

IMPERIAL COLLEGE LONDON
DEPARTMENT OF COMPUTING

**Probabilistic Models for Data Efficient
Reinforcement Learning**

Sanket Kamthe

Supervisor: Marc Deisenroth

This dissertation is submitted for the degree of
Doctor of Philosophy

October 25, 2021

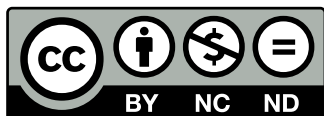
Declaration of Originality

I hereby declare that this thesis and the work to which it refers are the results of my own unaided work except where reviewed by my supervisor and anonymous reviewers. Any ideas, data, images or text resulting from the work of others (whether published or unpublished) are fully identified as such within the work and attributed to their originator in the text and bibliography. This thesis has not been submitted in whole or in part for any other academic degree or professional qualification.

Sanket Kamthe
October 25, 2021

Copyright Notice

The copyright of this thesis rests with the author. Unless otherwise indicated, its contents are licensed under a Creative Commons Attribution-Non Commercial 4.0 International Licence (CC BY-NC-ND). Under this licence, you may copy and redistribute the material in any medium or format. This is on the condition that: you credit the author and do not use it, or any derivative works, for a commercial purpose. When reusing or sharing this work, ensure you make the licence terms clear to others by naming the licence and linking to the licence text. Where a work has been adapted, you should indicate that the work has been changed and describe those changes. Please seek permission from the copyright holder for uses of this work that are not included in this licence or permitted under UK Copyright Law.



Acknowledgements

First of all, I would like to thank my supervisor Marc Deisenroth for his unwavering support over last seven years starting from my Masters at TU Darmstadt, Germany. He has gone above and beyond as a supervisor/mentor and I am forever indebted for his generosity not only in supporting me academically but also in personal life. This thesis constitutes three almost independent research themes and it is a testament to his support and encouragement for any research ideas I wished to pursue.

I would like to thank the incredible support staff, Amani and Hassan, at Imperial's Department of Computing, they worked tirelessly to ensure I was able to focus on my studies despite enormous strain put on by the 'Hostile Environment' approach of the UK Govt.

I was fortunate to start with great bunch of cohort at the HiPEDS CDT, our lunch time discussions and late evening slogs before deadlines are some of the most cherish-able moments of my life at Imperial. In the SML research group, I learned a great deal initially from Hugh Salimbeni and later from James Wilson. This thesis is uses a modified version of the $\text{\LaTeX}$ template shared by Hugh Salimbeni. I would also like to thank Yufei Wu who was a great collaborator and now an even better friend.

I spent the last year of my PhD at JP Morgan AI research London. I would like to thank my manager, Samuel Assefa, for supporting me to continue work on independent research projects. The last section of the thesis was almost entirely completed whilst at JP Morgan AI Research.

I would like to thank Imperial's Department of Computing for covering the budget shortfalls from the CDT scholarship for overseas students. Work during the thesis was also supported by the Google EMEA Scholarship and JP Morgan AI Research PhD Fellowship.

I would like to thank my parents for their support end encouragement to pursue higher studies at great personal financial and emotional burden to them.

Last but not the least, I would like to thank my wife, Marshneel, for her continuous and unwavering support through countless weekends and inhumane PhD schedule over the last three years. I may have never reached this stage without her. . . .

Abstract

Trial-and-error based reinforcement learning (RL) has seen rapid advancements in recent times, especially with the advent of deep neural networks. However, the standard deep learning methods often overlook the progress made in control theory by treating systems as black-box. We propose a model-based RL framework based on probabilistic Model Predictive Control (MPC). In particular, we propose to learn a probabilistic transition model using Gaussian Processes (GPs) to incorporate model uncertainty into long-term predictions, thereby, reducing the impact of model errors. We provide theoretical guarantees for first-order optimality in the GP-based transition models with deterministic approximate inference for long-term planning. We demonstrate that our approach not only achieves the state-of-the-art data efficiency, but also is a principled way for RL in constrained environments.

When the true state of the dynamical system cannot be fully observed the standard model based methods cannot be directly applied. For these systems an additional step of state estimation is needed. We propose distributed message passing for state estimation in non-linear dynamical systems. In particular, we propose to use expectation propagation (EP) to iteratively refine the state estimate, i.e., the Gaussian posterior distribution on the latent state. We show two things: (a) Classical Rauch-Tung-Striebel (RTS) smoothers, such as the extended Kalman smoother (EKS) or the unscented Kalman smoother (UKS), are special cases of our message passing scheme; (b) running the message passing scheme more than once can lead to significant improvements over the classical RTS smoothers. We show the explicit connection between message passing with EP and well-known RTS smoothers and provide a practical implementation of the suggested algorithm. Furthermore, we address convergence issues of EP by generalising this framework to damped updates and the consideration of general α -divergences.

Probabilistic models can also be used to generate synthetic data. In model based RL we use 'synthetic' data as a proxy to real environments and in order to achieve high data efficiency. The ability to generate high-fidelity synthetic data is crucial when available (real) data is limited as in RL or where privacy and data protection standards allow only for limited use of the given data, e.g., in medical and financial data-sets. Current state-of-the-art methods for synthetic data generation are based on generative models, such as Generative Adversarial Networks (GANs). Even though GANs have achieved

remarkable results in synthetic data generation, they are often challenging to interpret. Furthermore, GAN-based methods can suffer when used with mixed real and categorical variables. Moreover, the loss function (discriminator loss) design itself is problem specific, i.e., the generative model may not be useful for tasks it was not explicitly trained for. In this paper, we propose to use a probabilistic model as a synthetic data generator. Learning the probabilistic model for the data is equivalent to estimating the density of the data. Based on the copula theory, we divide the density estimation task into two parts, i.e., estimating univariate marginals and estimating the multivariate copula density over the univariate marginals. We use normalising flows to learn both the copula density and univariate marginals. We benchmark our method on both simulated and real data-sets in terms of density estimation as well as the ability to generate high-fidelity synthetic data.

Contents

Declaration of Originality	1
Copyright Notice	2
Acknowledgements	3
Abstract	4
List of Figures	10
List of Tables	10
1 Introduction	11
1.1 Motivation	12
1.2 Thesis Outline	13
1.3 Contributions	14
I Optimal Control with Gaussian Processes Dynamics	16
2 Background: Model Predictive Control for Reinforcement Learning	17
2.1 Building Blocks of MPC	18
2.1.1 Prediction Model	18
2.1.2 Objective Function	19
2.1.3 Optimal Control Law	20
2.2 Optimal Control	22
2.2.1 Hamilton-Jacobi-Bellman (HJB) Equation	23
2.2.2 Solving HJB Equations	24
2.3 Pontryagin's Maximum Principle	24
2.3.1 Statement of the Principle	25
2.3.2 Geometric Interpretation	25
2.3.3 Summary	31
3 Optimal Control for Gaussian Process Dynamical systems	34

3.1	Gaussian Process Predictive Model	35
3.1.1	Probabilistic Transition Model	35
3.1.2	Long-term Predictions	36
3.2	Objective Function	37
3.3	Probabilistic Maximum Principle for GP Dynamics	37
3.3.1	Existence/Uniqueness of a Local Solution	38
3.3.2	Pontryagin's Maximum Principle for GP Dynamics	39
3.3.3	Efficient Gradient Computation	41
3.4	Related Work	42
3.5	Experimental Results	43
3.5.1	Under-actuated Cart-Pole Swing-Up	43
3.5.2	Fully-actuated Double-Pendulum	43
3.5.3	Trials	44
3.5.4	Data Efficiency	44
3.5.5	State Constraints	46
3.6	Conclusion and Discussion	47
3.6.1	Future Work	47
II	State Space Models	49
4	State Estimation in Non-linear systems	50
4.1	Introduction	50
4.1.1	Contribution	53
4.1.2	Overview	53
4.2	Gaussian Filtering and Smoothing	53
4.2.1	Optimal Filtering and Smoothing	54
4.2.2	Implicit Linearisation in Non-linear Gaussian Filters and Smoothers	55
4.2.3	RTS Smoothing Applied to a Non-linear Benchmark	57
4.3	Expectation Propagation	57
4.3.1	EP Updates for Generalised Gaussian Filtering	60
4.3.2	Implicit Linearisation Requires Explicit Consideration	62
4.3.3	Derivative-based EP updates and Kalman smoother	63
4.3.4	Backward Message	66
4.3.5	Applying approximate EP to a non-linear benchmark	69
4.4	Improving EP by damping and Alpha-Divergence	70
4.4.1	Damped EP	70
4.4.2	Alpha Divergence	71
4.5	Results	73

4.5.1	Non-Linear Growth Model	73
4.5.2	Bearings Only Tracking	78
4.6	Conclusion and Discussion	79
III	Synthetic Data Generation	81
5	Copula Flows for Synthetic Data	82
5.1	Introduction	82
5.1.1	Related Literature	84
5.2	Background: Copula Theory and Normalising Flow	86
5.2.1	Notation	86
5.2.2	Copula Models	86
5.2.3	Generative Sampling	86
5.2.4	Normalising Flows	88
5.2.5	Flow Likelihood	88
5.3	Marginal Flows	90
5.3.1	Marginal Splines	90
5.3.2	Marginal Fitting: Discrete Data	90
5.4	Copula Flow	92
5.4.1	Splines for Copula	94
5.4.2	Copula for Discrete Marginals	95
5.5	Copula Flow: An Universal Density Approximator	95
5.5.1	Convergence of the Marginal Flow	96
5.5.2	Convergence of the Copula Flow	97
5.6	Experiments	97
5.6.1	Mixed Variable Copula Learning	98
5.6.2	Density Estimation	99
5.6.3	Synthetic Data	100
5.7	Conclusion	101
5.8	Future Work	101
IV	Conclusion and Future Work	102
6	Conclusion	103
A	Monotonic Rational Quadratic Transforms	105
A.1	Parametrisation of spline	105
B	Additional Plots	107

B.1 Bivariate Copulas	107
C Software Details	108
References	110

List of Figures

1.1 RL introduction	11
2.1 Tent K for the reachable set Ω with apex at x^*	28
2.2 Hamiltonian maximisation condition	30
2.3 Ad joint variables and reachable sets	31
2.4 Co-state trajectory	32
3.1 State constraints for RL benchmarks	43
3.2 Performance of RL algorithms	45
3.3 Performance of RL algorithms with state-space constraints	46
4.1 Expectation Propagation vs Kalman Filtering and Smoothing	56
4.2 Damped Expectation Propagation	71
4.3 Damped Power Expectation Propagation	73
4.4 Average Performance of Damped Power EP	74
4.5 Hyper-parameter Ablation study of damped power EP	76
4.6 RMSE for power and damping- Uniform non-linear growth model	77
4.7 Negative Log-Likelihood (NLL) for power and damping- Uniform non-linear growth model	77
4.8 RMSE for power and damping- Bearings only Tracking	77
4.9 Negative Log-Likelihood (NLL) for power and damping- Bearings only Tracking	78
5.1 Synthetic data schematic	83
5.2 Copula and Marginal for two rings data	84
5.3 Schematic for Copula Flow data generator	87
5.4 Copula flow generative model	88
5.5 Illustration of continuous Marginal learning	91
5.6 Illustration of discrete Marginal learning	92
5.7 Discrete marginal flow	93

5.8	Schematic for Marginal Learning	94
5.9	Distributional transform for statistical probability integral transform . .	96
5.10	Copula flow for modelling mixed density	98
5.11	Inverse flows for mixed density	99
A.1	Monotonic Rational Quadratic Splines	105
B.1	Learning bivariate Copulas using Copula Flow model. Each bi-variate density is learned by the same flow	107

List of Tables

3.1	State constraint violations	47
5.1	Test log-likelihoods (nats) on density estimation benchmarks	100
5.2	Kolmogorov-Smirnov KS test for marginals	100
5.3	Performance of the copula flow models for synthetic data generation. .	101
C.1	Hyperparameters for the density estimation results	109

Introduction

Reinforcement learning (RL) is a principled mathematical framework for experience-based autonomous learning of control policies. The trial-and-error learning process is one of the most distinguishing features of RL (Sutton and Barto 1998). Despite many recent advances in RL (Mnih et al. 2015; Silver et al. 2016; Yahya et al. 2016), a main limitation of current RL algorithms is their data inefficiency, i.e., the required number of interactions with the environment is impractically high. For example, many RL approaches in problems with low-dimensional state spaces and fairly benign dynamics require thousands of trials to learn (Mnih et al. 2015; Schulman, Levine, et al. 2015; Schulman, Wolski, et al. 2017). This *data inefficiency* makes learning in real control/robotic systems without task-specific priors impractical and prohibits RL approaches in more challenging scenarios.

Reinforcement learning can be described as a two step process outlined in the figure 1.1. The agent interacts with the environment to obtain a reward and the current state of the system. RL problem is then to optimise the agent behaviour in a way to

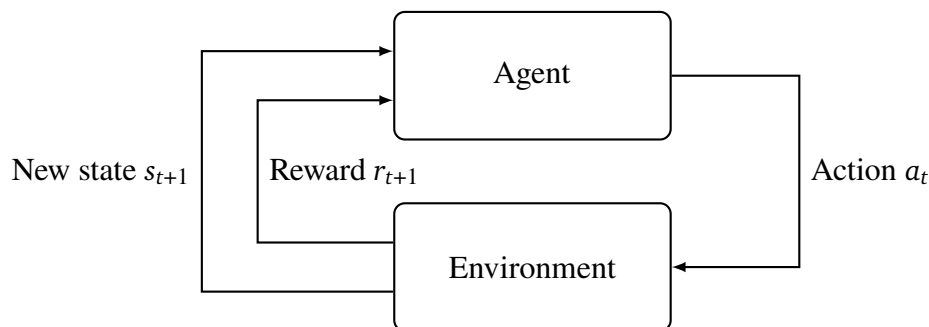


Figure 1.1: Basic reinforcement learning model diagram (Sutton and Barto 1998)

maximise the cumulative reward obtained by the agent. The efficiency gains can be achieved by changing how the agent uses the data obtained while interacting with the environment. In on-policy methods for example, we collect the data and optimise the agent behaviour to maximise the reward based on the data. We then immediately apply this policy to the environment and get new data and repeat this loop until we achieve desired accuracy.

1.1 Motivation

As a first step one may try to use the current data to estimate the rewards for a new optimised policy. This can help to increase the efficiency (Sutton and Barto 1998). However, the transitions observed in the past may not be feasible under the new policy (Kakade and Langford 2002). We can mitigate this by forcing the current policy to be close to the old policy. We can bound the expected error by limiting the change in the policy parameters. The limit is enforced via coupling in (Kakade and Langford 2002), a trust region in (Schulman, Levine, et al. 2015) and a heuristic to clip large changes in (Schulman, Wolski, et al. 2017).

Alternatively, one can increase the efficiency by using all the data available including the data from all the previous interactions (Haarnoja et al. 2018). This gives the algorithm an opportunity to exploit the additional information compared to the methods that rely on just the current data. However, for efficiency, this needs the algorithm to explore in the regions with very limited or no data resulting in large error in policy evaluations (Fujimoto, Meger, and Precup 2019; Agarwal, Schuurmans, and Norouzi 2020).

Another promising way to increase the data efficiency of RL without inserting task-specific prior knowledge is to learn models of the underlying system dynamics (Ng and Jordan 2000; R. S. Sutton 1990). When a good model is available, the exploration problem can be addressed by using models as a faithful proxy for the real environment, i.e., good policies can be obtained from the model without additional interactions with the real system.

However, modelling the underlying transition dynamics, especially non-linear, accurately is challenging and inevitably leads to model errors. To account for these model errors, one can use probabilistic models (Schneider 1997; Marc P Deisenroth and Carl E Rasmussen 2011). By explicitly taking model uncertainty into account, the number of interactions with the real system can be substantially reduced. For example, in (Marc P Deisenroth and Carl E Rasmussen 2011; Y. Pan and Theodorou 2014; Deisenroth, Fox, and Rasmussen 2015; Cutler and How 2015), the authors use Gaussian processes (GPs) to model the dynamics of the underlying system.

Standard model based methods in recent years have used models as a proxy for training a RL policy (G. Williams, Aldrich, and Theodorou 2015; Hafner et al. 2018;

Chua et al. 2018; Janner et al. 2019). These methods typically use models to generate pseudo-data on which policies are trained. For Gaussian process based dynamics, PILCO (Deisenroth, Fox, and Rasmussen 2015) for example, uses a differentiable model to train a policy via gradient decent. The main bottleneck for these methods is that very long term planning via the model often results in accumulation of model errors. Intuitively, this corresponds to the model queried around the unobserved values. The longer the horizon, the larger the probability of model error which affects the result.

To mitigate difficulties with long training horizons we propose to use Model Predictive Control (MPC) with significantly shorter planning horizons than those for model based RL problem. Unlike standard MPC, we do not have access to the true model of the system or good mathematical proxy for it available. We instead learn the model from the agent environment interactions. Moreover, unlike RL we do not directly learn a global policy function that maps given state to action, instead we solve an optimisation problem that takes in the current state of the system and predicts the optimal action. In the following we establish the exact problem formulation and also introduce background to the optimal control methods used in this work.

An additional advantage of the MPC is that the method allows to easily incorporate state as well as control constraints (David Q Mayne 2016). Control constraints are crucial to ensure that the agent does not execute unsafe action, whereas state constraints can be used to limit the robot movements to avoid an obstacles, for example. The dynamics model is a crucial part of the MPC methods. Crucial components of the the MPC-RL are model learning and generating roll-outs from the models that can then be used to improve agents performance.

1.2 Thesis Outline

In this work we address challenges associated with using data efficiently in reinforcement learning. We propose to use probabilistic models and MPC techniques to improve efficiency of RL. With learned dynamics models we can augment the data obtained from the agent environment interaction by simulating our dynamics models. For some robotics applications,

- We may not have full state information,
- State variables may be a combination of continuous and discrete variables.

In this thesis we tackle first challenge by developing iterative Kalman smoothing algorithm that can outperform standard Kalman Filter-Smoother. For the second challenge we build a probabilistic model with normalising flows and copula theory. Combined together the techniques discussed in this thesis can be used to solve a complex robotics tasks.

As an example consider training a limb prosthetic built with soft robotic components. To enable practical usage of this limb, we need:

1. **High data efficiency**, to avoid prosthetic user fatigue and mental trauma of repeated failures – We build data efficient RL models that can also deal with noisy readings.
2. **Unknown true state**, for soft robotic contacts the true state of the system may be impossible to determine – We develop an iterative smoother that can yield an powerful state estimator
3. **Discrete and continuous states spaces**, the objectives and the parameters can be discrete, type of grasp or motion as well as continuous, e.g., arm tendons – We build a probabilistic model capable of learning this joint distribution of state space and can be used to augment training data needed for RL control of the prosthetic.

The methods described above are state of the art results at the time of proposal for solving the above mentioned challenges. We divide detailed discussion of these methods in 3 parts.

Part I of the thesis deals with building data efficient RL algorithms using learned dynamics models. The MPC based RL method is described in detail along with the background on optimal control theory with focus on the Pontryagin’s Maximum Principle (PMP) and its geometric interpretation. We then extend the PMP to the Gaussian process dynamical models to develop a data efficient constrained RL algorithm.

In Part I we assume that the system is fully observable, however, often the true systems are not fully observable. For partially observable systems, model learning also known as system identification involves estimating true system states. The Kalman filter is both efficient and optimal for linear dynamical systems (Kalman 1960). For non-linear systems we need approximation or statistical linearisation to use the Kalman filter (Julier, Uhlmann, and Durrant-Whyte 2000; Lefebvre, Bruyninckx, and De Schutter 2004). In Part II of the thesis we propose an iterative non-linear Kalman filter based on the Expectation Propagation algorithm (T. P. Minka 2001a; Qi and T. Minka 2003). Part III of the thesis explores the application of probabilistic models for synthetic data generation. Synthetic data can also be used to augment real data to improve data efficiency of machine learning models (Patki, Wedge, and Veeramachaneni 2016). It can also alleviate data privacy concerns (Beaulieu-Jones et al. 2019).

1.3 Contributions

- **GP as Dynamics Model:** We show that the Gaussian process dynamics models used in PILCO can be interpreted as an deterministic Ordinary Differential (Difference) Equation (ODE). Using this interpretation we apply Picard–Lindelöf theorem to show the existence and uniqueness of the solutions to the GP dynamics ODE with moment matching approximation. The uniqueness of the evolved dynamics allows us to apply Pontryagin’s Maximum Principle (PMP) to Gaussian process dynamics. We propose probabilistic PMP for the Gaussian process dynamical models. Unlike

traditional methods the PMP can be used for constrained models. We show that the proposed method is able to not only get state of the data efficiency but also able to handle constraints in a principled manner.

- **Expectation Propagation for Non-Linear Dynamical Systems:**

1. We show that the standard Kalman-filter approximations can be used to perform numerically stable *approximate* Expectation Propagation (EP) for non-linear systems. We explicitly show the equivalence of the Gaussian RTS filter-smoother and the EP, i.e, derivative free Kalman filter like updates and derivative based EP updates are equivalent.
2. We further generalise the EP based scheme to develop Power EP, which can be used to interpolate between EP and variational Bayes (VB) for non-linear state-space models. Furthermore, we devise a scheme to use stable Kalman-filter based updates to overcome numerical instabilities often associated with approximate Power EP.

- **Copula Flow for Synthetic Data:**¹ Pervasive use of machine learning in day to day life poses a significant challenge to privacy in recent times. We propose to use synthetic data generator using Copula theory and normalising flows to build a probabilistic CopulaFlow model. A formidable challenge with Copula based models is the intractability of building them for mixed variables, i.e., continuous and discrete variables. We show that a distributional transform can be learned and used to build Copula based models with mixed variable types. We further prove that this model is an universal density estimator. Our approach is based on the normalising flows to estimate copulas. We show that our model is flexible enough to learn complex relationships and unlike traditional methods, does not rely on explicit learning of the graph or parametric bivariate copulas.

¹Patent Pending, filed at US patents office in December 2020

Part I

Optimal Control with Gaussian Processes Dynamics

Background: Model Predictive Control for Reinforcement Learning

Model predictive Control (MPC) is popular in process industries due to its simplicity and ability to handle large complex systems (Qin and Badgwell 2003). Intuitively MPC is simple to understand. This may be attributed to the way our brain works (Schacter, Addis, and Buckner 2007). Our brains simulate future outcomes of our actions. The future simulation is built on the memories from the past, in fact we use the same brain regions to plan future as we use to build our memories (Schacter, Addis, and Buckner 2007). In MPC as well, we use a model to predict/simulate the future state evolution of the system based on possible actions. Using this predicted state evolution we can compute the estimated cost of the future actions or control sequences. We can then find the optimal control/action sequences to minimise the future cost. All MPC based systems possess the following characteristics (Camacho and Bordons 2004):

- *Explicitly use model to predict the output at future time instances.* The length or the number of time steps into the future is called as the MPC horizon.
- *Calculate optimal control sequence* based on the optimisation criteria, e.g. objective function and constraints.
- *Receding Strategy*, that is at each instant we apply the first control signal and the horizon is shifted forward in the future. Then the optimal sequence is recalculated for the next horizon and this step is repeated.

MPC algorithms vary based on the dynamics model or the cost function to be minimised. For example the Linear MPC methods have linear system dynamics models. One can obtain a closed formed MPC solution for linear systems with quadratic costs (Brian D. O. Anderson and John B. Moore 1990).

2.1 Building Blocks of MPC

In standard MPC framework the model is known a-priori. For *MPC – RL* application, we do not have any a-priori information of the system dynamics. This presents us with two immediate choices: what type of model do we learn? How do we deal with the model errors? Model errors are dependent on the choice of model. In light of these design questions we establish building blocks common for all MPC methods. We can construct different MPC methods based on different choices in each of the following elements (Camacho and Bordons 2004)

1. **Prediction Model**
2. **Objective Function**
3. **Optimal Control law.**

Traditionally, the model and the objective function are well defined for the problems. Main task of MPC algorithms is then to efficiently obtain the optimal control law for the given specifications (Rawlings, D. Mayne, and Moritz Diehl 2017). Sometimes additional restrictions such as stability (David Q. Mayne et al. 2000) and robustness (David Q Mayne 2016) are also considered while designing MPC algorithms. In the following we describe each of the elements for MPC in detail.

2.1.1 Prediction Model

The prediction model is a crucial element of any MPC method. The quality of model significantly impacts the performance of the MPC algorithm. The model not only needs to closely mimic the true system but also needs to have a mechanism to evaluate future predictions. Furthermore, the future prediction mechanism should be amenable to further analysis. Traditional MPC methods rely on linear dynamical models as they are easy to interpret and analyse (Qin and Badgwell 2003). The industrial success of MPC relies on this simplicity, as linear models significantly reduce maintenance and training costs (Qin and Badgwell 2003; David Q. Mayne et al. 2000).

In this work we consider more generic approach by using a non-linear dynamical system as a prediction model. A non-linear system can be described by the difference equation:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) \quad f : \mathcal{X} \times \mathcal{U} \rightarrow \mathcal{X}, \quad (2.1)$$

where the $\mathbf{x}_t \in \mathcal{X} \subset \mathbb{R}^n$ is the current state, $\mathbf{u}_t \in \mathcal{U} \subset \mathbb{R}^m$ is the current control and $\mathbf{x}_{t+1}$ is the successor state. The sets $\mathcal{X}$ and $\mathcal{U}$ are assumed to be closed. We denote the control sequence $\mathbf{u}_0, \mathbf{u}_1, \dots, \mathbf{u}_T$ by $\mathbf{u}_{0:T-1}$. The solution of the Equation 2.1 at time t when the initial state at time zero is $\mathbf{x}$ and the control sequence is $\mathbf{u}_{0:T-1}$ is denoted by $\phi(t; \mathbf{x}, \mathbf{u}_{0:T-1})$. The solution at time t depends only on the control sequence $\mathbf{u}_0, \mathbf{u}_1, \dots, \mathbf{u}_{T-1}$. Without loss of generality we assume the continuity of system solutions, i.e.,

Proposition 2.1. Continuity of system solution

Suppose the function $f(\cdot)$ is continuous. Then for each integer $t \in \mathbb{N}$, the function $(\mathbf{x}, \mathbf{u}_{0:T-1}) \rightarrow \phi(t; \mathbf{x}, \mathbf{u}_{0:T-1})$ is continuous.

A linear dynamical system is just a special case of the non-linear version with $f(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k$ for a linear system. Unlike linear systems, it is far more challenging to learn a predictive non-linear model. In linear systems if know that input $\mathbf{u}_1$ and $\mathbf{u}_2$ produce outputs $\mathbf{x}_1$ and $\mathbf{x}_2$ then we can predict the output at combination of inputs using principle of superposition. A non-linear system does not obey the superposition principle. Hence, we need to query the function multiple times to learn the system model.

In *MPC – RL* we learn this function $\hat{f} : \mathcal{X} \times \mathcal{U} \rightarrow \mathcal{X}$ from the data collected whilst interacting with the environment. In the first episode we apply random control signals to collect trajectories of episode length N with states $\mathbf{X} = [\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_N]$, controls $\mathbf{U} = [\mathbf{u}_0, \mathbf{u}_1, \dots, \mathbf{u}_N]$. We can then learn the predictive model by any regression method by minimising loss $|\hat{f} - f|$. Some examples of possible regression methods include deep neural network, Gaussian Processes, etc.

When using learning dynamics model for MPC, there are two practical points to be considered. First, the system noise, it is an inherent part of the true systems and modelling error, second, the modelling error, it is function of the modelling tool employed but it is always non-zero in practice. To account for modelling errors and system noise we learn a probabilistic model based on Gaussian process regression. We detail this model learning and the Gaussian process based probabilistic MPC in the Chapter 3. In the remainder of this chapter we assume the model to be deterministic as this significantly eases notation and probabilistic model is discussed in detail in the next Chapter 3.

2.1.2 Objective Function

The next important building block for MPC is the objective (cost) function. The cost function is used as a performance metric for obtaining control law. The general structure of the objective function is that predicted states or system output $\mathbf{x}_{0:T-1}$ over the horizon T should follow a reference trajectory $\mathbf{x}_{ref}$ over the horizon T whilst minimising the control effort $\mathbf{u}_{0:T-1}$. Often the reference trajectory is just the target or the goal state. In process control applications, e.g., one often penalises the change in control instead of the magnitude of the control effort (Qin and Badgwell 2003; Camacho and Bordons 2004).

A system in state $\mathbf{x}$ when acted upon by control $\mathbf{u}$ incurs a cost of $\ell(\mathbf{x}, \mathbf{u})$. For practical considerations, we only consider finite time horizon T over which we seek to optimise the cost. The optimal control can be expressed as minimisation of the cumulative

cost,

$$\ell_{0:T-1} = \sum_{k=0}^{T-1} \ell(x_k, u_k) \quad (2.2)$$

with respect to the variables $(\mathbf{x}_{0:T-1}, \mathbf{u}_{0:T-1})$ subject to constrain that the variables satisfy the difference Equation (2.1), the initial condition $x_0 = x$ and state and control constraints.

Assumption 2.1. Continuity of system and cost

The functions $f : \mathcal{X} \times \mathcal{U} \rightarrow \mathcal{X}$ and $\ell : \mathcal{X} \times \mathcal{U} \rightarrow \mathbb{R}_{\geq 0}$ are continuous. In addition without loss of generalisation we assume $f(0, 0) = 0$ and $\ell(0, 0) = 0$.

In reinforcement learning context, the objective function is called as reward. We treat negative of the cost function as a reward function, hence typical control methods minimising cost can be used in RL with an assumption that the reward is the negative cost. Minimising cost, a performance metric in optimal control theory, is in principle same as maximising the reward, a performance metric in RL. Choice and structure of the objective function significantly influences the control law optimisation.

2.1.3 Optimal Control Law

We obtain the control law by minimising the objective function. We use the predictive model to compute future cost based on the control signal and the objective function. In the standard optimal control problem the objective is to find state and control sequence $\{\mathbf{x}_{1:T}, \mathbf{u}_{1:T}\}$ that steers dynamical system defined by Equation (2.1), from initial state $\{\mathbf{x}_0\}$ to the final state $\{\mathbf{x}_T\}$ by traversing a trajectory $\{\mathbf{x}_{0:T}, \mathbf{u}_{0:T}\}$ whilst minimising the cost function,

$$J(\mathbf{x}, \mathbf{u}) = \Phi(\mathbf{x}_T, \mathbf{u}_T) + \sum_{k=0}^{T-1} \ell(\mathbf{x}_k, \mathbf{u}_k). \quad (2.3)$$

The term $\Phi(\cdot)$ is called the terminal cost or the *Mayer cost*. The term $\ell(\cdot)$ is called the running cost or the *Lagrange cost*. When we have both terms as in Equation (2.3) it is called the *Bolza form*. When the terminal cost is zero $\Phi(\cdot) = 0$ the cost function is called of the Lagrange form. When we have only terminal cost J the problem is of the Mayer form. Both forms can be transformed interchangeably via suitable change of variables (Bliss 1946).

Definition 2.1. Optimal Control Problem

$$\begin{aligned}
\mathbf{u}^* = \arg \min_{\mathbf{u} \in \mathcal{U}} \quad & \mathcal{J}(\mathbf{x}, \mathbf{u}) := \Phi(\mathbf{x}_T) + \sum_{k=0}^{T-1} \ell(\mathbf{x}_k, \mathbf{u}_k) \\
\text{subject to} \quad & \mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) \quad (\text{dynamics constraint}), \\
& g_T(\mathbf{x}_T) = 0 \quad (\text{terminal constraint}), \\
& g(\mathbf{x}) \leq 0 \quad (\text{state constraint}), \\
& m(\mathbf{u}) = 0 \quad (\text{control constraint})
\end{aligned} \tag{2.4}$$

For simplicity we assume that the state sequence satisfies the difference equation $\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k)$, thereby eliminating $\mathbf{x}$ as control variable. The cost to be optimised in this case is now function of control sequence alone¹. The solution of the problem is an optimal sequence $\mathbf{u}_{0:T-1}^*$. In MPC, the first element of the sequence u_0^* is applied to the system. Then we obtain the next state of the system and repeat the optimisation procedure with the obtained state set as $\mathbf{x}_0$. We assume knowledge of the full state system to facilitate the analysis. In practice, however, we often have to employ state estimation techniques as discussed in Chapter 4.

As we only apply the first signal from the control sequence to the actual system. We can write a control law obtained from this procedure as,

$$\begin{aligned}
\mathbf{u} &= \mathbf{u}_{*0} \text{ of } \mathbf{u}_{0:T-1}^* \\
&= \mathbf{u}_0 \text{ of } \arg \min_{\mathbf{u}} \sum_{k=0}^{T-1} \ell(\mathbf{x}_k, \mathbf{u}_k) \\
&= \mathbf{u}_0 \text{ of } \mathcal{J}(\mathbf{x}) \\
\mathbf{u} &= \kappa(\mathbf{x})
\end{aligned} \tag{2.5}$$

While we can use the above control law to obtain the optimal control signal for all global states, this would be an intractable problem for even a moderate state space, especially for a non-linear systems. In linear systems with quadratic costs and no constraints on the control we can indeed obtain a global optimal solution in the form of linear quadratic regulator. For non-linear systems we have some techniques [rephrase]. In the following we consider standard methods to solve the optimal control problem. First we consider dynamical programming which can be used to solve the type of discrete control problems. Then we consider classic Hamilton-Jacobi-Bellman HJB equation for solving continuous control problems.

¹This not a strictly correct assumption. However, for simulated dynamics using the system function f it is practically convenient

2.1.3.1 Dynamic Programming

Dynamic Programming is an algorithm to efficiently solve optimisation problems similar to the one described above in equation (2.3). It is based on the principle that the subset of the optimal state action sequence is also optimal. The principle is attributed to Bellman and is referred to as Bellman's optimality principle (Richard E. Bellman 1954);

- Principle of Optimality: An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.
- Principle of Optimality: Suppose the optimal solution for a problem passes through some intermediate point (x_1, t_1) , then the optimal solution to the same problem starting at (x_1, t_1) must be the continuation of the same path.

The principle implies that future optimal actions are independent of the past choices that resulted in the current state. So, to find the optimal path one can start at the end trace back the best state-action pairs leading up-to the current state. The optimal value function is defined as

$$\nu(x) = \text{minimal cost of completing the task starting from state } x.$$

An algorithm yields an optimal action $u = \pi(x) \in \mathcal{U}(x)$ for every state x . The mapping of control function to a given state x is called as *control policy*. Once we have an optimal policy $\pi^*(x)$ we can apply that to the starting state x_0 and transition one step at a time to x_T . In our terminology, this would mean we apply control $u_0 = \pi^*(x_0)$ at state x_0 reach $x_1 = f(x_0, u_0)$. An optimal control policy satisfies

$$\pi^*(x) = \arg \min_{u \in \mathcal{U}} [\ell(x, u) + \nu(f(x, u))] \quad (2.6)$$

Many actions in the set $\mathcal{U}(x)$ may satisfy the above equations. However, the optimal value function is always unique and defined as

$$\nu^*(x) = \min_{u \in \mathcal{U}} [\ell(x, u) + \nu(f(x, u))] \quad (2.7)$$

If we know the $\nu(f(x, u)) \forall u$ then we can directly compute $\pi^*(x)$ and $\nu(x)$ from Bellman equations (2.6)–(2.7). In acyclic graphs we can start from the desired state x_T by setting $\nu(x_T) = 0$ and perform a backward pass to obtain an optimal trajectory. Setting the value function $\nu(x_T) = 0$ is equivalent to saying the minimum cost of completing the task is 0, i.e., we have completed the task.

2.2 Optimal Control

While discrete control strategies use dynamic programming based algorithms to iteratively improve control objective. Optimal control theory and advances predate modern computers and sampled systems. Continuous control strategies use calculus of variations and take advantages of rich mathematics that comes with the field (Clarke

1990).

For continuous time systems we define real valued vectors, the state as $\mathbf{x} \in \mathbb{R}^{n_x}$ and the control as $\mathbf{u} \in \mathcal{U} \subset \mathbb{R}^{n_u}$. Consider the stochastic differential equation

$$d\mathbf{x} = f(\mathbf{x}, \mathbf{u})dt + F(\mathbf{x}, \mathbf{u})d\mathbf{w} \quad (2.8)$$

where $d\mathbf{w}$ is a n_x dimensional Brownian motion. We assume $F(\mathbf{x}, \mathbf{u}) = 0$ and write $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$ as our system model. For stochastic systems and a derivation with $F(\mathbf{x}, \mathbf{u}) \neq 0$ see (Todorov 2006). The Euler discretisation of $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$ is

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \Delta f(\mathbf{x}_t, \mathbf{u}_t)$$

where Δ is the time step and $\mathbf{x}_t = \mathbf{x}(t\Delta)$

To derive the Hamilton-Jacobi-Bellman Equation for the control system described by equation (2.8) we define a continuous time cost function, $\ell(\mathbf{x}, \mathbf{u}, t) \geq 0$, and a final cost $\Phi(\mathbf{x}) \geq 0$ is the cost associated with the final state $\mathbf{x}(t_f)$. The total cost for a given state-control trajectory $[\mathbf{x}(t), \mathbf{u}(t) : 0 \leq t \leq t_f]$ is given by

$$J(\mathbf{x}, \mathbf{u}) = \Phi(\mathbf{x}(t_f)) + \int_0^{t_f} \ell(\mathbf{x}(t), \mathbf{u}(t), t)dt \quad (2.9)$$

2.2.1 Hamilton-Jacobi-Bellman (HJB) Equation

We can now apply the dynamic programming principle to the equation (2.9) for the discrete time version of the system $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$. We have the discrete time difference equation

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \Delta f(\mathbf{x}_t, \mathbf{u}_t) \quad (2.10)$$

Applying Bellman equation (2.7) to the transition model given by equation (2.10) we get

$$\begin{aligned} \nu(\mathbf{x}, t) &= \min_{\mathbf{u}} \{ \Delta \ell(\mathbf{x}_t, \mathbf{u}_t) + \nu(\mathbf{x}_{t+1}) \} \\ &= \min_{\mathbf{u}} \{ \Delta \ell(\mathbf{x}_t, \mathbf{u}_t) + \nu(\mathbf{x}_t + \Delta f(\mathbf{x}_t, \mathbf{u}_t)) \} \end{aligned} \quad (2.11)$$

To evaluate above equation we use second order Taylor-series expansion to obtain

$$\nu(\mathbf{x}) = \min_{\mathbf{u}} \{ \Delta \ell(\mathbf{x}_t, \mathbf{u}_t) + \nu(\mathbf{x}_{t+1}) + \left(\frac{\partial \nu(\mathbf{x}_t)}{\partial \mathbf{x}_t} \right)^T \Delta f(\mathbf{x}_t, \mathbf{u}_t) + o(\Delta^2) \} \quad (2.12)$$

$$= \min_{\mathbf{u}} \{ \Delta \ell(\mathbf{x}_t, \mathbf{u}_t) + \nu(\mathbf{x}_{t+1}) + \lambda^T \Delta f(\mathbf{x}_t, \mathbf{u}_t) + o(\Delta^2) \} \quad (2.13)$$

where we use $\lambda = \frac{\partial \nu(\mathbf{x}_t)}{\partial \mathbf{x}_t}$ to simplify the notation. Moving $\nu(\mathbf{x}_{t+1})$ to left, as it does not depend on the arguments of min, and dividing the both sides by Δ ,

$$\frac{\nu(\mathbf{x}_t) - \nu(\mathbf{x}_{t+1})}{\Delta} = \min_{\mathbf{u}} \{ \ell(\mathbf{x}_t, \mathbf{u}_t) + \lambda^T \Delta f(\mathbf{x}_t, \mathbf{u}_t) \} \quad (2.14)$$

Taking the limit $\Delta \rightarrow 0$ and treating $\nu(\mathbf{x}_t)$ as a continuous function we obtain

$$\frac{\nu(\mathbf{x}_t) - \nu(\mathbf{x}_{t+1})}{\Delta} = -\frac{\partial}{\partial t} \nu = -\dot{\nu}$$

Definition 2.2. Optimal Value Function

For the terminal cost $\nu(\mathbf{x}_{t_f}) = \Phi(\mathbf{x}_{t_f})$ and $0 \leq t \leq t_f$ the following identity holds:

$$-\dot{\nu} = \min_u \{ \ell(\mathbf{x}, \mathbf{u}) + \lambda^T f(\mathbf{x}_t, \mathbf{u}_t) \} \quad (2.15)$$

**Definition 2.3. Optimal Control Policy**

The optimal control policy $\pi^*(\mathbf{x})$ is a value of $\mathbf{u}$ that achieves the minimum in (2.15):

$$\pi^*(\mathbf{x}) = \arg \min_u \{ \ell(\mathbf{x}, \mathbf{u}) + \lambda^T f(\mathbf{x}, \mathbf{u}) \} \quad (2.16)$$



Equations (2.15)–(2.16) are the *Hamilton-Jacobi-Bellman* (HJB) equations.

