs during training, and using gradient descent optimization through the solver. While a relatively general and elegant approach, it can be computationally expensive and thus practically limited. In recent years, a number of works have proposed alternative approaches for inferring or learning SDEs (Kong et al. (2020), Solin et al. (2021), Kidger et al. (2021a), Kidger (2021), Zhang et al. (2023), Course and Nair (2024), and Bartosh et al. (2025)). We present an optimal control inspired method in Chapter 4 for efficient variational inference for (neural) SDEs. Under practically reasonable assumptions, we explicitly formulate the control term with linear and residual non-linear components and derive a closed-form control term for the linear part using stochastic optimal control. This model is shown to converge faster than a standard non-linear SDE, both for SDEs driven by BM and Markov-approximate fBM.

The preceding three paragraphs address the three research goals outlined in Sec. 1.8, thereby encapsulating the primary contributions of this dissertation: *learning from video in continuous-time*

using physics priors and fractional noise. The integration of visual data for dynamical system modeling furthermore holds significant promise as imaging technology continues to advance. Camera sensors are becoming increasingly affordable and powerful, offering rich information sources that can replace and enhance multiple sensor modalities at lower cost. Our research demonstrates that it is possible to simultaneously learn both Lagrangian dynamics and state estimator models from images in one end-to-end process. Secondly, our advances in variational inference for fractional processes open new avenues for modeling complex temporal phenomena with long-range dependencies. Lastly, we proposed a method inspired from optimal control theory to more efficiently learn SDEs from data.

5.2 Future Work

The humble progress presented in this dissertation is only a small step forward compared to the vast number of possible research directions. In fact, the more progress is made, the more questions arise. This future work section is a good opportunity to write down some of the most promising ideas. Some of these are relatively well thought out, while not emperically validated yet. We hope it might inspire future researchers to explore these directions[1].

5.2.1 KeyCLD

The work presented in Chapter 2 uses a given constraint manifold to learn constrained Lagrangian dynamics. Learning the constraint manifold (or constraint function) from data is a natural next step. Using a setup with multiple cameras, one could also learn to estimate keypoints in 3-d. Lastly, learning non-conservative forces such as friction and working towards real-world applications are important future work directions.

1. Feel free to contact me for further discussions or possible collaborations!

5.2.2 Variational inference for SDEs

The work presented in Chapter 3 is limited to variational inference for stochastic differential equations (SDEs) driven by fractional Brownian motion (fBM). Future work can explore SDEs driven by other types of noise, such as Lévy processes of fractional Lévy processes, which are better suited to model *long-tailed* noise.

5.2.3 Improving the Markov approximation of fractional Brownian motion with a constant drift

Specifically for fractional Brownian motion (fBM) Type I, the approximation quality is low for a Hurst index H close to 1 (see Fig. 3.14). Nevertheless, one would expect approximating fBM to be trivial close to $H = 1$, since in the limit of $H \to 1$, $B_H^{(I)}(t)$ converges to a straight line:

$$\lim_{H \to 1} B_H^{(I)}(t) = \varepsilon \frac{t}{T}, \quad \varepsilon \overset{d}{=} B_H^{(I)}(T)\,, \tag{5.1}$$

where $\overset{d}{=}$ denotes equality in distribution. In other words, by sampling one point ε at some time $T > 0$, we can approximate $B_H^{(I)}(t)$ close to $H = 1$ as a straight line. Additionally, realized paths of MA-fBM with H close to 1 show a *drifting* behaviour from the true fBM (see Fig. 3.12). These approximations would benefit from the simple addition of a constant drift term, that negates this error. Thus, we propose to explicitly add a straight line to the the approximation:

$$\hat{B}_H^{(I)}(t) = \sum_k \omega_k \left(Y_k(t) - Y_k(0)\right) + \omega_\varepsilon \varepsilon \frac{t}{T}\,, \tag{5.2}$$

where we have to find the optimal ω_ε together with the usual ω_k's. Since this approximation is also a weighted sum of terms, we can use the same method as before to find the optimal weights (see Sec. 3.3.1). Let us define $\boldsymbol{\omega}^{(\varepsilon)} = [\omega_1, \ldots, \omega_K, \omega_\varepsilon]$ and work out

$\boldsymbol{A}^{(\varepsilon)}$, $\boldsymbol{b}^{(\varepsilon)}$ as before (note that $c^{(\varepsilon)} \equiv c^{(I)}$).

$$\boldsymbol{A}^{(\varepsilon)}_{1:K,1:K} = \boldsymbol{A}^{(I)}\,, \tag{5.3}$$

$$\boldsymbol{A}^{(\varepsilon)}_{\varepsilon,\varepsilon} = \int_0^T \frac{t^2}{T^2}\mathbb{E}\left[\varepsilon^2\right]\mathrm{d}t \tag{5.4}$$

$$= \int_0^T \frac{t^2}{T^2}\mathbb{E}\left[B_H^{(I)}(T)^2\right]\mathrm{d}t \tag{5.5}$$

$$= V_H \frac{T^{2H+1}}{3}\,, \tag{5.6}$$

$$\boldsymbol{A}^{(\varepsilon)}_{1:K,\varepsilon} = \int_0^T \frac{t}{T}\mathbb{E}\left[B_H^{(I)}(T)\left(Y_k(t) - Y_k(0)\right)\right]\mathrm{d}t \tag{5.7}$$

$$= \frac{1}{\Gamma(H+1/2)}\int_0^T \frac{t}{T}\Bigg((e^{-\gamma_k t}-1) \cdot \int_{-\infty}^0 \left((T-s)^{H-1/2} - (-s)^{H-1/2}\right)e^{\gamma_k s}\,\mathrm{d}s + \int_0^t (T-s)^{H-1/2}e^{-\gamma_k(t-s)}\,\mathrm{d}s\Bigg)\mathrm{d}t \tag{5.8}$$

$$= \int_0^T \frac{t}{\gamma_k^{H+1/2}T}\Big(1 - e^{-\gamma_k t} - e^{\gamma_k T}Q(H+1/2,\gamma_k T) + e^{\gamma_k(T-t)}Q(H+1/2,\gamma_k(T-t))\Big)\mathrm{d}t \tag{5.9}$$

$$= \frac{1}{2\gamma_k^{H+5/2}T}\Big(-4 + \gamma_k T(\gamma_k T - 2) + (\gamma_k T)^{H+1/2}\frac{2\gamma_k T + 2H + 3}{\Gamma(H+5/2)} + 2e^{-\gamma_k T}(\gamma_k T + 1) - (\gamma_k^2 T^2 - 2)e^{\gamma_k T}Q(H+1/2,\gamma_k T)\Big)\,, \tag{5.10}$$