2.2.2 Solving HJB Equations

The HJB equation 2.15 is a non linear first order PDE. If there is a differentiable function ν^* satisfying the HJB equation everywhere then it is the unique global optimum. However, most often solutions to the equations do not satisfy the equation 2.15 everywhere. To work around this problem we often admit solution that satisfy HJB equations in an interval. Such solutions are called 'weak' solutions. If we decide to admit such weak solutions then we may have many solutions to the equations (Bertsekas 2010).

Crandall et al. (1983) introduced viscosity solutions for the HJB equations. The viscosity solution provides us with a consistency criterion to select one of the weak solutions. The optimal solution is also a viscosity solution, hence, it is the only optimal solution. We can devise (convergent) numerical methods to iteratively approximate the viscosity solutions (Todorov 2006).

However, the viscosity solutions rely on the discretisation of the state space and finer grid often yields better results (Todorov 2006). The downside of discretising methods is that they grow exponentially in the state dimensions. Bellman called this challenge as *the curse of dimensionality*. The Pontryagin's maximum principle can help to alleviate dimensionality curse.

2.3 Pontryagin's Maximum Principle

Pontryagin's maximum principle (PMP) was developed in the former Soviet Union during the heights of the cold war. Pontryagin, who was blind since the age of fourteen, introduced the maximum principle in 1956 and his work was later translated to English in (Pontryagin et al. 1962). The maximum principle establishes the necessary conditions for optimal control strategies. Unlike the HJB equations, Pontryagin's principle does not suffer from curse of dimensionality (Todorov 2006; Bertsekas 2010; Diehl 2011).

Pontryagin was a topologist and this influence is evident in the geometric nature of his original proof (Pontryagin et al. 1962).

2.3.1 Statement of the Principle

Consider the bracketed term in the optimal value function equation (2.15), namely $\ell(\mathbf{x}_t, \mathbf{u}_t) + \lambda^T f(\mathbf{x}_t, \mathbf{u}_t)$. This quantity, known as the *control Hamiltonian*, plays a key role in Pontryagin's Maximum Principle.

Definition 2.4. Control Hamiltonian

The control Hamiltonian for a system defined by the difference equation $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$ is given by

$$\mathcal{H}(\lambda, \mathbf{x}_t, \mathbf{u}_t) = \ell(\mathbf{x}_t, \mathbf{u}_t) + \lambda^T f(\mathbf{x}_t, \mathbf{u}_t) \quad (2.17)$$

Given a control system $(X, f, \mathcal{U})$ with dynamics f , controls $\mathbf{u} \in \mathcal{U}$ and states $\mathbf{x} \in X$, PMP states that:

Theorem 2.1. Pontryagin's Maximum Principle:

If $\{\mathbf{x}(t), \mathbf{u}(t) : 0 \leq t \leq T\}$ is an optimal state-control trajectory, then there exists an adjoint response, or co-state trajectory, $\lambda(t)$ satisfying the following conditions:

1. Adjoint Equation:

$$-\dot{\lambda}(t) = \frac{\partial \ell(\mathbf{x}(t), \mathbf{u}(t))}{\partial \mathbf{x}} + \lambda(t)^T \frac{\partial f(\mathbf{x}(t), \mathbf{u}(t))}{\partial \mathbf{x}} \quad (2.18)$$

2. Transversality Condition:

$$\lambda(T) = \frac{\partial \Phi(\mathbf{x}(T))}{\partial \mathbf{x}} \quad (2.19)$$

3. Minimum Condition:

$$\mathbf{u}^*(t) = \arg \min_{\mathbf{u}' \in \mathcal{U}} \{\ell(\mathbf{x}(t), \mathbf{u}') + \lambda^T f(\mathbf{x}(t), \mathbf{u}')\} \quad (2.20)$$

Equivalently, when written in terms of the Hamiltonian $\mathcal{H}$,

$$\mathcal{H}(\lambda, \mathbf{x}^*, \mathbf{u}^*) = \arg \min_{\mathbf{u}' \in \mathcal{U}} \mathcal{H}(\lambda, \mathbf{x}^*, \mathbf{u}') \quad (2.21)$$



Remark The boundary conditions $\Phi(\mathbf{x}(T))$, $\mathbf{x}(0)$ and time steps T are given.

2.3.2 Geometric Interpretation

While Pontryagin's maximum principle is detailed in many textbooks (Clarke 1990; Bertsekas 2010; Pontryagin et al. 1962; Schättler and Ledzewicz 2012), certain key concepts are often lost amidst mathematical definitions. Clarke (1990) provides theoretical treatment of PMP based on, e.g., the calculus of variations; Schättler and Ledzewicz (2012) approaches the topic from the lens of differential geometry; while, Lewis (2006) examines Pontryagin's principle from a control theoretical perspective. On

the applied front, works such as [I Michael Ross \(2015\)](#) convey an engineer's view of PMP and help to demonstrate how these tools apply to control, e.g., satellites ([I. Michael Ross and Karpenko 2012](#)).

In this section, we focus on the key insights that underlie an intuitive understanding of PMP. We defer proofs and rigorous treatment of this material to referenced texts. The following description is an amalgamation of the control perspective ([Lewis 2006](#)), engineering view point ([I Michael Ross 2015](#)) and the elegance of the original geometric insights in ([Vladimir Boltyanski 1993](#)).

Our main goal is to solve the Optimal Control Problem (OCP), defined by [2.1](#).

Definition 2.5. Minimisation Problem

Find the minimum of a function given in a finite dimensional space under constraints of equality type.

$$\arg \min_{\mathbf{x} \in \mathbb{R}^n} \ell(\mathbf{x}) \quad (2.22)$$

under the constraints

$$g_i(\mathbf{x}) = 0, \quad i = 1, \dots, s, \quad (2.23)$$

where $g_i : \mathbb{R}^n \rightarrow \mathbb{R}$ is differentiable on

$$\Omega_i := \{\mathbf{x} \in \mathbb{R}^n : g_i(\mathbf{x}) = 0\} \quad (2.24)$$



We can reformulate this problem: *find the minimum (or maximum) of the function $f(\mathbf{x})$ considered on the set $\Omega_1 \cap \dots \cap \Omega_s$.* The Lagrange theorem helps us solve the problems with equality constraints. For the inequality constraints in the OCP [2.1](#), we have similar sets i.e.

$$\Omega_i := \{\mathbf{x} \in \mathbb{R}^n : g_i(\mathbf{x}) \leq 0\}. \quad (2.25)$$

The inequality constraints can be solved by applying the Karush-Kuhn-Tucker (KKT) theorem.

To apply the Pontryagin maximum principle to this problem we generalise the minimisation problem [2.5](#). We begin with the definition of an abstract extremal problem.

Definition 2.6. Abstract extremal problem

Given sets $\Omega_1, \dots, \Omega_s \subseteq \mathbb{R}^n$ and a real-valued function ℓ with support on

$$\Sigma = \Omega_1 \cap \dots \cap \Omega_s,$$

find the minimum (and the minimisers) of the restriction of ℓ to Σ , denoted $\ell|_{\Sigma}$.



Assume that the point $x^* \in \Sigma$ minimises $\ell|_{\Sigma}$ and define the set

$$\Omega_0 = \{x^*\} \cup \{x : \ell(x) < \ell(x^*)\}.$$

Theorem 2.2. Criterion of Optimality

The point $x^* \in \Sigma$ is a minimiser of $\ell|_{\Sigma}$ if and only if

$$\Omega_0 \cap \Sigma = \Omega_0 \cap \Omega_1 \cap \dots \cap \Omega_s = \{x^*\} \quad (2.26)$$



Proof We prove the condition by contradiction. If there is a point

$$x' \in \Omega_0 \cap \Sigma$$

distinct from x^* , then $x' \in \Sigma$ and

$$\ell(x') < \ell(x^*),$$

contradicting the assumption that x^* is a minimiser. We can use similar arguments to show that x^* is the minimiser implies $(\Omega_0 \cap \Sigma = x^*)$ ■

The Theorem 2.2 allows us to write the extremal problem 2.6 as that of the following intersection problem (V. G. Boltyanski and Poznyak 2012)

Definition 2.7. Abstract intersection problem

For sets $\Omega_0 \cap \Omega_1 \cap \dots \cap \Omega_s$ in $\mathbb{R}^n$ with common point x^* , find a condition under which the intersection

$$\Omega_0 \cap \Omega_1 \cap \dots \cap \Omega_s$$

consists only of the point x^* .



Herein so far we have described optimal control problem in terms of sets. The main purpose of this discussion is that we can apply the concepts to the reachable sets for dynamical systems. Reachable sets have a very intuitive definition for dynamical systems. Consider a system with initial conditions $x_0 = 0$, the corresponding set of states that can be reached by time T when said system is subjected to an unconstrained control sequence $u_{0:T} \in \mathcal{U}$ is known as the *reachable set*. More formally,

Definition 2.8. Reachable Set

Given a reference state x_0 and starting time t_0 , the reachable set over a period of length τ is defined as

$$\mathcal{R}(x_0, t_0, \tau) = \{\phi(\tau, x_0, U) | U \in \mathcal{U}\} \quad (2.27)$$

where $U = [u_{t_0}, \dots, u_\tau]$ and the $\phi(\tau, x_0, U)$ is the solution to the dynamics equation (2.1).



Any solution to the optimal control problem should satisfy the constraints, i.e., the solution must be in the The main goal of the abstract extremal problem setup is that the optimal or the extremal solution we are looking for is in the intersection set of the constraints and the reachable sets.

Let us denote the reachable set starting from x_0 to be $\Omega_2 = \mathcal{R}(x_0, \mathcal{U}, \tau)$ and

the terminal set/goal set we wish to reach by Ω_1 then the optimal control problem Definition 2.1 is an abstract intersection problem:

$$\Omega_0 \cap \Omega_1 \cap \Omega_2 = \{x^*\}$$

For non-linear systems, reachable sets are often complex. Recognising this fact, Boltyanski (1975) proposed the *tent method* to help simplify associated analysis. In the tent method, we approximate a complex reachable set anchored at x_0 by a *tent*, a convex cone with apex x_0 . We follow (V. G. Boltyanski and Poznyak 2012) to formally define the tent K as follows:

Definition 2.9. Tent K

The tent K is a tangent cone (Clarke 1990) for set Ω with apex at a point x^* . To formalise notation, let $\Omega \subset \mathbb{R}^n$ be a set containing a point x^* and let K be a closed, convex cone with apex x^* . The cone K is said to be a tent of Ω at the point x^* if there exists a continuous mapping $\varphi : Q \rightarrow \mathbb{R}^n$ where Q is a neighbourhood of the point x^* such that

$$\begin{aligned}\varphi(x) &= x + o(x - x^*), \\ \varphi(K \cap Q) &\subset \Omega.\end{aligned}$$



The Figure 2.1, shows the reachable set Ω and its approximating cone K .

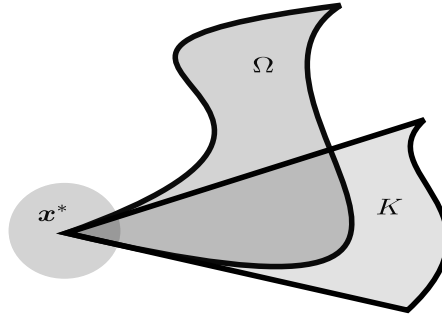


Figure 2.1: Tent K for the reachable set Ω with apex at x^*

Definition 2.10. Separability

A system $K_0, K_1, \dots, K_s$ of closed, convex cones with a common apex x^* in $\mathbb{R}^n$ is said to be separable if there exists a hyperplane Γ through x^* that separates one of the cones from the intersection of the others.



Theorem 2.3. The criterion of separability

Given closed, convex cones $K_0, K_1, \dots, K_s$ with a common apex x^* in $\mathbb{R}^n$, a necessary and sufficient condition for separability is the existence of s dual vectors α_i such that

$$\langle \alpha_i, x - x^* \rangle \leq 0 \quad \forall x \in K_i \quad (2.28)$$

and at least one of which is distinct from zero and such that

$$\alpha_0 + \alpha_1 + \dots + \alpha_s = 0 \quad (2.29)$$



Proof We state the theorem without the proof here. The proof relies on convex set theory and Farkas lemma (Farkas 1902). Detailed exposition of either is not required in the subsequent discussions. For detailed proof see (V. G. Boltyanski and Poznyak 2012; Boltyanski 1958).

The following theorem establishes the rationale behind the introduction of the tent method. It essentially says that the separability of cones is equivalent to the intersection of sets problem, which in turn solves the optimal control problem by Theorem 2.2 and the Definition 2.6.

Theorem 2.4. A necessary condition of separability

Let $\Omega_0, \Omega_1, \dots, \Omega_s \subset \mathbb{R}^n$ share a point $\mathbf{x}^*$ in common and let $K_0, K_1, \dots, K_s$ be their corresponding tents, each with apex $\mathbf{x}^*$. Assume that at least one of the tents is distinct from its affine hull (not a flat). If the cones $K_0, K_1, \dots, K_s$ are nonseparable, then there exists a point

$$\mathbf{x}' \in \Omega_0 \cap \Omega_1 \cap \dots \cap \Omega_s \text{ such that } \mathbf{x}' \neq \mathbf{x}^*.$$

Hence, separability is a necessary condition for

$$\Omega_0 \cap \Omega_1 \cap \dots \cap \Omega_s = \{\mathbf{x}^*\}$$



Proof Original proof (in Russian) was presented in (Boltyanski 1975), the above theorem follows the notation from (V. G. Boltyanski and Poznyak 2012).

Pontryagin's maximum principle says that the (control) Hamiltonian should be maximised as a necessary condition for the optimal trajectory. The following Lemma shows the link between the criterion of separability 2.3 and the Hamiltonian 2.4.

Lemma 2.1. Property of Maximum Hamiltonian

For control system with dynamics f and control set $\mathcal{U}$ consider point $(\mathbf{x}, \lambda, \hat{\mathbf{u}}) \in \mathcal{X} \times \mathbb{R}^n \times \mathcal{U}$. Then the Hamiltonian $\mathcal{H}(\mathbf{x}, \lambda, \hat{\mathbf{u}}) = \max_{\mathbf{u} \in \mathcal{U}} \mathcal{H}(\mathbf{x}, \lambda, \mathbf{u})$ if and only if

$$\langle \lambda, \mathbf{v} \rangle \leq 0, \quad \mathbf{v} \in \{f(\mathbf{x}, \mathbf{u}) - f(\mathbf{x}, \hat{\mathbf{u}}) \mid \forall \mathbf{u} \in \mathcal{U}\}.$$



Proof See Chapter 6 (Lewis 2006)

The important aspect of the Lemma 2.1 is illustrated in the Figure 2.2. The Hamiltonian is maximised when the vector λ is orthogonal to the supporting hyperplane for the reachable set of the system $(\mathcal{X}, f, \mathcal{U})$. If such a (co-)vector λ exists then the control maximising the Hamiltonian is on the boundary of the set $\{f(\mathbf{x}, \mathbf{u}) \mid \forall \mathbf{u} \in \mathcal{U}\}$.

We now show how the concepts introduced above can be used to visualise the Pontryagin principle geometrically. The concept is illustrated in the Figure 2.3a. For the

fixed interval ending in at time T , the reachable set is shown as the shaded area. The final point is on the boundary of this reachable set. If we arrive at this point earlier than time T our cost functions dictates that for optimality we stay at the edge of the reachable set at time T . Said another way, if the cost is not at its minimum, i.e, 0, for optimality we should keep moving toward lower cost. If we reached the target state ahead of time T , we stay there as any movement would increase cost.

Any variation of the trajectory around the final point would move inwards. Hence its approximating cone or tent will also point into the reachable state with its apex at the terminal point. From Lemma 2.1 we know that Hamiltonian at the terminal state is maximised if there exists an adjoint vector λ_T orthogonal to the tangent hyperplane at the boundary.

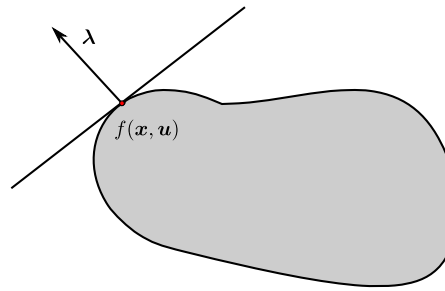


Figure 2.2: The diagram shows maximisation condition of the Hamiltonian described by the Lemma 2.1. The shaded region shows the reachable set or the region $\{f(x, u) \mid \forall u \in \mathcal{U}\}$

For every time step t in between 0 to T , there exists a reachable set, where the procedure or similar arguments can be repeated. That is at every time step t , there exists an adjoint vector λ_t , such that the Hamiltonian is maximised as shown in the Figure 2.3b. The final value of the adjoint response is thus translated to the initial point x_0 . The adjoint vector thus moves backwards in time starting from the final state x_T and moving towards the origin.

The steps above only considered the dynamics equations as the constraints in the original optimal control problem 2.1. The other conditions can be easily added to the abstract extremal problem 2.6 and solved as an intersection problem for all the sets 2.7. Thus we can apply the above framework to any non-linear constrained optimisation problem with each constraint s represented by the corresponding set Ω_s and its approximating cone K_s . This result holds true for functions in Banach spaces (V. G. Boltyanski and Poznyak 2012). When only considering dynamics equations as constraints the adjoint vector is essentially same as the Lagrangian vector for constrained optimisation. Moreover, the Karush-Kuhn-Tucker (KKT) conditions are easily derived from application of cone separation Theorem 2.3 with convex cones formed for each equality and inequality constraints.

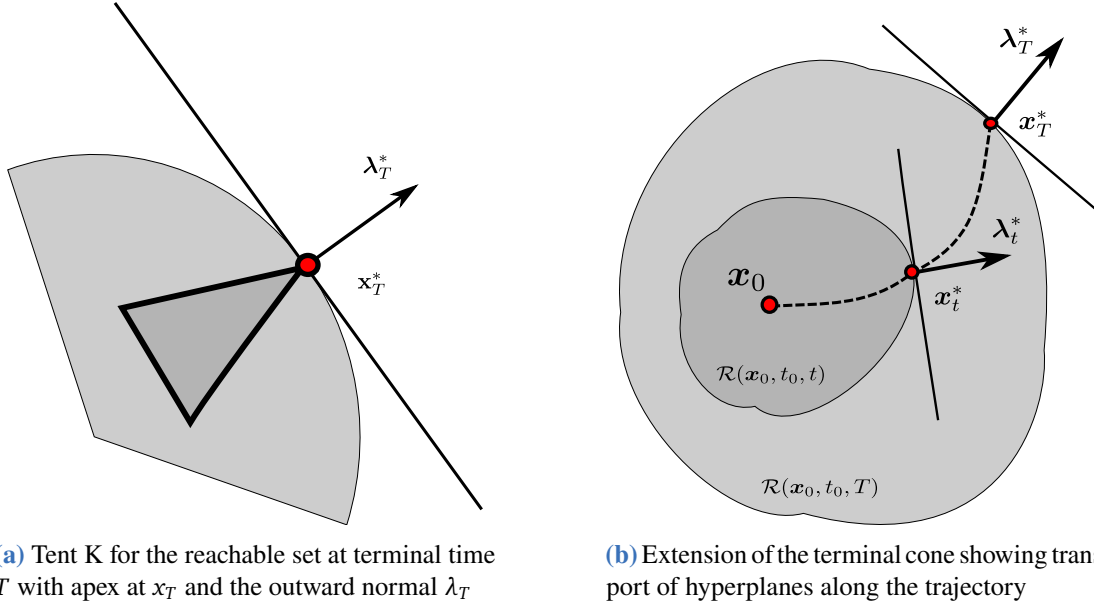


Figure 2.3: A figure demonstrating the adjoint variables λ in relation to the reachable sets of the control system $(X, f, \mathcal{U})$.

2.3.3 Summary

In the following we summarise the steps we followed so far in interpreting Pontryagin's maximum principle.



Remark

1. We defined an abstract geometric extremal problem for the optimal control problem 2.1. We then showed that the intersection of the constraint set and the reachable set gives the the solution to this extremal problem.
2. However, this extremal problem cannot be easily solved as the sets can be of varying complex shapes. So, we introduced a concept of tents K as approximations to these complex shapes. we then showed that separability criteria for these cones K formed using tent method is same as the intersection problem for the original complex set.
3. The intersection problem for cones is solved by the tangent hyperplanes for the cones. We show how the Hamiltonian is maximised by an adjoint vector orthogonal to these separating hyperplanes for the reachable sets. The Hamiltonian definition and the cone separability criteria both establish that for optimality there exists some adjoint vector λ orthogonal to the tangent hyperplane for the reachable set.
4. The condition for a full trajectory can broken into set of smaller problems of finding separating hyperplanes for reachable sets for each time instant $t_0 \leq t \leq T$. The overall trajectory is optimal if we have each of these smaller problems locally optimal. Which links back to the dynamical programming approach.
5. However, an important caveat is that each sub-problem is only locally optimal and we have only local or first order optimality with PMP. For engineering systems this

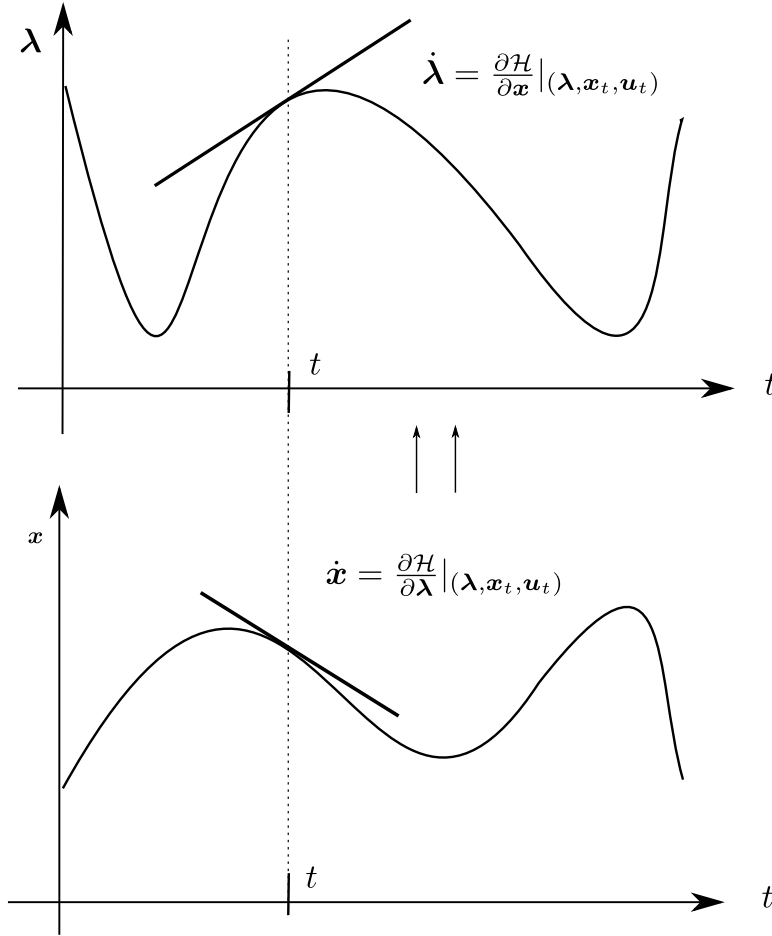


Figure 2.4: Co-state or the adjoint trajectory and its relation the state trajectory

is often good enough to solve practical problems (Todorov 2006; I Michael Ross 2015)

Remark We have two different ODEs one in state x with initial condition as starting state x_0 and one in adjoint space λ moving in the backward direction starting from the terminal state. The Figure 2.4 demonstrates this relation. The PMP states that for optimality the necessary condition is that the difference equations defined in Figure 2.4 are met for each $0 \geq t \leq T$. The adjoint vector can be viewed as a dual space in Lagrangian formulation, with control Hamiltonian as the Lagrangian of the optimal control problem.

Remark Equations $\dot{x} = \frac{\partial \mathcal{H}}{\partial \lambda} | (\lambda, x_t, u_t)$ and $\dot{\lambda} = \frac{\partial \mathcal{H}}{\partial x} | (\lambda, x_t, u_t)$ are similar to the Hamiltonian equations in classic mechanics, however, their interpretation is not same as the classic mechanics. The key notion is that the time derivative of one variable is linked to the partial derivative of Hamiltonian with respect to another variable.

Remark

- The PMP has $2n$ variable and $2n$ equations compared to only n in HJB equation, where n is the dimensionality of the state. However, HJB is a PDE and therefore significantly more challenging to solve compared to ODEs especially when n becomes large. HJB gives us global solution to the problem whereas the PMP

solution is valid only for given initial state and terminal set. Any change in either of those needs recomputing the optimal solutions.

- The proposed *MPC – RL* method recomputes the optimal solution for each new initial state. We show how this method can be used to develop an RL algorithm in the next chapter 3.

Optimal Control for Gaussian Process Dynamical systems

In this chapter we apply the proposed *MPC – RL* algorithm to non linear dynamical systems. For non-linear models it is very challenging to learn the transition dynamics accurately with limited number of observations. This often leads to modelling errors which if unaccounted for can degrade the performance of MPC (Camacho and Bordons 2004; Rawlings, D. Mayne, and Moritz Diehl 2017). We can build a probabilistic model to tackle the modelling errors (Schneider 1997; Marc P Deisenroth and Carl E Rasmussen 2011). In the following we build a Gaussian process dynamical model similar to one used in the the PILCO algorithm (Marc P Deisenroth and Carl E Rasmussen 2011) to tackle model uncertainty. PILCO algorithm propagates uncertainty through time for long-term planning and learns parameters of a feedback policy by means of gradient-based policy search. It achieves an unprecedented data efficiency for learning control policies from scratch.

While the PILCO algorithm is data efficient, it has few shortcomings: 1) Learning closed-loop feedback policies needs the full planning horizon to stabilise the system, which results in a significant computational burden; 2) It requires us to specify a parametrised policy a priori, often with hundreds of parameters; 3) It cannot handle state constraints; 4) Control constraints are enforced by using a differentiable squashing function that is applied to the RBF policy. This allows PILCO to explicitly take control constraints into account during planning. However, this kind of constraint handling can produce unreliable predictions near constraint boundaries (David Q. Mayne et al. 2000; Schättler and Ledzewicz 2012; Naidu and Naidu, Subbaram/Dorf 2003).

In this chapter, we develop an MPC based RL algorithm that is a) data efficient,

b) does not require to look at the full planning horizon, c) handles constraints naturally, d) does not require a parametrised policy, e) is theoretically justified. The key idea is to reformulate the optimal control problem with learned GP models as an equivalent deterministic problem, an idea similar to (Ng and Jordan 2000). This reformulation allows us to exploit Pontryagin’s maximum principle to find optimal control signals while handling constraints in a principled way.

We propose probabilistic model predictive control (MPC) with learned GP models, while propagating uncertainty through time. The MPC formulation allows to plan ahead for relatively short horizons, which limits the computational burden and allows for infinite-horizon control applications. Our approach can find optimal trajectories in constrained settings, offers an increased robustness to model errors and an unprecedented data efficiency compared to the state of the art. To summarise in terms of the basic MPC ingredients described in the previous chapter our formulation has the following structure,

- Gaussian process based probabilistic **predictive model**,
- expected value of the cumulative cost as an **objective function** and
- application of Pontryagin’s maximum principle for obtaining **optimal control**.

In the following we describe each of the items above in detail. We first build a probabilistic predictive model. The predictive cumulative cost computed based on the probabilistic is also a probabilistic and we treat *expected cost* as the objective function. Pontryagin’s principle cannot be directly applied to the probabilistic model, we show that the moment matching transition method proposed in Marc P Deisenroth and Carl E Rasmussen 2011 can be used to interpret GP dynamics model as deterministic ODE in an augmented space. We show that the PMP is valid for this augmented space with GP dynamics ODE. The proposed formulation can not only achieve state of the art results compared to PILCO it can also perform well with state constraints.

3.1 Gaussian Process Predictive Model

3.1.1 Probabilistic Transition Model

We learn a probabilistic model of the unknown underlying dynamics f to be robust to model errors (Schneider 1997; Marc P Deisenroth and Carl E Rasmussen 2011). In particular, we use a Gaussian process (GP) as a prior $p(f)$ over plausible transition functions f .

A GP is a probabilistic non-parametric model for regression. In a GP, any finite number of function values is jointly Gaussian distributed (Carl E Rasmussen and C. K. I. Williams 2006). A GP is fully specified by a mean function $m(\cdot)$ and a covariance function (kernel) $k(\cdot, \cdot)$.

The inputs for the dynamics GP are given by tuples $\tilde{\mathbf{x}}_t := (\mathbf{x}_t, \mathbf{u}_t)$, and the corre-

sponding targets are $\mathbf{x}_{t+1}$. We denote the collections of training inputs and targets by $\tilde{\mathbf{X}}, \mathbf{y}$, respectively. Furthermore, we assume a Gaussian (RBF, squared exponential) covariance function

$$k(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j) = \sigma_f^2 \exp \left(-\frac{1}{2} (\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j)^T \mathbf{L}^{-1} (\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j) \right), \quad (3.1)$$

where σ_f^2 is the signal variance and $\mathbf{L} = \text{diag}(l_1, \dots, l_{D+U})$ is a diagonal matrix of length-scales $l_1, \dots, l_{D+U}$. The GP is trained via the standard procedure of evidence maximisation (Carl E Rasmussen and C. K. I. Williams 2006; MacKay 1998).

We make the standard assumption that the GPs for each target dimension of the transition function $f : \mathbb{R}^D \times \mathcal{U} \rightarrow \mathbb{R}^D$ are independent. For given hyper-parameters, training inputs $\tilde{\mathbf{X}}$, training targets $\mathbf{y}$ and a new test input $\tilde{\mathbf{x}}_*$, the GP yields the predictive distribution $p(f(\tilde{\mathbf{x}}_*) | \tilde{\mathbf{X}}, \mathbf{y}) = \mathcal{N}(f(\tilde{\mathbf{x}}_*) | m(\tilde{\mathbf{x}}_*), \Sigma(\tilde{\mathbf{x}}_*))$, where

$$m(\tilde{\mathbf{x}}_*) = [m_1(\tilde{\mathbf{x}}_*), \dots, m_D(\tilde{\mathbf{x}}_*)]^T \quad (3.2)$$

$$m_d(\tilde{\mathbf{x}}_*) = k_d(\tilde{\mathbf{x}}_*, \tilde{\mathbf{X}}) (\mathbf{K}_d + \sigma_d^2 \mathbf{I})^{-1} \mathbf{y}_d \quad (3.3)$$

$$\Sigma(\tilde{\mathbf{x}}_*) = \text{diag}(\sigma_1^2(\tilde{\mathbf{x}}_*), \dots, \sigma_D^2(\tilde{\mathbf{x}}_*)) \quad (3.4)$$

$$\sigma_d^2 = \sigma_{fd}^2 - k_d(\tilde{\mathbf{x}}_*, \tilde{\mathbf{X}}) (\mathbf{K}_d + \sigma_d^2 \mathbf{I})^{-1} k_d(\tilde{\mathbf{X}}, \tilde{\mathbf{x}}_*), \quad (3.5)$$

for all predictive dimensions $d = 1, \dots, D$.

3.1.2 Long-term Predictions

To obtain the state distributions $p(\mathbf{x}_1), \dots, p(\mathbf{x}_T)$ for a given control sequence $\mathbf{u}_0, \dots, \mathbf{u}_{T-1}$, we iteratively predict

$$p(\mathbf{x}_{t+1} | \mathbf{u}_t) = \iint p(\mathbf{x}_{t+1} | \mathbf{x}_t, \mathbf{u}_t) p(\mathbf{x}_t) p(f) df d\mathbf{x}_t \quad (3.6)$$

for $t = 0, \dots, T-1$, by making a *deterministic* Gaussian approximation to $p(\mathbf{x}_{t+1} | \mathbf{u}_t)$ using moment matching (Marc P Deisenroth and Carl E Rasmussen 2011; Girard et al. 2003; Quinonero-Candela and C. Rasmussen 2005). This approximation has been shown to work well in practice in RL contexts (Marc P Deisenroth and Carl E Rasmussen 2011; Y. Pan and Theodorou 2014; Deisenroth, Fox, and Rasmussen 2015; Cutler and How 2015; Bastian Bischoff et al. 2013; B. Bischoff et al. 2014; Y. Pan, Theodorou, and Kontitsis 2015) and can be computed in closed form when using the Gaussian kernel (3.1).

A key property that we exploit is that moment matching allows us to formulate the uncertainty propagation in (3.6) as a ‘deterministic system function’

$$\mathbf{z}_{t+1} = f_{MM}(\mathbf{z}_t, \mathbf{u}_t), \quad \mathbf{z}_t := [\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t], \quad (3.7)$$

where $\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t$ are the mean and the covariance of $p(\mathbf{x}_t)$. For a deterministic control signal $\mathbf{u}_t$ we further define the moments of the control-augmented distribution $p(\mathbf{x}_t, \mathbf{u}_t)$ as

$$\tilde{\mathbf{z}}_t := [\tilde{\boldsymbol{\mu}}_t, \tilde{\boldsymbol{\Sigma}}_t], \quad \tilde{\boldsymbol{\mu}}_t = [\boldsymbol{\mu}_t, \mathbf{u}_t], \quad \tilde{\boldsymbol{\Sigma}}_t = \text{blkdiag}[\boldsymbol{\Sigma}_t, \mathbf{0}], \quad (3.8)$$

such that (3.7) can equivalently be written as the deterministic system equation

$$\mathbf{z}_{t+1} = f_{MM}(\tilde{\mathbf{z}}_t). \quad (3.9)$$

3.2 Objective Function

To find the optimal open-loop sequence $\mathbf{u}_0^*, \dots, \mathbf{u}_{T-1}^*$, we first compute the expected long-term cost J in (2.3) using the Gaussian approximations $p(\mathbf{x}_1), \dots, p(\mathbf{x}_T)$ obtained via (3.6) for a given open-loop control sequence $\mathbf{u}_0, \dots, \mathbf{u}_{T-1}$. Second, we find a control sequence that minimises the expected long-term cost (2.3). In the following, we detail these steps.

3.2.0.0.1 Computing the Expected Long-Term Cost To compute the expected long-term cost in (2.3), we sum up the expected immediate costs

$$\mathbb{E}[\ell(\mathbf{x}_t, \mathbf{u}_t)] = \int \ell(\tilde{\mathbf{x}}_t) \mathcal{N}(\tilde{\mathbf{x}}_t | \tilde{\boldsymbol{\mu}}_t, \tilde{\boldsymbol{\Sigma}}_t) d\tilde{\mathbf{x}}_t \quad (3.10)$$

for $t = 0, \dots, T-1$. We choose ℓ , such that this expectation and the partial derivatives $\partial \mathbb{E}[\ell(\mathbf{x}_t, \mathbf{u}_t)] / \partial \mathbf{x}_t$, $\partial \mathbb{E}[\ell(\mathbf{x}_t, \mathbf{u}_t)] / \partial \mathbf{u}_t$ can be computed analytically.¹

Similar to (3.7), this allows us to define deterministic mappings

$$\ell_{MM}(\mathbf{z}_t, \mathbf{u}_t) = \ell_{MM}(\tilde{\mathbf{z}}_t) := \mathbb{E}[\ell(\mathbf{x}_t, \mathbf{u}_t)] \quad (3.11)$$

$$\Phi_{MM}(\mathbf{z}_T) := \mathbb{E}[\Phi(\mathbf{x}_T)] \quad (3.12)$$

that map the mean and covariance of $\tilde{\mathbf{x}}$ onto the corresponding expected costs in (2.3).



Remark The open-loop optimisation turns out to be sparse (Bock and Plitt 1984). However, optimisation via the value function or dynamic programming is valid only for unconstrained controls. To address this practical shortcoming, we define Pontryagin's Maximum Principle that allows us to formulate the constrained problem while maintaining the sparsity.

3.3 Probabilistic Maximum Principle for GP Dynamics

To facilitate our discussion we first define some notation. Practical control signals are often constrained. We formally define a class of *admissible controls* $\mathcal{U}$ that are piecewise continuous functions defined on a compact space $U \subset \mathbb{R}^U$. This definition is fairly general, and commonly used zero-order-hold or first-order-hold signals satisfy this requirement. Applying admissible controls to the deterministic system dynamics f_{MM} defined in (3.9) yields a set $\mathcal{Z}$ of *admissible controlled trajectories*. We define the tuple $(\mathcal{Z}, f_{MM}, \mathcal{U})$ as our control system. For a single admissible control trajectory $\mathbf{u}_{0:H-1}$,

¹Choices for ℓ include the standard quadratic (polynomial) cost, but also costs expressed as Fourier series expansions or radial basis function networks with Gaussian basis function.

there will be a unique trajectory $\mathbf{z}_{0:H}$, and the pair $(\mathbf{z}_{0:H}, \mathbf{u}_{0:H-1})$ is called an admissible controlled trajectory (Schättler and Ledzewicz 2012).

Definition 3.1. Control Hamiltonian

We now define the control-Hamiltonian (Clarke 1990; Schättler and Ledzewicz 2012; Emanuel Todorov 2009) for this control system as

$$\mathcal{H}(\lambda_{t+1}, \mathbf{z}_t, \mathbf{u}_t) = \ell_{MM}(\mathbf{z}_t, \mathbf{u}_t) + \lambda_{t+1}^T f_{MM}(\mathbf{z}_t, \mathbf{u}_t). \quad (3.13)$$

A special term λ_0 may be considered for correct indexing, as the



This formulation of the control-Hamiltonian is the centre piece of the Pontryagin's approach to the OCP.

3.3.1 Existence/Uniqueness of a Local Solution

To apply the PMP we need to extend some of the important characteristics of ODEs to our system. In particular, we need to show the existence and uniqueness of a (local) solution to our difference equation (3.9).

For existence of a solution we need to satisfy the difference equation point-wise over the entire horizon and for uniqueness we need the system to have only one singularity. For our discrete-time system equation (via the moment-matching approximation) in (3.7) we have the following

Lemma 3.1. Lipschitz continuity

The moment matching mapping f_{MM} is Lipschitz continuous for controls defined over a compact set $\mathcal{U}$.



Proof

Lipschitz continuity requires that the gradient $\partial f_{MM} / \partial \mathbf{u}_t$ is bounded. The gradient is

$$\frac{\partial f_{MM}}{\partial \mathbf{u}_t} = \frac{\partial \mathbf{z}_{t+1}}{\partial \mathbf{u}_t} = \left[\frac{\partial \boldsymbol{\mu}_{t+1}}{\partial \mathbf{u}_t}, \frac{\partial \boldsymbol{\Sigma}_{t+1}}{\partial \mathbf{u}_t} \right]. \quad (3.14)$$

The derivatives $\left[\frac{\partial \boldsymbol{\mu}_{t+1}}{\partial \mathbf{u}_t}, \frac{\partial \boldsymbol{\Sigma}_{t+1}}{\partial \mathbf{u}_t} \right]$ can be computed analytically (Deisenroth, Fox, and Rasmussen 2015).

We first show that the derivative $\partial \boldsymbol{\mu}_{t+1} / \partial \mathbf{u}_t$ is bounded. Defining

$$\boldsymbol{\beta}_d := (\mathbf{K}_d + \sigma_{f_d}^2 \mathbf{I})^{-1} \mathbf{y}_d,$$

from (Deisenroth, Fox, and Rasmussen 2015), we obtain for all state dimensions $d =$

$1, \dots, D$

$$\mu_{t+1}^d = \sum_{i=1}^N \beta_{d_i} q_{d_i}, \quad (3.15)$$

$$q_{d_i} = \sigma_{f_d}^2 |\mathbf{I} + \mathbf{L}_d^{-1} \tilde{\Sigma}_t|^{-\frac{1}{2}} \times \quad (3.16)$$

$$\exp \left(-\frac{1}{2} (\tilde{\mathbf{x}}_i - \tilde{\boldsymbol{\mu}}_t)^\top (\mathbf{L}_d + \tilde{\Sigma}_t)^{-1} (\tilde{\mathbf{x}}_i - \tilde{\boldsymbol{\mu}}_t) \right), \quad (3.17)$$

where N is the size of the training set of the dynamics GP and $\tilde{\mathbf{x}}_i$ the i th training input. The corresponding gradient w.r.t. $\mathbf{u}_t$ is given by the last F elements of

$$\frac{\partial \mu_{t+1}^d}{\partial \tilde{\boldsymbol{\mu}}_t} = \sum_{i=1}^N \beta_{d_i} \frac{\partial q_{d_i}}{\partial \tilde{\boldsymbol{\mu}}_t} \quad (3.18)$$

$$= \sum_{i=1}^N \beta_{d_i} q_{d_i} (\tilde{\mathbf{x}}_i - \tilde{\boldsymbol{\mu}}_t)^\top (\tilde{\Sigma}_t + \mathbf{L}_d)^{-1} \in \mathbb{R}^{1 \times (D+F)} \quad (3.19)$$

Let us examine the individual terms in the sum on the rhs in (3.19): For a given trained GP $\|\beta_d\| < \infty$ is constant. The definition of q_{d_i} in (3.17) contains an exponentiated negative quadratic term, which is bounded between $[0, 1]$. Since $\mathbf{I} + \mathbf{L}_d^{-1} \tilde{\Sigma}_t$ is positive definite, the inverse determinant is defined and bounded. Finally, $\sigma_{f_d}^2 < \infty$, which makes $q_{d_i} < \infty$. The remaining term in (3.19) is a vector-matrix product. The matrix is regular and its inverse exists and is bounded (and constant as a function of $\mathbf{u}_t$. Since $\mathbf{u}_t \in \mathcal{U}$ where $\mathcal{U}$ is compact, we can also conclude that the vector difference in (3.19) is finite, which overall proves that f_{MM} is (locally) Lipschitz continuous and Lemma 3.1.

Existence and uniqueness of the trajectories for the moment matching difference equation are given by

Lemma 3.2. Unique Solution

A solution of $\mathbf{z}_{t+1} = f_{MM}(\mathbf{z}_t, \mathbf{u}_t)$ exists and is unique.



Proof Difference equations always yield an answer for a given input. Therefore, a solution trivially exists. Uniqueness directly follows from the Picard-Lindelöf theorem², which we can apply due to Lemma 3.1. This theorem requires the discrete-time system function to be deterministic (see Appendix B of (Schättler and Ledzewicz 2012)). Due to our re-formulation of the system dynamics (3.7), this follows directly, such that the $\mathbf{z}_{1:T}$ for a given control sequence $\mathbf{u}_{0:T-1}$ are unique.

3.3.2 Pontryagin's Maximum Principle for GP Dynamics

With Lemmas 3.1 3.2 and the definition of the control-Hamiltonian 3.13 we can now state PMP for the control system $(\mathbf{Z}, f_{MM}, \mathcal{U})$ as follows:

²For discrete difference equation uniqueness is trivial. Here, however, we interpret the dynamics as an Ordinary Differential Equation (ODE). With an ODE interpretation for the GP dynamics the Lemma allows us to say that the solution will exist and is unique.

Theorem 3.1. Pontryagin's Maximum Principle for Moment Matching GP

Let $(\mathbf{z}_t^*, \mathbf{u}_t^*)$, $0 \leq t \leq H - 1$ be an admissible controlled trajectory defined over the horizon H . If $(\mathbf{z}_{0:H}^*, \mathbf{u}_{0:H-1}^*)$ is optimal, then there exists an ad-joint vector $\lambda_t \in \mathbb{R}^D \setminus \{\mathbf{0}\}$ satisfying the following conditions:

1. Ad-joint equation: The ad-joint vector λ_t is a solution to the discrete difference equation

$$\lambda_t^T = \frac{\partial}{\partial \mathbf{z}_t} \ell_{MM}(\mathbf{z}_t, \mathbf{u}_t) + \lambda_{t+1}^T \frac{\partial f_{MM}(\mathbf{z}_t, \mathbf{u}_t)}{\partial \mathbf{z}_t}. \quad (3.20)$$

2. Transversality condition: At the endpoint the ad-joint vector λ_H satisfies

$$\lambda_H = \frac{\partial}{\partial \mathbf{z}_H} \Phi_{MM}(\mathbf{z}_H). \quad (3.21)$$

3. Minimum Condition: For $t = 0, \dots, H - 1$, we have

$$\mathcal{H}(\lambda_{t+1}, \mathbf{z}_t^*, \mathbf{u}_t^*) = \min_u \mathcal{H}(\lambda_{t+1}, \mathbf{z}_t^*, \mathbf{u}) \quad (3.22)$$

for all $\mathbf{u} \in \mathcal{U}$.



Remark The minimum condition (3.22) can be used to find an optimal control. The Hamiltonian is minimised point-wise over the admissible control set $\mathcal{U}$: For every $t = 0, \dots, H - 1$ we find optimal controls $\mathbf{u}_t^* \in \arg \min_{\mathbf{u}} \mathcal{H}(\lambda_{t+1}, \mathbf{z}_t^*, \mathbf{u})$. The minimisation problem possesses additional variables λ_{t+1} . These variables can be interpreted as Lagrange multipliers for the optimisation. They capture the impact of the control $\mathbf{u}_t$ over the whole trajectory and, hence, these variables make the optimisation problem sparse (Moritz Diehl 2014). For the GP dynamics we compute the multipliers λ_t in closed form, thereby, significantly reducing the computational burden to minimise the expected long-term cost J in (2.3). We detail this calculation in section 3.3.3.



Remark In the optimal control problem, we aim to find an admissible control trajectory that minimizes the cost subject to possibly additional constraints. PMP gives first-order optimality conditions over these admissible controlled trajectories and can be generalised to handle additional state and control constraints (Schättler and Ledzewicz 2012; Naidu and Naidu, Subbaram/Dorf 2003; Emanuel Todorov 2009).



Remark The Hamiltonian $\mathcal{H}$ in (3.22) is constant for unconstrained controls in time-invariant dynamics and equals 0 everywhere when the final time H is not fixed (Schättler and Ledzewicz 2012).



Remark For linear dynamics the proposed method is a generalisation of iLQG (Emanuel Todorov and W. Li 2005): The moment matching transition f_{MM} implicitly linearises the transition dynamics at each time step, whereas in iLQG an explicit local linear approximation is made. For a linear f_{MM} and a quadratic cost ℓ we can write the LQG case as shown in Theorem 1 in (Toussaint 2009). If we iterate with successive corrections to the linear approximations we obtain iLQG.

3.3.3 Efficient Gradient Computation

With the definition of the Hamiltonian $\mathcal{H}$ in (3.13) we can efficiently calculate the gradient of the expected total cost J . For a time horizon H we can write the accumulated cost as the Bellman recursion (Richard E Bellman 1957)

$$J_H(\mathbf{z}_H) := \Phi_{MM}(\mathbf{z}_H), \quad (3.23)$$

$$J_t(\mathbf{z}_t) := \ell_{MM}(\mathbf{z}_t, \mathbf{u}_t) + J_{t+1}(\mathbf{z}_{t+1}) \quad (3.24)$$

for $t = H - 1, \dots, 0$. Since the (open-loop) control $\mathbf{u}_t$ only term cost w.r.t. $\mathbf{u}_t$ can be efficiently computed using the Hamiltonian (Emanuel Todorov 2009). Next we show that the substitution $\lambda_{t+1}^T = \frac{\partial J_{t+1}}{\partial \mathbf{z}_{t+1}}$ is valid for the entire horizon H . For the terminal cost $\Phi_{MM}(\mathbf{z}_H)$ this is valid by the transversality condition (3.21). For other time impacts the future costs via $\mathbf{z}_{t+1} = f_{MM}(\mathbf{z}_t, \mathbf{u}_t)$ the derivative of the total cost with $\mathbf{u}_t$ is given by

$$\frac{\partial J_t}{\partial \mathbf{u}_t} = \frac{\partial \ell_{MM}(\mathbf{z}_t, \mathbf{u}_t)}{\partial \mathbf{u}_t} + \frac{\partial J_{t+1}}{\partial \mathbf{z}_{t+1}} \frac{\partial f_{MM}(\mathbf{z}_t, \mathbf{u}_t)}{\partial \mathbf{u}_t}. \quad (3.25)$$

Comparing this expression with the definition of the Hamiltonian (3.13), we see that if we make the substitution $\lambda_{t+1}^T = \frac{\partial J_{t+1}}{\partial \mathbf{z}_{t+1}}$ we obtain

$$\frac{\partial J_t}{\partial \mathbf{u}_t} = \frac{\partial \ell_{MM}(\mathbf{z}_t, \mathbf{u}_t)}{\partial \mathbf{u}_t} + \lambda_{t+1}^T \frac{\partial f_{MM}(\mathbf{z}_t, \mathbf{u}_t)}{\partial \mathbf{u}_t} = \frac{\mathcal{H}}{\partial \mathbf{u}_t}. \quad (3.26)$$

This implies that the gradient of the expected long-tps we differentiate (3.23) w.r.t. $\mathbf{z}_t$, which yields

$$\frac{\partial J_t}{\partial \mathbf{z}_t} = \frac{\partial \ell_{MM}(\mathbf{z}_t, \mathbf{u}_t)}{\partial \mathbf{z}_t} + \frac{\partial J_{t+1}}{\partial \mathbf{z}_{t+1}} \frac{\partial \mathbf{z}_{t+1}}{\partial \mathbf{z}_t} \quad (3.27)$$

$$= \frac{\partial \ell_{MM}(\mathbf{z}_t, \mathbf{u}_t)}{\partial \mathbf{z}_t} + \lambda_{t+1}^T \frac{\partial f_{MM}(\mathbf{z}_t, \mathbf{u}_t)}{\partial \mathbf{z}_t}, \quad (3.28)$$

which is identical to the ad-joint equation (3.20). Hence, in our setting, PMP implies that *gradient descent on the Hamiltonian $\mathcal{H}$ is equivalent to gradient descent on the total cost (2.3)* (Schättler and Ledzewicz 2012; Moritz Diehl 2014).

Algorithmically, in an RL setting, we find the optimal control sequence $\mathbf{u}_0^*, \dots, \mathbf{u}_{H-1}^*$ as follows:

1. For a given initial (random) control sequence $\mathbf{u}_{0:H-1}$ we follow the steps described in section 3.1.2 to determine the corresponding trajectory $\mathbf{z}_{1:H}$. Additionally, we compute Lagrange multipliers $\lambda_{t+1}^T = \frac{\partial J_{t+1}}{\partial \mathbf{z}_{t+1}}$ during the forward propagation. Note that traditionally ad-joint equations are propagated backward to find the multipliers (Clarke 1990; Schättler and Ledzewicz 2012).
2. Given λ_t and a cost function ℓ_{MM} we can determine the Hamiltonians $\mathcal{H}_{1:H}$. Then we find a new control sequence $\mathbf{u}_{0:H-1}^*$ via any gradient descent method using (3.26).
3. Return to 1 or exit when converged.

We use Sequential Quadratic Programming (SQP) with BFGS for Hessian updates (Nocedal and Wright 2006). The Lagrangian of SQP is a partially separable function (Griewank and Toint 1982). In the PMP, this separation is explicit via the

Hamiltonians, i.e., the $\mathcal{H}_t$ is a function of variables with index t or $t + 1$. This leads to a block-diagonal Hessian of SQP Lagrangian (Griewank and Toint 1982). The structure can be exploited to approximate the Hessian via block-updates within BFGS (Bock and Plitt 1984; Griewank and Toint 1982)

3.4 Related Work

Model-based RL: A recent survey of model based RL in robotics (Polydoros and Nalpantidis 2017) highlights the importance of models for building adaptable robots. Instead of GP dynamics model with a zero prior mean (as used in this paper) an RBF network and linear mean functions are proposed (Cutler and How 2015; B. Bischoff et al. 2014). This accelerates learning and facilitates transferring a learned model from simulation to a real robot. Even implicit model learning can be beneficial: The UNREAL learner proposed in (Jaderberg et al. 2016) learns a predictive model for the environment as an auxiliary task, which helps learning.

MPC with GP transition models: GP-based predictive control was used for boiler and building control (Grancharova, Kocijan, and Johansen 2008; Nghiem and Jones 2017), but the model uncertainty was discarded. In (Klenske et al. 2016), the predictive variances were used within a GP-MPC scheme to actively reject periodic disturbances, although not in an RL setting. Similarly, in (C. Ostafew et al. 2015; C. J. Ostafew, Schoellig, and Barfoot 2016), the authors used a GP prior to model additive noise and the model is improved episodically. In (Nghiem and Jones 2017; Boedecker et al. 2014), the authors considered MPC problems with GP models, where only the GP’s posterior mean was used while ignoring the variance for planning. MPC methods with deterministic models are useful only when model errors and system noise can be neglected in the problem (Girard et al. 2003; Kocijan 2016).

Optimal Control: The application of optimal control theory for the models based on GP dynamics employs some structure in the transition model, i.e., there is an explicit assumption of control affinity (Y. Pan and Theodorou 2014; Y. Pan, Theodorou, and Kontitsis 2015; Boedecker et al. 2014; Hennig 2011) and linearisation via locally quadratic approximations (Y. Pan and Theodorou 2014; Boedecker et al. 2014). The AICO model (Toussaint 2009) uses approximate inference with (known) locally linear models. The probabilistic trajectories for model-free RL in (Rawlik, Toussaint, and Vijayakumar 2012) are obtained by reformulating the stochastic optimal control problem as KL divergence minimisation. We implicitly linearise the transition dynamic via moment matching approximation.

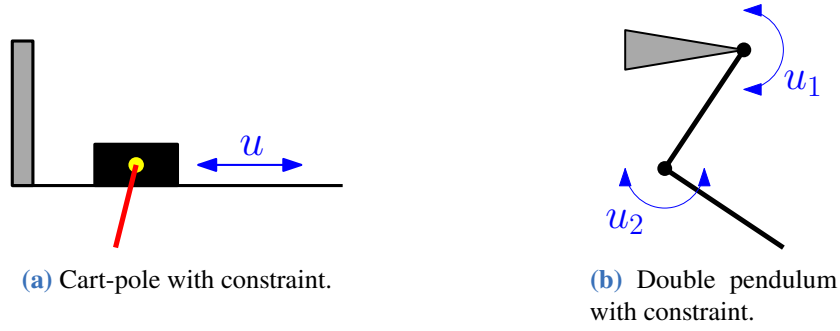


Figure 3.1: State constraints for RL benchmarks. **a** The position of the cart is constrained on the left side by a wall. **b** The angle of the inner pendulum cannot enter the grey region.

3.5 Experimental Results

We evaluate the quality of our algorithm in two ways: First, we assess whether probabilistic MPC leads to faster learning compared with PILCO, the current state of the art in terms of data efficiency. Second, we assess the impact of state constraints while performing the same task.

We consider two RL benchmark problems: the under-actuated cart-pole-swing-up and the fully actuated double-pendulum swing-up. In both tasks, PILCO is the most data-efficient RL algorithm to date (Deisenroth, Fox, and Rasmussen 2015).

3.5.1 Under-actuated Cart-Pole Swing-Up

The cart pole system is an under-actuated system with a freely swinging pendulum of 50 cm mounted on a cart. The swing-up and balancing task cannot be solved using a linear model (Raiko and Tornio 2009). The cart-pole system state space consists of the position of the cart x , cart velocity $\dot{x}$, the angle θ of the pendulum and the angular velocity $\dot{\theta}$. A horizontal force $u \in [-10, 10]$ N can be applied to the cart. Starting in a position where the pendulum hangs downwards, the objective is to automatically learn a controller that swings the pendulum up and balances it in the inverted position in the middle of the track.

Constrained Cart-Pole Swing-Up For the state-space constraint experiment we place a wall on the track near the target, see Fig. 3.1a. The wall is at -70 cm, which, along with force limitations, requires the system to swing from the right side.

3.5.2 Fully-actuated Double-Pendulum

The double pendulum system is a two-link robot arm (links lengths: 1 m) with two actuators at each joint. The state space consists of 2 angles and 2 angular velocities $[\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2]$ (Deisenroth, Fox, and Rasmussen 2015). The torques u_1 and u_2 are

limited to $[-2, 2]$ Nm. Starting from a position where both links are in a downwards position, the objective is to learn a control strategy that swings the double-pendulum up and balances it in the inverted position.

Constrained Double-Pendulum The double-pendulum has a constraint on the angle of the inner pendulum, so that it only has a 340° motion range, i.e., it cannot spin through, see Fig. 3.1b. The constraint blocks all clockwise swing-ups. The system is underpowered, and it has to swing clockwise first for a counter-clockwise swing-up without violating the constraints.

3.5.3 Trials

The general setting is as follows: All RL algorithms start off with a single short random trajectory, which is used for learning the dynamics model. As in (Marc P Deisenroth and Carl E Rasmussen 2011; Deisenroth, Fox, and Rasmussen 2015) the GP is used to predict state differences $\mathbf{x}_{t+1} - \mathbf{x}_t$. The learned GP dynamics model is then used to determine a controller based on iterated moment matching (3.6), which is then applied to the system, starting from $\mathbf{x}_0 \sim p(\mathbf{x}_0)$. Model learning, controller learning and application of the controller to the system constitute a ‘trial’. After each trial, the hyper-parameters of the model are updated with the newly acquired experience and learning continues.

3.5.3.1 Baselines

We compare our GP-MPC approach with the following baselines: the PILCO algorithm (Marc P Deisenroth and Carl E Rasmussen 2011; Deisenroth, Fox, and Rasmussen 2015) and a zero-variance GP-MPC algorithm (in the flavour of (Nghiem and Jones 2017; Boedecker et al. 2014)) for RL, where the GP’s predictive variances are discarded. Due to the lack of exploration, such a zero-variance approach within PILCO (a policy search method) does not learn anything useful as already demonstrated in (Deisenroth, Fox, and Rasmussen 2015), and we do not include this baseline.

We average over 10 independent experiments, where every algorithm is initialised with the same first (random) trajectory. The performance differences of the RL algorithms are therefore due to different approaches to controller learning and the induced exploration.

3.5.4 Data Efficiency

In both benchmark experiments (cart-pole and double pendulum), we use the exact saturating cost from (Deisenroth, Fox, and Rasmussen 2015), which penalises the

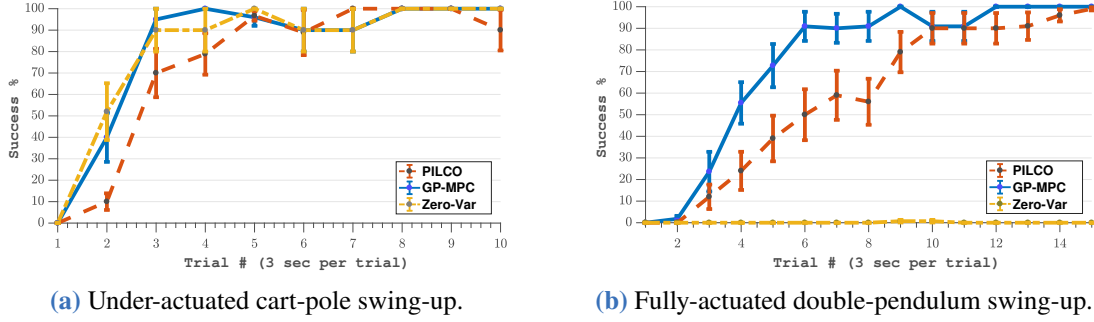


Figure 3.2: Performance of RL algorithms. Error bars represent the standard error. **a** Cart-pole; **b** Double pendulum. GP-MPC (blue) consistently outperforms PILCO (red) and the zero-variance MPC approach (yellow) in terms of data efficiency. While the zero-variance MPC approach works well on the cart-pole task, it fails in the double-pendulum task. We attribute this to the inability to explore the state space sufficiently well.

Euclidean distance of the tip of the (outer) pendulum from the target position, i.e., we are in a setting in which PILCO performs very well.

Fig. 3.2a shows that both our MPC-based controller (blue) and the zero-variance approach successfully³ complete the task in fewer trials than the state-of-the-art PILCO method (red). From the repeated trials we see that GP-MPC learns faster and more reliably than PILCO. In particular, GP-MPC and the zero-variance approach can solve the cart-pole task with high probability (90%) after three trials (9 seconds), where the first trial was random. PILCO needs two additional trials. The reason why the zero-variance approach (no model uncertainties) works in an MPC context but not within a policy search setting is that we include every observed state transition immediately in the GP dynamics model, which makes MPC fairly robust to model errors. It even allows for model-based RL with deterministic models in simple settings.

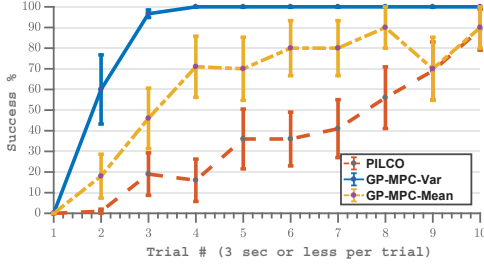
Fig. 3.2b highlights that our proposed GP-MPC approach (yellow) requires on average only six trials (18 s) of experience to achieve a 90% success rate⁴, including the first random trial. PILCO requires four additional trials, whereas the zero-variance MPC approach completely fails in this RL setting. The reason for this failure is that the deterministic predictions with a poor model in this complicated state space do not allow for sufficient exploration. We also observe that GP-MPC is more robust to the variations amongst trials.

In both experiments, our proposed GP-MPC requires only 60% of PILCO’s experience, such that we report an unprecedented learning speed for these benchmarks, even with settings for which PILCO performs very well.

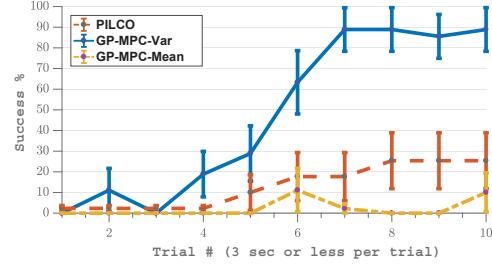
We identify two key ingredients that are responsible for the success and learning speed of our approach: the ability to (1) immediately react to observed states by adjusting

³We define ‘success’ if the pendulum tip is closer than 8 cm to the target position for ten consecutive time steps.

⁴The tip of outer pendulum is closer than 22 cm to the target.



(a) Cart-pole with constraint.



(b) Double pendulum with constraint.

Figure 3.3: Performance with state-space constraints. Error bars represent the standard error. **a** Cart-pole; **b** Double pendulum. GP-MPC with chance constraints. GP-MPC-Var (blue) is the only method that is able to consistently solve the problem. Expected violations constraint GP-MPC-Mean (yellow) fails in cart-pole. PILCO (red) violates state constraints and struggles to complete the task.

the long-term plan and (2) augment the training set of the GP model on the fly as soon as a new state transition is observed (hyper-parameters are not updated at every time step). These properties turn out to be crucial in the very early stages of learning when very little information is available. If we ignored the on-the-fly updates of the GP dynamics model, our approach would still successfully learn, although the learning efficiency would be slightly decreased.

3.5.5 State Constraints

A scenario in which PILCO struggles is a setting with state space constraints. We modify the cart-pole and the double-pendulum tasks to such a setting. Both tasks are symmetric, and we impose state constraints in such a way that only one direction of the swing-up is feasible. For the cart-pole system, we place a wall near the target position of the cart, see Fig. 3.1a; the double pendulum has a constraint on the angle of the inner pendulum, so that it only has a 340° motion range, i.e., it cannot spin through, see Fig. 3.1b. These constraints constitute linear constraints on the state.

We use a quadratic cost that penalises the Euclidean distance between the tip of the pendulum and the target. This, along with the ‘implicit’ linearisation, makes the optimal control problem an ‘implicit’ QP. If a rollout violates the state constraint we immediately abort that trial and move on to the next trial. We use the same experimental set-up as data efficiency experiments.

The state constraints are implemented as *expected violations*, i.e., $\mathbb{E}[\mathbf{x}_t] < \mathbf{x}_{\text{limit}}$ and *chance constraints* $p(\mathbf{x}_t < \mathbf{x}_{\text{limit}}) \geq 0.95$. Fig. 3.3 shows that our MPC-based controller with chance constraint (blue) successfully completes the task with a small acceptable number of violations, see Table 3.1. The expected violation approach, which only considers the predicted mean (yellow) fails to complete the task due to repeated constraint violations. PILCO uses its saturating cost (there is little hope for learning with

Experiment	Cart-pole	Double Pendulum
PILCO	16/100	23/100
GP-MPC-Mean	21/100	26/100
GP-MPC-Var	3/100	11/100

Table 3.1: State constraint violations. The number of trials that resulted in state constraint violation corresponding to the trial data shown in the the Fig. 3.3.

quadratic cost (Marc P Deisenroth and Carl E Rasmussen 2010)) and has partial success in completing the task, but it struggles, especially during initial trials due to repeated state violations.

One of the key points we observe from the Table 3.1 is that the incorporation of uncertainty into planning is again crucial for successful learning. If we use only predicted means to determine whether the constraint is violated, learning is not reliably ‘safe’. Incorporation of the predictive variance, however, results in significantly fewer constraint violations.

3.6 Conclusion and Discussion

We proposed an algorithm for data-efficient RL that is based on probabilistic MPC with learned transition models using Gaussian processes. By exploiting Pontryagin’s maximum principle our algorithm can naturally deal with state and control constraints. Key to this theoretical underpinning of a practical algorithm was the re-formulation of the optimal control problem with uncertainty propagation via moment matching into an deterministic optimal control problem. MPC allows the learned model to be updated immediately, which leads to an increased robustness with respect to model inaccuracies. We provided empirical evidence that our framework is not only theoretically sound, but also extremely data efficient, while being able to learn in settings with hard state constraints.

One of the most critical components of our approach is the incorporation of model uncertainty into modelling and planning. In complex environments, model uncertainty drives targeted exploration. It additionally allows us to account for constraints in a risk-averse way, which is important in the early stages of learning.

3.6.1 Future Work

The proposed method could be significantly improved by considering a feedback policy instead of the open loop policy employed by the MPC. However, we would lose the (local) optimality guarantees for such a system. This feedback policy would be variation of PILCO (Deisenroth, Fox, and Rasmussen 2015) with a modified cost function to account for chance constraints. Another way we can improve the performance by creating GP

dynamics models with control affine structure. Affine structure can be exploited to obtain closed form approximation to control signals.

Part II

Iterative Kalman Filtering for Non-linear Dynamical Systems

State Estimation in Non-linear systems

4.1 Introduction

Estimating the latent state of a stochastic system given noisy measurements is a challenging task with many practical applications in signal processing and control. We are often interested in estimating state probability densities given all the measurements. For linear systems with Gaussian noise, an optimal estimate of the state probability densities conditioned on the observations up to the current time is given in closed form by the Kalman Filter ([Kalman 1960](#)). The estimate conditioned on all (including future) measurements of a finite time series is given by the Rauch-Tung-Striebel (RTS) ([Rauch, Tung, and Striebel 1965](#)) smoother.

The Kalman filter is optimal in linear Gaussian systems ([Brian D O Anderson and John B Moore 2005](#)). The efficiency of the RTS smoother stems from the linearity property of Gaussian distribution, i.e., a linear/affine transformation of a Gaussian remains Gaussian. Moreover, given a joint Gaussian, we can get a closed-form expression for the conditional density. The Gaussian conditioning property allows us to derive recursive filtering and smoothing equations in closed form.

In contrast, in non-linear systems, these computationally convenient properties no longer hold. Typically, we tackle this challenge either by approximating a non-Gaussian density by a Gaussian density or using sampling-based methods, e.g., the particle filter ([Doucet, Godsill, and Andrieu 2000](#)), to represent non-Gaussian densities. In this paper, we focus on the former, i.e., we use a Gaussian approximation for the true non-Gaussian densities. As this assumption is only exact for linear systems, we either explicitly, as in the extended Kalman filter (EKF) or implicitly, as in the unscented

Kalman filter (UKF) (Julier, Uhlmann, and Durrant-Whyte 2000), linearise our non-linear system. The same linearisation can be used to construct an RTS smoother for non-linear systems, e.g., the extended and unscented Kalman smoothers (Särkkä 2008).

The quality of these approximate state estimators depends on the mechanism used for the linearisation. A crude approximation may introduce large errors. One can alleviate some of these errors by iteratively correcting previous estimates (Lefebvre, Bruyninckx, and De Schutter 2004). For example, in the iterative extended Kalman filter (IEKF) (Gelb 1974), measurement updates are carried out iteratively until convergence. The IEKF is a Gauß–Newton method for MAP estimation (Bell and F. Cathey 1993). Recent progress (Raitoharju et al. 2018; Gultekin and Paisley 2017) shows that the Bayesian framework and iterative updates can give significant improvement over standard filtering and smoothing.

All Gaussian filtering systems make an assumption that we can project the true posterior to a Gaussian (Thrun, Burgard, and D. Fox 2005), with a reasonable approximation error. More generally, this can be interpreted as an approximate assumed density filter (ADF) with a Gaussian assumption on densities. In the ADF, the approximate posterior is obtained by minimising the Kullback–Leibler ($\mathcal{KL}$) divergence between true posterior and the assumed density. For a Gaussian assumed density, this amounts to computing the first two moments of the true posterior and approximate it with a Gaussian that possesses the same first two moments. In the ADF, as well in the iterative Kalman filters, there is no mechanism to revisit previously filtered estimates and correct them based on all the observations.

This shortcoming was addressed by expectation propagation (EP) proposed in (T. P. Minka 2001a; b; Oppen and Winther 2001). EP extends the ADF by adding a mechanism to iteratively update previous computations without double counting observations. For linear Gaussian systems, one full iteration of EP is identical to the Kalman filter-smoother (T. P. Minka 2001a; Qi and T. Minka 2003; Heskes and Zoeter 2002). Hence, EP is optimal for linear systems.

One can extend EP to non-linear systems similar to the generalised Gaussian filters. This extension is motivated by results in (Dehaene and Barthelmé 2016), which show that EP has an order of magnitude lower asymptotic error bound compared to a canonical Gaussian approximation for non-linear systems. Furthermore, applying EP to non-linear systems allows us to perform multiple sweeps of RTS smoothing, iterating over measurement and transition non-linearities conditioned on all available data, whereas the IEKF systems can only iterate over measurement non-linearities. Moreover, EP can be implemented as message passing algorithm. Hence, it can be used as a distributed (and asynchronous) state estimator and is not limited to the sequential nature of iterative Kalman filters.

One challenge with implementing EP is that its efficacy relies on the ease of

computing moments with respect to the true posterior; for non-linear systems the exact posterior is often intractable. In this work, we propose to approximate the true posterior by linearisation methods, such as the unscented transform (Julier, Uhlmann, and Durrant-Whyte 2000). Hence, we are only implementing *approximate* EP when we apply EP to non-linear systems. We prove that this approximate EP exactly recovers the respective RTS smoother, e.g., with unscented transform approximation we obtain the unscented Kalman smoother (Särkkä 2008). This allows us to extend EP to all non-linear Gaussian filters.

However, note that EP, an iterative algorithm, is not guaranteed to converge, but may end up in oscillations and limit cycles (Heskes and Zoeter 2003). By extension, the approximate EP version proposed in this paper is also prone to numerical instability (Gultekin and Paisley 2017) and limit cycles. We address this problem by using a fixed damping rate.

An important consequence of using the $\mathcal{KL}$ divergence with true posterior for EP iteration is that when the true posterior is also Gaussian, e.g., in the linear case, EP recovers the optimal solution, i.e., the RTS smoother. For the non-linear case, the true posterior is non-Gaussian and often multimodal. With approximate EP we try to match moments of the non-Gaussian density as closely as possible, often resulting in averaging modes of true multi-modal posterior. For some applications, one may want a different behaviour from the estimator, picking/favouring one mode as against averaging over the entire distribution. This behaviour can be achieved by using a generalisation of $\mathcal{KL}$ divergence, i.e., an α divergence (Zhu and Rohwer 1995; T. T. Minka 2005) as a divergence measure for running EP. The α -divergence is a family of divergences indexed by $\alpha \in (-\infty, +\infty)$. For $\alpha = 1$ we get the $\mathcal{KL}(p||q)$ divergence. For $\alpha = 0$ we exactly recover variational Bayes (VB) (T. T. Minka 2005; Piché, Särkkä, and Hartikainen 2012).

Different values for α correspond to different filtering algorithms for state-space models; $\alpha = 0$ corresponds to variational message passing (T. T. Minka 2005; Jaakkola 2000), while $\alpha = 1$ is equivalent to running belief propagation or EP. Values $0 < \alpha < 1$ produce a blend of either algorithms (equivalent to fractional belief propagation) (T. T. Minka 2005). An important aspect of the α -divergence EP is that one can use $D_{\alpha'}(p||q)$ to minimise any other $D_{\alpha}(p||q)$ within a scaling factor. Hence, we can exploit stable Kalman-filter based EP ($\alpha = 1$) to minimise fractional α -EP which can be unstable. Owing to this instability, authors in (Gultekin and Paisley 2017) only use a single iteration of EP. Our proposed method on the other hand suffers from no such limitations, thereby allowing for better estimates of the posterior distribution.

4.1.1 Contribution

1. We show that standard Kalman-filter approximations can be used to perform numerically stable *approximate* EP by explicitly showing the equivalence of the Gaussian RTS filter-smoother of derivative free and derivative based EP updates.
2. We further generalise EP based scheme to Power EP, which allows to interpolate between EP and variational Bayes (VB) for non-linear state-space models. Furthermore, we devise a scheme to use stable Kalman-filter based updates to overcome numerical instabilities often associated with approximate Power EP
3. Any approximation method used to compute moments can be used to create novel algorithms for inference in non-linear dynamical systems within a single framework. On the other hand all Kalman-filter-smoother methods can be seen as special case of our framework.

We use a classic non-linear system benchmark, the uniform nonlinear growth model (UNGM) (Doucet, Godsill, and Andrieu 2000) with a fixed seed as a running example to demonstrate benefits of the iterative smoothing scheme.

4.1.2 Overview

The chapter is organised as follows: First in Section 4.2, we establish the notation and review classic Kalman filtering and smoothing, which we then apply to our UNGM benchmark. In Section 4.3, we show how we can use numerically stable non-linear Kalman based updates in place of potentially unstable approximate EP updates, by algebraically showing their equivalence. Next, we apply these Kalman-based EP iterations to further improve the standard smoothing results of the UNGM example. Section 4.4.2 then tackles the limit-cycle behaviour of EP by using damped EP (Heskes and Zoeter 2003), and then further improves the results by using α -divergence EP. We then analyse the efficacy of our method and analyse its sensitivity with respect to α , the damping parameter, and the linearisation technique in Section 4.5.

4.2 Gaussian Filtering and Smoothing

We consider dynamical systems of the form

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}) + \mathbf{w}_t \quad (4.1)$$

$$\mathbf{y}_t = h(\mathbf{x}_t) + \mathbf{v}_t, \quad (4.2)$$

where $\mathbf{x}_t \in \mathbb{R}^D$ is the unobserved/latent state and $\mathbf{y}_t \in \mathbb{R}^E$ is the measurement at time step $t = 1, \dots, T$. Moreover, f and h are (non-linear) transition and measurement functions, respectively. The noise processes $\mathbf{w}_t$ and $\mathbf{v}_t$ are i.i.d. zero-mean Gaussian with diagonal covariances $\mathbf{Q}$ and $\mathbf{R}$, respectively. The initial state distribution is given

by $p(\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_0 | \boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$. The purpose of filtering and smoothing is to find posterior distributions $p(\mathbf{x}_t | \mathbf{y}_{1:\tau})$, where a subscript $1 : \tau$ abbreviates $1, \dots, \tau$, with $\tau = t$ for filtering and $\tau = T$ for smoothing.