$$\boldsymbol{b}^{(\varepsilon)}_{1:K} = \boldsymbol{b}^{(I)}\,, \tag{5.11}$$

$$\boldsymbol{b}^{(\varepsilon)}_{\varepsilon} = \int_0^T \frac{t}{T}\mathbb{E}\left[B_H^{(I)}(t)B_H^{(I)}(T)\right]\mathrm{d}t \tag{5.12}$$

$$= \frac{1}{2}V_H\int_0^T \frac{t}{T}(t^{2H} + T^{2H} - (T-t)^{2H})\,\mathrm{d}t \tag{5.13}$$

$$= V_H\frac{2H^2+5H+1}{4(2H+1)(H+1)}T^{2H+1}\,. \tag{5.14}$$

Would that this where to be used as a Markov-Approximation, we need to be able to write it as a system of SDEs. Remember that we *first* sample ε from a Gaussian distribution, and then start solving the system of SDEs. This means we need to condition on ε, which will be (approximated by) $\hat{B}_H^{(I)}(T)$, the value at the endpoint T. For that we introduce a new process $Y_\varepsilon(t)$:

$$Y_\varepsilon(0) = \frac{1}{\Gamma(H+1/2)} \int_{-\infty}^{0} \left((T-t)^{H-1/2} - (-t)^{H-1/2} \right) \mathrm{d}W(t)\,, \tag{5.15}$$

$$\mathrm{d}Y_\varepsilon(t) = G_H(t)\,\mathrm{d}W(t)\,,\ G_H(t) = \frac{(T-t)^{H-1/2}}{\Gamma(H+1/2)}\,, \tag{5.16}$$

for which the solution at time T is obviously (see Eq. (3.7))

$$Y_\varepsilon(T) \overset{d}{=} B_H^{(I)}(T) \overset{d}{=} \varepsilon\,. \tag{5.17}$$

We now consider the system of SDEs

$$\mathrm{d}\hat{B}_H^{(I)}(t) = \sum_k \omega_k\,\mathrm{d}Y_k(t) + \omega_\varepsilon \frac{\varepsilon}{T}\,\mathrm{d}t\,, \tag{5.18}$$

$$\mathrm{d}Y_k(t) = -\gamma_k Y_k(t)\,\mathrm{d}t + \mathrm{d}W(t)\,, \quad k = 1, \ldots, K\,, \tag{5.19}$$

$$\mathrm{d}Y_\varepsilon(t) = G_H(t)\,\mathrm{d}W(t)\,. \tag{5.20}$$

To condition the SDEs on ε, we introduce an extra control term $u_\varepsilon\left(Y_\varepsilon(t), t\right)$:

$$\mathrm{d}\hat{B}_H^{(I)}(t) = \sum_k \omega_k\,\mathrm{d}Y_k(t) + \omega_\varepsilon \frac{\varepsilon}{T}\,\mathrm{d}t \tag{5.21}$$

$$\mathrm{d}Y_k(t) = -\gamma_k Y_k(t)\,\mathrm{d}t + u_\varepsilon\left(Y_\varepsilon(t), t\right)\mathrm{d}t + \mathrm{d}W(t), \quad k = 1, \ldots, K\,, \tag{5.22}$$

$$\mathrm{d}Y_\varepsilon(t) = G_H(t) u_\varepsilon\left(Y_\varepsilon(t), t\right)\mathrm{d}t + G_H(t)\,\mathrm{d}W(t)\,. \tag{5.23}$$

As usual (see Sec. 4.3.1)

$$u_\varepsilon\left(Y_\varepsilon(t), t\right) = \partial_{Y_\varepsilon} \log p\left(Y_\varepsilon(T) = \varepsilon | Y_\varepsilon(t)\right)\,, \tag{5.24}$$

where the probability density p is with respect to the unconditioned SDEs. Note, this is only a function of Y_ε, not the other Y_k's.

We have:

$$Y_\varepsilon(T) = Y_\varepsilon(t) + \int_t^T G_H(t)\,\mathrm{d}W(t)\,, \tag{5.25}$$

$$\mathbb{E}\left[Y_\varepsilon(T)|Y_\varepsilon(t)\right] = Y_\varepsilon(t)\,, \tag{5.26}$$

$$v_t = \mathrm{Var}(Y_\varepsilon(T)|Y_\varepsilon(t)) \tag{5.27}$$

$$= \int_t^T G_H(s)^2\,\mathrm{d}s \tag{5.28}$$

$$= \frac{(T-t)^{2H}}{2H\Gamma(H+1/2)^2}\,. \tag{5.29}$$

Hence

$$\log p\left(Y_\varepsilon(T) = \varepsilon|Y_\varepsilon(t)\right) = -\frac{(\varepsilon - Y_\varepsilon(t))^2}{2v_t} + \text{const.}\,, \tag{5.30}$$

and

$$u_\varepsilon\left(Y_\varepsilon(t), t\right) = \frac{\varepsilon - Y_\varepsilon(t)}{v_t}\,. \tag{5.31}$$

$Y_\varepsilon(0)$ can be sampled together with the usual $Y_1(0), \ldots, Y_K(0)$ (see Dfn. 4) from their joint Gaussian distributions using:

$$\mathrm{Var}(\varepsilon) = V_H T^{2H}\,, \tag{5.32}$$

$$\mathrm{Var}(Y_\varepsilon(0)) = \mathrm{Var}(\varepsilon) - v_0 \tag{5.33}$$

$$= V_H T^{2H} - v_0\,, \tag{5.34}$$

$$\mathrm{Cov}(Y_\varepsilon(0), Y_k(0)) = \frac{1}{\Gamma(H+1/2)}\mathbb{E}\Bigg[\int_{-\infty}^0 e^{\gamma_k s}\,\mathrm{d}W(s) \cdot \left(\int_{-\infty}^0 (T-s)^{H-1/2}\,\mathrm{d}W(s) - \int_{-\infty}^0 (-s)^{H-1/2}\,\mathrm{d}W(s)\right)\Bigg] \tag{5.35}$$

$$= \frac{1}{\Gamma(H+1/2)}\int_{-\infty}^0 e^{\gamma_k s}\Big((T-s)^{H-1/2} - (-s)^{H-1/2}\Big)\,\mathrm{d}s \tag{5.36}$$

$$= \frac{e^{\gamma_k T} Q(H+1/2, \gamma_k T) - 1}{\gamma_k^{H+1/2}}\,. \tag{5.37}$$

Finally, we can sample ε conditionally on $Y_\varepsilon(0)$:

$$\mathbb{E}\left[\varepsilon|Y_\varepsilon(0)\right] = Y_\varepsilon(0) \tag{5.38}$$

$$\mathrm{Var}(\varepsilon|Y_\varepsilon(0)) = v_0 \tag{5.39}$$

$$\varepsilon|Y_\varepsilon(0) \sim \mathcal{N}(Y_\varepsilon(0), v_0)\,. \tag{5.40}$$

This ultimately leads to an adapted augmented system SDEs driven by ε-MA-fBM.

Definition 10 (ε-Markov-Approximate fBMSDE (ε-MA-fBMSDE))**.** *Substituting the fBM, $B_H(t)$, in Dfn. 6 by the finite linear combination of OU-processes $\hat{B}_H(t)$ and an extra constant drift term, we define ε-MA-fBMSDE as:*

$$\mathrm{d}X(t) = b_\theta\left(X(t), t\right)\mathrm{d}t + \sigma_\theta\left(X(t), t\right)\mathrm{d}\hat{B}_H(t), \tag{5.41}$$

where (cf. see above)

$$\mathrm{d}\hat{B}_H(t) = \sum_{k=1}^{K}\omega_k\,\mathrm{d}Y_k(t) + \omega_\varepsilon\frac{\varepsilon}{T}\,\mathrm{d}t \tag{5.42}$$

$$= \sum_{k=1}^{K}\omega_k(-\gamma_k Y_k(t)\,\mathrm{d}t + u_\varepsilon\left(Y_\varepsilon(t), t\right)\mathrm{d}t + \mathrm{d}W(t)) + \omega_\varepsilon\frac{\varepsilon}{T}\,\mathrm{d}t \tag{5.43}$$

$$= \left(\bar{\omega}u_\varepsilon\left(Y_\varepsilon(t), t\right) + \omega_\varepsilon\frac{\varepsilon}{T} - \sum_{k=1}^{K}\omega_k\gamma_k Y_k(t)\right)\mathrm{d}t + \bar{\omega}\,\mathrm{d}W(t)\,, \tag{5.44}$$

$$\mathrm{d}Y_k(t) = \left(-\gamma_k Y_k(t) + u_\varepsilon\left(Y_\varepsilon(t), t\right)\right)\mathrm{d}t + \mathrm{d}W(t)\,, \tag{5.45}$$

$$\mathrm{d}Y_\varepsilon(t) = G_H(t)u_\varepsilon\left(Y_\varepsilon(t), t\right)\mathrm{d}t + G_H(t)\,\mathrm{d}W(t)\,. \tag{5.46}$$

Proposition 9 (Augmented Markov SDE system for ε-MA-fBM including ε method)**.** *$X(t)$ can be augmented by the finite number of Markov processes $Y_k(t)$ and the constant drift term $Y_\varepsilon(t)$ (approximating $B_H(t)$) to a higher dimensional state variable of the form $Z(t) \doteq (X(t), Y_1(t), \ldots Y_K(t), Y_\varepsilon(t)) \in \mathbb{R}^{(K+2)\times D}$, such that the joint process of the augmented system becomes Markovian and can be described by an 'ordinary' SDE:*

$$\mathrm{d}Z(t) = h_\theta\left(Z(t), t\right)\mathrm{d}t + \Sigma_\theta\left(Z(t), t\right)\mathrm{d}W(t), \tag{5.47}$$

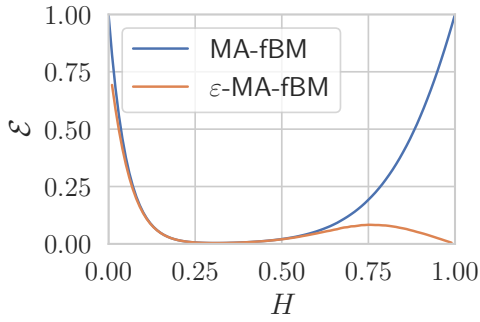


Figure 5.1 Comparison of the approximation error of MA-fBM and ε-MA-fBM in function of H and with a time horizon $T = 10$, $\gamma_{\min} = 10^{-2}$ and $\gamma_{\max} = 10^{2}$.

where the augmented drift vector $h_\theta \in \mathbb{R}^{(K+2)\times D}$ *and the augmented diffusion matrix* $\Sigma_\theta(Z,t) \in \mathbb{R}^{(K+2)\times D\times D}$ *are given by*

$$h_\theta(Z,t) = \begin{pmatrix} b_\theta(X,t) + \sigma_\theta(X,t)\left(\omega_\varepsilon \frac{\varepsilon}{T} - \sum_k \omega_k \gamma_k Y_k\right) \\ -\gamma_1 Y_1 \\ \dots \\ -\gamma_K Y_K \\ 0 \end{pmatrix} + \begin{pmatrix} \bar{\omega}\sigma_\theta(X,t) \\ 1 \\ \dots \\ 1 \\ G_H(t) \end{pmatrix} u_\varepsilon(Y_\varepsilon, t), \tag{5.48}$$

$$\Sigma_\theta(Z,t) = \begin{pmatrix} \bar{\omega}\sigma_\theta(X,t) \\ I_D \\ \vdots \\ I_D \\ G_H(t) I_D \end{pmatrix}, \tag{5.49}$$

where $I_D \in \mathbb{R}^{D\times D}$ *is the identity matrix, and where* $\varepsilon \in \mathbb{R}^D$ *and* $u_\varepsilon(Y_\varepsilon, t) : \mathbb{R}^D \times \mathbb{R} \to \mathbb{R}^D$ *are generalized to vectors.*

Fig. 5.1 illustrates the approximation error of MA-fBM compared to the presented ε-MA-fBM. The ε-MA-fBM error is uniformly

lower than the MA-fBM for any H, and significantly lower for $H > 0.5$. As can be expected, the error goes down to 0 for the limit $H \to 1$, since the true fBM is a straight line there. But also for H around 0.6, the error is already lower thanks to the addition of the constant drift term. This shows the promise of the ε-MA-fBM approach.

Variational Inference

Since this approximation is again a Markovian system of SDEs, with one extra process $Y_\varepsilon(t)$, we can use the same variational inference approach as in Chapter 3. Note that, in principle, $Y_\varepsilon(0)$ and ε should be modeled with a posterior distribution.