4.2.1 Optimal Filtering and Smoothing

Bayesian filtering corresponds to finding a posterior distribution $p(\mathbf{x}_t | \mathbf{y}_{1:t})$ on the unobserved latent state. It proceeds by sequentially computing a time update and a measurement update (Brian D O Anderson and John B Moore 2005). Given a filtering distribution $p(\mathbf{x}_{t-1} | \mathbf{y}_{1:t-1})$, the time update

$$p(\mathbf{x}_t | \mathbf{y}_{1:t-1}) = \int p(\mathbf{x}_t | \mathbf{x}_{t-1}) p(\mathbf{x}_{t-1} | \mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1} \quad (4.3)$$

computes a predictive distribution of the state $\mathbf{x}_t$ given observations up to time step $t - 1$. Here, $p(\mathbf{x}_t | \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t | f(\mathbf{x}_{t-1}), \mathbf{Q})$ is the transition probability. The measurement update then incorporates the observation $\mathbf{y}_t$ at time t to yield the filtering distribution

$$p(\mathbf{x}_t | \mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_t | \mathbf{x}_t) p(\mathbf{x}_t | \mathbf{y}_{1:t-1})}{\int p(\mathbf{y}_t | \mathbf{x}_t) p(\mathbf{x}_t | \mathbf{y}_{1:t-1}) d\mathbf{x}_t} \propto p(\mathbf{y}_t | \mathbf{x}_t) p(\mathbf{x}_t | \mathbf{y}_{1:t-1}) \quad (4.4)$$

at time t .

The smoother computes a posterior distribution $p(\mathbf{x}_t | \mathbf{y}_{1:T})$ of the latent state $\mathbf{x}_t$ given all observations in the time series. In this article, we consider the Rauch-Tung-Striebel (RTS) smoother, which computes this posterior according to

$$p(\mathbf{x}_t | \mathbf{y}_{1:T}) = \int p(\mathbf{x}_t | \mathbf{x}_{t+1}, \mathbf{y}_{1:T}) p(\mathbf{x}_{t+1} | \mathbf{y}_{1:T}) d\mathbf{x}_{t+1}, \quad (4.5)$$

where we have used Markov property $p(\mathbf{x}_t | \mathbf{x}_{t+1}, \mathbf{y}_{1:t}) = p(\mathbf{x}_t | \mathbf{x}_{t+1}, \mathbf{y}_{1:T})$.

The expressions in (4.3)–(4.5) are generally intractable for non-linear systems. However, if we focus on Gaussian approximations to the filtering and smoothing distributions, we can derive both smoothing and filtering equations by forming joint Gaussian densities $p(\mathbf{x}_t, \mathbf{y}_t | \mathbf{y}_{1:t-1})$ and $p(\mathbf{x}_t, \mathbf{x}_{t+1} | \mathbf{y}_{1:t})$ (Marc P Deisenroth 2009; Särkkä, Hartikainen, et al. 2015). The joint distribution $p(\mathbf{x}_t, \mathbf{y}_t | \mathbf{y}_{1:t-1})$ can be written as

$$\mathcal{N} \left(\begin{pmatrix} \mathbf{x}_t \\ \mathbf{y}_t \end{pmatrix} \middle| \begin{pmatrix} \boldsymbol{\mu}_{t|1:t-1}^x \\ \boldsymbol{\mu}_{t|1:t-1}^y \end{pmatrix}, \begin{pmatrix} \boldsymbol{\Sigma}_{t|1:t-1}^x & \boldsymbol{\Sigma}_{t|1:t-1}^{xy} \\ \boldsymbol{\Sigma}_{t|1:t-1}^{yx} & \boldsymbol{\Sigma}_{t|1:t-1}^y \end{pmatrix} \right). \quad (4.6)$$

By exploiting the joint Gaussianity, the measurement update can then be computed by Gaussian conditioning, and we obtain filter update $p(\mathbf{x}_t | \mathbf{y}_{1:t}) \approx \mathcal{N}(\mathbf{x}_t | \boldsymbol{\mu}_{t|t}^x, \boldsymbol{\Sigma}_{t|t}^x)$, where

$$\boldsymbol{\mu}_{t|t}^x = \boldsymbol{\mu}_{t|t-1}^x + \mathbf{K}(\mathbf{y}_t - \boldsymbol{\mu}_{t|t-1}^y), \quad (4.7)$$

$$\boldsymbol{\Sigma}_{t|t}^x = \boldsymbol{\Sigma}_{t|t-1}^x - \mathbf{K} \boldsymbol{\Sigma}_{t|t-1}^{yx}. \quad (4.8)$$

Here, the matrix $\mathbf{K} := \boldsymbol{\Sigma}_{t|t-1}^{xy} (\boldsymbol{\Sigma}_{t|t-1}^y)^{-1}$ corresponds to the Kalman gain in linear systems. The means and covariances for this update or entries in the joint equation (4.6), can be

computed by the following equations:

$$\boldsymbol{\mu}_{t|t-1}^x = \int f(\mathbf{x}_{t-1}) p(\mathbf{x}_{t-1} | \mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1}, \quad (4.9)$$

$$\boldsymbol{\Sigma}_{t|t-1}^x = \int f(\mathbf{x}_{t-1}) (f(\mathbf{x}_{t-1}))^T p(\mathbf{x}_{t-1} | \mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t|t-1}^x (\boldsymbol{\mu}_{t|t-1}^x)^T + \mathbf{Q}, \quad (4.10)$$

$$\boldsymbol{\mu}_{t|t-1}^y = \int h(\mathbf{x}_t) p(\mathbf{x}_t | \mathbf{y}_{1:t-1}) d\mathbf{x}_t, \quad (4.11)$$

$$\boldsymbol{\Sigma}_{t|t-1}^y = \int h(\mathbf{x}_t) (h(\mathbf{x}_t))^T p(\mathbf{x}_t | \mathbf{y}_{1:t-1}) d\mathbf{x}_t - \boldsymbol{\mu}_{t|t-1}^y (\boldsymbol{\mu}_{t|t-1}^y)^T + \mathbf{R}, \quad (4.12)$$

$$\boldsymbol{\Sigma}_{t|t-1}^{xy} = \int \mathbf{x}_t (h(\mathbf{x}_t))^T p(\mathbf{x}_t | \mathbf{y}_{1:t-1}) d\mathbf{x}_t - \boldsymbol{\mu}_{t|t-1}^x (\boldsymbol{\mu}_{t|t-1}^y)^T + \mathbf{R}. \quad (4.13)$$

These equations cannot be computed in closed form when f and h are non-linear. Any method that approximates these integrals can be used to form Gaussian filters (Marc P Deisenroth and Ohlsson 2011; Särkkä 2013). A similar update can be written for the RTS smoother by approximating the joint distribution $p(\mathbf{x}_t, \mathbf{x}_{t+1} | \mathbf{y}_{1:t})$ by a Gaussian (M. Deisenroth et al. 2012), such that the mean and the covariance matrix of the smoothing distribution are given by

$$\boldsymbol{\mu}_{t|T}^x = \boldsymbol{\mu}_{t|t}^x + \mathbf{L}(\boldsymbol{\mu}_{t+1|T}^x - \boldsymbol{\mu}_{t+1|t}^x), \quad (4.14)$$

$$\boldsymbol{\Sigma}_{t|T}^x = \boldsymbol{\Sigma}_{t+1|t+1}^x + \mathbf{L} \left(\boldsymbol{\Sigma}_{t+1|T}^x - \boldsymbol{\Sigma}_{t+1|t}^x \right) \mathbf{L}^T, \quad (4.15)$$

respectively. Here, the matrix $\mathbf{L} = \boldsymbol{\Sigma}_{t,t+1|t}^x \left(\boldsymbol{\Sigma}_{t+1|t}^x \right)^{-1}$ is called the smoothing gain. The cross-covariance $\boldsymbol{\Sigma}_{t,t+1|t}^x$ between $\mathbf{x}_t$ and $\mathbf{x}_{t+1}$ is given by

$$\boldsymbol{\Sigma}_{t,t+1|t}^x = \int \mathbf{x}_t (f(\mathbf{x}_t))^T p(\mathbf{x}_t | \mathbf{y}_{1:t}) d\mathbf{x}_t - \boldsymbol{\mu}_{t|t}^x (\boldsymbol{\mu}_{t+1|t}^x)^T. \quad (4.16)$$

For a non-linear transition function f , we need to approximate this integral, e.g., by means of linearisation or the unscented transformation.

4.2.2 Implicit Linearisation in Non-linear Gaussian Filters and Smoothers

Any method used to approximate these integrals implicitly linearises the respective transition f and measurement h functions as two variables are jointly Gaussian if and only if they have a linear/affine relationship. For example, a joint Gaussian formed between $\mathbf{x}_t$ and $\mathbf{y}_t$ linearises the measurement function.

Commonly used Bayesian filters and smoothers for non-linear systems are the extended Kalman filter, the unscented Kalman filter (Julier, Uhlmann, and Durrant-Whyte 2000), and the cubature Kalman filter (Arasaratnam and Haykin 2009), all of which compute means and covariances of the unobserved state. Since all update equations of these non-linear filters and smoothers can equally be written in terms of (4.7), (4.8), (4.14), and (4.15), these methods implicitly linearise the dynamical system (4.1)–(4.2).

We may improve the filtering/smoothing results by iteratively refining the state

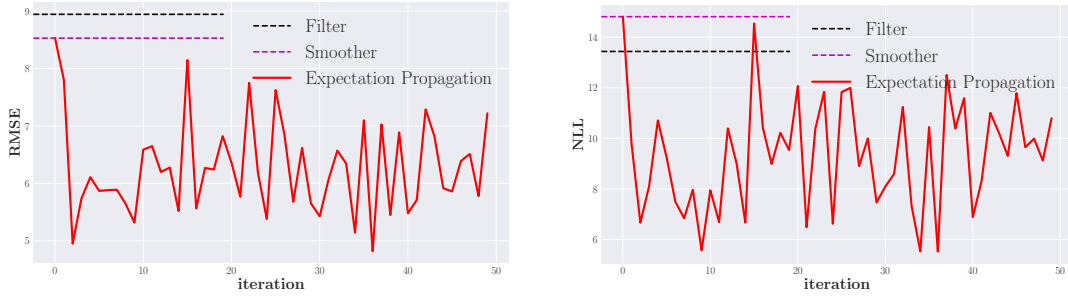


Figure 4.1: In this figure we compare the Root Mean Square Error (RMSE) performance of estimation algorithms applied to UNGM. The (unscented) Kalman filter, (unscented) RTS smoother form the base lines. The iterative EP starts from the RTS smoothing base-line as 1st iteration of EP is same as RTS smoother. Smoothing improves filtering by taking into account all the measurements. Most non-linear RTS smoothers, rely on local linearisation based on current estimate, with iterative application these estimates may be different for each iteration. In the non-linear EP proposed here we exploit this whilst avoiding counting the observations multiple times. We can see that the EP estimate is consistently better than the smoother. The oscillatory behaviour is addressed by using damping Section 4.4.1 and Power EP Section 4.4

estimate. In the iterative extend Kalman filter (IEKF) (Senne 1972), e.g., an iterative measurement update is used to improve the standard EKF solution. The IEKF formulates the measurement update as a non-linear least squares problem, which can then be solved using the Gauß-Newton method (F. W. Cathey 1993), the damped Gauß-Newton (More 1977) or a Quasi-Newton (Skoglund, Hendebly, and Axehill 2015). For comparison of various methods for iterated filtering, see (Havlík and Straka 2015).

These iterative methods, as well as recently proposed posterior linearisation filter (Garcia-Fernandez et al. 2015), show that iteratively modifying linearisation improves filtering results if done correctly (Raitoharju et al. 2018); similar results for transition functions are not well explored. We would expect similar improvements as obtained in filtering by iteratively (implicitly) linearising the transition function as well. This is challenging, as, unlike the measurement function that only depends on measurement at time t , the smoothing result depends on the whole chain from the current time t to the end T . Expectation propagation is an algorithm that can be used to iteratively refine the approximations (T. P. Minka 2001a). Standard EP is only applicable for linear functions and it exactly recovers the Kalman and RTS smoother for linear systems. Similar to Kalman-based iterative filters we expect the iterative EP updates to iteratively improve results. In the following section, we demonstrate performance of EP on an instance of non-linear benchmark. The equations and algorithm is detailed in section 4.3.

4.2.3 RTS Smoothing Applied to a Non-linear Benchmark

To illustrate the benefits of iterative refinement we apply it to a well-known non-linear benchmark

$$x_t = \frac{x_{t-1}}{2} + \frac{25x_{t-1}}{1 + x_{t-1}^2} + 8 \cos(1.2(t - 1)) + w_t \quad (4.17)$$

$$y_t = \frac{x_t^2}{20} + v_t \quad (4.18)$$

using the same setup as in (Doucet, Godsill, and Andrieu 2000): $w_t \sim \mathcal{N}(0, 1)$, $w_t \sim \mathcal{N}(0, 10)$ and $p(x_0) = \mathcal{N}(x_0 | 0, 5)$. This benchmark is challenging for state estimation as the transition density around $x \approx 0$ is bimodal and symmetric. The scaled quadratic measurement function makes these modes indistinguishable.

We then apply the unscented Kalman filter and smoother (Merwe and Wan 2004) to this task. Results are shown in Figure 4.1.¹ We can see that smoothing improves the root mean square error (RMSE), and iterations further improve the results significantly. The RTS smoothing result and the first pass of EP are identical. We see that iterative refinement is almost always better than the results obtained by a single pass. In the next section, we develop the expectation propagation algorithm used to generate the solid red curve in Figure 4.1.

4.3 Expectation Propagation

Expectation propagation (EP) (T. P. Minka 2001a; b) is a Bayesian inference algorithm that approximates complex and an often intractable joint distribution $p(\mathbf{X})$ by a distribution $q(\mathbf{X})$, where $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_T\}$ which is a product of simpler and tractable individual/marginal distributions $q(\mathbf{X}) = \prod_i q_i(\mathbf{x}_i)$. The underlying assumption is that if we can approximate these individual distributions q_i well we may get better estimates of the intractable joint. We restrict the factors $q_i(\mathbf{x})$ to family of distributions, such that they are closed under multiplication and division (Seeger 2005).

Definition 4.1. KL Projection

The $\mathcal{KL}$ projections of p onto $\mathcal{F}$ is a distribution q that minimises the $\mathcal{KL}$ - divergence to p over a distribution family $\mathcal{F}$. We use the shorthand notation to define the projection operator $\text{proj}[\cdot]$ as

$$\text{proj}[p || q] = \arg \min_{q \in \mathcal{F}} \mathcal{KL}(p || q). \quad (4.19)$$



¹The UKS is chosen here as it demonstrates the advantages of the proposed method. A more thorough comparison across different filters and seeds is discussed in Section 4.5. We chose the UT for implicit linearisation as it is numerically more stable than the EKF, and sampling-based linearisation is inherently more accurate and makes it challenging to demonstrate different aspects of the method proposed.

In EP, we assume that the approximations q are from the exponential family. In the exponential family, the $\text{proj}[p, \|\cdot\|q]$ or minimum $\mathcal{KL}$ -divergence is obtained when q matches the moments of p . We estimate $q(\mathbf{x}_t)$ by iteratively refining each of the factors q_i . First, we compute the *cavity distribution*

$$q^{\setminus i}(\mathbf{x}_t) := q(\mathbf{x}_t)/q_i(\mathbf{x}_t), \quad (4.20)$$

which contains the “rest of the distribution” (T. P. Minka 2004), i.e., all the information except from the i^{th} factor. Then, we minimise the $\mathcal{KL}$ -divergence, implicitly via moment matching, over $q_i(\mathbf{x}_t)$, holding the cavity distribution $q^{\setminus i}(\mathbf{x}_t)$ fixed. This yields the updated i^{th} factor

$$q_i(\mathbf{x}_t)^{\text{new}} = \arg \min \mathcal{KL} \left(f_i(\mathbf{x}_t) q^{\setminus i}(\mathbf{x}_t) \| q_i(\mathbf{x}_t) q^{\setminus i}(\mathbf{x}_t) \right) \quad (4.21)$$

$$= \text{proj} \left[f_i(\mathbf{x}_t) q^{\setminus i}(\mathbf{x}_t) \| q_i(\mathbf{x}_t) q^{\setminus i}(\mathbf{x}_t) \right] / q^{\setminus i}(\mathbf{x}_t). \quad (4.22)$$

Next, we describe how to chose the q factorisation for the state estimation problem. While there is no restriction on how to structure the individual factors $q_i(\mathbf{x}_t)$ it is easier to chose a factorisation that exploits the structure of the task we are planning to solve. For the state estimation problem described in section 4.2.1, e.g., the joint density can be written as (Särkkä and Sarmavuori 2013),

$$p(\mathbf{x}_{1:T}, \mathbf{y}_{1:T}) = p(\mathbf{x}_0) \prod_{t=1}^T p(\mathbf{x}_t | \mathbf{x}_{t-1}) p(\mathbf{y}_t | \mathbf{x}_t), \quad (4.23)$$

where we have used Markov property and conditional independence of measurements, i.e., $p(\mathbf{y}_t | \mathbf{x}_t) = p(\mathbf{y}_t | \mathbf{x}_t, \mathbf{y}_{1:t-1})$, to write the joint density as a product of densities. We are interested in the smoothing density $p(\mathbf{x}_{1:T} | \mathbf{y}_{1:T})$, which can be obtained by applying Bayes’ theorem to the joint density in (4.23), i.e.,

$$p(\mathbf{x}_{1:T} | \mathbf{y}_{1:T}) = \frac{p(\mathbf{x}_0) \prod_{t=1}^T p(\mathbf{x}_t | \mathbf{x}_{t-1}) p(\mathbf{y}_t | \mathbf{x}_t)}{\int p(\mathbf{x}_{1:T}, \mathbf{y}_{1:T}) d\mathbf{x}_{1:T}} \quad (4.24)$$

$$= Z p(\mathbf{x}_0) \prod_{t=1}^T p(\mathbf{x}_t | \mathbf{x}_{t-1}) p(\mathbf{y}_t | \mathbf{x}_t), \quad (4.25)$$

where $Z = 1 / \int p(\mathbf{x}_{1:T}, \mathbf{y}_{1:T}) d\mathbf{x}_{1:T}$.

The joint density in (4.23) can be approximated well if we can find a good approximation for each of the individual terms on the right-hand side of the equation. We are interested in approximations that allow us to factorise $p(\mathbf{x}_{1:T} | \mathbf{y}_{1:T})$ into $\prod_{t=1}^T \tilde{p}(\mathbf{x}_t | \mathbf{y}_{1:T})$, which can be achieved by rewriting (4.25) as

$$p(\mathbf{x}_{1:T} | \mathbf{y}_{1:T}) = Z p(\mathbf{x}_0) \prod_{t=1}^T p(\mathbf{x}_t | \mathbf{x}_{t-1}) p(\mathbf{y}_t | \mathbf{x}_t) \quad (4.26)$$

$$\approx Z p(\mathbf{x}_0) \prod_{t=1}^T f_{\triangleright \triangleleft}(\mathbf{x}_t, \mathbf{x}_{t-1}) f_{\Delta}(\mathbf{x}_t, \mathbf{y}_t). \quad (4.27)$$

Here, $f_{\triangleright \triangleleft}(\mathbf{x}_t, \mathbf{x}_{t-1})$ approximates the transition density, which we further decouple to

write

$$f_{\triangleright\triangleleft}(\mathbf{x}_t, \mathbf{x}_{t-1}) = f_{\triangleright}(\mathbf{x}_t) f_{\triangleleft}(\mathbf{x}_{t-1}). \quad (4.28)$$

The forward factor $f_{\triangleright}$ is a message/information that is carried from all the latent states and observations before the current latent state, i.e., it carries the history of all observations. The future information on the other hand is captured by $f_{\triangleleft}$. This decoupling allows us to write the latent state distributions as isolated marginals. To achieve this, we substitute $f_{\triangleright}(\mathbf{x}_0) \equiv p(\mathbf{x}_0)$ and $f_{\triangleright}(\mathbf{x}_T) \equiv 1$ in (4.27) to obtain

$$p(\mathbf{x}_{1:T} | \mathbf{y}_{1:T}) = Z \prod_{t=0}^T f_{\triangleright}(\mathbf{x}_t) f_{\Delta}(\mathbf{x}_t) f_{\triangleleft}(\mathbf{x}_t) \quad (4.29)$$

$$= Z \prod_{t=0}^T \tilde{p}(\mathbf{x}_t | \mathbf{y}_{1:T}). \quad (4.30)$$

Now, each state distribution is decoupled and can be written as product of factors/messages

$$\begin{aligned} \tilde{p}(\mathbf{x}_t | \mathbf{y}_{1:T}) &= f_{\triangleright}(\mathbf{x}_t) f_{\Delta}(\mathbf{x}_t) f_{\triangleleft}(\mathbf{x}_t) \\ &= (\text{forward}) (\text{measurement}) (\text{backward}). \end{aligned} \quad (4.31)$$

The smoothing distribution $p(\mathbf{x}_t | \mathbf{y}_{1:T})$ is represented as the product of factors $f_i(\mathbf{x}_t)$, i.e., $p(\mathbf{x}_t | \mathbf{y}_{1:T}) = \prod_i f_i(\mathbf{x}_t)$ as in (4.31).

So far we only assumed that the smoothing distribution p factorises in a specific form, but we did not impose any restrictions on the factors f_i . Each of these factors represents a specific information that can be used to update our state estimate/belief. The current observation $\mathbf{y}_t$ influences the observation or up factor f_{Δ} . Owing to the Markovian structure of the problem, The forward factor $f_{\triangleright}$, the backward factor $f_{\triangleleft}$ and the up factor f_{Δ} is all the information we need to compute $p(\mathbf{x}_t | \mathbf{y}_{1:T})$. For linear systems, these factors are unique and optimal (T. P. Minka 2001a). The Kalman smoother obtains these results in one forward-backward sweep. For non-linear systems, however, these factors are intractable. We can use approximations to compute these factors (Qi and T. Minka 2003). The above interpretation of the factors is valid only for the first iteration, i.e., one forward pass and one backward pass both done sequentially. If we iterate multiple times, the forward factor for example now also has information from the *future* observations. When correctly setup we hope that with each iteration we improve the estimate of the factors. The premise of the EP algorithm is that a better estimate of each one of the them may lead to a better joint estimate (T. T. Minka 2005).

For the non-linear system we consider, true factors $f_i(\mathbf{x}_t)$ and thereby $p(\mathbf{x}_t)$ are intractable. We approximate the true distribution p by a proxy q . To facilitate computation we choose q that factorises in the same way as p , i.e., we write q as

$$q(\mathbf{x}_{1:T}) = \prod_{t=1}^T q(\mathbf{x}_t) = \prod_{t=0}^T q_{\triangleright}(\mathbf{x}_t) q_{\Delta}(\mathbf{x}_t) q_{\triangleleft}(\mathbf{x}_t), \quad (4.32)$$

Algorithm 1 Gaussian EP for Dynamical Systems

-
- 1: **Init:** Set all factors q_i to $\mathcal{N}(\mathbf{0}, \infty I)$; Set $q(\mathbf{x}_0) = p(\mathbf{x}_0)$ and marginals $q(\mathbf{x}_{t \neq 1}) = \mathcal{N}(\mathbf{0}, 10^{10} I)$
 - 2: **repeat**
 - 3: **for** $t = 1$ to T **do**
 - 4: **for** all factors $q_i(\mathbf{x}_t)$, where $i = \triangleright, \Delta, \triangleleft$ **do**
 - 5: **Form a Cavity:**

$$q^{\setminus i}(\mathbf{x}_t) = q(\mathbf{x}_t) / q_i(\mathbf{x}_t), \quad (4.33)$$
 - 6: **Project:**

$$q'(\mathbf{x}_t) = \text{proj} \left[f_i(\mathbf{x}_t) q^{\setminus i}(\mathbf{x}_t) \parallel q_i(\mathbf{x}_t) q^{\setminus i}(\mathbf{x}_t) \right] \quad (4.34)$$
 - 7: **Update Factor:**

$$q_i(\mathbf{x}_t) = q(\mathbf{x}_t) / q^{\setminus i}(\mathbf{x}_t) \quad (4.35)$$
 - 8: **Update Posterior:**

$$q(\mathbf{x}_t)^{\text{new}} = q'(\mathbf{x}_t) q_i(\mathbf{x}_t) \quad (4.36)$$
 - 9: **end for**
 - 10: **end for**
 - 11: **until** Convergence or maximum number of iterations exceeded
-

which mimics the factorisation of p in (4.31). In the following, we detail the computation for updating the factors q_i , for all $i \in \{\triangleright, \Delta, \triangleleft\}$, for Gaussian beliefs.

4.3.1 EP Updates for Generalised Gaussian Filtering

In generalised Gaussian filtering, we assume that all densities are Gaussian as described in Section 4.2.1. To facilitate computation we chose our approximate densities q_i , $i \in \{\triangleright, \Delta, \triangleleft\}$, to be Gaussian. Moreover, Gaussians are closed under multiplication and division. The pseudo-code in Alg. 1 describes the main steps of Gaussian EP for dynamical systems. The algorithm updates the marginal $q(\mathbf{x}_t)$ and the messages $q_i(\mathbf{x}_t)$, which form this marginal, in three steps.

- **Form a Cavity:** First, the *cavity distribution* $q^{\setminus i}(\mathbf{x}_t)$ is computed (step 5 in Alg. 1) by removing $q_i(\mathbf{x}_t)$ from the marginal $q(\mathbf{x}_t)$. For Gaussian beliefs both $q(\mathbf{x}_t)$ and $q_i(\mathbf{x}_t)$ are Gaussian, hence, the cavity distribution is also a (scaled) Gaussian,

$$q^{\setminus i}(\mathbf{x}_t) = q(\mathbf{x}_t) / q_i(\mathbf{x}_t) \propto \mathcal{N}(\mathbf{x}_t \mid \boldsymbol{\mu}^{\setminus i}, \boldsymbol{\Sigma}^{\setminus i}). \quad (4.37)$$

The terms $\boldsymbol{\mu}^{\setminus i}$, $\boldsymbol{\Sigma}^{\setminus i}$ can be computed via the standard Gaussian division rule (Brandt

Petersen et al. 2012),

$$\Sigma^{\setminus i} = (\Sigma_t^{-1} - \Sigma_i^{-1})^{-1}, \quad (4.38)$$

$$\mu^{\setminus i} = \Sigma^{\setminus i}(\Sigma_t^{-1}\mu_t - \Sigma_i^{-1}\mu_i). \quad (4.39)$$

- **Projection:** In the projection step (step 6 in Alg. 1), we minimise the $\mathcal{KL}$ divergence between true posterior $p(\mathbf{x}_t)$ and the approximation $q(\mathbf{x}_t)$ while holding the cavity distribution $q^{\setminus i}(\mathbf{x}_t)$ constant. The idea is to assume that while correcting the i^{th} factor, all other factors (which form the cavity distribution) provide the context of this correction, i.e., $p(\mathbf{x}_t) = \prod_i f_i(\mathbf{x}_t) = f_i(\mathbf{x}_t)f^{\setminus i}(\mathbf{x}_t) \approx f_i(\mathbf{x}_t)q^{\setminus i}(\mathbf{x}_t)$. For exponential families, the $\mathcal{KL}$ minimisation is equivalent to moment matching (Seeger 2005). Hence, we need to compute the moments of $f_i(\mathbf{x}_t)q^{\setminus i}(\mathbf{x}_t)$. Furthermore, in the exponential family, the required moments can be computed using the derivatives of the log-partition function (normalising constant) $\log Z_i$ of $f_i(\mathbf{x}_t)q^{\setminus i}(\mathbf{x}_t)$ (T. P. Minka 2001a; b; 2008). Hence, for the projection $\text{proj}[\cdot]$ we determine moments of $f_i(\mathbf{x}_t)q^{\setminus i}(\mathbf{x}_t)$, e.g., via the derivatives of

$$\log Z_i(\mu^{\setminus i}, \Sigma^{\setminus i}) = \log \int f_i(\mathbf{x}_t)q^{\setminus i}(\mathbf{x}_t)d\mathbf{x}_t. \quad (4.40)$$

For non-linear systems we cannot compute above integral in closed form. Instead we approximate the integral $\int f_i(\mathbf{x}_t)q^{\setminus i}(\mathbf{x}_t)d\mathbf{x}_t$ via any prediction method (e.g. unscented transform, linearisation etc.). By approximating this integral in Algorithm 1, we are essentially doing an approximate EP in non-linear systems. We compute the moments (T. P. Minka 2008) of the temporary factor $q'(\mathbf{x}_t)$ via

$$\mu_t = \mu^{\setminus i} + \Sigma^{\setminus i}\nabla_\mu^\top, \quad (4.41)$$

$$\Sigma_t = \Sigma^{\setminus i} - \Sigma^{\setminus i}(\nabla_\mu^\top \nabla_\mu - 2\nabla_\Sigma)\Sigma^{\setminus i} \quad (4.42)$$

$$\nabla_\mu := d \log Z_i / d\mu^{\setminus i}, \quad (4.43)$$

$$\nabla_\Sigma := d \log Z_i / d\Sigma^{\setminus i} \quad (4.44)$$

- **Update:** We construct a Gaussian from the moments of $f_i(\mathbf{x}_t)q^{\setminus i}(\mathbf{x}_t)$ from the projection step above. This Gaussian represents the new marginal $q^{\text{new}}(\mathbf{x}_t)$, and we update the approximate factor with new value $q_i^{\text{new}}(\mathbf{x}_t) = q^{\text{new}}(\mathbf{x}_t)/q^{\setminus i}(\mathbf{x}_t)$.

Here, we explicitly compute a Gaussian approximation to the integral $\int f_i(\mathbf{x}_t)q^{\setminus i}(\mathbf{x}_t)d\mathbf{x}_t$, which allows us to show the equivalence of non-linear Kalman smoothers and (approximate) EP. For linear systems with Gaussian noise, the true factors $f_i(\mathbf{x}_t)$ are also Gaussian, and the Gaussian approximation is exact.

We are primarily interested in non-linear systems for which, by making a Gaussian approximation, we implicitly linearise the factors $f_i(\mathbf{x}_t)$ (as discussed in Section 4.2.2). We need to explicitly account for this implicit linearisation in our (approximate) EP update scheme.

4.3.2 Implicit Linearisation Requires Explicit Consideration

As discussed in the projection step of EP, in the exponential family, the required moments can be computed using the derivatives of the log-partition function $\log Z_i$ in (4.40) (Seeger 2005). However, for non-linear transition and measurement models in (4.1)–(4.2), computing Z_i is intractable. If we instead approximate the integral as a Gaussian, the (approximate) log partition function is then the log normaliser of this Gaussian. This approximation introduces an implicit linearisation 4.2.2. When computing Gaussian EP updates using the derivatives ∇_μ and ∇_Σ according to (4.42), it is crucial to explicitly account for the implicit linearisation assumption in the derivatives. Otherwise, the EP updates are inconsistent (Marc Peter Deisenroth and Mohamed 2012).

For example, in the measurement update, we form a joint Gaussian $\tilde{p}(\mathbf{x}_t, \mathbf{y}_t)$ as in (4.6). We implicitly linearise the measurement function h and determine the moments of $f_\Delta(\mathbf{x}_t)q^{\setminus\Delta}(\mathbf{x}_t)$, e.g., via the derivatives of

$$\log Z_\Delta(\boldsymbol{\mu}^{y\setminus\Delta}, \boldsymbol{\Sigma}^{y\setminus\Delta}) = \log \int f_\Delta(\mathbf{x}_t)q^{\setminus\Delta}(\mathbf{x}_t)d\mathbf{x}_t. \quad (4.45)$$

The implicit linearisation gives $\mathbf{y}_t = \mathbf{J}\mathbf{x}_t$, and the corresponding gradients are (Marc Peter Deisenroth and Mohamed 2012)

$$\nabla_\mu = \frac{d \log Z_\Delta}{d\boldsymbol{\mu}_t^{y\setminus\Delta}} = \frac{d \log Z_\Delta}{d\boldsymbol{\mu}_t^{y\setminus\Delta}} \frac{\partial \boldsymbol{\mu}_t^{y\setminus\Delta}}{\partial \boldsymbol{\mu}_t^{x\setminus\Delta}} = (\mathbf{y}_t - \boldsymbol{\mu}_t^{y\setminus\Delta})^\top (\boldsymbol{\Sigma}_t^{y\setminus\Delta})^{-1} \mathbf{J}^\top \in \mathbb{R}^{1 \times D}, \quad (4.46)$$

$$\nabla_\Sigma = \frac{d \log Z_\Delta}{d\boldsymbol{\Sigma}_t^{y\setminus\Delta}} = \frac{d \log Z_\Delta}{d\boldsymbol{\Sigma}_t^{y\setminus\Delta}} \frac{\partial \boldsymbol{\Sigma}_t^{y\setminus\Delta}}{\partial \boldsymbol{\Sigma}_t^{x\setminus\Delta}} = \frac{1}{2} \left(\frac{d \log Z_\Delta}{d\boldsymbol{\mu}_t^{y\setminus\Delta\top}} \frac{d \log Z_\Delta}{d\boldsymbol{\mu}_t^{y\setminus\Delta}} - (\boldsymbol{\Sigma}_t^{y\setminus\Delta})^{-1} \right) \frac{\partial \boldsymbol{\Sigma}_t^{y\setminus\Delta}}{\partial \boldsymbol{\Sigma}_t^{x\setminus\Delta}} \in \mathbb{R}^{D \times D}, \quad (4.47)$$

$$\frac{\partial \boldsymbol{\mu}_t^{y\setminus\Delta}}{\partial \boldsymbol{\mu}_t^{x\setminus\Delta}} = \mathbf{J}^\top \in \mathbb{R}^{E \times D}, \quad (4.48)$$

$$\frac{\partial \boldsymbol{\Sigma}_t^{y\setminus\Delta}}{\partial \boldsymbol{\Sigma}_t^{x\setminus\Delta}} = \mathbf{J} \mathbb{I}_4 \mathbf{J}^\top \in \mathbb{R}^{E \times E \times D \times D}, \quad (4.49)$$

where $\mathbb{I}_4 \in \mathbb{R}^{D \times D \times D \times D}$ is an identity tensor. Note that with the implicit linear model $\mathbf{y}_t = \mathbf{J}\mathbf{x}_t$, the derivatives $\partial \boldsymbol{\mu}_t^{y\setminus\Delta} / \partial \boldsymbol{\Sigma}_t^{x\setminus\Delta}$ and $\partial \boldsymbol{\Sigma}_t^{x\setminus\Delta} / \partial \boldsymbol{\mu}_t^{y\setminus\Delta}$ vanish. Although for unscented transform $\boldsymbol{\mu}^{y\setminus\Delta}$ is a function of both mean $\boldsymbol{\mu}^{x\setminus\Delta}$ and the covariance $\boldsymbol{\Sigma}^{y\setminus\Delta}$ the derivatives $\partial \boldsymbol{\mu}^{y\setminus\Delta} / \partial \boldsymbol{\mu}^{x\setminus\Delta}$ and $\partial \boldsymbol{\Sigma}^{y\setminus\Delta} / \partial \boldsymbol{\Sigma}^{x\setminus\Delta}$ must equal the corresponding partial derivatives in (4.49), and $\partial \boldsymbol{\mu}^{y\setminus\Delta} / \partial \boldsymbol{\Sigma}^{x\setminus\Delta}$ and $\partial \boldsymbol{\Sigma}^{y\setminus\Delta} / \partial \boldsymbol{\mu}^{x\setminus\Delta}$ must be set to $\mathbf{0}$. Hence, the implicit linearisation expressed by the Gaussian approximation $Z_{\setminus\Delta}$ must be explicitly taken into account in the derivatives to guarantee consistent EP updates (Marc Peter Deisenroth and Mohamed 2012).

4.3.3 Derivative-based EP updates and Kalman smoother

An important advantage of taking these gradients explicitly into account is that we can interpret the EP computations in the context of classical Gaussian filtering and smoothing (Brian D O Anderson and John B Moore 2005). When the EP updates are performed in a specific order, a single iteration of EP is equivalent to Kalman smoothing. As in Kalman filtering, we start with the time update or “forward factor” ($\triangleright$) update, followed by a measurement update or “up factor” (Δ) update and repeat until we reach $t = T$. Then we move backwards until we reach $t = 0$ by updating the “back factor” $\triangleleft$ similar to backward pass of an RTS smoother.

As shown in Alg. 1, we initialise all the factors $q_i(\mathbf{x}_t)$, $i \in \{\triangleright, \Delta, \triangleleft\}$ with zero mean and infinite variance (zero precision). This implies that the division and multiplication by these densities corresponds to subtracting and adding zeros to precision matrices, respectively. Therefore, for the first pass, the cavity formation step leaves densities intact as division by a zero-precision Gaussian (subtracting zero) has no effect.

We could implement EP by using derivative based updates (4.44) or using Kalman filter equivalent terms (4.7)–(4.14). Updating the moments of the approximate distribution $q(\mathbf{x}_t)$ via the derivatives of the log-partition function recovers exactly the standard Gaussian EP updates in explicitly linearised dynamical systems (Qi and T. Minka 2003). In the following, we show that the approximate EP with implicit linearisation via any approximation method, when explicitly accounted for this linearisation, also recovers standard Kalman filter smoother updates.

In the following, we show equivalence of these updates and Kalman filters analytically. For the measurement update, the cavity distribution $q_{\setminus\Delta}(\mathbf{x}_t)$ corresponds to the time update $p(\mathbf{x}_t|\mathbf{y}_{1:t-1})$ during the forward pass. When performing the backward pass removing $q_{\triangleleft}$ has no effect in the first pass and the $q(\mathbf{x}_t) = q_{\setminus\triangleleft}(\mathbf{x}_t)$ corresponds to the filter distribution $p(\mathbf{x}_t|\mathbf{y}_{1:t})$. For the forward update, we have to follow a different procedure as we update the values $q_{\Delta}(\mathbf{x}_t)$ in context to be consistent. (Removing the zero precision $\triangleright$ factor has no effect)

4.3.3.1 Forward Message

For the forward message, the projection step involves computing the partition function

$$Z_{\triangleright}(\boldsymbol{\mu}_t^{\setminus\triangleright}, \boldsymbol{\Sigma}_t^{\setminus\triangleright}) = \int f_{\triangleright}(\mathbf{x}_t) q_{\setminus\triangleright}(\mathbf{x}_t) d\mathbf{x}_t, \quad (4.50)$$

where the forward factor is given by,

$$f_{\triangleright}(\mathbf{x}_t) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1}) q_{\setminus\triangleleft}(\mathbf{x}_{t-1}) d\mathbf{x}_{t-1} \quad (4.51)$$

We propose to approximate $f_{\triangleright}(\mathbf{x}_t)$ directly by a Gaussian $q_{\triangleright}(\mathbf{x}_t) \propto \mathcal{N}(\tilde{\boldsymbol{\mu}}^{\triangleright}, \tilde{\boldsymbol{\Sigma}}^{\triangleright})$. This approximation implicitly linearises the transition function f as the cavity $q_{\triangleleft}(\mathbf{x}_{t-1})$ is Gaussian. We then update the posterior $q(\mathbf{x}_t)$ by Gaussian multiplication, i.e., $q(\mathbf{x}_t) \propto q_{\triangleright}(\mathbf{x}_t)q_{\triangleleft}(\mathbf{x}_t)$.