Linear SDE driven by ε-MA-fBM

In the case of a linear 1-d SDE, *e.g.*, for fractional diffusion models (Nobis et al. (2024)) or fractional Schrödinger bridges (Nobis et al. (2025)), we have

$$\mathrm{d}X(t) = -\theta_t(X(t) - \mu_t)\,\mathrm{d}t + \sigma_t\,\mathrm{d}\hat{B}_H(t)\,, \tag{5.50}$$

which can be written in augmented form as

$$\mathrm{d}Z(t) = F(t)Z(t)\,\mathrm{d}t + \varepsilon u(t)\,\mathrm{d}t + L(t)\,\mathrm{d}W(t)\,, \tag{5.51}$$

where

$$F(t) = \begin{pmatrix} -\theta_t & -\sigma_t\omega_1\gamma_1 & \cdots & -\sigma_t\omega_K\gamma_K & -\sigma_t\bar{\omega}/v_t \\ 0 & -\gamma_1 & & & -1/v_t \\ \vdots & & \ddots & & \vdots \\ 0 & & & -\gamma_K & -1/v_t \\ 0 & 0 & \cdots & 0 & -G_H(t)/v_t \end{pmatrix}, \tag{5.52}$$

$$u(t) = \begin{pmatrix} \theta_t\mu_t + \sigma_t\left(\omega_\varepsilon/T + \bar{\omega}/v_t\right) \\ 1/v_t \\ \vdots \\ 1/v_t \\ G_H(t)/v_t \end{pmatrix}, \tag{5.53}$$

$$L(t) = \begin{pmatrix} \sigma_t \bar{\omega} \\ 1 \\ \vdots \\ 1 \\ G_H(t) \end{pmatrix} . \tag{5.54}$$

The mean and covariance can be solved using ODEs (Särkkä and Solin (2019)):

$$\frac{\mathrm{d}m}{\mathrm{d}t} = F(t)m + \varepsilon u(t)\,, \tag{5.55}$$

$$\frac{\mathrm{d}P}{\mathrm{d}t} = F(t)P + PF^\top(t) + L(t)L^\top(t)\,, \tag{5.56}$$

with initial values $m(0)$ and $P(0)$. We will assume that $\mathbb{E}[X(0)] = 0$ and we know that $\mathbb{E}[Y(0)]$ for all Y processes (see above), thus $m(0) = 0$. We also assume $\mathrm{Var}(X(0)) = 0$ and $\mathrm{Cov}(X(0), Y(0)) = 0$ for all Y processes, thus all other entries in $P(0)$ are given above.

However, the mean is conditioned on ε, which is not very convenient because the ODEs would need to be solved every time again, for every sampled ε. Still, one can express the solution using the transition matrix $\Psi(\tau, t)$:

$$m(t) = \Psi(t, 0)m_0 + \int_0^t \Psi(t, \tau)\varepsilon u(\tau)\,\mathrm{d}\tau \tag{5.57}$$

$$= \varepsilon \int_0^t \Psi(t, \tau)u(\tau)\,\mathrm{d}\tau\,, \tag{5.58}$$

because $m_0 = 0$. This means, for a general choice of $\theta_t, \mu_t, \sigma_t$

$$m_p(t) = \int_0^t \Psi(t, \tau)u(\tau)\,\mathrm{d}\tau, \tag{5.59}$$

which is the solution of the ODE

$$\frac{\mathrm{d}m_p}{\mathrm{d}t} = F(t)m_p + u(t), \quad m_p(0) = 0, \tag{5.60}$$

should be solved numerically. This leads to a general solution

$$m(t) = \varepsilon m_p(t)\,. \tag{5.61}$$

Likewise, $P(t)$ should be solved numerically (it does not depend on ε). Finally, we have

$$p(Z(t)|\varepsilon) = \mathcal{N}(\varepsilon m_p(t), P(t)), \tag{5.62}$$

$$p(Z(t)) = \int p(Z(t)|\varepsilon)p(\varepsilon)\,\mathrm{d}\varepsilon \tag{5.63}$$

$$= \mathcal{N}\left(m_p(t)\mathbb{E}[\varepsilon], P(t) + m_p(t)\mathrm{Var}(\varepsilon)m_p^\top(t)\right) \tag{5.64}$$

$$= \mathcal{N}\left(0, P(t) + V_H T^{2H} m_p(t)m_p^\top(t)\right). \tag{5.65}$$

As said above, this solution is for $X(0) = 0$. As usual, for a given datapoint x_0:

$$p(Z(t)|X(0) = x_0) = \mathcal{N}\left([x_0, 0, \ldots, 0]^\top, P(t) + V_H T^{2H} m_p(t)m_p^\top(t)\right). \tag{5.66}$$

5.2.4 Efficient learning of stochastic differential equations

In Chapter 4 we propose a method for more efficient learning of stochastic differential equations (SDEs) that is based on optimal stochastic control. A possible improvement of this method is the generalization to multidimensional SDEs. In this case, $\lambda_\theta \in \mathbb{R}^{D\times D}$ and $\eta_\theta \in \mathbb{R}^D$, and the non-linear drift function $b_\theta(X(t), t) : \mathbb{R}^D \times \mathbb{R} \to \mathbb{R}^D$. Note that Prop. 8 is already valid for the multidimensional case. The multidimensional prior process is:

$$\mathrm{d}X(t) = (-\lambda X(t) + \eta)\,\mathrm{d}t + \varsigma\,\mathrm{d}B(t), \tag{5.67}$$

with $\lambda \in \mathbb{R}^{D\times D}$, $\eta \in \mathbb{R}^D$ and $\varsigma \in \mathbb{R}^{D\times D}$. Deriving the full Gaussian expression for $p(X(t_1), \ldots, X(t_M)|x)$ can be done by defining a concatenization of the states at the different observation times $(t_1, \ldots, t_M)$. This could be a practical hurdle due to the increased size of the covariance matrix. An alternative approach could be to apply backwards filtering, *i.e.*, solving the backward Kolmogorov equations as described in Archambeau and Opper (2011).

Another improvement is to define a local linearization scheme instead of the explicit residual formulation (Sec. 4.3.2). Using the explicit residual formulation, there is no guarantee that the non-linear term contains no linear terms, which are then *not* taken

into account by the optimal linear control term. Instead, we can define a general, non-linear prior drift function $b_\theta(X(t), t)$, and use a local linearization. Concretely, the prior drift function is locally linearized at current time t^* and state x using

$$b_\theta(X(t), t) \approx -\lambda X(t) + \eta\,, \tag{5.68}$$

$$\lambda = -\frac{\partial b_\theta}{\partial X}(X(t), t^*)\Big|_{X(t)=x}\,, \tag{5.69}$$

$$\eta = b_\theta(x, t^*) + \lambda x\,. \tag{5.70}$$

λ and η derived this way can be analoguously used to calculate the prior distribution used in the formulation of the optimal control term (Eq. (4.5)). In the context of learning SDEs with gradient descent, $b_\theta(X(t), t)$ should be differentiable with respect to $X(t)$, thus calculating λ and η is trivial using the specific tools provided by the machine learning library used for the implementation. The downside of this method is the possible increase in computation time, as the local linearization has to be calculated at each time step of the SDE solver.