4.3.3.2 Measurement Message

For the measurement message, the true factor $f_{\Delta}(\mathbf{x}_t)$ is an integral over the measurement function h :

$$f_{\Delta}(\mathbf{x}_t) = p(\mathbf{x}_t | \mathbf{y}_t) \propto p(\mathbf{y}_t | \mathbf{x}_t) p(\mathbf{x}_t), \quad (4.52)$$

$$= \int h(\mathbf{x}_t) \mathcal{N}(\mathbf{x}_t | \boldsymbol{\mu}_{t|t-1}^x, \boldsymbol{\Sigma}_{t|t-1}^x + \mathbf{R}). \quad (4.53)$$

The integral (4.53) is intractable. However, we can make a Gaussian approximation $\mathcal{N}(\mathbf{y} | \boldsymbol{\mu}_{t|t-1}^y, \boldsymbol{\Sigma}_{t|t-1}^y)$ to $f_{\Delta}(\mathbf{x}_t)$ via any moment matching transform. The partition function for the measurement function can then be written as

$$Z_{\Delta}(\boldsymbol{\mu}_t^{y \setminus \Delta}, \boldsymbol{\Sigma}_t^{y \setminus \Delta}) = \int f_{\Delta}(\mathbf{x}_t) q_{\triangleleft}(\mathbf{x}_t) d\mathbf{x}_t \propto \int f_{\Delta}(\mathbf{x}_t) \mathcal{N}(\mathbf{x}_t | \boldsymbol{\mu}_t^{x \setminus \Delta}, \boldsymbol{\Sigma}_t^{x \setminus \Delta}) d\mathbf{x}_t \quad (4.54)$$

We can now use derivatives ∇_{μ} and ∇_{Σ} as defined in (4.46) and (4.47), respectively, to update $q(\mathbf{x}_t)$.

Theorem 4.1. Filtering and EP

Applying EP forward and measurement updates in sequence, is equivalent to Kalman filtering. That is the moments of the $f_{\Delta}(\mathbf{x}_t)q_{\triangleleft}(\mathbf{x}_t)$ via derivatives of Z_{Δ}

$$\boldsymbol{\mu}_t^x = \boldsymbol{\mu}_t^{x \setminus \Delta} + \boldsymbol{\Sigma}_t^{x \setminus \Delta} \nabla_{\mu}^{\top}, \quad (4.55)$$

$$\boldsymbol{\Sigma}_t^x = \boldsymbol{\Sigma}_t^{x \setminus \Delta} - (\boldsymbol{\Sigma}_t^{x \setminus \Delta}) \left[\nabla_{\mu}^{\top} \nabla_{\mu} - 2 \nabla_{\Sigma} \right] (\boldsymbol{\Sigma}_t^{x \setminus \Delta}), \quad (4.56)$$

are identical to those obtained by any Gaussian filter (4.8)

$$\boldsymbol{\mu}_t^x = \boldsymbol{\mu}_t^{x \setminus \Delta} + K_t(\mathbf{y}_t - \boldsymbol{\mu}_t^{y \setminus \Delta}), \quad (4.57)$$

$$\boldsymbol{\Sigma}_t^x = \boldsymbol{\Sigma}_t^{x \setminus \Delta} - K_t \boldsymbol{\Sigma}_t^{y \setminus \Delta}, \quad (4.58)$$

where

$$K_t = \boldsymbol{\Sigma}_t^{xy \setminus \Delta} (\boldsymbol{\Sigma}_t^{y \setminus \Delta})^{-1}. \quad (4.59)$$

Proof The derivative-free or the standard Kalman update is

$$\boldsymbol{\mu}_{t|t}^x = \boldsymbol{\mu}_{t|t-1}^x + K(\mathbf{y}_t - \boldsymbol{\mu}_{t|t-1}^y), \quad (4.60)$$

$$\boldsymbol{\Sigma}_{t|t}^x = \boldsymbol{\Sigma}_{t|t-1}^x - K \boldsymbol{\Sigma}_{t|t-1}^{yx}. \quad (4.61)$$

Here, $K_t = \boldsymbol{\Sigma}_{t|t-1}^{xy} (\boldsymbol{\Sigma}_{t|t-1}^y)^{-1}$ is called the Kalman gain. For EP we work with cavity

distributions and we can state the standard Kalman update with cavity distributions as,

$$\boldsymbol{\mu}_t^x = \boldsymbol{\mu}_t^{x \setminus \Delta} + \mathbf{K}_t(\mathbf{y}_t - \boldsymbol{\mu}_t^{y \setminus \Delta}), \quad (4.62)$$

$$\boldsymbol{\Sigma}_t = \boldsymbol{\Sigma}_t^{x \setminus \Delta} - \mathbf{K}_t(\boldsymbol{\Sigma}_t^{yx \setminus \Delta}), \quad (4.63)$$

$$\text{where } \mathbf{K}_t = \boldsymbol{\Sigma}_t^{xy \setminus \Delta} (\boldsymbol{\Sigma}_t^{y \setminus \Delta})^{-1}. \quad (4.64)$$

In the following, we show that the derivative-free update equations and the derivative-based updates are identical for Gaussian smoothers: The log-partition function for the equation (4.54)

The mean update with derivative is

$$\boldsymbol{\mu}_t^x = \boldsymbol{\mu}_t^{x \setminus \Delta} + \boldsymbol{\Sigma}_t^{x \setminus \Delta} \boldsymbol{\nabla}_\mu^\top, \quad (4.65)$$

where

$$\boldsymbol{\nabla}_\mu = \frac{d \log Z_\Delta}{d \boldsymbol{\mu}_t^{x \setminus \Delta}}. \quad (4.66)$$

In the following, we show that the derivative-free update equations and the derivative-based updates are identical for Gaussian smoothers: The log-partition function for the equation

$$\log Z_\Delta = \log \mathcal{N}(\mathbf{y}_t | \boldsymbol{\mu}_t^{y \setminus \Delta}, \boldsymbol{\Sigma}_t^{y \setminus \Delta}) \quad (4.67)$$

and its derivative $\boldsymbol{\nabla}_\mu$ with respect to the mean $\boldsymbol{\mu}_t^{x \setminus \Delta}$ of the cavity distribution is given as

$$\frac{d \log Z_\Delta}{d \boldsymbol{\mu}_t^{x \setminus \Delta}} = \frac{\partial \log Z_\Delta}{\partial \boldsymbol{\mu}_t^{y \setminus \Delta}} \frac{\partial \boldsymbol{\mu}_t^{y \setminus \Delta}}{\partial \boldsymbol{\mu}_t^{x \setminus \Delta}} = (\mathbf{y}_t - \boldsymbol{\mu}_t^{y \setminus \Delta})^\top (\boldsymbol{\Sigma}_t^{y \setminus \Delta})^{-1} \boldsymbol{\Sigma}_t^{yx \setminus \Delta} (\boldsymbol{\Sigma}_t^{x \setminus \Delta})^{-1} \quad (4.68)$$

since

$$\frac{\partial \log Z_\Delta}{\partial \boldsymbol{\mu}_t^{y \setminus \Delta}} = (\mathbf{y}_t - \boldsymbol{\mu}_t^{y \setminus \Delta})^\top (\boldsymbol{\Sigma}_t^{y \setminus \Delta})^{-1}, \quad (4.69)$$

$$\frac{\partial \boldsymbol{\mu}_t^{y \setminus \Delta}}{\partial \boldsymbol{\mu}_t^{x \setminus \Delta}} = \boldsymbol{\Sigma}_t^{yx \setminus \Delta} (\boldsymbol{\Sigma}_t^{x \setminus \Delta})^{-1}. \quad (4.70)$$

Using these results in the derivative-based update of the mean, we obtain

$$\boldsymbol{\mu}_t^x = \boldsymbol{\mu}_t^{x \setminus \Delta} + \boldsymbol{\Sigma}_t^{x \setminus \Delta} \boldsymbol{\nabla}_\mu^\top \quad (4.71)$$

$$= \boldsymbol{\mu}_t^{x \setminus \Delta} + \boldsymbol{\Sigma}_t^{x \setminus \Delta} (\boldsymbol{\Sigma}_t^{x \setminus \Delta})^{-1} \boldsymbol{\Sigma}_t^{xy \setminus \Delta} (\boldsymbol{\Sigma}_t^{y \setminus \Delta})^{-1} (\mathbf{y}_t - \boldsymbol{\mu}_t^{y \setminus \Delta}) \quad (4.72)$$

$$= \boldsymbol{\mu}_t^{x \setminus \Delta} + \boldsymbol{\Sigma}_t^{xy \setminus \Delta} (\boldsymbol{\Sigma}_t^{y \setminus \Delta})^{-1} (\mathbf{y}_t - \boldsymbol{\mu}_t^{y \setminus \Delta}) \quad (4.73)$$

$$= \boldsymbol{\mu}_t^{x \setminus \Delta} + \mathbf{K}(\mathbf{y}_t - \boldsymbol{\mu}_t^{y \setminus \Delta}), \quad (4.74)$$

This equation is identical to the derivative-free mean update (4.63).

The derivative $\boldsymbol{\nabla}_\Sigma$ with respect to the variance $\boldsymbol{\Sigma}_t^{x \setminus \Delta}$ of the cavity distribution is given by

$$\boldsymbol{\nabla}_\Sigma = \frac{d \log Z_\Delta}{d \boldsymbol{\Sigma}_t^{x \setminus \Delta}} = \frac{\partial \log Z_\Delta}{\partial \boldsymbol{\Sigma}_t^{y \setminus \Delta}} \frac{\partial \boldsymbol{\Sigma}_t^{y \setminus \Delta}}{\partial \boldsymbol{\Sigma}_t^{x \setminus \Delta}}. \quad (4.75)$$

Since $\boldsymbol{\Sigma}_t^{y \setminus \Delta} = \mathbf{J} \boldsymbol{\Sigma}_t^{x \setminus \Delta} \mathbf{J}^\top$ we can write the above derivative by writing the chain rule

from the inner variables² as,

$$\nabla_{\Sigma} = J^{\top} \left[\frac{\partial \log Z_{\Delta}}{\partial \Sigma_t^{y \setminus \Delta}} \right] J \quad (4.76)$$

$$= J^{\top} \left[\Sigma_t^{y \setminus \Delta^{-1}} - \left(\Sigma_t^{y \setminus \Delta^{-1}} \right) (y_t - \mu_t^{y \setminus \Delta}) (y_t - \mu_t^{y \setminus \Delta})^{\top} \left(\Sigma_t^{y \setminus \Delta^{-1}} \right) \right] J \quad (4.77)$$

Using the definition of $\frac{\partial \log Z_{\Delta}}{\partial \mu_t^{y \setminus \Delta}}$ in (4.69) and rearranging terms we get

$$\nabla_{\Sigma} = \frac{1}{2} \left[\underbrace{J^{\top} \frac{\partial \log Z_{\Delta}}{(\partial \mu_t^{y \setminus \Delta})^{\top}}}_{\nabla_{\mu}^{\top}} \underbrace{\frac{\partial \log Z_{\Delta}}{\partial \mu_t^{y \setminus \Delta}} J}_{\nabla_{\mu}} - J^{\top} (\Sigma_t^{y \setminus \Delta})^{-1} J \right] \quad (4.78)$$

$$= \frac{1}{2} \left[\nabla_{\mu}^{\top} \nabla_{\mu} - J^{\top} (\Sigma_t^{y \setminus \Delta})^{-1} J \right] \quad (4.79)$$

We first compute the intermediate term $\nabla_{\mu}^{\top} \nabla_{\mu} - 2\nabla_{\Sigma}$ from the derivative-based update in the algorithm. Substituting the value of ∇_{Σ} from (4.79), we obtain

$$\nabla_{\mu}^{\top} \nabla_{\mu} - 2\nabla_{\Sigma} = \nabla_{\mu}^{\top} \nabla_{\mu} - \nabla_{\mu}^{\top} \nabla_{\mu} + J^{\top} (\Sigma_t^{y \setminus \Delta})^{-1} J \quad (4.80)$$

$$= J^{\top} (\Sigma_t^{y \setminus \Delta})^{-1} J \quad (4.81)$$

Building on this we compute

$$(\Sigma_t^{x \setminus \Delta}) \left[\nabla_{\mu}^{\top} \nabla_{\mu} - 2\nabla_{\Sigma} \right] (\Sigma_t^{x \setminus \Delta}) = (\Sigma_t^{x \setminus \Delta}) \left[\underbrace{(\Sigma_t^{x \setminus \Delta})^{-1} \Sigma_t^{xy \setminus \Delta} (\Sigma_t^{y \setminus \Delta})^{-1} \Sigma_t^{yx \setminus \Delta} (\Sigma_t^{x \setminus \Delta})^{-1}}_{J^{\top}} \underbrace{J}_{J} \right] (\Sigma_t^{x \setminus \Delta}) \quad (4.82)$$

$$= \Sigma_t^{xy \setminus \Delta} (\Sigma_t^{y \setminus \Delta})^{-1} \Sigma_t^{yx \setminus \Delta} \quad (4.83)$$

$$= K_t \Sigma_t^{yx \setminus \Delta}, \quad (4.84)$$

where we used the result from (4.81).

The derivative-based update is

$$\Sigma_t^x = \Sigma_t^{x \setminus \Delta} - (\Sigma_t^{x \setminus \Delta}) \left[\nabla_{\mu}^{\top} \nabla_{\mu} - 2\nabla_{\Sigma} \right] (\Sigma_t^{x \setminus \Delta}) = \Sigma_t^{x \setminus \Delta} - K_t \Sigma_t^{yx \setminus \Delta}, \quad (4.85)$$

which is same as the classical Kalman filter update, where K_t is the Kalman gain. ■

4.3.4 Backward Message

If we compute the backward message updates after a full forward sweep, i.e., a full Kalman filtering forward pass, then the backward update, when starting from at the final time step T , is equivalent to the smoothing update in RTS smoothers. If we have only performed a single forward and measurement update for all observations, then the back factor $q_{\triangleleft}(\mathbf{x}_t)$ has zero precision, i.e., the cavity distribution $q_{n \triangleleft}(\mathbf{x}_t) = q(\mathbf{x}_t)/q_{\triangleleft}(\mathbf{x}_t)$ is identical to the the filtering distribution $p(\mathbf{x}_t | \mathbf{y}_{1:t})$. To compute the backward message

²Reverse mode gradient

$q_{\triangleleft}(\mathbf{x}_t)$, we require the partition function

$$Z_{\triangleleft}(\boldsymbol{\mu}_t^{\triangleleft}, \boldsymbol{\Sigma}_t^{\triangleleft}) = \int f_{\triangleleft}(\mathbf{x}_t) q_{\triangleleft}(\mathbf{x}_t) d\mathbf{x}_t \propto \int f_{\triangleleft}(\mathbf{x}_t) \mathcal{N}(\mathbf{x}_t | \boldsymbol{\mu}_t^{\triangleleft}, \boldsymbol{\Sigma}_t^{\triangleleft}) d\mathbf{x}_t, \quad (4.86)$$

where

$$f_{\triangleleft}(\mathbf{x}_t) = \int p(\mathbf{x}_{t+1} | \mathbf{x}_t) q_{\triangleright}(\mathbf{x}_{t+1}) d\mathbf{x}_{t+1}. \quad (4.87)$$

The true factor $f_{\triangleleft}(\mathbf{x}_t)$ in (4.87) accounts for the coupling between $\mathbf{x}_t$ and $\mathbf{x}_{t+1}$, via averaging over the $\mathbf{x}_{t+1}$ to obtain the back factor (4.87), which was lost in assuming the full factorization in (4.28). Using (4.87) in (4.86) and reordering the integration yields

$$Z_{\triangleleft}(\boldsymbol{\mu}_t^{\triangleleft}, \boldsymbol{\Sigma}_t^{\triangleleft}) \propto \int q_{\triangleright}(\mathbf{x}_{t+1}) \int p(\mathbf{x}_{t+1} | \mathbf{x}_t) q_{\triangleleft}(\mathbf{x}_t) d\mathbf{x}_t d\mathbf{x}_{t+1}. \quad (4.88)$$

We approximate the inner integral in (4.88) as $\mathcal{N}(\mathbf{x}_{t+1} | \tilde{\boldsymbol{\mu}}^{\triangleleft}, \tilde{\boldsymbol{\Sigma}}^{\triangleleft})$ by moment matching. Note that $\tilde{\boldsymbol{\mu}}^{\triangleleft}$ and $\tilde{\boldsymbol{\Sigma}}^{\triangleleft}$ are functions of $\boldsymbol{\mu}_t^{\triangleleft}$ and $\boldsymbol{\Sigma}_t^{\triangleleft}$. This Gaussian approximation implicitly linearises transition function f . Now, (4.88) can be computed analytically, and we obtain a Gaussian approximation

$$\tilde{Z}_{\triangleleft} = Z_{\triangleleft} \approx \mathcal{N}(\boldsymbol{\mu}_{t+1}^{\triangleright} | \tilde{\boldsymbol{\mu}}^{\triangleleft}, \tilde{\boldsymbol{\Sigma}}^{\triangleleft} + \boldsymbol{\Sigma}_{t+1}^{\triangleright}) \quad (4.89)$$

that allows us to update the moments of $q(\mathbf{x}_t)$ and the message $q_{\triangleleft}(\mathbf{x}_t)$ via derivatives of $\tilde{Z}_{\triangleleft}$.

Theorem 4.2. EP and RTS Smoother

The backward message updates via the derivatives of $\tilde{Z}_{\triangleleft}$

$$\boldsymbol{\mu}_t = \boldsymbol{\mu}_t^{\triangleleft} + \boldsymbol{\Sigma}_t^{\triangleleft} \boldsymbol{\nabla}_{\boldsymbol{\mu}}^{\top}, \quad (4.90)$$

$$\boldsymbol{\Sigma}_t = (\boldsymbol{\Sigma}_t^{\triangleleft}) - (\boldsymbol{\Sigma}_t^{\triangleleft}) \left[\boldsymbol{\nabla}_{\boldsymbol{\mu}}^{\top} \boldsymbol{\nabla}_{\boldsymbol{\mu}} - 2 \boldsymbol{\nabla}_{\boldsymbol{\Sigma}} \right] (\boldsymbol{\Sigma}_t^{\triangleleft}), \quad (4.91)$$

are identical to those obtained by the RTS smoothing update

$$\boldsymbol{\mu}_t^x = \boldsymbol{\mu}_t^{x\triangleleft} + L_t (\boldsymbol{\mu}_{t+1} - \boldsymbol{\mu}_{t+1|t}^{x\triangleleft}), \quad (4.92)$$

$$\boldsymbol{\Sigma}_{t|T}^x = \boldsymbol{\Sigma}_{t|t}^{x\triangleleft} + L_t \left(\boldsymbol{\Sigma}_{t+1|T}^x - \boldsymbol{\Sigma}_{t+1|t}^{x\triangleleft} \right) L_t^{\top}, \quad (4.93)$$

where

$$L_t = \left[\boldsymbol{\Sigma}_{t,t+1|t}^x \left(\boldsymbol{\Sigma}_{t+1|t}^x \right)^{-1} \right]. \quad (4.94)$$

Proof The derivative-free update is

$$\boldsymbol{\mu}_t^x = \boldsymbol{\mu}_t^{x\triangleleft} + L_t (\boldsymbol{\mu}_{t+1} - \boldsymbol{\mu}_{t+1}^{x\triangleleft}), \quad (4.95)$$

$$\boldsymbol{\Sigma}_{t|T}^x = \boldsymbol{\Sigma}_{t|t}^{x\triangleleft} + L_t \left(\boldsymbol{\Sigma}_{t+1|T}^x - \boldsymbol{\Sigma}_{t+1|t}^{x\triangleleft} \right) L_t^{\top}, \quad (4.96)$$

where

$$L_t = \boldsymbol{\Sigma}_{t,t+1|T}^x (\boldsymbol{\Sigma}_{t+1}^{x\triangleleft})^{-1} \quad (4.97)$$

The Gaussian approximation for the log partition function is

$$\tilde{Z}_{\triangleleft} = \mathcal{N}(\boldsymbol{\mu}_{t+1}^{\triangleleft} | \tilde{\boldsymbol{\mu}}^{\triangleleft}, \tilde{\boldsymbol{\Sigma}}^{\triangleleft} + \boldsymbol{\Sigma}_{t+1}^{\triangleleft}) \quad (4.98)$$

Its derivative $\nabla_{\boldsymbol{\mu}}$ with respect to the mean $\boldsymbol{\mu}_t^{x\triangleleft}$ of the cavity distribution is given by

$$\frac{d \log Z_{\triangleleft}}{d \boldsymbol{\mu}_t^{x\triangleleft}} = \frac{\partial \log Z_{\triangleleft}}{\partial \boldsymbol{\mu}_{t+1}^{x\triangleleft}} \frac{\partial \boldsymbol{\mu}_{t+1}^{x\triangleleft}}{\partial \boldsymbol{\mu}_t^{x\triangleleft}} = (\boldsymbol{\Sigma}_{t,t+1|T}^x (\tilde{\boldsymbol{\Sigma}}_t^{\triangleleft})^{-1})^\top \Gamma^{-1} (\boldsymbol{\mu}_{t+1}^{\triangleleft} - \tilde{\boldsymbol{\mu}}_t^{\triangleleft}), \quad (4.99)$$

where we defined Γ as

$$\Gamma = (\tilde{\boldsymbol{\Sigma}}^{\triangleleft} + \boldsymbol{\Sigma}_t^{x\triangleleft}) \quad (4.100)$$

to simplify the notation. In the following we expand Γ . First, we multiply by identity matrices on both sides,

$$\Gamma^{-1} = [I] (\tilde{\boldsymbol{\Sigma}}^{\triangleleft} + \boldsymbol{\Sigma}_{t+1}^{x\triangleleft})^{-1} [I] \quad (4.101)$$

and then expand identities to write,

$$\Gamma^{-1} = \left[(\tilde{\boldsymbol{\Sigma}}^{\triangleleft})^{-1} \tilde{\boldsymbol{\Sigma}}^{\triangleleft} \right] \left(\tilde{\boldsymbol{\Sigma}}^{\triangleleft} + \boldsymbol{\Sigma}_{t+1}^{x\triangleleft} \right)^{-1} \left[\boldsymbol{\Sigma}_{t+1}^{x\triangleleft} (\boldsymbol{\Sigma}_{t+1}^{\triangleleft})^{-1} \right]. \quad (4.102)$$

Rearranging the brackets we get,

$$\Gamma^{-1} = (\tilde{\boldsymbol{\Sigma}}^{\triangleleft})^{-1} \left[\tilde{\boldsymbol{\Sigma}}^{\triangleleft} \left(\tilde{\boldsymbol{\Sigma}}^{\triangleleft} + \boldsymbol{\Sigma}_{t+1}^{x\triangleleft} \right)^{-1} \boldsymbol{\Sigma}_{t+1}^{x\triangleleft} \right] (\boldsymbol{\Sigma}_{t+1}^{\triangleleft})^{-1} \quad (4.103)$$

The term in the square bracket resembles the matrix identity,

$$A (A + B)^{-1} B = \left(A^{-1} + B^{-1} \right)^{-1}.$$

With $A = \tilde{\boldsymbol{\Sigma}}^{\triangleleft}$ and $B = \boldsymbol{\Sigma}_{t+1}^{x\triangleleft}$ the terms in the square bracket can be rewritten as,

$$\Gamma^{-1} = (\tilde{\boldsymbol{\Sigma}}^{\triangleleft})^{-1} \left[\left((\tilde{\boldsymbol{\Sigma}}^{\triangleleft})^{-1} + (\boldsymbol{\Sigma}_{t+1}^{x\triangleleft})^{-1} \right)^{-1} \right] (\boldsymbol{\Sigma}_{t+1}^{\triangleleft})^{-1}. \quad (4.104)$$

If the next state approximation is given as by the smoothing density $\mathcal{N}(\mathbf{x}_{t+1} | \boldsymbol{\mu}_{t+1}, \boldsymbol{\Sigma}_{t+1})$ our Gaussian estimate for the next state must match, i.e., $(\tilde{\boldsymbol{\Sigma}}^{\triangleleft})^{-1} + (\boldsymbol{\Sigma}_{t+1}^{x\triangleleft})^{-1} = \boldsymbol{\Sigma}_{t+1}^{-1}$. To obtain forward cavity we need to remove back cavity from the approximate factorisation, i.e., $q_{\triangleleft}(x_t) \propto q(x_{t+1})/q_{\triangleleft}(x_t)$. Ignoring normalisation constant we can use Gaussian division formula ([Brandt Petersen et al. 2012](#))

$$(\boldsymbol{\Sigma}_{t+1}^{x\triangleleft})^{-1} = \boldsymbol{\Sigma}_{t+1}^{-1} - (\tilde{\boldsymbol{\Sigma}}^{\triangleleft})^{-1}, \quad (4.105)$$

$$(\boldsymbol{\Sigma}_{t+1}^{x\triangleleft})^{-1} \boldsymbol{\mu}_{t+1}^{\triangleleft} = \boldsymbol{\Sigma}_{t+1}^{-1} \boldsymbol{\mu}_{t+1} - (\tilde{\boldsymbol{\Sigma}}^{\triangleleft})^{-1} \tilde{\boldsymbol{\mu}}_t^{\triangleleft} \quad (4.106)$$

Right multiplying Γ^{-1} by $(\boldsymbol{\mu}_{t+1}^{\triangleleft} - \tilde{\boldsymbol{\mu}}_t^{\triangleleft})$,

$$\begin{aligned} \Gamma^{-1} (\boldsymbol{\mu}_{t+1}^{\triangleleft} - \tilde{\boldsymbol{\mu}}_t^{\triangleleft}) &= (\tilde{\boldsymbol{\Sigma}}^{\triangleleft})^{-1} \boldsymbol{\Sigma}_{t+1} \left[(\boldsymbol{\Sigma}_{t+1}^{x\triangleleft})^{-1} \boldsymbol{\mu}_{t+1}^{\triangleleft} - (\boldsymbol{\Sigma}_{t+1}^{x\triangleleft})^{-1} \tilde{\boldsymbol{\mu}}_t^{\triangleleft} \right] \\ &= (\tilde{\boldsymbol{\Sigma}}^{\triangleleft})^{-1} \boldsymbol{\Sigma}_{t+1} \left[\boldsymbol{\Sigma}_{t+1}^{-1} \boldsymbol{\mu}_{t+1} - (\tilde{\boldsymbol{\Sigma}}^{\triangleleft})^{-1} \tilde{\boldsymbol{\mu}}_t^{\triangleleft} - (\boldsymbol{\Sigma}_{t+1}^{x\triangleleft})^{-1} \tilde{\boldsymbol{\mu}}_t^{\triangleleft} \right] \end{aligned} \quad (4.107)$$

where we used equation (4.106) to expand the first term in the square bracket.

$$\Gamma^{-1} (\boldsymbol{\mu}_{t+1}^{\triangleleft} - \tilde{\boldsymbol{\mu}}_t^{\triangleleft}) = (\tilde{\boldsymbol{\Sigma}}^{\triangleleft})^{-1} \boldsymbol{\Sigma}_{t+1} \left[\boldsymbol{\Sigma}_{t+1}^{-1} \boldsymbol{\mu}_{t+1} - ((\tilde{\boldsymbol{\Sigma}}^{\triangleleft})^{-1} - (\boldsymbol{\Sigma}_{t+1}^{x\triangleleft})^{-1}) \tilde{\boldsymbol{\mu}}_t^{\triangleleft} \right] \quad (4.108)$$

For this transition transformation $\mathbf{J} = \boldsymbol{\Sigma}_{t,t+1|T}^x (\boldsymbol{\Sigma}_t^{\triangleleft})^{-1}$ and we similar to derivation

used to obtain (4.81) we have the square bracket term

$$\left[\nabla_{\mu}^{\top} \nabla_{\mu} - 2 \nabla_{\Sigma} \right] = J^{\top} (\tilde{\Sigma}^{\setminus \triangleleft} + \Sigma_{t+1}^{\setminus \triangleright})^{-1} J \quad (4.109)$$

The covariance update with derivatives

$$\begin{aligned} (\Sigma_t^{\setminus \triangleleft}) \left[\nabla_{\mu}^{\top} \nabla_{\mu} - 2 \nabla_{\Sigma} \right] (\Sigma_t^{\setminus \triangleleft}) &= (\Sigma_t^{\setminus \triangleleft}) \left[\underbrace{(\Sigma_t^{\setminus \triangleleft})^{-1} \Sigma_{t,t+1|t}^{x \setminus \triangleleft \top}}_{\Gamma^{-1}} (\tilde{\Sigma}^{\setminus \triangleleft} + \Sigma_{t+1}^{\setminus \triangleright})^{-1} \underbrace{\Sigma_{t,t+1|t}^{x \setminus \triangleleft}}_J (\Sigma_t^{\setminus \triangleleft})^{-1} \right] (\Sigma_t^{\setminus \triangleleft}), \\ &= \Sigma_{t,t+1|T}^x \underbrace{(\tilde{\Sigma}^{\setminus \triangleleft} + \Sigma_{t+1}^{\setminus \triangleright})^{-1}}_{\Gamma^{-1}} \Sigma_{t,t+1|t}^{x \setminus \triangleleft} \end{aligned} \quad (4.110)$$

The central bracket term is Γ^{-1} and from equation (4.108) we can now rewrite Γ^{-1} as,

$$\begin{aligned} \Gamma^{-1} &= (\tilde{\Sigma}^{\setminus \triangleleft})^{-1} [\Sigma_{t+1}] (\Sigma_{t+1}^{\setminus \triangleright})^{-1}, \\ &= (\tilde{\Sigma}^{\setminus \triangleleft})^{-1} [\Sigma_{t+1}] \left[(\Sigma_{t+1})^{-1} - (\tilde{\Sigma}^{\setminus \triangleleft})^{-1} \right], \\ &= (\tilde{\Sigma}^{\setminus \triangleleft})^{-1} \left[I - \Sigma_{t+1} (\tilde{\Sigma}^{\setminus \triangleleft})^{-1} \right], \\ &= (\tilde{\Sigma}^{\setminus \triangleleft})^{-1} \left[\tilde{\Sigma}^{\setminus \triangleleft} - \Sigma_{t+1} \right] (\tilde{\Sigma}^{\setminus \triangleleft})^{-1}. \end{aligned} \quad (4.111)$$

Substituting (4.111) in (4.110)

$$(\Sigma_t^{\setminus \triangleleft}) \left[\nabla_{\mu}^{\top} \nabla_{\mu} - 2 \nabla_{\Sigma} \right] (\Sigma_t^{\setminus \triangleleft}) = L \left[\tilde{\Sigma}^{\setminus \triangleleft} - \Sigma_{t+1} \right] L^{\top} \quad (4.112)$$

The derivative based update is,

$$\Sigma_t = (\Sigma_t^{\setminus \triangleleft}) - (\Sigma_t^{\setminus \triangleleft}) \left[\nabla_{\mu}^{\top} \nabla_{\mu} - 2 \nabla_{\Sigma} \right] (\Sigma_t^{\setminus \triangleleft}) \quad (4.113)$$

$$= (\Sigma_t^{\setminus \triangleleft}) - L \left[\tilde{\Sigma}^{\setminus \triangleleft} - \Sigma_{t+1} \right] L^{\top} \quad (4.114)$$

$$= (\Sigma_t^{\setminus \triangleleft}) + L \left[\Sigma_{t+1} - \tilde{\Sigma}^{\setminus \triangleleft} \right] L^{\top} \quad (4.115)$$

Which exactly same as derivative free smoothing/ RTS smoothing update. $\blacksquare$

The Kalman filtering updates and the derivatives based updates with implicit linearisation are identical as shown by the Theorems 4.1 and 4.2. Thus we can use Kalman filtering updates which are numerically stable to run EP algorithm. EP algorithm is be numerically unstable for non-linear approximations (Seeger 2005), whereas if we set the partial derivatives in (4.49) we get numerically stable algorithm for EP updates.

4.3.5 Applying approximate EP to a non-linear benchmark

We apply the EP Algorithm to the non-linear benchmark (4.17). We use the unscented transform to approximate the intractable integrals. Results are shown in figure 4.1. The first pass of EP produces the same result as the RTS smoother. Subsequent iterations improve the result compared to the smoothing. However, we clearly see oscillations, not convergence to a fixed point. It is also important to note that in case of non-linear systems, approximate EP may deteriorate and produce results inferior than

the RTS smoother. It may even diverge and fail to perform the estimation task. One can employ heuristics to determine termination conditions, but there are no established stopping or convergence criteria. In the next section, we review some techniques that can help alleviate this oscillatory behaviour³.

4.4 Improving EP by damping and Alpha-Divergence

4.4.1 Damped EP

Oscillatory behaviour of EP observed in the Figure 4.1 is well-known (Heskes and Zoeter 2003). Standard expectation propagation as an iteration scheme has no termination criteria and it is not guaranteed to converge (T. P. Minka 2001a). EP operates on one factor at a time. However, one can establish a convergence criterion based on the covariance of all the factors in algorithm. This is called double-loop EP (Heskes and Zoeter 2003). In double-loop EP, the inner loop is standard EP with damped updates for each factor. The outer loop based on all the factors is used to pick the optimal damping rate for the inner EP update. However, this outer loop is prohibitively expensive for most practical applications. Moreover, non-linear systems discussed in this article use *approximate* EP, and the approximation would lose any convergence guarantees associated with the double-loop version. However, instead of computing an optimal optimal damping for each iteration, we can use a fixed damping for the EP updates, which results in the so-called damped EP (T. P. Minka 2004). This damping helps the algorithm avoid limit cycles (Heskes and Zoeter 2003). We This changes the EP update to a two step process, similar to equation (4.22) we compute the new $q(\mathbf{x}_t)'$ distribution.

$$q_i(\mathbf{x}_t)' = \text{proj} \left[f_i(\mathbf{x}_t) q^{\setminus i}(\mathbf{x}_t) \parallel q_i(\mathbf{x}_t) q^{\setminus i}(\mathbf{x}_t) \right] / q^{\setminus i}(\mathbf{x}_t) \quad (4.116)$$

However, this $q(\mathbf{x}_t)'$ is not the the new distribution as in (4.22), we now apply damping to obtain final $q(\mathbf{x}_t)^{new}$ as,

$$q_i(\mathbf{x}_t)^{new} = (q_i(\mathbf{x}_t)')^\gamma \left(q_i(\mathbf{x}_t)^{old} \right)^{1-\gamma}. \quad (4.117)$$

The damping factor $\gamma \in (0, 1]$ is a hyper-parameter, which can be tuned depending on the filtering task. In the Figure 4.2, we see that a damping factor of 0.8 reduces oscillations and stabilises EP in this particular example. We chose 0.8 to illustrate the effect of damping on oscillations, in the experiment section we study the impact of varying damping γ .

³In approximate EP, the approximation for non-linear systems may dominate the performance over EP or advanced techniques discussed in the section 4.4.2.

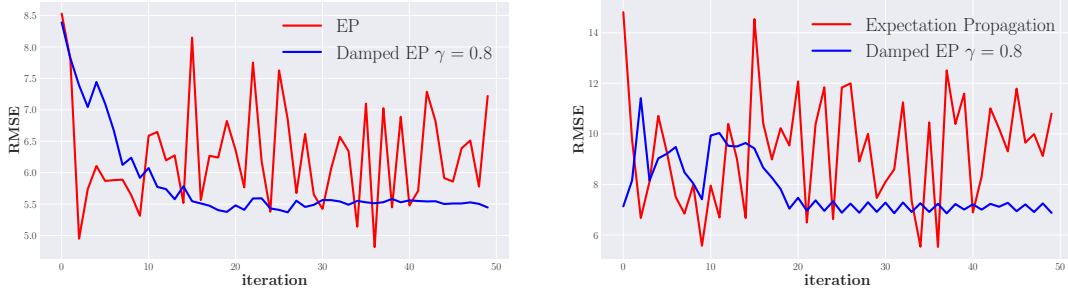


Figure 4.2: In this figure we demonstrate that damping reduces oscillatory behaviour of the algorithm and stabilises estimator. The settings are identical to those in the Figure 4.1. The filtering and smoothing results from Figure 4.1 are omitted for clarity. We see that damped EP significantly improves the estimator in terms of both the Root mean square error (RMSE) and Negative Log-Likelihood (NLL) metrics.

4.4.2 Alpha Divergence

In some cases, the damping factor alone may not necessarily prevent oscillations. This is particularly true for non-linear systems, where the true posteriors are often multi-modal. An iterative Gaussian approximation to the multi-modal density may alternate among different modes. The standard EP used up until now tends to cover the mass and approximately match the first two moments of the posterior. Depending on the application, one may want to encourage algorithm to cover most of the probability mass or a mode averaging behaviour. On the other hand, it may be desirable that the algorithm picks one of the modes rather than averaging them, e.g., in case of tracking a car moving past a junction. We can modify the optimisation objective to encourage the desired behaviour by using the power EP instead (T. P. Minka 2004). Power EP minimises the α -divergence (Amari 1985) instead of the $\mathcal{KL}$ -divergence used by standard EP (T. P. Minka 2004).

Definition 4.2. Alpha Divergence

The α -divergence is a generalisation of $\mathcal{KL}$ divergence (Zhu and Rohwer 1995).

We follow (Zhu and Rohwer 1995; T. T. Minka 2005) to define the α -divergence as

$$D_{\alpha}(p||q) := \frac{\int \alpha p(x) + (1 - \alpha)q(x) - p(x)^{\alpha}q(x)^{1-\alpha} dx}{\alpha(1 - \alpha)}. \quad (4.118)$$

The α -divergence is zero if and only if $p = q$ and positive otherwise. Hence, we can use this as a cost function in Definition 4.1 of $\text{proj}[\cdot]$. For $\alpha = 1$ we obtain the $\mathcal{KL}$ -divergence (Zhu and Rohwer 1995), which is the approximate EP algorithm we considered in the previous section 4.3. For $\alpha = 0.5$, the objective corresponds to the Hellinger distance.

While typically it would be difficult to compute or even approximate (4.118), we can work around this challenge by using the following property of α divergences:

Algorithm 2 Power EP for Dynamical Systems

- 1: **Init:** Set all factors f_i^α to $\mathcal{N}(\mathbf{0}, \infty I)$; Set $q(\mathbf{x}_1) = p(\mathbf{x}_1)$ and marginals $q(\mathbf{x}_{t \neq 1}) = \mathcal{N}(\mathbf{0}, 10^{10} I)$
- 2: **repeat**
- 3: **for** $t = 1$ to T **do**
- 4: **for** all factors f_i^α , where $i = \triangleright, \Delta, \triangleleft$ **do**
- 5: Compute cavity distribution

$$q_{\alpha}^{\setminus i}(\mathbf{x}_t) = q(\mathbf{x}_t) / f_i^\alpha(\mathbf{x}_t) \quad (4.119)$$

with

- 6: **Project:**

$$q'(\mathbf{x}_t) = \text{proj} \left[f_i^\alpha(\mathbf{x}_t) q_{\alpha}^{\setminus i}(\mathbf{x}_t) \parallel q_i(\mathbf{x}_t) q_{\alpha}^{\setminus i}(\mathbf{x}_t) \right] \quad (4.120)$$

- 7: **Update factor:**

$$f_i^\alpha(\mathbf{x}_t)^{\text{new}} = f_i^\alpha(\mathbf{x}_t)^{1-\gamma} \left(\frac{q'(\mathbf{x}_t)}{q_{\alpha}^{\setminus i}(\mathbf{x}_t)} \right)^\gamma = f_i^\alpha(\mathbf{x}_t) \left(\frac{q'(\mathbf{x}_t)}{q(\mathbf{x}_t)} \right)^\gamma. \quad (4.121)$$

- 8: **Update posterior:**

$$q(\mathbf{x}_t)^{\text{new}} = q(\mathbf{x}_t) \left(\frac{f_i^\alpha(\mathbf{x}_t)^{\text{new}}}{f_i^\alpha(\mathbf{x}_t)} \right)^{1/\alpha} = q(\mathbf{x}_t) \left(\frac{q'(\mathbf{x}_t)}{q(\mathbf{x}_t)} \right)^{\gamma/\alpha}. \quad (4.122)$$

- 9: **end for**

- 10: **end for**

- 11: **until** Convergence or maximum number of iterations exceeded
-

Proposition 4.1

Any stationary point q of $D_\alpha(p||q)$ is also a stationary point of $\text{proj} [p(x)^\alpha \parallel q(x)^{1-\alpha}]$ by appropriate scaling factor that normalises the respective distributions (T. T. Minka 2005). More generally, we can minimise any $D_\alpha(p||q)$ by using any other $D_{\alpha'}(p||q)$ (T. T. Minka 2005; Yedidia, Freeman, and Weiss 2005; Hernandez-Lobato et al. 2016).



Hence, we can apply the Kalman filter equation in Section 4.3.1, which implements $\alpha = 1$ to minimise any $D_\alpha(p||q)$ up to a scaling factor. Since we make a Gaussian approximation to the posterior density, we can calculate this scaling factor that normalises the Gaussian. Algorithm 2 describes EP using the α -divergence. The $\text{proj}[\cdot]$ by equation (4.120) in step 6 is given by

$$q'(x) = \text{proj} [p(x)^\alpha \parallel q(x)^{1-\alpha}] \quad (4.123)$$

$$q(x)^{\text{new}} = q(x)^\gamma q'(x)^{1-\gamma}. \quad (4.124)$$

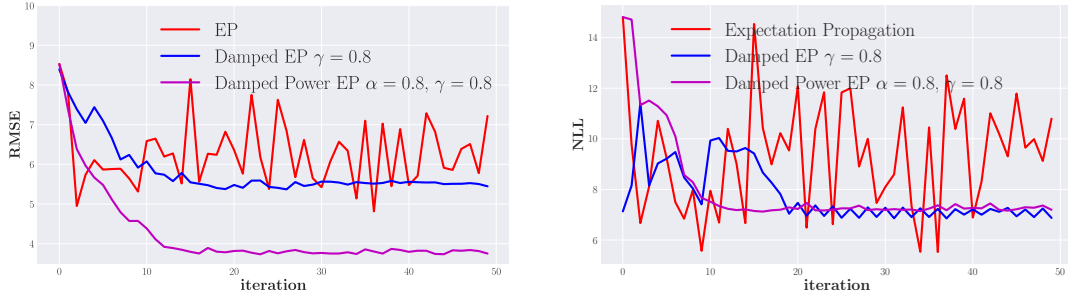


Figure 4.3: In this figure we build on result from the previous figures with once again using the same settings and seeds as in the Figure 4.1. In addition to the damping as in Figure 4.2, we now add power (fractional) EP results in this figure. We see that using power EP we further improve the performance of the standard EP iterations. The power of EP and damping factor are identical, which improves numerical stability as the power and damping cancel out, see Equation 4.122 in Algorithm 2.

The advantage of applying α divergence EP to the same problem as before is visible in Figure 4.3. The oscillations are substantially reduced in the log-likelihood, and the RMSE is lower than that of the damped EP. The same choice of damping and α is convenient mathematically as the power and damping cancel each other and this simplifies the posterior update step 8 in Algorithm 2. This further improves the stability of the algorithm as we not have to raise a Gaussian to a fractional power, which can be numerically unstable in the non-linear systems.

4.5 Results

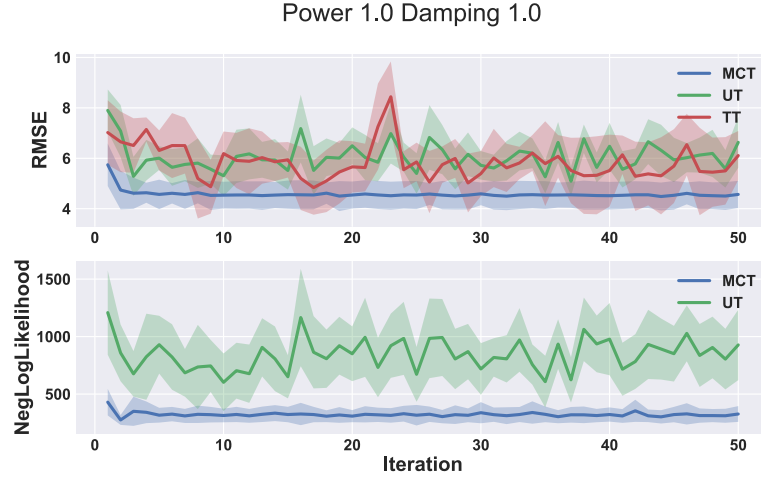
In this section we evaluate the proposed iterative smoother on two benchmark problems.

4.5.1 Non-Linear Growth Model

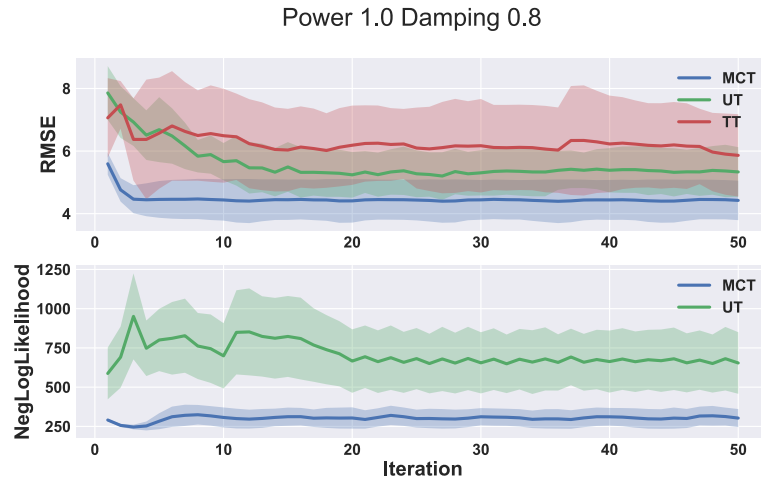
First we consider the non-linear benchmark from section 4.2.3. The example shown in Figure 4.3 is using the unscented transform for a single seed. We now repeat the same experiment with 10 different random seeds to evaluate the robustness of different algorithms. Our first goal is to evaluate power EP for different values of power α on the non-linear benchmark problem, across different transforms and EP iterations.

4.5.1.1 Linearisation Methods and Power EP

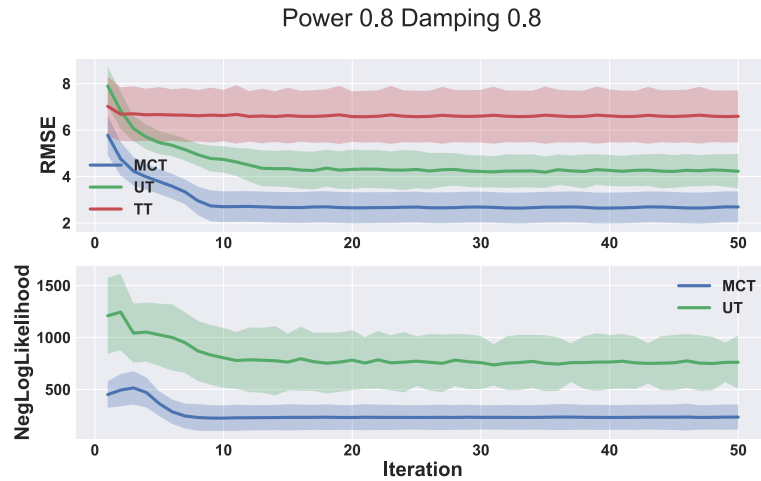
We test α -divergence EP with three different linearisation techniques, Taylor transform (TT), unscented transform (UT) and Monte-Carlo transform (MCT). In Figure 4.4 a) we run standard approximate EP, i.e., we set the power to $\alpha = 1$ and the damping to $\gamma = 1$ as well. In this setting, the first iteration is identical to the respective Kalman



(a) Standard EP UNGM



(b) Damped EP UNGM



(c) Damped Power EP UNGM

Figure 4.4: Average performance over multiple runs of the experimental set-up in Figures 4.2 and 4.3. other linearisation methods namely, Monte Carlo transform (MCT), first-order Taylor transform (TT) are shown along the unscented transform (UT). Solid lines show average performance across 10 independent evaluations; shaded regions show standard deviation. The MCT transform is more stable and performs well for a wide range of damping factors and powers.

smoother. We see that both TT and UT based iterations, in terms of RMSE and NLL, are oscillatory, whereas, for MCT the performance improves quickly in the beginning; subsequent updates do not improve the results further. Note that the higher accuracy and stability of MCT is owing to a large number of samples (10^6) used for statistical linearisation in this one-dimensional system.

Changing the damping to $\gamma = 0.8$ as used in the running example, we see that there are significantly fewer oscillations for both UT and TT in the RMSE. Moreover, the UT log-likelihood also stabilises in about 20–25 iterations. Note that damping does not improve the RMSE significantly. Its main impact is in reducing oscillations.

Next in Figure 4.4 c), we apply power $\alpha = 0.8$ and damping $\gamma = 0.8$ as used in the example in Figure 4.3. All transforms show stable RMSE curves, and the performance, in terms of both RMSE and NLL, is significantly improved for compared to Figures 4.4 a) and b). This demonstrates that applying power EP iterations gives significant benefits.

4.5.1.2 Iterations Needed

Next, we analyse the sensitivity with respect of another hyper parameter in the algorithm namely, the number of EP iterations. Here we look at the same experiments as above at different EP iterations, i.e., in the first Figure 4.5, we plot the second iteration and sweep across different power values $\alpha \in [0.1, 1.0]$. We repeat the same procedure with different EP iterations. We see that for lower iterations the algorithm is sensitive to the choice of power α . However, for larger iterations wide range of power values give similar results. This coincides with the observations by (T. T. Minka 2005), i.e, the change in power changes how fast we reach a local minimum. It should be noted that these intuitions are reasonable in cases where the non-linearity is not severe. For highly non-linear systems the choice of linearisation transform may dominate the results.

4.5.1.3 Sweep of Parameters

Up until now we fixed the values of power or damping and studied its impact on the estimator performance. We now perform full sweep of both power and damping with fixed number of 50 iterations. We plot the mean of both RMSE and NLL of ten different seeds and all three transforms in Figure 4.6 and Figure 4.7. For RMSE in Figure 4.6b, the optimal choice of power and damping are around 0.8, whereas for MCT in Figure 4.6c, the optimal choice is around 0.5 for both damping and power. TT is unstable for low damping and with powers above 0.8. In terms of NLL for both UT and MCT powers above 0.8 are better choice as seen in Figure 4.7b and Figure 4.7c respectively.

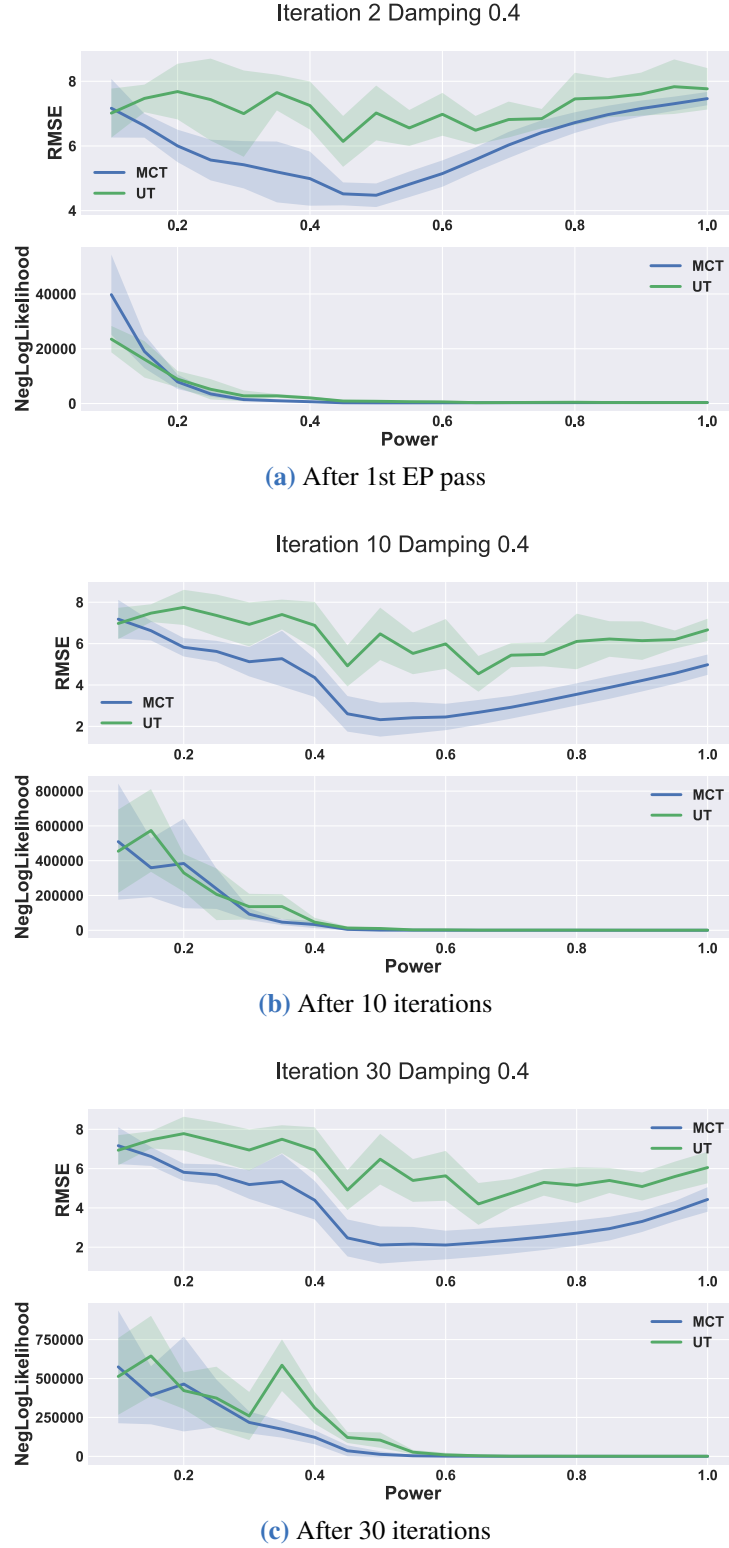


Figure 4.5: Sensitivity analysis of the proposed algorithm across different choices of power. The damping is constant for all the experiments. The power of the EP algorithm is varied, and the corresponding performance at different iterations is shown. We average across 10 different seeds with shaded areas showing the standard deviation. For the first iteration (leftmost figure), we see that the power of around 0.5 performs well. In the middle and the rightmost figure, which capture the 10th and 30th iteration respectively, we see that the impact of power is less pronounced. For a large number of iterations, we get similar RMSE and NLL values for wider range of power.

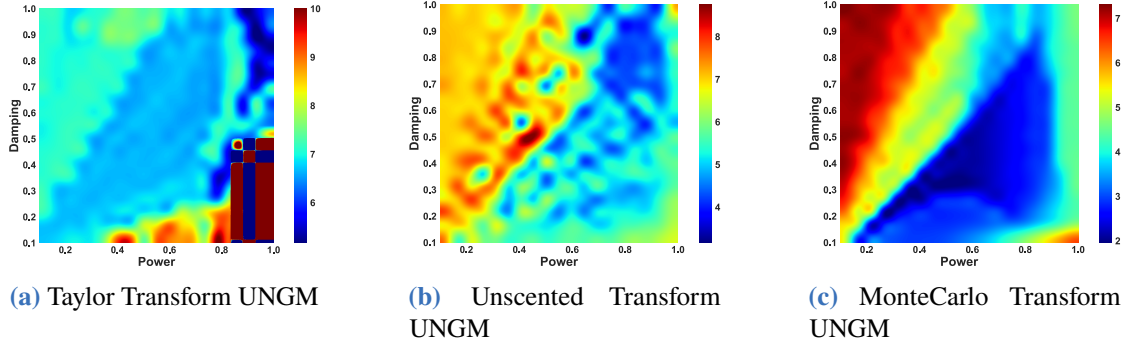


Figure 4.6: In these figures, we fixed the number of iterations to 50 and plot the average Root Mean Square Error (RMSE) over ten independent runs of the proposed method over varying the power and damping factor. We vary both power and damping between $[0.1, 1.0]$ for different transforms that approximate integral in Equation (4.40). Note that the y-axis ranges are different for each transform. For Taylor transform the estimator became unstable for certain values resulting in artifacts in the figure. For MCT we see that the filter is numerical stable over a large range of power and damping factors.

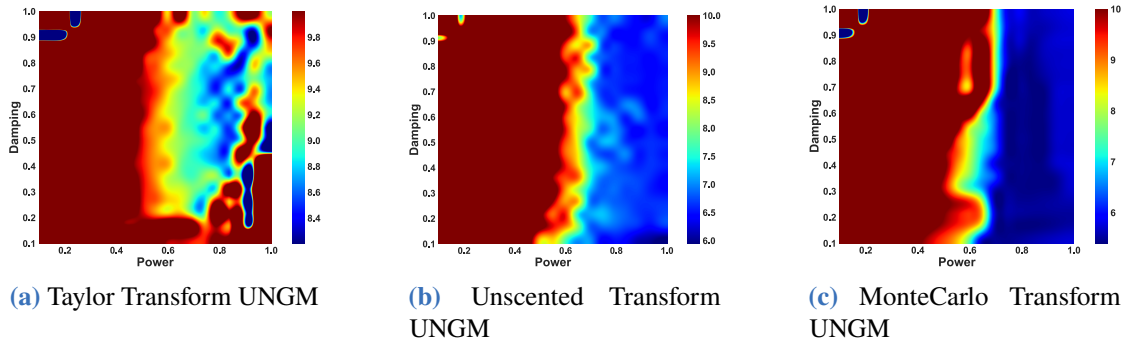


Figure 4.7: These figures plot the Negative Log-Likelihood (NLL) in log scale for similar setting as described in Figure 4.6. A consistent theme in this plots is that for NLL the power in the range above 0.7 performs well. Dark red regions in the Taylor Transform indicate that filter/smoothen losses track, whereas we can choose any damping factor for Unscented Transform and MCT as long as the power is above 0.8.

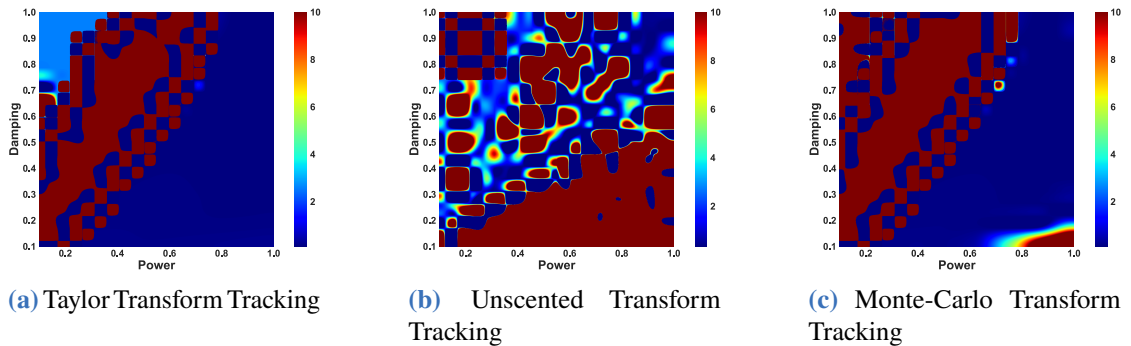


Figure 4.8: In this plot we replicate the set up used to generate Figure 4.6 for the bearings only tracking problem described in Section 4.5.2. We observe similar results as that for the UNGM, but now the Unscented Transform performs worst. Unscented transform is known to perform poorly with trigonometric non-linearities (Marc P Deisenroth and Ohlsson 2011).

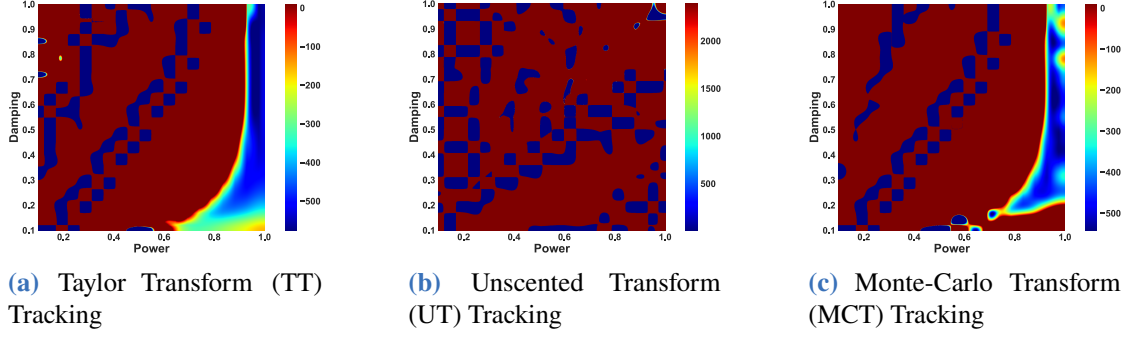


Figure 4.9: Negative log-likelihood (NLL) for a setting similar to the one described in Figure 4.6. Unlike for the UNGM, powers lower than 0.9 often lead to losing track of the target. Unlike the RMSE results in Figure 4.8, for the NLL, the unscented transform fails to estimate states irrespective of the choice of power and damping. (A different scaling of the UT may result in better performance, we do not explore it here, for details see (S. Julier 2002)).

4.5.2 Bearings Only Tracking

We test our proposed method for classic bearing only tracking problem that is commonly encountered in passive sensor environments. In this problem we assume that a single target is doing a curved motion with setup similar to car problem in (Särkkä 2013). The challenge in this system is due to highly non-linear, measurement function. The system dynamics can be approximated by the following transition equation,

$$\begin{aligned} \mathbf{x}_t &= \mathbf{A}\mathbf{x}_{t-1} + \mathbf{w}_{t-1}, \quad \mathbf{w}_{t-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}) \\ \mathbf{y}_t &= h(\mathbf{x}_t) + \mathbf{v}_t, \quad \mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}), \end{aligned} \quad (4.125)$$

where the state is four-dimensional, $\mathbf{x} = (x_1, x_2, x_3, x_4)$, such that the position of the target is (x_1, x_2) and the corresponding velocities are (x_3, x_4) . The state transition matrix is given by

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (4.126)$$

and the corresponding noise covariance matrix is given by,

$$\mathbf{Q} = \begin{pmatrix} q_1^c \Delta t^3 / 3 & 0 & q_1^c \Delta t^2 / 2 & 0 \\ 0 & q_2^c \Delta t^3 / 3 & 0 & q_2^c \Delta t^2 / 2 \\ q_1^c \Delta t^2 / 2 & 0 & q_1^c \Delta t & 0 \\ 0 & q_2^c \Delta t^2 / 2 & 0 & q_2^c \Delta t \end{pmatrix} \quad (4.127)$$

where the diffusion coefficients are set as $q_2^c = q_1^c = 0.1$

The measurement function $h(\mathbf{x}_t)$ is an inverse tangent function computed based on position parameters as and each of the sensors $s^i, i \in 1, 2, 3, 4$,

$$y_t^i = \tan^{-1} \left(\frac{x_{2t} - s_{x_2}^i}{x_{1t} - s_{x_1}^i} \right) + v_t, \quad (4.128)$$

where the four sensors are placed at four corners of square of size 1.0 centred at origin. The measurement noise v_t is isotropic Gaussian with standard deviation $\sigma = 0.05$ radians. We set the sampling frequency as $\Delta_t = 0.1$ s, and all sensors produce measurements at each time step.

Similar to the UNGM model, we perform a full sweep of the power and damping for the tracking problem. We observe that the algorithm is unstable and loses track over very low values of power as well as damping. In Figure 4.8 we plot RMSE for tracking performance. For Taylor transform and Monte-Carlo setting we see that tracking is stable over wide range of values with a damping greater than 0.2 and power greater than 0.8 performing the best. For Unscented transform however, we see that filter loses track for almost any choice of the damping and or power, this is a well known phenomenon for standard Unscented transform (Marc P Deisenroth and Ohlsson 2011). Note we can alleviate some of this effects by using scaled unscented transform and choosing appropriate scaling weights to counter non-linearities induced by trigonometric functions (S. Julier 2002).

For Negative Log-Likelihood NLL plots in the Figure 4.9 we observe similar behaviour as for the RMSE, i.e., there is a wide range of values for which both Taylor transform and the MonteCarlo transform perform well. The Unscented transform however struggles in terms of the NLL values as well with poor performance for every choice of the damping and power, scaled Unscented transform (S. Julier 2002) might be the best way to mitigate the impact of trigonometric measurement functions.

4.6 Conclusion and Discussion

We proposed an iterative state estimator for non-linear dynamical systems based on Expectation Propagation (EP). The standard derivative based EP method is numerically unstable for non-linear systems (Seeger 2005). We show that by explicitly accounting for the implicit linearisation, typically used in Kalman smoothers, we can use Kalman smoothing equations to build an iterative state estimator. The proposed method gets best of both worlds, numerical stability of Kalman based estimators and iterative improvements of the EP algorithm. We further improve the algorithm by using damping and power EP. With these extensions, any Kalman filter or smoother can be shown as special case of the proposed method.

With the set of experiments we show the efficacy of the proposed method to improve on the standard filtering and smoothing methods on two benchmark problems. The iterative method still needs the user to choose the linearisation algorithm, in the two examples we showcase that Unscented Transform struggles with trigonometric non-linearities, where as in some cases explicit linearisation by Taylor series can result in estimator losing track and diverging completely.

Overall, to apply the iterative estimator the user can begin by choosing appropriate statistical linearisation method, e.g., unscented transform. If the resulting filter is small amount of damping can applies to stabilise the oscillatory behaviour of the estimator. Then one can try to improve the results further by decreasing power of the EP approximation by choosing $\alpha \approx 0.9$ which tends to work for broad range of algorithm hyper-parameters like, linearisation, damping value and number of iterations etc. The alpha divergence of $\alpha = 0.5$ works well if the estimator is not very sensitive to other choices in the algorithm. The $\alpha = 0.5$ corresponds to the Hellinger distance, which is the only α -divergence that obeys triangle inequality.

Part III

Copula Based Probabilistic Models

Copula Flows for Synthetic Data

5.1 Introduction

Recent advances in hardware capabilities and ubiquity of data has led to the explosion of machine learning applications in various domains. Machine learning is now an integral part of the daily decision making process, from personalised medicine to credit lending in banks and financial institutions. The downside of such pervasive use of machine learning in daily life is the privacy concern associated with personal data used by such models (Elliot and Domingo-Ferrer 2018). In some applications such as medicine, sharing data with privacy in mind is fundamental for advancing the science (Beaulieu-Jones et al. 2019). However, the available data may not be sufficient to realistically train state of the art machine learning algorithms. Synthetic data is one way of mitigating this challenge, see Figure 5.1. In this paper we propose a probabilistic synthetic data generator that is interpretable and can model arbitrarily complex datasets. We can use the learned probabilistic model not only for synthetic data generation but also for other tasks such as inference.

Current state-of-the-art methods for synthetic data generation, such as Generative Adversarial Networks (GANs) (Goodfellow et al. 2014), use complex deep generative networks to produce high-quality synthetic data for a large variety of problems (Choi et al. 2017; Xu et al. 2019). However, GANs are known to be unstable and delicate to train (Arjovsky and Bottou 2017). Moreover, GANs are also notoriously difficult to interpret, and finding an appropriate loss function is an active area of research (Wang, She, and Ward 2019). Another class of models, variational auto encoders (VAEs) (Diederik P Kingma and Welling 2013; Rezende and Mohamed 2015) have also been used to generate

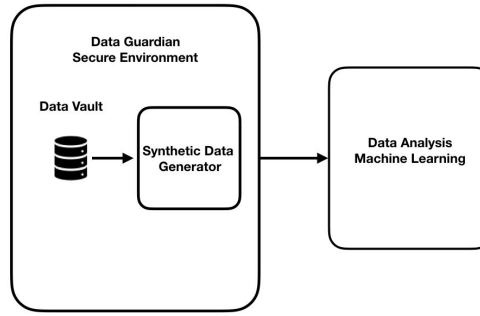


Figure 5.1: The figure shows typical application of the proposed synthetic data generator. The data vault is protected by a regulator or the access is restricted for privacy protection. In order to do machine learning and analysis on the data the synthetic data generator needs to produce results with statistical integrity. We show with the experiments that the copula flow method proposed in this paper can act as a proxy for private data vault for machine learning tasks.

data, although their primary focus is finding a lower-dimensional data representation. While deep generative models, such as VAEs and GANs, can generate high-fidelity synthetic data, they are both difficult to interpret. Due to the complex composition of deep networks, it is nearly impossible to characterise the impact of varying weights on the density of the generated data. For VAEs, a latent space may be used to interpret different aspects of the data.

In this work, we focus on probabilistic models based on copulas. Sklar showed that any multivariate distribution function can be expressed as a function of its univariate marginals and a copula (Sklar 1959). However, estimating copulas for multivariate mixed data types is typically hard, owing to 1) very few parametric copulas have multivariate formulations and 2) copulas are not unique for discrete or count data (Nikolouloupoulos and Karlis 2009). To address the first challenge, the current standard is to use a pairwise copula construction (Aas et al. 2009), which involves two steps, first identifying the pair copula and second finding a tree structure defining pairwise relationship (Elidan 2010; Lopez-paz, Hernández-lobato, and Ghahramani 2013; Czado 2010; Chang, S. Pan, and Joe 2019).

In the following, we propose a probabilistic synthetic data generator that is interpretable and can model arbitrarily complex data-sets. Our approach is based on normalising flows to estimate copulas. We show that our model is flexible enough to learn complex relationships and does not rely on explicit learning of the graph or parametric bivariate copulas. Figure 5.2 illustrates the flexibility of our proposed method by modelling copula with a complex structure. We are not aware of any copula based method able to learn the copula shown in Figure 5.2 a).

To deal with count data, we propose a modified spline flow (Durkan et al. 2019) as a continuous relaxation of discrete probability mass functions. This formulation allows us

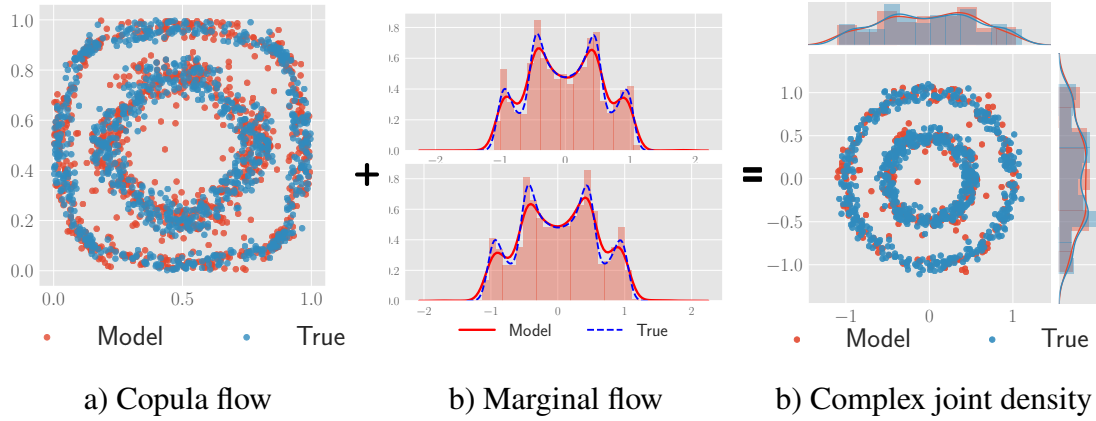


Figure 5.2: Learning the joint and marginal distributions for ‘2 rings’ data. This figure demonstrates how copulas and marginals can be combined to obtain complex joint distributions. The copula associated with the ‘2 rings’ distribution cannot be modelled by any standard bi-variate copula, whereas our proposed copula flow method can learn this structure.

to build complex multivariate models with mixed data types that can learn copula-based flow models, which can be used for the following tasks:

1. *Synthetic data generation:* Use the estimated model to generate new data-sets that have a distribution similar to the training set.
2. *Inferential statistics:* When the copula has learned the relationship between the variables correctly, one can change the marginals to study the effects of change. For example, if we estimated a copula flow based on a UK dataset with age as one of the variables, we can modify the marginal of the age distribution to generate synthetic data for a different country. To generate data for Germany, we would use Germany’s age distribution as a marginal for the data-generating process.
3. *Privacy preservation:* The data generated from the copula flow model is fully synthetic, i.e., the generator does not rely on actual observations for generating new data. One can perturb the generated data based on differential privacy mechanisms to prevent privacy attacks based on a large number of identical queries.

5.1.1 Related Literature

The Synthetic Data Vault (SDV) project ([Patki, Wedge, and Veeramachaneni 2016](#)) is very close in spirit of this paper. In the SDV, the univariate marginals are learned by using a Gaussian mixture model, and the multivariate copula is learned as a Gaussian copula.

5.1.1.1 Copula-based Modelling

Unlike flow-based estimation, proposed in this work, copula-based models typically follow a two-step procedure, the pair-copula construction or *vine copulas*, by first constructing a pairwise dependence structure and then learning pair-copulas for it ([Aas](#)

et al. 2009). Copula Bayesian networks (Elidan 2010) use conditional copulas and learn a Bayesian network over the variables. In (Chang, S. Pan, and Joe 2019), the authors propose to use Monte-Carlo tree search for finding the network structure. Such models have limited flexibility, e.g., no standard copula model can learn the structure in figure 5.2, which our model (copula flow) can learn. Compared to auto-encoder based model proposed in (Tagasovska, Ackerer, and Vatter 2019) our approach can model both continuous and discrete variables. Also, an auto-encoder might make it challenging to interpret relationships between variables. If we were to apply our model to synthetic image generation then (Tagasovska, Ackerer, and Vatter 2019) would be a special version of our method. Copula based density modelling in (Wiese, Knobloch, and Korn 2019) only speaks about bi-variate copulas, and doesn't address the challenge of discrete variables.

5.1.1.2 GAN-based Models

GAN-based methods are the workhorse of the majority of recent advances in the synthetic data generation, e.g., the conditional GAN (Xu et al. 2019), builds a conditional generator to tackle mixed-variable types, whereas the tableGAN (Park et al. 2018) builds GANs for privacy preservation. Unlike the tableGAN, the PATE-GAN (Jordon, Yoon, and Van Der Schaar 2019) produces synthetic data with differential privacy guarantees. Data generators for medical records (Choi et al. 2017; Che et al. 2017) are also based on GANs.

5.1.1.3 Normalising Flow based Models

Normalising flow methods have had very rapid growth in terms of the continuous improvements on benchmarks introduced by (Papamakarios, Pavlakou, and Murray 2017). Our work relies heavily on the NSF (Durkan et al. 2019; Müller et al. 2019). Instead of training models with a likelihood loss, in (Nash and Durkan 2019), the authors use auto-regressive energy models for density estimation. While most methods either rely on coupling (Dinh, Krueger, and Y. Bengio 2015; Dinh, Sohl-Dickstein, and S. Bengio 2019) or auto-regressive (Papamakarios, Pavlakou, and Murray 2017; Diederik P. Kingma et al. 2016) models, transformation networks (Oliva et al. 2018) use a combination of both. The neural auto-regressive flows (Huang et al. 2018), which have monotonic function guarantees, can be used as a drop-in replacement for the spline-based copula network used in this paper.

5.2 Background: Copula Theory and Normalising Flow

5.2.1 Notation

We denote the domain of a function F by $\mathbf{dom}(F)$ and range by $\mathbf{ran}(F)$. Here, the range $\mathbf{ran}(F)$, implies *image* of the function F and not its co-domain. We use capital letters X, Y to denote random variables and lower case x, y to represent their realisations. Bold symbols $\mathbf{X} = [X_1, \dots, X_d]$ and corresponding $\mathbf{x} = [x_1, \dots, x_d]$ represent vectors of random variables and their realisations, respectively. We denote the distribution function (also known as the Cumulative Distribution Function (CDF)) of a random variable X by F_X , and the corresponding Probability Density Function (PDF) by f_X . Here, the uniform marginals are obtained by the probability integral transform, i.e., $U_X = F_X(X)$ and $U_Y = F_Y(Y)$.

5.2.2 Copula Models

For a pair of random variables, the marginal CDFs F_X, F_Y describe how each variable is individually distributed. The joint CDF $F_{X,Y}$ tells us how they are jointly distributed. Sklar's seminal result (Sklar 1959) allows us to write the joint CDF as a function of the univariate marginals F_X and F_Y , i.e.,

Theorem 5.1. Sklar's theorem

The joint distribution function $F_{X,Y}$ with marginals F_X and F_Y can be written as

$$F_{X,Y}(X, Y) = C(U_X, U_Y) = C(F_X(X), F_Y(Y)), \quad (5.1)$$

where C is a copula.



Here, the uniform marginals are obtained by the probability integral transform, i.e., $U_X = F_X(X)$ and $U_Y = F_Y(Y)$.