Finally, the method is not immediatly applicable to latent SDEs, since a tractable observation likelihood needs to be available in latent space. If the encoding from the observation space to latent space is linear, the observation likelihood can be mapped to the latent space. For non-linear encodings, the observation likelihood has to be approximated in the latent space, *e.g.*, by using particle filters.

Bibliography

Alemi, Alexander, Ben Poole, Ian Fischer, Joshua Dillon, Rif A Saurous, and Kevin Murphy. 2018. "Fixing a broken ELBO." In *International conference on machine learning,* 159–168. PMLR.

Alfonsi, Aurélien, and Ahmed Kebaier. 2021. "Approximation of Stochastic Volterra Equations with kernels of completely monotone type." *arXiv preprint arXiv:2102.13505*.

Ali, Moayed Haji, Andrew Bond, Tolga Birdal, Duygu Ceylan, Levent Karacan, Erkut Erdem, and Aykut Erdem. 2023. "VidStyleODE: Disentangled Video Editing via StyleGAN and NeuralODEs." In *International Conference on Computer Vision (ICCV)*.

Allen-Blanchette, Christine, Sushant Veer, Anirudha Majumdar, and Naomi Ehrich Leonard. 2020. "LagNetVip: A lagrangian neural network for video prediction." *arXiv preprint arXiv:2010.12932*.

Allouche, Michaël, Stéphane Girard, and Emmanuel Gobet. 2022. "A generative model for fBm with deep ReLU neural networks." *Journal of Complexity* 73:101667.

Andersson, Joel A E, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. 2019. "CasADi – A software framework for nonlinear optimization and optimal control." *Mathematical Programming Computation* 11 (1): 1–36. https://doi.org/10.1007/s12532-018-0139-4.

Archambeau, Cedric, and Manfred Opper. 2011. "Approximate inference for continuous-time Markov processes." *Bayesian time series models,* 125–140.

Asseman, Alexis, Tomasz Kornuta, and Ahmet Ozcan. 2018. "Learning beyond simulated physics." In *Modeling and Decision-making in the Spatiotemporal Domain Workshop,* in *Neural Information Processing Systems.* https://openreview.net/pdf?id=HylajWsRF7.

Babaeizadeh, Mohammad, Chelsea Finn, Dumitru Erhan, Roy H Campbell, and Sergey Levine. 2018. "Stochastic Variational Video Prediction." In *International Conference on Learning Representations.*

Babuschkin, Igor, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, et al. 2020. *The DeepMind JAX Ecosystem.* http://github.com/deepmind.

Barron, Jonathan T. 2017. "Continuously differentiable exponential linear units." *arXiv preprint arXiv:1704.07483.*

Bartosh, Grigory, Dmitry Vetrov, and Christian A Naesseth. 2025. "SDE Matching: Scalable and Simulation-Free Training of Latent Stochastic Differential Equations." *arXiv preprint arXiv:2502.02472.*

Bayer, Christian, and Simon Breneis. 2023a. "Markovian approximations of stochastic Volterra equations with the fractional kernel." *Quantitative Finance* 23 (1): 53–70.

———. 2023b. "Weak Markovian approximations of rough Heston." *arXiv preprint arXiv:2309.07023.*

Betsch, Peter. 2005. "The discrete null space method for the energy consistent integration of constrained mechanical systems: Part I: Holonomic constraints." *Computer Methods in Applied Mechanics and Engineering* 194 (50-52): 5159–5190.

Bishop, Christopher M, and Nasser M Nasrabadi. 2006. *Pattern recognition and machine learning.* Vol. 4. Springer.

Blei, David M, Alp Kucukelbir, and Jon D McAuliffe. 2017. "Variational inference: A review for statisticians." *Journal of the American statistical Association* 112 (518): 859–877.

Botev, Aleksandar, Andrew Jaegle, Peter Wirnsberger, Daniel Hennes, and Irina Higgins. 2021. "Which priors matter? Benchmarking models for learning latent dynamics." *arXiv e-prints,* arXiv–2111.

Bradbury, James, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, et al. 2018. *JAX: composable transformations of Python+NumPy programs.* V. 0.3.13. http://github.com/google/jax.

Cao, Zhe, Tomas Simon, Shih-En Wei, and Yaser Sheikh. 2017. "Realtime multi-person 2d pose estimation using part affinity fields." In *Proceedings of the IEEE conference on computer vision and pattern recognition,* 7291–7299.

Carmona, Philippe, and Laure Coutin. 1998a. "Fractional Brownian motion and the Markov property." *Electronic Communications in Probability [electronic only]* 3:95–107.

———. 1998b. "Simultaneous approximation of a family of (stochastic) differential equations." *Unpublished, June* 915 (10.1051).

Carmona, Philippe, Laure Coutin, and Gérard Montseny. 2000. "Approximation of some Gaussian processes." *Statistical inference for stochastic processes* 3:161–171.

Chen, Boyuan, Pieter Abbeel, and Deepak Pathak. 2021. "Unsupervised learning of visual 3D keypoints for control." In *International Conference on Machine Learning,* 1539–1549. PMLR.

Chen, Boyuan, Kuang Huang, Sunand Raghupathi, Ishaan Chandratreya, Qiang Du, and Hod Lipson. 2022. "Automated discovery of fundamental variables hidden in experimental data." *Nature Computational Science* 2 (7): 433–442.

Chen, Ricky TQ, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. 2018. "Neural ordinary differential equations." In *Proceedings of the 32nd International Conference on Neural Information Processing Systems,* 6572–6583.

Ciregan, Dan, Ueli Meier, and Jürgen Schmidhuber. 2012. "Multi-column deep neural networks for image classification." In *2012 IEEE conference on computer vision and pattern recognition,* 3642–3649. IEEE.

Course, Kevin, and Prasanth Nair. 2024. "Amortized reparametrization: efficient and scalable variational inference for latent SDEs." In *Advances in Neural Information Processing Systems.*

Cranmer, Miles, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. 2020. "Lagrangian Neural Networks." In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations.*

Daems, Rembert, Manfred Opper, Guillaume Crevecoeur, and Tolga Birdal. 2024. "Variational Inference for SDEs Driven by Fractional Noise." In *The Twelfth International Conference on Learning Representations.* https://openreview.net/forum?id=rtx8B94JMS.

———. 2025. "Efficient Training of Neural SDEs Using Stochastic Optimal Control." In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN).*

Daems, Rembert, Jeroen Taets, Guillaume Crevecoeur, et al. 2024. "KeyCLD: Learning constrained Lagrangian dynamics in keypoint coordinates from images." *Neurocomputing* 573:127175.

Davidson, James, and Nigar Hashimzade. 2009. "Type I and type II fractional Brownian motions: A reconsideration." *Computational Statistics & Data Analysis* 53 (6): 2089–2106.

Denton, Emily, and Rob Fergus. 2018. "Stochastic video generation with a learned prior." In *International conference on machine learning,* 1174–1183. PMLR.

Finzi, Marc, Ke Alexander Wang, and Andrew G Wilson. 2020. "Simplifying Hamiltonian and Lagrangian neural networks via explicit constraints." *Advances in neural information processing systems* 33:13880–13889.

Franceschi, Jean-Yves, Edouard Delasalles, Mickaël Chen, Sylvain Lamprier, and Patrick Gallinari. 2020. "Stochastic latent residual video prediction." In *International Conference on Machine Learning,* 3233–3246. PMLR.

Gallier, Jean, et al. 2010. "The Schur complement and symmetric positive semidefinite (and definite) matrices." *Penn Engineering,* 1–12.

Ganguly, Ankush, and Samuel WF Earp. 2021. "An introduction to variational inference." *arXiv preprint arXiv:2108.13083.*

Gardiner, Crispin W, et al. 1985. *Handbook of stochastic methods.* Vol. 3. springer Berlin.

Gatheral, Jim, Thibault Jaisson, and Mathieu Rosenbaum. 2018. "Volatility is rough." *Quantitative finance* 18 (6): 933–949.

Gojcic, Zan, Or Litany, Andreas Wieser, Leonidas J Guibas, and Tolga Birdal. 2021. "Weakly supervised learning of rigid 3D scene flow." In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition,* 5692–5703.

Gomez-Estern, Fabio, Romeo Ortega, Francisco R Rubio, and Javier Aracil. 2001. "Stabilization of a class of underactuated mechanical systems via total energy shaping." In *Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No. 01CH37228),* 2:1137–1143. IEEE.

Goodfellow, Ian, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. *Deep learning.* Vol. 1. 2. MIT press Cambridge.

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. "Generative adversarial nets." *Advances in neural information processing systems* 27.

Gordon, Cade, and Natalie Parde. 2021. "Latent neural differential equations for video generation." In *NeurIPS 2020 Workshop on Pre-registration in Machine Learning,* 73–86. PMLR.

Greydanus, Sam, and Andrew Sosanya. 2022. "Dissipative Hamiltonian Neural Networks: Learning Dissipative and Conservative Dynamics Separately." *arXiv preprint arXiv:2201.10085.*

Greydanus, Samuel, Misko Dzamba, and Jason Yosinski. 2019. "Hamiltonian Neural Networks." In *Advances in Neural Information Processing Systems,* edited by H Wallach, H Larochelle, A Beygelzimer, F d'Alché-Buc, E Fox, and R Garnett, vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/26cd8ecadce0d4efd6cc8a8725cbd1f8-Paper.pdf.

Gruver, Nate, Marc Anton Finzi, Samuel Don Stanton, and Andrew Gordon Wilson. 2021. "Deconstructing the Inductive Biases of Hamiltonian Neural Networks." In *International Conference on Learning Representations.*

Gu, Zaiwang, Jun Cheng, Huazhu Fu, Kang Zhou, Huaying Hao, Yitian Zhao, Tianyang Zhang, Shenghua Gao, and Jiang Liu. 2019. "Ce-net: Context encoder network for 2d medical image segmentation." *IEEE transactions on medical imaging* 38 (10): 2281–2292.

Guerra, Joao, and David Nualart. 2008. "Stochastic differential equations driven by fractional Brownian motion and standard Brownian motion." *Stochastic analysis and applications* 26:1053–1075.

Harms, Philipp. 2020. "Strong convergence rates for markovian representations of fractional processes." *Discrete and Continuous Dynamical Systems-B* 26 (10): 5567–5579.

Harms, Philipp, and David Stefanovits. 2019. "Affine representations of fractional processes with applications in mathematical finance." *Stochastic Processes and their Applications* 129:1185–1228.

Hayashi, Kohei, and Kei Nakagawa. 2022. "Fractional SDE-Net: Generation of time series data with long-term memory." In *2022 IEEE 9th International Conference on Data Science and Advanced Analytics (DSAA),* 1–10. IEEE.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. "Deep residual learning for image recognition." In *Proceedings of the IEEE conference on computer vision and pattern recognition,* 770–778.

Heek, Jonathan, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. 2023. *Flax: A neural network library and ecosystem for JAX*. V. 0.7.4. http://github.com/google/flax.

Hessel, Matteo, David Budden, Fabio Viola, Mihaela Rosca, Eren Sezener, and Tom Hennigan. 2020. "Optax: composable gradient transformation and optimisation, in JAX!," http://github.com/deepmind/optax.

Ho, Jonathan, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. 2022. "Video diffusion models." *arXiv:2204.03458*.

Jakab, Tomas, Ankush Gupta, Hakan Bilen, and Andrea Vedaldi. 2018. "Unsupervised learning of object landmarks through conditional image generation." In *Proceedings of the 32nd International Conference on Neural Information Processing Systems,* 4020–4031.

Jaques, Miguel, Martin Asenov, Michael Burke, and Timothy Hospedales. 2021. "Vision-based system identification and 3D keypoint discovery using dynamics constraints." *arXiv preprint arXiv:2109.05928*.

Jaques, Miguel, Michael Burke, and Timothy Hospedales. 2019. "Physics-as-Inverse-Graphics: Unsupervised Physical Parameter Estimation from Video." In *International Conference on Learning Representations*.

Jin, Pengzhan, Zhen Zhang, Aiqing Zhu, Yifa Tang, and George Em Karniadakis. 2020. "SympNets: Intrinsic structure-preserving symplectic networks for identifying Hamiltonian systems." *Neural Networks* 132:166–179.

Kappen, Hilbert J. 2005. "Linear theory for control of nonlinear stochastic systems." *Physical review letters* 95 (20): 200201.

Kappen, Hilbert Johan, and Hans Christian Ruiz. 2016. "Adaptive importance sampling for control and inference." *Journal of Statistical Physics* 162:1244–1266.

Kidger, Patrick. 2021. "On Neural Differential Equations." PhD diss., University of Oxford.

Kidger, Patrick, James Foster, Xuechen Li, and Terry J Lyons. 2021a. "Neural sdes as infinite-dimensional gans." In *International conference on machine learning,* 5453–5463. PMLR.