For continuous marginals F_X and F_Y , the copula C is unique, whereas for discrete marginals it is unique only on the $\mathbf{ran}(F_X) \times \mathbf{ran}(F_Y)$. A corollary of Sklar's theorem is that X and Y are independent if and only if their copula is an independent copula, i.e., $C(F_X(X), F_Y(Y)) = F_X(X) F_Y(Y)$.

5.2.3 Generative Sampling

We can use the constructive inverse of Sklar's theorem for building a generative model. The key concept is that the inverse of the probability integral transform, called *inverse transform sampling*, allows us to generate random samples starting from a uniform distribution. The procedure to generate a random sample x distributed as F_X is to first sample a random variable $u \sim \text{unif}[0, 1]$ and second to set $x := F_X^{(-1)}(u)$. Here, the the

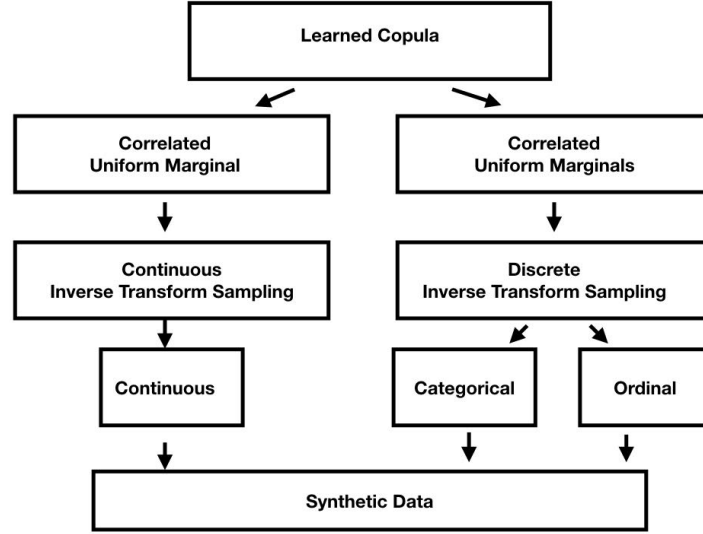


Figure 5.3: The figure demonstrates the synthetic data generation starting from copula model. We start from learned Copula model, see Figure 5.8, to generate synthetic data. The Copula generates correlated marginals. Then we can generate data from these marginals using inverse transform sampling. For discrete variables there is an additional step depending on the nature of the discrete variable.

function $F^{(-1)}$ is a quasi-inverse function.

Definition 5.1. Quasi-inverse function

Let F be a CDF. Then the function $F^{(-1)}$ is a quasi-inverse function of F with domain $[0, 1]$ such that $F(F^{(-1)}(y)) = y \quad \forall y \in \mathbf{ran}(F)$ and $F^{(-1)}(y) = \inf \{x | F(x) \geq y\} \quad \forall x \in \mathbf{dom}(F)$, and $y \notin \mathbf{ran}(F)$. For strictly monotonically increasing F , the quasi-inverse becomes the regular inverse CDF F^{-1} (Nelsen 2006). 

The inverse function, also called as the *quantile function*, maps a random variable from a uniform distribution to F -distributed random variables, i.e., $X = F^{-1}(U)$ or $F^{-1} : \text{unif}[0, 1] \rightarrow \mathbf{ran}(F)$. Hence, we can present the problem of synthetic data generation as the problem of estimating the (quasi)-inverse function for the given data.

To generate samples from F_X using a copula-based model, we start with a pair of uniformly distributed random variables, then pass them through the inverse of the copula CDF to obtain correlated uniform marginals; see Figure 5.4 b). Note that the copulas are defined over uniform marginals, i.e., the random variables are uniformly distributed. We use the correlated univariate marginals to subsequently generate F -distributed random variables via inverse transform sampling; see Figure 5.4 c). Hence, if we can learn the CDFs for the copula as well as the marginals, we can then combine these two models to generate correlated random samples that are distributed similar to the training data. The procedure is illustrated in Figure 5.4 and for mixed variables in Figure 5.3.

The inverse function used to generate synthetic data can be described as a flow

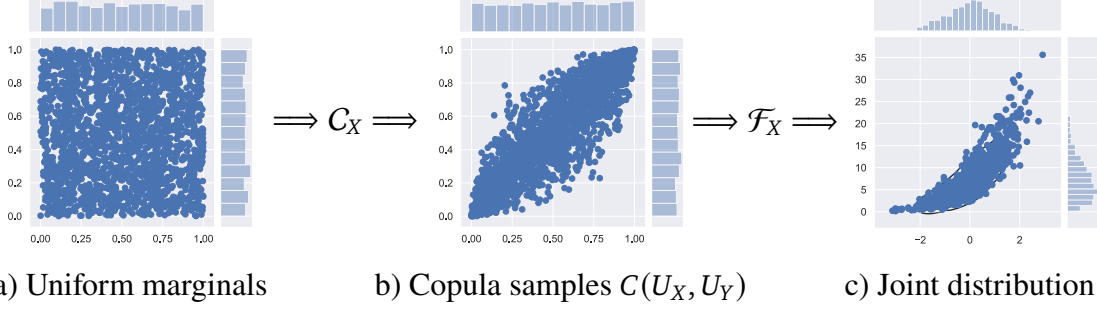


Figure 5.4: Copula flow generative model. Left: We start with uniformly distributed independent variable samples. These samples are passed through the copula flow network to generate correlated uniformly distributed random variables (middle). The correlated variables are transformed to the desired distribution by using univariate marginal flows $\mathcal{F}_X$ (right).

function $\mathcal{F}_X \approx F_X^{(-1)}$ that transforms a uniform random variable U into $X \sim F_X$, see Figure 5.4. We can interpret the (quasi)-inverse function as a normalising flow (Rezende and Mohamed 2015; Tabak and Vanden-Eijnden 2010; Papamakarios, Nalisnick, et al. 2019) that transforms a uniform density into the one described by the copula distribution function C . We refer it as *copula flow*.

5.2.4 Normalising Flows

Normalising flows are compositions of smooth, invertible mappings that transform one density into another (Rezende and Mohamed 2015; Papamakarios, Nalisnick, et al. 2019). The approximation in $\mathcal{F}_X \approx F_X^{(-1)}$ indicates that we are estimating a (quasi) inverse distribution function by using a flow function $\mathcal{F}_X$. If we learn the true CDF F_X , we have

$$F_X(x) = \mathcal{F}_X^{-1}(x) = u, \quad u \sim \text{unif}[0, 1], \quad \forall x \in \text{dom}(F_X), \quad (5.2)$$

due to uniqueness of the CDF. An advantage of the normalising flow formulation is that we can construct arbitrarily complex maps as long as they are invertible and differentiable.

Consider an invertible flow function $\mathcal{F} : \mathbb{R}^D \rightarrow \mathbb{R}^D$, which transforms a random variable as $\mathbf{X} = \mathcal{F}(\mathbf{U})$. By using the change of variables formula, the density f_X of variable $\mathbf{X}$ is obtained as

$$f_X(\mathbf{X}) = f_U(\mathcal{F}^{-1}(\mathbf{X})) \left| \det \left(\frac{\partial \mathcal{F}^{-1}(\mathbf{X})}{\partial \mathbf{X}} \right) \right| = \left| \det \left(\frac{\partial \mathcal{F}^{-1}(\mathbf{X})}{\partial \mathbf{X}} \right) \right|, \quad (5.3)$$

where $f_U(\mathcal{F}^{-1}(\mathbf{X})) = f_U(\mathbf{U}) = 1.0$, since $f_U = \text{unif}[0, 1]$.

5.2.5 Flow Likelihood

We can derive the flow-based likelihood by starting from the generative procedure for the samples. We start with a d -dimensional independently distributed random vector $\mathbf{U} \sim \text{unif}[0, 1]^d$ and map it through the copula flow C_X to obtain the random vector $\mathbf{U}_X = C_X(\mathbf{U})$. The joint vector is then mapped through the marginal flows $\mathcal{F}_X$ to obtain a random vector $\mathbf{X} = \mathcal{F}_X(\mathbf{U}_X)$. The combined formulation can be written as a composition

of two flows to obtain $\mathbf{X} = \mathcal{F}_X(C_X(\mathbf{U}))$, which is also a valid flow function (Rezende and Mohamed 2015). The likelihood for this flow function can be written as

$$f_X(\mathbf{X}) = f_{U_X}(\mathcal{F}_X^{-1}(\mathbf{X})) \left| \det \left(\frac{\partial \mathcal{F}_X^{-1}(\mathbf{X})}{\partial \mathbf{X}} \right) \right|. \quad (5.4)$$

The marginal flows are independent for each dimension of the random vector $\mathbf{X}$. Hence, the determinant can be expressed as a product of the diagonal terms to obtain the likelihood

$$f_X(\mathbf{X}) = f_{U_X}(\mathcal{F}_X^{-1}(\mathbf{X})) \prod_{i=1}^d \left| \frac{\partial \mathcal{F}_{X_i}^{-1}(x_i)}{\partial x_i} \right|. \quad (5.5)$$

The quantity $f_{U_X}(\mathcal{F}_X^{-1}(\mathbf{X}))$ is essentially the flow likelihood for the copula density, which we can write as

$$f_{U_X}(\mathcal{F}_X^{-1}(\mathbf{X})) = f_{U_X}(C_X(\mathbf{U})) \quad (5.6)$$

$$= f_U(C_X^{-1}(\mathbf{U}_X)) \left| \det \left(\frac{\partial C_X^{-1}(\mathbf{U}_X)}{\partial \mathbf{U}_X} \right) \right| \quad (5.7)$$

$$= \left| \det \left(\frac{\partial C_X^{-1}(\mathbf{U}_X)}{\partial \mathbf{U}_X} \right) \right|, \quad (5.8)$$

where $f_U(C_X^{-1}(\mathbf{U}_X)) = f_U(\mathbf{U}) = 1.0$, since $f_U = \text{unif}[0, 1]$. Using the the copula likelihood in (5.4) in the total likelihood (5.8) yields

$$f_X(\mathbf{X}) = \left| \det \left(\frac{\partial C_X^{-1}(\mathcal{F}_X^{-1}(\mathbf{X}))}{\partial \mathcal{F}_X^{-1}(\mathbf{X})} \right) \right| \prod_{i=1}^d \left| \frac{\partial \mathcal{F}_{X_i}^{-1}(x_i)}{\partial x_i} \right|. \quad (5.9)$$

In normalising flows computing determinant of the Jacobian, $\frac{\partial C_X^{-1}}{\partial \mathbf{U}_X}$, is the most expensive operation. For the autoregressive formulation this Jacobian is lower triangular, hence, its determinant can be easily computed as trace of the matrix.

5.2.5.1 Sklar's Theorem

As discussed in section 5.2.3, we can interpret the inverse of the flow function as a CDF:

$$\mathbf{X} = \mathcal{F}_X(C_X(\mathbf{U})), \quad (5.10)$$

$$C_X^{-1}(\mathcal{F}_X^{-1}(\mathbf{X})) = \mathbf{U}. \quad (5.11)$$

If we replace the flow function with the true marginal CDF F_X and the true copula CDF C_X , we can rewrite it as

$$C_X(F_X(\mathbf{X})) = \mathbf{U}. \quad (5.12)$$

By the probability integral transform we know that if H_X is the CDF of $\mathbf{X}$ then $H_X(\mathbf{X}) = \mathbf{U}$. Hence, we can write the joint CDF in terms of marginal CDFs as

$$H_X(\mathbf{X}) = C_X(F_X(\mathbf{X})). \quad (5.13)$$

This is essentially Sklar’s theorem (Sklar 1959). Note we reached this result by starting from the normalising flow formulation, whereas in section 5.2.3 we used Sklar’s theorem for the generative model.

5.3 Marginal Flows

To estimate univariate marginals, we can use both parametric and non-parametric density estimation methods as the dimensionality is not a challenge. However, for training the copula, we require models that can be inverted easily to obtain uniform marginals, i.e., we need a well-defined CDF for the methods we employ for the density estimation. Simultaneously, we need a model that can be used for generating data via inverse transform sampling.

5.3.1 Marginal Splines

Both objectives can be achieved by using the normalising flow based methods as we learn a bijective function. In this paper we use neural splines (Durkan et al. 2019) to build flow bijective function. In particular we use a rational-quadratic function which takes the form of a quotient of two quadratic polynomials. Rational quadratic functions are easily differentiable, and since we consider only monotonic segments of these functions, they are also analytically invertible. Details of the spline originally proposed by (Gregory and Delbourgo 1982) and adapted for neural density estimation by (Durkan et al. 2019) are included in the Appendix A

With the a sufficiently dense spline, i.e., a large number of knot positions, see Appendix A for details, we can learn arbitrarily complex marginal distributions. As we directly attempt to learn the quasi-inverse function $\mathcal{F} : \text{unif}[0, 1] \rightarrow \mathbf{ran}(F)$, where F is the *true* CDF, a single parameter vector $\theta_d = [\theta_d^w, \theta_d^h, \theta_d^s]$ describing the width, height and slope parameters, respectively, is sufficient.

5.3.2 Marginal Fitting: Discrete Data

Modelling mixed variables via normalising flows is challenging (Onken and Panzeri 2016; Ziegler and Rush 2019; Tran et al. 2019). Discrete data poses challenges for both copula learning as well as learning flow functions. For marginals, i.e., univariate flow functions, the input is a uniform distribution continuous in $[0, 1]$, whereas the output is discrete and hence discontinuous. For copula learning, we need uniform marginals for the given training data. With discrete inputs to the inverse flow function (CDF), the output is discontinuous in $[0, 1]$.

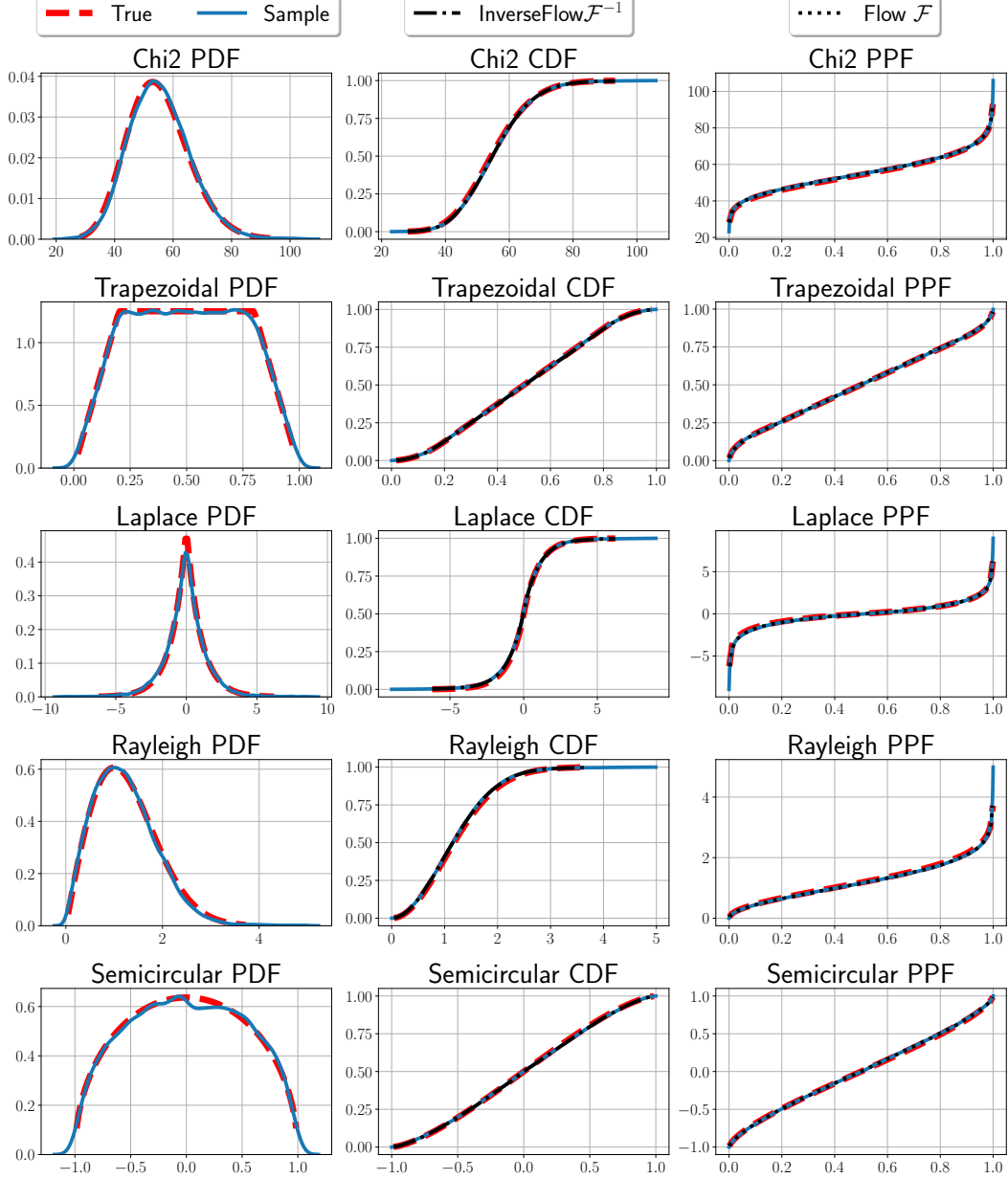


Figure 5.5: Illustration of continuous Marginal learning with the spline flow network. We demonstrate the ability of the proposed spline network to learn correct distributions. The illustrations are, left column: Probability Distribution Function PDF, centre column: Cumulative Distribution Function (CDF), right column: Percent Point Function (PPF) for various random variables. The network is able to learn sharp transitions in the Laplace density, third row, as well as smooth changes in the semicircular PDF in the last row. Furthermore, we can see that the inverse flow function and the true CDF in the middle column coincide over the whole $\text{dom}(\cdot)$ of the random variable. Similarly the flow function $\mathcal{F}$ matches the true the quantile function or PPF

We first focus on learning the univariate flow maps, i.e., marginal learning. Ordinal variables have a natural order, which we can use directly. For categorical data, we propose to assign each class a unique integer in $\{0, \dots, n-1\}$, where n is the number of classes. We define a CDF over these integer values. As this assignment is not unique, the same category assignment needs to be maintained for training and data generation.

For discrete data generation, we round the data output of the flow function to the

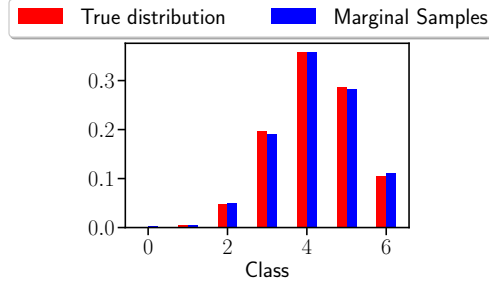


Figure 5.6: Discrete samples generated from the proposed marginal. The histograms are almost identical for the True distribution and the data sampled from the proposed discrete marginal flow.

next higher integer, i.e., we can consider random variable $Y = \text{ceil}(X)$. However, this procedure does not yield a valid density. To ensure that the samples are properly discretised, we use a quantised distribution (Salimans et al. 2019; Dillon et al. 2017). With quantised distribution the density learning is formulated as a quantile learning procedure.

We assign a quantile range for a given class or ordinal integer. We use the same spline-based flow functions as the one used for continuous marginals, but with quantisation as the last step. The advantage of this discrete flow is that our quasi-inverse, i.e., the flow function is a continuous and monotonic function, which can be trained by maximising the likelihood. Figure 5.7 shows the spline-based flow function learned for a hypergeometric distribution. The continuous flow function (blue curve in Figure 5.7) learned for the discrete marginal, light orange step function in Figure 5.7, allows us to generate discrete data via inverse transform sampling.

5.4 Copula Flow

Copula is a CDF defined over uniform marginals, i.e., $\text{unif}[0, 1]^d \rightarrow \text{unif}[0, 1]$. We can use the inverse of this CDF to generate samples via inverse transform sampling. For the multivariate case, we can use the conditional generation procedure (Hansen 1994). Let $\mathbf{U}_X = [U_{X_1}, \dots, U_{X_d}]$ be a random vector with copula C_X , and $\mathbf{U} = U_1, \dots, U_d$ be i.i.d. $\text{unif}[0, 1]$ random variables. Then the multivariate flow transform $\mathbf{U}_X := C_X(\mathbf{U})$ can be defined recursively as

$$\begin{aligned} U_{X_1} &:= C_{X_1}(U_1), \\ U_{X_k} &:= C_{X_k|1,\dots,k-1}(U_k|U_{X_1}, \dots, U_{X_{k-1}}), \quad 2 \leq k \leq d, \end{aligned} \quad (5.14)$$

where $C_{X_k|1,\dots,k-1} = C_{X_k|1,\dots,k-1}^{-1}$ is the flow function conditioned on all the variables up to the index k , i.e., $U_{X_1}, \dots, U_{X_{k-1}}$. The key concept is the interpretation of the inverse of the (normalising) copula flow C_X^{-1} as a conditional CDF. We can learn the conditional flow function $C_{X_k|1,\dots,k-1}$ as,

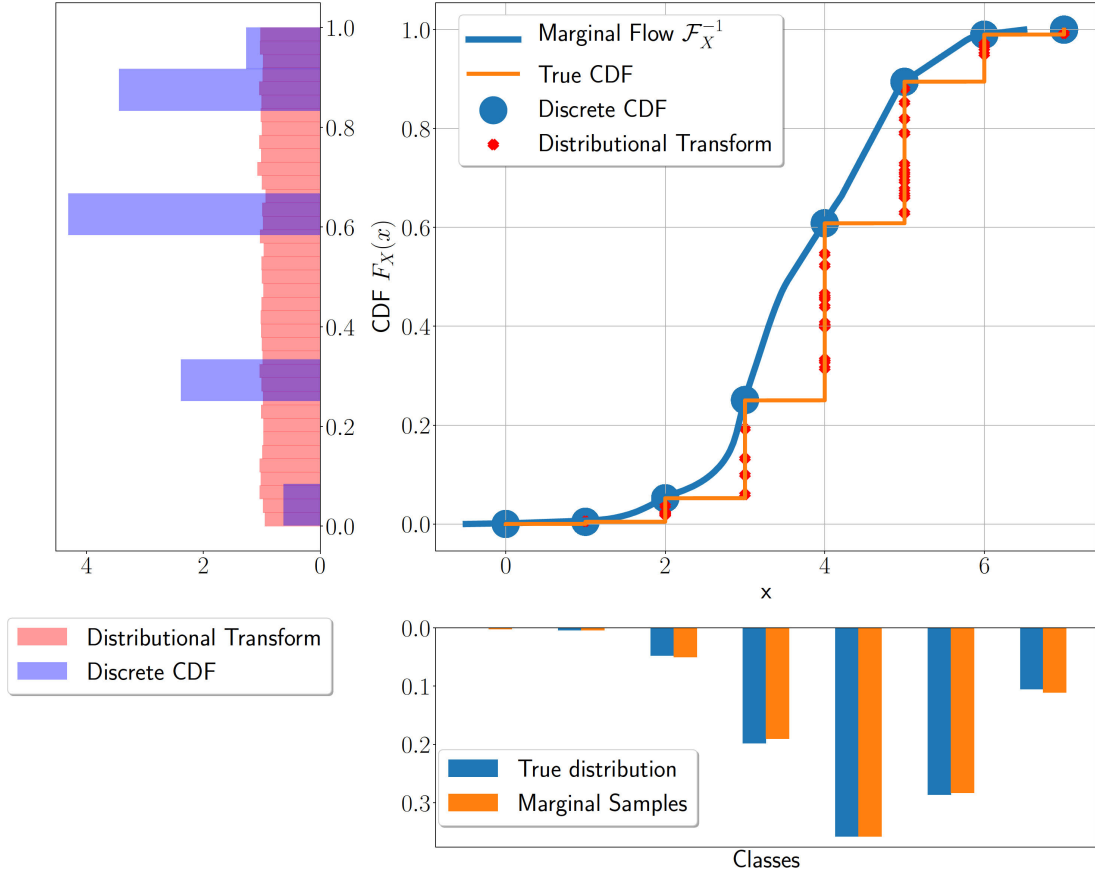


Figure 5.7: Illustration of discrete marginal flow for hypergeometric distribution used in the experiments 5.6. The marginal flow (blue) gives discrete samples when rounded up. The distributional transform is uniform along the vertical line of the true CDF (orange). The true and generated marginals have almost identical data distribution

$$Y_1 := g_{\theta_1}(U_1), \quad (5.15)$$

where, g is a flow function built using neural network with parameters θ_1 . For the first dimension, i.e. , for θ_1 the function g is essentially same as univariate marginal. Moreover, as the copula is defined over uniform marginals the g for the first dimension is identity. For conditional generation the parameters for the next dimension are computed using a neural network $\theta_2 = NN(y_1)$. The flow function then gives us $Y_2 := g_{\theta_2}(U_2)$. We can expand this autoregressive network as,

$$\begin{aligned} \theta_k &= NN(Y_1, \dots, Y_{k-1}), \\ Y_k &:= g_{\theta_k}(U_k), \quad 2 \leq k \leq d, \end{aligned} \quad (5.16)$$

Learning d neural networks for each dimension would be highly inefficient. Ideally, we should share the network weights that extract features from specific dimension. [Germain et al. \(2015\)](#) proposed a neural network architecture that outputs all the flow parameters $\theta_1, \dots, \theta_d$ in one pass. The networks weights are masked in such a way that

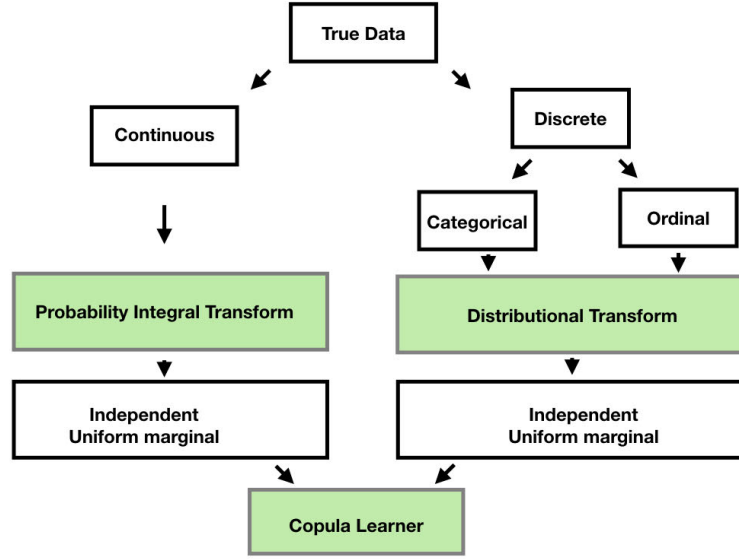


Figure 5.8: The figure shows how the data flows in the model for learning the copula. The green blocks represent the part of the process that are novel and proposed in this work.

the autoregressive property is preserved for the network, i.e., k^{th} output only depends on $1, \dots, k - 1$ dimensions.

Advantage of using this architecture is that we can train the network with only one pass over the autoregressive network, speeding up the training by d times. Note that this only saves training time, sampling is still autoregressive process and hence, d times slower (Germain et al. 2015). For faster sampling one can use the inverse model such as used in Inverse Autoregressive Flow (IAF) (Diederik P. Kingma et al. 2016). For IAF sampling is d times faster which allows it to scale variational inference to large dimensions.

5.4.1 Splines for Copula

While many different methods can be used to create g see (Papamakarios, Pavlakou, and Murray 2017; Papamakarios, Nalisnick, et al. 2019) and references therein for details. We propose to use neural spline network (Durkan et al. 2019), where function g is a rational quadratic spline and the NN encodes the spline variables. The splines are modelled as per original work of (Gregory and Delbourgo 1982). The splines are monotonic by construction, we can construct complex structures by stacking many of them together whilst still preserving the monotonicity. The procedure above is similar to autoregressive density estimators (Papamakarios, Pavlakou, and Murray 2017; Papamakarios, Nalisnick, et al. 2019) with Masked Autoregressive Flow (MAF) structure (Papamakarios, Pavlakou, and Murray 2017; Germain et al. 2015). Similar to MAF we can stack such multiple flows to create flexible multivariate copula flow C_X . In the copula literature this multivariate extension is convenient for Archimedean copulas (Nelsen 2006), where the

generators for such copulas can be interpreted as conditional flow functions.

5.4.2 Copula for Discrete Marginals

Learning copulas for discrete variables is a challenging task (Nelsen 2006). The section 5.3.2 described how to use flow function to learn the discrete random variable marginal. However, the inverse of this flow function, i.e., the CDF of the marginal, results in discontinuous values at the locations of the training inputs (blue circles in Figure 5.7). Copulas are unique and well defined only for continuous univariate marginals. An elegant way to find continuous univariate marginals for the discrete variable is via the distributional transform (Rüschendorf 2009).

Definition 5.2. Distributional Transform

Let $X \sim F_X$. The modified CDF is defined as

$$F_X(x, V) := p(X < x) + Vp(X = x). \quad (5.17)$$

where V is uniformly distributed independent of X . Then the distributional transform (i.e., the modified CDF) $X \rightarrow U$ of X is

$$u := \mathcal{F}_X^{-1}(x) = F_X(x, V) \quad (5.18)$$

$$= F_{X-}(x) + V(F_X(x) - F_{X-}(x)), \quad (5.19)$$

where $F_{X-}(x) = p(X < x)$, $F_X(x) = p(X \leq x)$ and $p(X = x) = F_X(x) - F_{X-}(x)$.

With this, we have $U \sim \text{unif}[0, 1]$ and $X = \mathcal{F}_X(U)$ a.s. (Rüschendorf 2009). 

This distributional transform behaves similar to the probability integral transform for continuous distributions, i.e., a discrete random variable X mapped through the distributional transform gives an uniform marginal. However, unlike for continuous variables, the distributional map is stochastic and hence not unique. With the distributional transform, the copula model does not need a special treatment for discrete variables; see section 2 in (Rüschendorf 2009) for details. In Figure 5.9 the blue curve is obtained by forward transform followed by distributional transform. We can see that we recover original uniform distribution. note that the inverse flow function $\mathcal{F}_X^{-1}$, i.e., is not same as the analytic inverse of the forward flow function $\mathcal{F}_X$.

In Figure 5.7, the red cross marks show the values from distributional transform. All samples along the y-axis that share same x-value.

5.5 Copula Flow: An Universal Density Approximator

The copula flow model can be used to represent any distribution function, i.e., the model is a universal density approximator. The inverse of the flow model can be interpreted as a CDF. By the probability integral transform and and universality definition

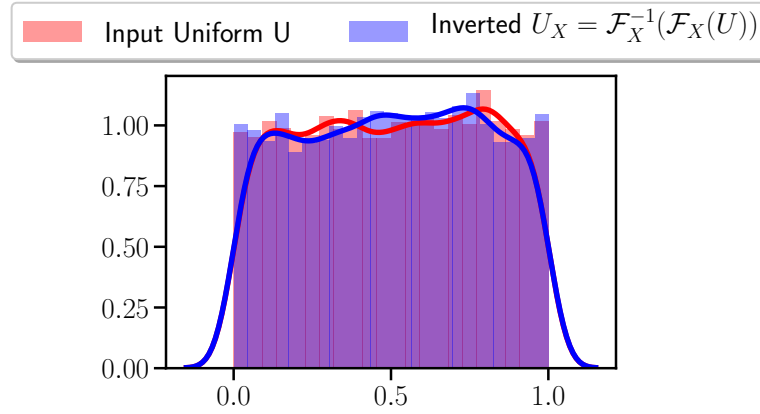


Figure 5.9: This plots demonstrates that the distributional transform indeed recovers uniform distribution for learned discrete distribution. This plot is generated by starting with uniform distribution to obtain discrete samples plotted in Figure 5.6, these samples are then again passed through the proposed distributional transform, see text for details, to generate uniform distribution. That is with distributional transform we have *probability integral transform* allowing us to apply Sklar’s theorem 5.2.2 to discrete variables without any modifications.

in Theorem 1 in (Hyvärinen and Pajunen 1999), we know that if H_X is the CDF of the random variable then $H_X(X)$ is uniformly distributed in the hyper-cube $[0, 1]^d$, where d is the dimension of the vector.

Theorem 5.2. Universality

The proposed Copula flow model is an universal density estimator



Proof

Any invertible non-linear function that can take i.i.d. unif $[0, 1]$ data and map it onto a density via monotone functions is a universal approximator (Hyvärinen and Pajunen 1999).

In the following, we show that with the combination of marginal and copula flows we can learn any CDF with desired accuracy. Then the inverse of this CDF, i.e., the quantile function, can be used to generate random variables distributed according to X .

5.5.1 Convergence of the Marginal Flow

Proposition 5.1. Universality of Splines

Monotonic rational quadratic splines can universally approximate any monotonic function with appropriate derivative approximations, see Theorem 3.1 in (Gregory and Delbourgo 1982)



As discussed in section 5.2.3, a (quasi) inverse function F^{-1} can be used to transform random variable $U \sim \text{unif}[0, 1]$ to the random variable $X \sim F$. The inverse function is a monotonic function, and from Proposition 5.5.1 we can say that

Lemma 5.1. Splines are universal approximators

Given a flow map $\mathcal{F}_X : U \rightarrow \tilde{X}$, a sequence of normalising flow mappings, such that $\mathcal{F}_X$ converges point-wise to the true inverse function F_X^{-1} , the transformed random variable $\tilde{X} = \mathcal{F}_X(U)$ converges in distribution to the distribution $X = F_X^{-1}(U)$. 

Proof: This is Portmanteau's Theorem applied to flow functions based on Proposition 5.5.1

5.5.2 Convergence of the Copula Flow

For any arbitrary ordering of the variables we have $U = C_X(U_X)$. We can write

$$U_1 \doteq C_{X_1}(U_{X_1}, \emptyset) = p(U_{X_1} \leq u_{X_1} | \emptyset) \approx C_{X_1}^{-1}(U_{X_1}, \emptyset) \quad (5.20)$$

$$U_k \doteq C_{X_k}(U_{X_k}, U_{X_{1:k-1}}) \quad (5.21)$$

$$= p(U_{X_k} \leq u_{X_k} | U_{X_{1:k-1}}) \quad (5.22)$$

$$\approx C_{X_k}^{-1}(U_{X_k}, U_{X_{1:k-1}}), \quad 2 < k < d, \quad (5.23)$$

where $\emptyset$ indicates null set. According to (Hyvärinen and Pajunen 1999), the variables $U_1, \dots, U_d$ are independently and uniformly distributed. The conditional copula flow is also modelled as a rational quadratic spline. From Lemma 5.1 the $C_X^{-1}(U_X)$ converges to the true copula CDF C_X . Therefore, $C_X(U)$ converges in distribution to U_X .

We can extend the conditional copula flow by adding the marginal flow $U_{X_k} = F_{X_k}(X_k)$ to obtain

$$U_k \doteq C_{X_k}(F_{X_k}(X_k), F_{X_{1:k-1}}(X_{1:k-1})) = H_{X_k}(X_k, X_{1:k-1}) \quad (5.24)$$

$$= p(X_k \leq x_k | X_{1:k-1}) \quad (5.25)$$

$$\approx C_{X_k}^{-1}(\mathcal{F}_{X_k}^{-1}(X_k), \mathcal{F}_{X_{1:k-1}}^{-1}(X_{1:k-1})), \quad 2 < k < d \quad (5.26)$$

From Theorem 1 in (Hyvärinen and Pajunen 1999), the variables $U_1, \dots, U_d$ are independently and uniformly distributed. Moreover, the combined flow is monotonic as both the copula flow and marginal flow are monotonic. Hence, the inverse of the combined flow can transform any random vector X to uniform independent samples in the cube $\text{unif}[0, 1]^d$.

The flows are learned as invertible monotonic functions. Therefore, using Lemma 5.1 we obtain that the distribution $\mathcal{F}_X^{-1}(C_X(U))$ converges in distribution to X . We can generate any random vector X starting from i.i.d. random variables. Hence, the copula flow model is universal density approximator. 

5.6 Experiments

5.6.1 Mixed Variable Copula Learning

We start with an illustrative example by constructing a generative model with discrete and continuous variables. We demonstrate that the copula flow model can indeed learn the structure as well as the marginal densities. We construct a vine model with three different marginals: 1) X_1 , continuous, Half Normal; 2) X_2 discrete, hypergeometric; 3) X_3 , continuous, Gamma. Since the copula describes a symmetric relationship, we have three different bivariate relations between three variables. We assign three widely used bivariate copulas: a Gaussian for X_1 and X_2 , a Clayton for X_2 and X_3 , and a Gumbel copula for X_1 and X_3 . This model is inspired by the mixed vine data model described in (Onken and Panzeri 2016), where authors propose to build a likelihood using CDFs and vine for mixed data. We use this model as a baseline for comparison.

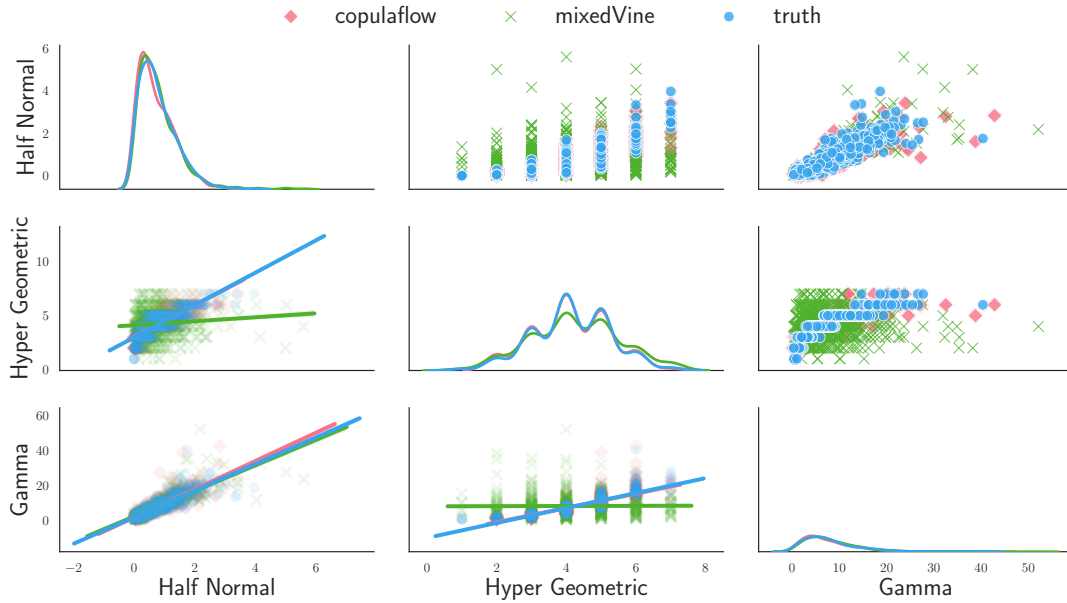


Figure 5.10: Mixed variable copula learning: joint marginals, joint scatter plots, regression plots. Fitting a mixed type vine (Onken and Panzeri 2016). X_2 is a discrete Hyper geometric, X_1 is Half Normal, and X_3 is Gamma distributed. Pairwise copulas (X_1 and X_2) Gaussian, (X_2 and X_3) Clayton, (X_1 and X_3) Gumbel. The diagonal plots show the marginals for each variable, whereas the upper triangle shows joint scatter plots. The lower triangle shows regression plots for each variable. Here, we overlay a linear regression fit and data for the pair of variables. For continuous variables X_1 and X_3 all three methods produce regression lines similar to each other. For relations between discrete and continuous variables, the copula flow and the ground truth align. However, for both discrete to continuous relations, the mixed vine fails to capture the relation as evident by the misaligned linear regression fit.