Kidger, Patrick, James Foster, Xuechen Chen Li, and Terry Lyons. 2021b. "Efficient and accurate gradients for neural sdes." *Advances in Neural Information Processing Systems* 34:18747–18761.

Kingma, Diederik P, and Jimmy Ba. 2015. "Adam: A Method for Stochastic Optimization." In *International Conference on Learning Representations (ICLR).*

Kolter, Zico, David Duvenaud, and Matt Johnson. 2020. "Deep implicit layers-neural odes, deep equilibirum models, and beyond." *URL http://implicit-layers-tutorial. org/(visited on 2025-03-03).*

Kong, Lingkai, Jimeng Sun, and Chao Zhang. 2020. "SDE-Net: equipping deep neural networks with uncertainty estimates." In *Proceedings of the 37th International Conference on Machine Learning,* 5405–5415.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton. 2012. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems* 25.

Lanczos, Cornelius. 2020. "The variational principles of mechanics." In *The Variational Principles of Mechanics.* University of Toronto press.

Laus, L. P., and J. M. Selig. 2020. "Rigid body dynamics using equimomental systems of point-masses." *Acta Mechanica* 231 (1): 221–236. ISSN: 0001-5970, 1619-6937. https://doi.org/10.1007/s00707-019-02543-3.

Levina, Elizaveta, and Peter Bickel. 2004. "Maximum likelihood estimation of intrinsic dimension." *Advances in neural information processing systems* 17.

Li, Xuechen, Ting-Kam Leonard Wong, Ricky TQ Chen, and David K Duvenaud. 2020. "Scalable gradients and variational inference for stochastic differential equations." In *Symposium on Advances in Approximate Bayesian Inference,* 1–28. PMLR.

Liao, Shujian, Terry Lyons, Weixin Yang, and Hao Ni. 2019. "Learning stochastic differential equations using RNN with log signature features." *arXiv preprint arXiv:1908.08286.*

Lim, SC, and VM Sithi. 1995. "Asymptotic properties of the fractional Brownian motion of Riemann-Liouville type." *Physics Letters A* 206 (5-6): 311–317.

Liu, Xuanqing, Tesi Xiao, Si Si, Qin Cao, Sanjiv Kumar, and Cho-Jui Hsieh. 2019. "Neural sde: Stabilizing neural ode networks with stochastic noise." *arXiv preprint arXiv:1906.02355.*

Long, Jonathan, Evan Shelhamer, and Trevor Darrell. 2015. "Fully convolutional networks for semantic segmentation." In *Proceedings of the IEEE conference on computer vision and pattern recognition,* 3431–3440.

Luo, Zhengxiong, Dayou Chen, Yingya Zhang, Yan Huang, Liang Wang, Yujun Shen, Deli Zhao, Jingren Zhou, and Tieniu Tan. 2023. "VideoFusion: Decomposed Diffusion Models for High-Quality Video Generation." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,* 10209–10218.

Lutter, M, K Listmann, and J Peters. 2019. "Deep Lagrangian Networks for end-to-end learning of energy-based control for under-actuated systems." In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2019),* 7718–7725. IEEE.

Lutter, Michael, and Jan Peters. 2021. "Combining Physics and Deep Learning to learn Continuous-Time Dynamics Models." *arXiv e-prints,* arXiv–2110.

Lutter, Michael, Christian Ritter, and Jan Peters. 2018. "Deep Lagrangian Networks: Using Physics as Model Prior for Deep Learning." In *International Conference on Learning Representations.*

Lysy, Martin, and Natesh S Pillai. 2013. "Statistical inference for stochastic differential equations with memory." *arXiv preprint arXiv:1307.1164.*

Mandelbrot, Benoit B, and John W Van Ness. 1968. "Fractional Brownian motions, fractional noises and applications." *SIAM review* 10 (4): 422–437.

Maoutsa, Dimitra, and Manfred Opper. 2022. "Deterministic particle flows for constraining stochastic nonlinear systems." *Physical Review Research* 4 (4): 043035.

Minderer, Matthias, Chen Sun, Ruben Villegas, Forrester Cole, Kevin P Murphy, and Honglak Lee. 2019. "Unsupervised learning of object structure and dynamics from videos." *Advances in Neural Information Processing Systems* 32.

Morrill, James, Cristopher Salvi, Patrick Kidger, and James Foster. 2021. "Neural rough differential equations for long time series." In *International Conference on Machine Learning,* 7829–7838. PMLR.

Newell, Alejandro, Kaiyu Yang, and Jia Deng. 2016. "Stacked hourglass networks for human pose estimation." In *European conference on computer vision,* 483–499. Springer.

Nguyen-Phuoc, Thu, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. 2019. "HoloGAN: Unsupervised learning of 3D representations from natural images." In *Proceedings of the IEEE/CVF International Conference on Computer Vision,* 7588–7597.

Nobis, Gabriel, Arina Belova, Maximilian Springenberg, Rembert Daems, Christoph Knochenhauer, Manfred Opper, Tolga Birdal, and Wojciech Samek. 2025. "Fractional Brownian Bridges for Aligned Data." In *Learning Meaningful Representations of Life (LMRL) Workshop at ICLR 2025.*

Nobis, Gabriel, Maximilian Springenberg, Marco Aversa, Michael Detzel, Rembert Daems, Roderick Murray-Smith, Shinichi Nakajima, et al. 2024. "Generative Fractional Diffusion Models." In *Advances in Neural Information Processing Systems,* edited by A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, 37:25469–25509. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2024/file/2d2cf241331d7e71a6f20e9ed798a06b-Paper-Conference.pdf.

Øksendal, Bernt. 2003. *Stochastic differential equations.* Springer.

Opper, Manfred. 2019. "Variational inference for stochastic differential equations." *Annalen der Physik* 531 (3): 1800233.

Ortega, Romeo, Arjan J Van Der Schaft, Iven Mareels, and Bernhard Maschke. 2001. "Putting energy back in control." *IEEE Control Systems Magazine* 21 (2): 18–33.

Park, Sung Woo, Kyungjae Lee, and Junseok Kwon. 2021. "Neural markov controlled SDE: Stochastic optimization for continuous-time data." In *International Conference on Learning Representations.*

Park, Sunghyun, Kangyeol Kim, Junsoo Lee, Jaegul Choo, Joonseok Lee, Sookyung Kim, and Edward Choi. 2021. "Vid-ode: Continuous-time video generation with neural ordinary differential equation." In *Proceedings of the AAAI Conference on Artificial Intelligence,* vol. 35.

Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, et al. 2019. "PyTorch: An Imperative Style, High-Performance Deep Learning Library." In *Advances in Neural Information Processing Systems 32,* 8024–8035. Curran Associates, Inc. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Peltier, Romain-François, and Jacques Lévy Véhel. 1995. "Multifractional Brownian motion: definition and preliminary results." PhD diss., INRIA.

Poole, Ben, Sherjil Ozair, Aaron Van Den Oord, Alex Alemi, and George Tucker. 2019. "On variational bounds of mutual information." In *International conference on machine learning,* 5171–5180. PMLR.

Quetelet, Adolphe. 1842. *A treatise on man and the development of his faculties.* W. / R. Chambers.

Rasmussen, Carl Edward, Christopher KI Williams, et al. 2006. *Gaussian processes for machine learning.* Vol. 1. Springer.

Redmon, Joseph, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. "You only look once: Unified, real-time object detection." In *Proceedings of the IEEE conference on computer vision and pattern recognition,* 779–788.

Rempe, Davis, Tolga Birdal, Aaron Hertzmann, Jimei Yang, Srinath Sridhar, and Leonidas J Guibas. 2021. "Humor: 3d human motion model for robust pose estimation." In *Proceedings of the IEEE/CVF international conference on computer vision,* 11488–11499.

Rempe, Davis, Tolga Birdal, Yongheng Zhao, Zan Gojcic, Srinath Sridhar, and Leonidas J. Guibas. 2020. "CaSPR: Learning Canonical Spatiotemporal Point Cloud Representations." In *Advances in Neural Information Processing Systems (NeurIPS).*

Rombach, Robin, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. "High-resolution image synthesis with latent diffusion models." In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition,* 10684–10695.

Ryder, Tom, Andrew Golightly, A Stephen McGough, and Dennis Prangle. 2018. "Black-box variational inference for stochastic differential equations." In *International Conference on Machine Learning,* 4423–4432. PMLR.

Saemundsson, Steindor, Alexander Terenin, Katja Hofmann, and Marc Deisenroth. 2020. "Variational integrator networks for physically structured embeddings." In *International Conference on Artificial Intelligence and Statistics,* 3078–3087. PMLR.

Särkkä, Simo, and Arno Solin. 2019. *Applied stochastic differential equations.* Vol. 10. Cambridge University Press.

Solin, Arno, Ella Tamir, and Prakhar Verma. 2021. "Scalable inference in SDEs by direct matching of the Fokker–Planck–Kolmogorov equation." *Advances in Neural Information Processing Systems* 34:417–429.

Song, Yang, and Stefano Ermon. 2019. "Generative modeling by estimating gradients of the data distribution." *Advances in neural information processing systems* 32.

Spong, Mark W, and Mathukumalli Vidyasagar. 2008. *Robot dynamics and control.* John Wiley & Sons.

Theodorou, Evangelos A. 2015. "Nonlinear stochastic control and information theoretic dualities: Connections, interdependencies and thermodynamic interpretations." *Entropy* 17 (5): 3352–3375.

Todorov, Emanuel, Tom Erez, and Yuval Tassa. 2012. "Mujoco: A physics engine for model-based control." In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems,* 5026–5033. IEEE.

Tong, Anh, Thanh Nguyen-Tang, Toan Tran, and Jaesik Choi. 2022. "Learning Fractional White Noises in Neural Stochastic Differential Equations." In *Advances in Neural Information Processing Systems.*

Tong, David. 2005. "Classical dynamics." *Wilberforce Road.*

Toth, Peter, Danilo J Rezende, Andrew Jaegle, Sébastien Racanière, Aleksandar Botev, and Irina Higgins. 2020. "Hamiltonian Generative Networks." In *International Conference on Learning Representations.*

Tunyasuvunakool, Saran, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. 2020. "dm_control: Software and tasks for continuous control." *Software Impacts* 6:100022. ISSN: 2665-9638. https://doi.org/https://doi.org/10.1016/j.simpa.2020.100022. https://www.sciencedirect.com/science/article/pii/S2665963820300099.

Tzen, Belinda, and Maxim Raginsky. 2019. "Neural stochastic differential equations: Deep latent gaussian models in the diffusion limit." *arXiv preprint arXiv:1905.09883.*

Vecerik, Mel, Jean-Baptiste Regli, Oleg Sushkov, David Barker, Rugile Pevceviciute, Thomas Rothörl, Raia Hadsell, Lourdes Agapito, and Jonathan Scholz. 2021. "S3K: Self-Supervised Semantic Keypoints for Robotic Manipulation via Multi-View Consistency." In *Conference on Robot Learning,* 449–460. PMLR.

Virtanen, Pauli, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, et al. 2020. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python." *Nature Methods* 17:261–272. https://doi.org/10.1038/s41592-019-0686-2.

Yang, Ling, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. 2023. "Diffusion models: A comprehensive survey of methods and applications." *ACM Computing Surveys* 56 (4): 1–39.

Yang, Luxuan, Ting Gao, Yubin Lu, Jinqiao Duan, and Tao Liu. 2023. "Neural network stochastic differential equation models with applications to financial data forecasting." *Applied Mathematical Modelling* 115:279–299.

Yang, Ruihan, Prakhar Srivastava, and Stephan Mandt. 2022. "Diffusion probabilistic modeling for video generation." *arXiv preprint arXiv:2203.09481*.

Yu, Zhengdi, Shaoli Huang, Yongkang Cheng, and Tolga Birdal. 2023. "SignAvatars: A Large-scale 3D Sign Language Holistic Motion Dataset and Benchmark." *arXiv preprint arXiv:2310.20436*.

Zhang, Xiao, Wei Wei, Zhen Zhang, Lei Zhang, and Wei Li. 2023. "Milstein-driven neural stochastic differential equation model with uncertainty estimates." *Pattern Recognition Letters*.

Zheng, Ce, Wenhan Wu, Taojiannan Yang, Sijie Zhu, Chen Chen, Ruixu Liu, Ju Shen, Nasser Kehtarnavaz, and Mubarak Shah. 2020. "Deep learning-based human pose estimation: A survey." *arXiv preprint arXiv:2012.13392*.

Zhong, Yaofeng Desmond, Biswadip Dey, and Amit Chakraborty. 2020. "Dissipative SymODEN: Encoding Hamiltonian Dynamics with Dissipation and Control into Deep Learning." In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*.

Zhong, Yaofeng Desmond, and Naomi Leonard. 2020. "Unsupervised learning of lagrangian dynamics from images for prediction and control." *Advances in Neural Information Processing Systems* 33.

Zhou, Xingyi, Dequan Wang, and Philipp Krähenbühl. 2019. "Objects as points." *arXiv preprint arXiv:1904.07850*.

A realization of two-dimensional fractional Brownian motion with Hurst index 0.8. The long-range, spatial correlations resemble images of clouds.