The copula flow can learn all three marginals, which are plotted along the diagonal of Figure 5.10, while the mixed vine copula struggles a bit (second row, second column). The copula flow also successfully learns copulas between discrete and continuous variables, as evident by the linear regression line in the lower half of the plots in Figure 5.10. Both the mixed vine model (Onken and Panzeri 2016) and the copula flow can learn

the continuous copula well as evident by the left lowermost regression plot, but the mixed vine copula struggles with mixed variable copulas (third row, second column of Figure 5.10). Scatter plots in the upper diagonal show that the samples from mixed vine model are more spread out while copula flow samples align with the true samples.

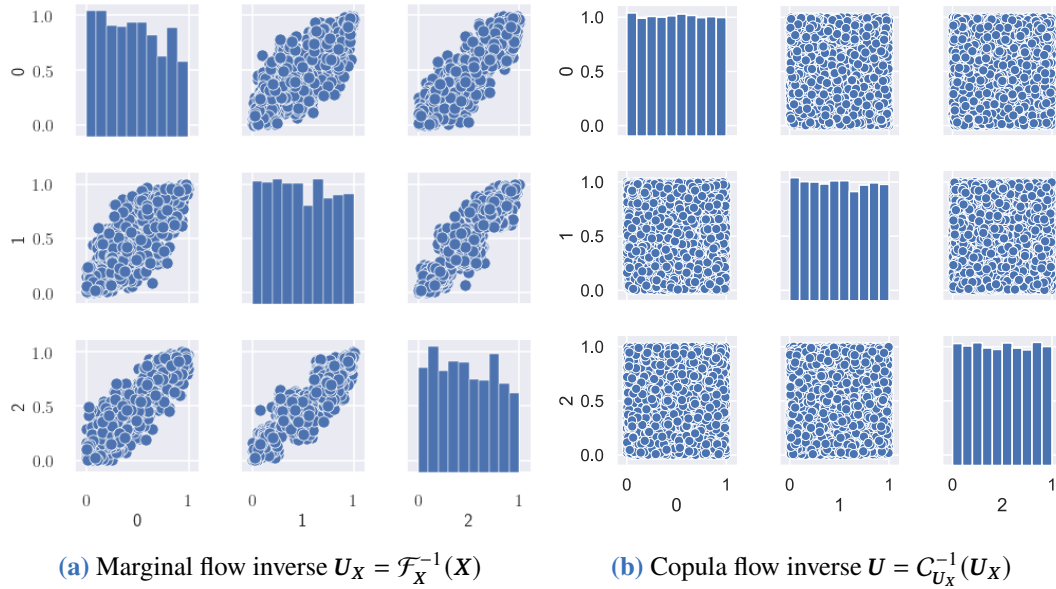


Figure 5.11: The figure depicts inverse flow transformation of the 3 dimensional mixed vine example. In the Figure 5.11a we can clearly see the copula structures in the off-diagonal scatter plots. The almost uniform distribution along the diagonal indicates that $\mathcal{F}_X^{-1} \approx F_X$. The Figure 5.11b on the right shows inverse transform applied to the output of inverse flow, i.e. $C_X^{-1}(\mathcal{F}_X^{-1} \approx F_{CX})$. As the evident from scatter plots the copula flow and marginal flow together closely match true CDF.

5.6.2 Density Estimation

For density estimation, we compare the performance against the current state of the art neural density estimators. The standard benchmark introduced by (Papamakarios, Pavlakou, and Murray 2017) modifies the original data with discrete marginals. For example, in the UCI power data-set, the authors add a uniform random noise to the sub-metering columns. This is equivalent to using the distributional transform, Definition 5.2, with a fixed seed. These ad-hoc changes defeat the purpose of interpretability and transparency. However, to maintain the exact comparison across different density estimators we keep the data, training, test, and validation splits exactly as in (Papamakarios, Pavlakou, and Murray 2017), with the code provided by (Oliva et al. 2018). We use CDF splines flow for estimating the marginals, which is equivalent to estimating the model with an independence copula. We obtain the univariate uniform marginals by inverting the flow, and then fit a copula flow model to obtain the joint density. We summarise the density estimation results in Table 5.1. The marginal likelihood scores ignore the relationships amongst the variables, whereas the copula likelihood shows information

Model	Power	Gas	Hepmass	Miniboone
FFJORD	0.46 ± 0.01	8.59 ± 0.12	-14.92 ± 0.08	-10.43 ± 0.04
RQ-NSF (AR)	0.66 ± 0.01	13.09 ± 0.02	-14.01 ± 0.03	-9.22 ± 0.48
MAF	0.45 ± 0.01	12.35 ± 0.02	-17.03 ± 0.02	-10.92 ± 0.46
Marginal Flows $\mathcal{F}$	-0.80 ± 0.02	-6.67 ± 0.02	-26.42 ± 0.05	-53.17 ± 0.06
Copula Flow $\mathcal{C}$	1.39 ± 0.03	15.6 ± 0.67	5.4 ± 0.10	37.77 ± 0.21
Joint Model $\mathcal{C} + \mathcal{F}$	0.59 ± 0.03	8.05 ± 0.68	-19.6 ± 0.12	-14.83 ± 0.21

Table 5.1: Test log-likelihoods (nats) on density estimation benchmarks. Exact replication of the dataset and splits as provided by (Papamakarios, Pavlakou, and Murray 2017). The algorithms are MAF (Papamakarios, Pavlakou, and Murray 2017), RQ-NSF (AR) (Durkan et al. 2019).

	Power		Gas	
	KS Stat	P value	KS Stat	P value
1	0.012	0.474	0.009	0.755
2	0.011	0.582	0.017	0.099
3	0.012	0.521	0.017	0.39
4	0.014	0.320	0.016	0.509
5	0.012	0.452	0.014	0.256
6	0.009	0.792	0.016	0.148
			0.010	0.685
			0.011	0.590

Table 5.2: The table shows the Kolmogorov-Smirnov KS test for each of the six dimensions. We perform a two-sample KS test. A total of 10,000 samples were selected as the two-sample KS test is exact up to 10,000 data points; it uses approximations for large sample sizes. Low values of test statistic and high p-values indicates we cannot reject the hypothesis that two samples are from the same distribution.

gained by using copula. The joint likelihood can be expressed as sum of marginal and copula likelihoods, see (5.9) The copula flow based model achieves a performance that is close to the state of the art. Further improvements may be achieved by fine tuning the copula flow structure as the marginal CDFs are close to the empirical CDF as verified by using the Kolmogorov-Smirnov (KS) test ; see Table 5.2

5.6.3 Synthetic Data

Estimating the performance of data generators is a challenging process (Theis, Van Den Oord, and Bethge 2016). Merely having very high likelihood on the training or test set is not necessarily the best metric. One measure to look at is the machine learning efficacy of the generated data, i.e., whether can we use the synthetic data to effectively perform machine learning tasks (e.g., regression or classification), which we wished to accomplish with the true data (Xu et al. 2019; Jordon, Yoon, and Van Der Schaar 2019). Here, we test the synthetic data generator using the framework introduced in (Xu et al. 2019). The copula flow model can capture the relations between different variables while learning a density model as it is evident by the high scores in the ML tasks benchmark;

see Table 5.3. For Gaussian mixture (GM) based models, the copula flow model performs very close to the ‘Identity’ transform, which uses the true data. This indicates a high similarity of the true and the synthetic data-set. For real data, the model performs very close to the current state of the art (Xu et al. 2019) for classification tasks and outperforms it on regression tasks on aggregate.

Method	GM Sim.		BN Sim.		Real	
	$\mathcal{L}_{syn}$	$\mathcal{L}_{test}$	$\mathcal{L}_{syn}$	$\mathcal{L}_{test}$	clf	reg
Identity	-2.61	-2.61	-9.33	-9.36	0.743	0.14
CLBN (Che et al. 2017)	-3.06	7.31	-10.66	-9.92	0.382	-6.28
PrivBN (Zhang et al. 2017)	-3.38	-12.42	-12.97	-10.90	0.225	-4.49
MedGAN (Choi et al. 2017)	-7.27	-60.03	-11.14	-12.15	0.137	-8.80
VeeGAN (Srivastava et al. 2017)	-10.06	-4.22	-15.40	-13.86	0.143	-6.5e6
TableGAN (Park et al. 2018)	-8.24	-4.12	-11.84	-10.47	0.162	-3.09
TVAE (Xu et al. 2019)	-2.65	-5.42	-6.76	-9.59	0.519	-0.20
CTGAN (Xu et al. 2019)	-5.72	-3.40	-11.67	-10.60	0.469	-0.43
CopulaFlow	-3.17	-2.98	-10.51	-9.69	0.431	-0.18

Table 5.3: Performance of the copula flow models for synthetic data generation. We benchmark by using the synthetic data to perform the machine learning (ML) task, classification or regression, associated with the data-set. The metric is generated for the original test data. Gaussian mixture (GM) and Bayesian network (BN) simulations correspond to data generated from known GM or BN models. Log-likelihoods $\mathcal{L}_{syn}$ of the true model given synthetic data and $\mathcal{L}_{test}$ of the generative model given the test data are shown. All values except the copula flow are from the framework in (Xu et al. 2019).

5.7 Conclusion

In this work we proposed a new probabilistic model to generate high fidelity synthetic data. Our model is based on copula theory which allows us to build interpretable and flexible model for the data generation. We used distributional transform to address the challenges associated with learning copulas for discrete data. Experimental results show that our proposed model is capable of learning mixed variable copulas required for tabular data-sets. We show that synthetic data generated from a variety of real data-sets is capable of capturing the relationship amongst the mixed variables.

5.8 Future Work

Our model can be extended to generate data with differential privacy guarantees (H. Li, Xiong, and Jiang 2014). We consider this as an important extension and future work. Moreover, our copula flow model can also be used, where copula models are used, e.g., in finance and actuarial science.

Part IV

Conclusion

Conclusion

In this work we proposed an algorithm for data-efficient RL that is based on probabilistic MPC. To tackle the uncertainty of the true environment we learned the transition models using Gaussian processes. These transition models are stochastic, however, we re-formulate the optimal control problem with uncertainty propagation via moment matching into an deterministic optimal control problem. Such a reformulation allows us to exploit Pontryagin’s maximum principle to build an algorithm that can naturally deal with state and control constraints. MPC and non-parametric nature of Gaussian processes allows the learned model to be updated immediately, which leads to an increased robustness with respect to model inaccuracies. We provided empirical evidence that our framework is not only theoretically sound, but also extremely data efficient, while being able to learn in settings with hard state constraints.

For scenarios where entire state may not be observable we proposed an iterative state estimator for non-linear dynamical systems based on Expectation Propagation (EP). The standard derivative based EP method is numerically unstable for non-linear systems ([Seeger 2005](#)). Similar to Kalman smoothers we can use implicit linearisation to build an iterative estimator using Expectation propagation. We showed that to get the correct estimates one needs to explicitly account for the implicit linearisation. The proposed method gets best of both worlds, numerical stability of Kalman based estimators and iterative improvements of the EP algorithm. We further improve the algorithm by using damping and power EP. With these extensions, any Kalman filter or smoother can be shown as special case of the proposed method.

In final section we proposed a new probabilistic model to generate high fidelity synthetic data based on copula theory. Model based reinforcement learning method

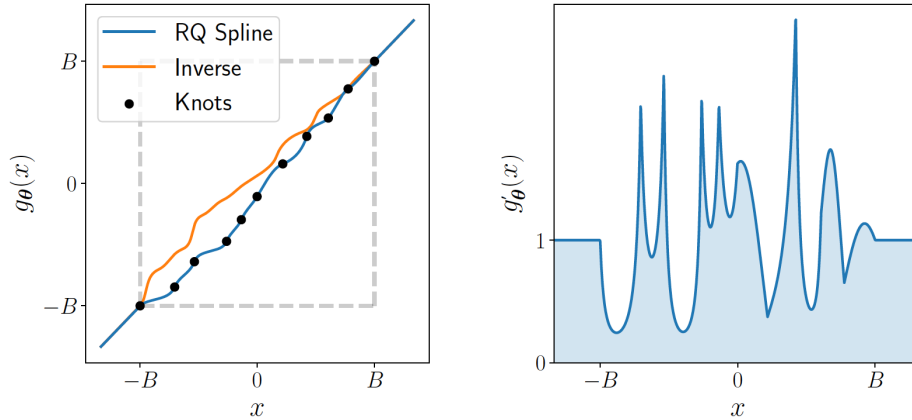
proposed in the first section relies on 'synthetic' data generated by the model to develop optimal control solution. Model here acts as a proxy to the real environment. The applications of such an data generator can go beyond reinforcement learning. We proposed an interpretable and flexible model for the data generation based on copula theory and recent advances in Normalising flows. We used distributional transform to address the challenges associated with learning copulas for discrete data. Experimental results show that our proposed model is capable of learning mixed variable copulas required for tabular data-sets. We show that synthetic data generated from a variety of real data-sets is capable of capturing the relationship amongst the mixed variables.

In real world applications we are often faced with 1) limited data 2) partial observability and 3) privacy concerns. Each of the sections in the thesis attempt to solve these problems in a novel way. All or selective parts of the work presented in this theses can be used to build a real world data efficient reinforcement learning algorithm.

Monotonic Rational Quadratic Transforms

This section is replica of the Appendix in (Durkan et al. 2019), reproduced here to make this chapter complete on its own.

A.1 Parametrisation of spline



$$\left(x^{(0)}, y^{(0)}\right) = (-B, -B) \quad (\text{A.1})$$

$$\left(x^{(K)}, y^{(K)}\right) = (B, B) \quad (\text{A.2})$$

$$x^{(k)} < x^{(k+1)} \text{ and } y^{(k)} < y^{(k+1)} \quad \text{for all } k = 0, \dots, K-1 \quad (\text{A.3})$$

Let $\{\delta^{(k)}\}_{k=0}^K$ be the non-negative derivative values at these knot points (we take $\delta^{(0)} = \delta^{(K)} = 1$ so that the spline matches the derivative of the linear tails). Given these quantities, the algorithm of (Gregory and Delbourgo 1982) defines a monotonic rational-quadratic spline which passes through each knot and has the given derivative value at each knot as follows:

1. Let $w^{(k)} = x^{(k+1)} - x^{(k)}$ be the bin widths, and $s^{(k)} = (y^{(k+1)} - y^{(k)})/w^{(k)}$ be the slopes of the lines joining the co-ordinates.
2. For $x \in [x^{(k)}, x^{(k+1)}]$, let $\xi = (x - x^{(k)})/w^{(k)}$, so that $\xi \in [0, 1]$.
3. Then, for $x \in [x^{(k)}, x^{(k+1)}]$, $k = 0, \dots, K-1$, define

$$g(x) = \frac{\alpha^{(k)}(\xi)}{\beta^{(k)}(\xi)} \quad (\text{A.4})$$

where

$$\alpha^{(k)}(\xi) = s^{(k)}y^{(k+1)}\xi^2 + \left[y^{(k)}\delta^{(k+1)} + y^{(k+1)}\delta^{(k)}\right]\xi(1-\xi) + s^{(k)}y^{(k)}(1-\xi)^2 \quad (\text{A.5})$$

$$\beta^{(k)}(\xi) = s^{(k)}\xi^2 + \left[\delta^{(k+1)} + \delta^{(k)}\right]\xi(1-\xi) + s^{(k)}(1-\xi)^2 \quad (\text{A.6})$$

As noted in (Gregory and Delbourgo 1982) $\beta^{(k)}(\xi)$ can be rewritten as

$$\beta^{(k)}(\xi) = s^{(k)} + \left[\delta^{(k+1)} + \delta^{(k)} - 2s^{(k)}\right]\xi(1-\xi) \quad (\text{A.7})$$

so that the quotient can be written as

$$\frac{\alpha^{(k)}(\xi)}{\beta^{(k)}(\xi)} = y^{(k)} + \frac{(y^{(k+1)} - y^{(k)})[s^{(k)}\xi^2 + \delta^{(k)}\xi(1-\xi)]}{s^{(k)} + [\delta^{(k+1)} + \delta^{(k)} - 2s^{(k)}]\xi(1-\xi)} \quad (\text{A.8})$$

which is less prone to numerical issues, especially for small values of $s^{(k)}$. Authors in (Gregory and Delbourgo 1982) show that the spline defined by Equation (A.4) interpolates between the given knots, satisfies the derivative constraints at the knot points, and is monotonic on each bin. Rational-quadratic functions also provide flexibility over previous approaches: it is not possible to match arbitrary values and derivatives of a function at two boundary knots with a quadratic polynomial, or a monotonic segment of a cubic polynomial.

Additional Plots

B.1 Bivariate Copulas

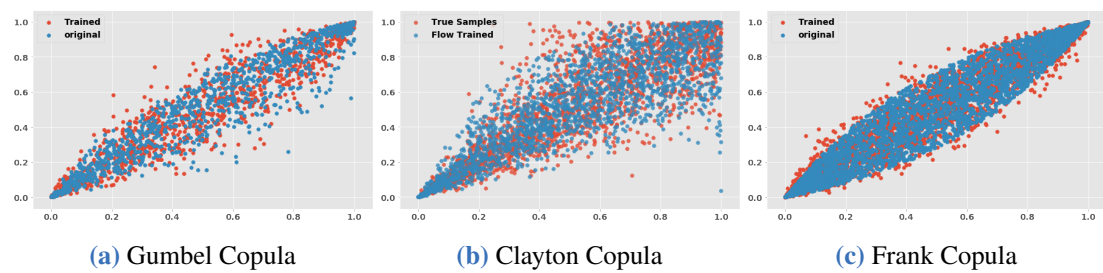


Figure B.1: Learning bivariate Copulas using Copula Flow model. Each bi-variate density is learned by the same flow

Software Details

A key point of our approach is the interpretation of normalising flow functions as a quantile function transforming a uniform density to the desired density via inverse transform sampling. As quantile functions are monotonic, we use monotone rational quadratic splines described in (Durkan et al. 2019) as a normalising flow function. However, we make changes to the original splines. These changes are necessary to ensure that the flow is learning quantile function.

- The univariate flow $\mathcal{F}$ maps from the unif $[0, 1]$ to $\mathbf{ran}(F_X)$ of the random variable X . Hence, unlike (Durkan et al. 2019), our splines are asymmetric in their support.
- We modify the standard spline network to build a map as $(0, 1) \rightarrow (B_{\text{lower}}, B_{\text{upper}})$ where B is the range of the marginal. Infinite support can be added with $B \rightarrow \infty$.
- We do not parametrise the knot positions by a neural network. We instead treat them as a free vector, i.e., $\theta_d = [\theta_d^w, \theta_d^h, \theta_d^s]$ are the width, height and slope parameters, respectively, that characterise the flow function for the independent marginals of data dimension $\mathcal{F}_{\theta_d}$
- We map out-of-bound values back into the range and set the gradients of the flow map to 0 at these locations.

For copula flow we use same network as autoregressive neural spline flows (Durkan et al. 2019). As copula is a CDF defined over uniform densities, the copula flow spline maps unif $[0, 1] \rightarrow \text{unif}[0, 1]$. Apart from the change in the range of the splines, the copula flow architecture is same as that of autoregressive neural spline flows (Durkan et al. 2019).

		Power	Gas	Hepmass	Miniboone	Synthetic Data
	Dimension	6	8	21	43	Various
	Train Data Points	1,615,917	852,174	315,123	29,556	Various
Marginal Flow	Batch Size	2048	1024	1024	512	1024
	Epochs	50	50	50	50	100
	Bins	512	512	512	512	512
	Learning Rate	1e-3	1e-3	1e-3	1e-3	1e-3
	Batch Size	512	512	512	512	512
Copula Flow	Epochs	100	100	100	50	100
	Bins	16	16	16	8	16
	Learning Rate	1e-4	1e-4	1e-4	1e-4	1e-4
	Hidden Features	[512, 512]	[256, 256]	[256, 256]	[128, 128]	[512, 512]
	Number of Flows	5	10	15	10	10

Table C.1: Hyperparameters for the density estimation results

References

- Sutton and Barto (1998). *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning. Cambridge, MA, USA: The MIT Press (cit. on pp. 11, 12).
- Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dhharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis (2015). “Human-Level Control through Deep Reinforcement Learning”. In: *Nature* (cit. on p. 11).
- Silver, David, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis (2016). “Mastering the Game of Go with Deep Neural Networks and Tree Search”. In: *Nature* (cit. on p. 11).
- Yahya, Ali, Adrian Li, Mrinal Kalakrishnan, Yevgen Chebotar, and Sergey Levine (2016). “Collective Robot Reinforcement Learning with Distributed Asynchronous Guided Policy Search”. In: *arXiv preprint arXiv:1610.00673* (cit. on p. 11).
- Schulman, John, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel (2015). “Trust Region Policy Optimization”. In: arXiv: 1502.05477 (cit. on pp. 11, 12).
- Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov (2017). “Proximal Policy Optimization Algorithms”. In: arXiv: 1707.06347 (cit. on pp. 11, 12).
- Kakade, Sham and John Langford (2002). “Approximately Optimal Approximate Reinforcement Learning”. In: *19th International Conference on Machine Learning*, pp. 267–274 (cit. on p. 12).
- Haarnoja, Tuomas, Aurick Zhou, Pieter Abbeel, and Sergey Levine (2018). “Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor”. In: *35th International Conference on Machine Learning, ICML 2018*, pp. 2976–2989. eprint: 1801.01290 (cit. on p. 12).

- Fujimoto, Scott, David Meger, and Doina Precup (2019). “Off-Policy Deep Reinforcement Learning without Exploration”. In: *36th International Conference on Machine Learning, ICML 2019*, pp. 3599–3609. eprint: [1812.02900](#) (cit. on p. 12).
- Agarwal, Rishabh, Dale Schuurmans, and Mohammad Norouzi (2020). “An Optimistic Perspective on Offline Reinforcement Learning”. In: *Proceedings of the 37th International Conference on Machine Learning*. Proceedings of Machine Learning Research. Virtual: PMLR, pp. 104–114 (cit. on p. 12).
- Ng, Andrew Y and Michael Jordan (2000). “Pegasus: A Policy Search Method for Large MDPs and POMDPs”. In: *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pp. 406–415 (cit. on pp. 12, 35).
- Sutton, Richard S. (1990). “Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming”. In: *Machine Learning Proceedings 1990*. Morgan Kaufmann, pp. 216–224 (cit. on p. 12).
- Schneider, Jeff G (1997). “Exploiting Model Uncertainty Estimates for Safe Dynamic Control Learning”. In: *Advances in Neural Information Processing Systems*. Morgan Kaufman Publishers (cit. on pp. 12, 34, 35).
- Deisenroth, Marc P and Carl E Rasmussen (2011). “PILCO: A Model-Based and Data-Efficient Approach to Policy Search”. In: *Proceedings of the International Conference on Machine Learning*. ACM, pp. 465–472 (cit. on pp. 12, 34–36, 44).
- Pan, Yunpeng and Evangelos Theodorou (2014). “Probabilistic Differential Dynamic Programming”. In: *Advances in Neural Information Processing Systems* (cit. on pp. 12, 36, 42).
- Deisenroth, Fox, and Rasmussen (Feb. 2015). “Gaussian Processes for Data-Efficient Learning in Robotics and Control.” In: *IEEE transactions on pattern analysis and machine intelligence* 2, pp. 408–23. eprint: [1502.02860v1](#) (cit. on pp. 12, 13, 36, 38, 43, 44, 47).
- Cutler, Mark and Jonathan P. How (2015). “Efficient Reinforcement Learning for Robots using Informative Simulated Priors”. In: *2015 IEEE International Conference on Robotics and Automation*. IEEE, pp. 2605–2612 (cit. on pp. 12, 36, 42).
- Williams, Grady, Andrew Aldrich, and Evangelos Theodorou (2015). “Model Predictive Path Integral Control using Covariance Variable Importance Sampling”. In: arXiv: [1509.01149](#) (cit. on p. 12).
- Hafner, Danijar, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson (2018). “Learning Latent Dynamics for Planning from Pixels”. In: *36th International Conference on Machine Learning, ICML 2019*, pp. 4528–4547. eprint: [1811.04551](#) (cit. on p. 12).
- Chua, Kurtland, Roberto Calandra, Rowan McAllister, and Sergey Levine (2018). “Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models”.

- In: *Advances in Neural Information Processing Systems*, pp. 4754–4765. arXiv: [1805.12114](#) (cit. on p. [12](#)).
- Janner, Michael, Justin Fu, Marvin Zhang, and Sergey Levine (2019). “When to Trust Your Model: Model-Based Policy Optimization”. In: arXiv: [1906.08253](#) (cit. on p. [13](#)).
- Mayne, David Q (2016). “Robust and stochastic model predictive control: Are we going in the right direction?” In: *Annual Reviews in Control*, pp. 184–192 (cit. on pp. [13](#), [18](#)).
- Kalman, Rudolf E (1960). “A New Approach to Linear Filtering and Prediction Problems”. In: *Transactions of the ASME — Journal of Basic Engineering* 82.Series D, pp. 35–45 (cit. on pp. [14](#), [50](#)).
- Julier, Uhlmann, and Durrant-Whyte (2000). “A New Method for the Nonlinear Transformation of Means and Covariances in Filters and Estimators”. In: *IEEE Transactions on Automatic Control* 45.3, pp. 477–482 (cit. on pp. [14](#), [51](#), [52](#), [55](#)).
- Lefebvre, Tine, Herman Bruyninckx, and Joris De Schutter (2004). “Kalman Filters for Non-linear systems: A comparison of Performance”. In: *International Journal of Control* 77.7, pp. 639–653 (cit. on pp. [14](#), [51](#)).
- Minka, Thomas P (2001a). “A Family of Algorithms for Approximate Bayesian Inference”. PhD thesis. Cambridge, MA, USA: Massachusetts Institute of Technology (cit. on pp. [14](#), [51](#), [56](#), [57](#), [59](#), [61](#), [70](#)).
- Qi, Yuan and Thomas Minka (2003). “Expectation Propagation for Signal Detection in Flat-Fading Channels”. In: *Proceedings of the IEEE International Symposium on Information Theory* (cit. on pp. [14](#), [51](#), [59](#), [63](#)).
- Patki, Neha, Roy Wedge, and Kalyan Veeramachaneni (2016). “The Synthetic Data Vault”. In: *3rd IEEE International Conference on Data Science and Advanced Analytics, DSAA* (cit. on pp. [14](#), [84](#)).
- Beaulieu-Jones, Brett K, Zhiwei Steven Wu, Chris Williams, Ran Lee, Sanjeev P Bhavnani, James Brian Byrd, and Casey S Greene (2019). “Privacy-Preserving Generative Deep Neural Networks Support Clinical Data Sharing”. In: *Circulation. Cardiovascular quality and outcomes* (cit. on pp. [14](#), [82](#)).
- Qin, S. Joe and Thomas A. Badgwell (2003). “A survey of industrial model predictive control technology”. In: *Control Engineering Practice* 7, pp. 733–764 (cit. on pp. [17–19](#)).
- Schacter, Daniel L., Donna Rose Addis, and Randy L. Buckner (2007). “Remembering the Past to Imagine the Future: The Prospective Brain”. In: *Nature Reviews Neuroscience* 9, pp. 657–661 (cit. on p. [17](#)).
- Camacho, EF and C Bordons (2004). *Model Predictive Control*. 2nd ed. Springer (cit. on pp. [17–19](#), [34](#)).

- Anderson, Brian D. O. and John B. Moore (1990). *Optimal Control: Linear Quadratic Methods*. Prentice-Hall, Inc. (cit. on p. 17).
- Rawlings, James, Dvid. Mayne, and Moritz Diehl (2017). *Model Predictive Control: Theory, Computation, and Design*. 2nd. y Nob Hill Publishing, LLC (cit. on pp. 18, 34).
- Mayne, David Q., James B. Rawlings, Crhistopher V. Rao, and Pierre O.M. Scokaert (2000). “Constrained Model Predictive Control: Stability and Optimality”. In: *Automatica* 6, pp. 789–814 (cit. on pp. 18, 34).
- Bliss, Gilbert A (1946). *Lectures on the calculus of variations*. Chicago Univ. Press (cit. on p. 20).
- Bellman, Richard E. (1954). *The Theory of Dynamic Programming* (cit. on p. 22).
- Clarke, Frank H. (1990). *Optimization and Non-Smooth Analysis*. Society for Industrial and Applied Mathematics SIAM (cit. on pp. 22, 25, 28, 38, 41).
- Todorov, E (2006). “Optimal Control Theory”. In: *Bayesian Brain: Probabilistic Approaches to Neural Networks*. MIT Press. Chap. 06 (cit. on pp. 23, 24, 32).
- Bertsekas, Dimitri P (2010). *Dynamic Programming and Optimal Control*. 3rd. Athena Scientific, pp. 1–200 (cit. on pp. 24, 25).
- Crandall, Michael G., Pierre-Louis Lions, Sadakazu Aizawa, Viorel Barbu, Anatole Beck, Jr. Benton, Stanley H., Jean-Michel Bony, E. D. Conway, and E. Hopf (1983). “Viscosity solutions of Hamilton-Jacobi equations”. In: *Transactions of the American Mathematical Society* 1, pp. 1–1 (cit. on p. 24).
- Pontryagin, Lev Semenovich, E F Mishchenko, V G Boltyanskii, and R V Gamkrelidze (1962). *The Mathematical Theory of Optimal Processes*. Wiley (cit. on pp. 24, 25).
- Diehl, M (2011). *Numerical Optimal Control*. Tech. rep. ESAT-SCD, KU (cit. on p. 24).
- Schättler, Heinz M. and Urszula. Ledzewicz (2012). *Geometric Optimal Control: Theory, Methods and Examples*. 9. Springer, pp. 1689–1699 (cit. on pp. 25, 34, 38–41).
- Lewis, Andrew D. (2006). *The Maximum Principle of Pontryagin in Control and in Optimal Control*. Tech. rep. Department of Mathematics and Statistics, Queen’s University, KIngston, Canada (cit. on pp. 25, 26, 29).
- Ross, I Michael (2015). *A Primer on Pontryagin’s Principle in Optimal Control*. Collegiate publishers (cit. on pp. 26, 32).
- Ross, I. Michael and Mark Karpenko (2012). “A review of pseudospectral optimal control: From theory to flight”. In: *Annual Reviews in Control* 2, pp. 182–197 (cit. on p. 26).
- Boltyanski, Vladimir (1993). “Tent Method in Optimal Control Theory.” In: *Optimal Control*. ISNM International Series of Numerical Mathematics, Birkhauser Basel (cit. on p. 26).
- Boltyanski, Vladimir G. and Alexander S. Poznyak (2012). *The Robust Maximum Principle*. Boston: Birkhäuser Boston (cit. on pp. 27–30).

- Boltyanski, V (1975). “The Tent method in the Theory of Extremal Problems (in Russian)”. In: *Usp. Mat. Nauk* 30 (cit. on pp. 28, 29).
- Farkas, Julius (1902). “Theorie der Einfachen Ungleichungen”. In: *Journal für die Reine und Angewandte Mathematik* 1902.124, pp. 1–27 (cit. on p. 29).
- Boltyanski, V (1958). “The Maximum Principle in the Theory of Optimal Processes (in Russian)”. In: *Dokl. Akad. Nauk SSSR* (cit. on p. 29).
- Naidu, D. S. (Desineni S.) and Richard C. Naidu, Subbaram/Dorf (2003). *Optimal Control Systems*. CRC Press, p. 433 (cit. on pp. 34, 40).
- Rasmussen, Carl E and Christopher K I Williams (2006). *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. Cambridge, MA, USA: The MIT Press (cit. on pp. 35, 36).
- MacKay, David J C (1998). “Introduction to Gaussian Processes”. In: *Neural Networks and Machine Learning*. Berlin, Germany: Springer, pp. 133–165 (cit. on p. 36).
- Girard, Agathe, Carl E Rasmussen, Joaquin Quiñero~Candela, and Roderick Murray-Smith (2003). “Gaussian Process Priors with Uncertain Inputs—Application to Multiple-Step Ahead Time Series Forecasting”. In: *Advances in Neural Information Processing Systems 15*. Cambridge, MA, USA: The MIT Press, pp. 529–536 (cit. on pp. 36, 42).
- Quinonero-Candela, J and CE Rasmussen (2005). “A Unifying View of Sparse Approximate Gaussian process Regression”. In: *The Journal of Machine Learning Research* (cit. on p. 36).
- Bischoff, Bastian, Duy Nguyen-Tuong, Torsten Koller, Heiner Markert, and Alois Knoll (2013). “Learning Throttle Valve Control Using Policy Search”. In: *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases* (cit. on p. 36).
- Bischoff, B., D. Nguyen-Tuong, H. van Hoof, A. McHutchon, C. E. Rasmussen, A. Knoll, J. Peters, and M.P. Deisenroth (2014). “Policy Search for Learning Robot Control using Sparse Data”. In: *2014 IEEE International Conference on Robotics and Automation*. IEEE, pp. 3882–3887 (cit. on pp. 36, 42).
- Pan, Yunpeng, Evangelos Theodorou, and Michail Kontitsis (2015). *Sample Efficient Path Integral Control under Uncertainty* (cit. on pp. 36, 42).
- Bock, Hans G and Karl J Plitt (1984). “A Multiple Shooting Algorithm for Direct Solution of Optimal Control Problems”. In: *Proceedings 9th IFAC World Congress Budapest*. Pergamon Press, pp. 243–247 (cit. on pp. 37, 42).
- Todorov, Emanuel (2009). “Efficient Computation of Optimal Actions.” In: *Proceedings of the National Academy of Sciences of the United States of America* 28, pp. 11478–83 (cit. on pp. 38, 40, 41).
- Diehl, Moritz (2014). *Lecture Notes on Optimal Control and Estimation*. Tech. rep., p. 97 (cit. on pp. 40, 41).

- Todorov, Emanuel and Weiwei Li (2005). “A Generalized Iterative LQG Method for Locally-Optimal Feedback Control of Constrained Nonlinear Stochastic Systems”. In: *Proceedings of the American Control Conference* (cit. on p. 40).
- Toussaint, Marc (2009). “Robot Trajectory Optimization using Approximate Inference”. In: *Proceedings of the 26th International Conference on Machine Learning*. Montreal, QC, Canada (cit. on pp. 40, 42).
- Bellman, Richard E (1957). *Dynamic Programming*. Princeton, NJ, USA: Princeton University Press (cit. on p. 41).
- Nocedal, Jorge and Stephen J Wright (2006). *Numerical Optimization*. Springer, XXII, 664 s. 85 illus. eprint: [NIHMS150003](#) (cit. on p. 41).
- Griewank, A. and Ph L. Toint (1982). “Partitioned Variable Metric Updates for Large Structured Optimization Problems”. In: *Numerische Mathematik* 1, pp. 119–137 (cit. on pp. 41, 42).
- Polydoros, Athanasios S. and Lazaros Nalpantidis (2017). “Survey of Model-Based Reinforcement Learning: Applications on Robotics”. In: *Journal of Intelligent and Robotic Systems* 2, pp. 153–173 (cit. on p. 42).
- Jaderberg, Max, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu (2016). “Reinforcement Learning with Unsupervised Auxiliary Tasks”. In: *5th International Conference on Learning Representations*. eprint: [1611.05397](#) (cit. on p. 42).
- Grancharova, Alexandra, Juš Kocijan, and Tor A Johansen (June 2008). “Explicit Stochastic Predictive Control of Combustion Plants based on Gaussian Process Models”. In: *Automatica* 6, pp. 1621–1631 (cit. on p. 42).
- Nghiem, Truong X and Colin N Jones (2017). “Data-driven Demand Response Modeling and Control of Buildings with Gaussian Processes”. In: *Proceedings of the American Control Conference* (cit. on pp. 42, 44).
- Klenske, Edgar D., Melanie N. Zeilinger, Bernhard Schölkopf, and Philipp Hennig (2016). “Gaussian Process-Based Predictive Control for Periodic Error Correction”. In: *IEEE Transactions on Control Systems Technology* 1, pp. 110–121 (cit. on p. 42).
- Ostafew, Ch.J., A.P. Schoellig, T.D. Barfoot, and J. Collier (2015). “Learning-based Nonlinear Model Predictive Control to Improve Vision-based Mobile Robot Path Tracking”. In: *Journal of Field Robotics* 1, pp. 133–152 (cit. on p. 42).
- Ostafew, Chris J., Angela P. Schoellig, and Timothy D. Barfoot (2016). “Robust Constrained Learning-based NMPC enabling reliable mobile robot path tracking”. In: *International Journal of Robotics Research* 13, pp. 1547–1563 (cit. on p. 42).
- Boedecker, Joschka, Jost Tobias Springenberg, Jan Wülfing, and Martin Riedmiller (2014). “Approximate Real-Time Optimal Control Based on Sparse Gaussian Process Models”. In: *Proceedings of Adaptive Dynamic Programming and Reinforcement Learning* (cit. on pp. 42, 44).

- Kocijan, Juš (2016). *Modelling and Control of Dynamic Systems Using Gaussian Process Models*. Advances in Industrial Control. Cham: Springer International Publishing (cit. on p. 42).
- Hennig, Philipp (2011). “Optimal Reinforcement Learning for Gaussian Systems”. In: *Advances in Neural Information Processing Systems 24*. Curran Associates, Inc., pp. 325–333 (cit. on p. 42).
- Rawlik, Konrad, Marc Toussaint, and Sethu Vijayakumar (2012). “On Stochastic Optimal Control and Reinforcement Learning by Approximate Inference”. In: *Robotics: science and systems*, pp. 3052–3056 (cit. on p. 42).
- Raiko, Tapani and Matti Törnio (2009). “Variational Bayesian Learning of Nonlinear Hidden State-Space Models for Model Predictive Control”. In: *Neurocomputing* 16–18, pp. 3702–3712 (cit. on p. 43).
- Deisenroth, Marc P and Carl E Rasmussen (2010). *A Practical and Conceptual Framework for Learning in Control*. Tech. rep. UW-CSE-10-06-01. 185 Stevens Way, Seattle, WA, 98195, USA: Department of Computer Science && Engineering, University of Washington (cit. on p. 47).
- Rauch, H. E., F. Tung, and C. T. Striebel (1965). “Maximum Likelihood Estimates of Linear Dynamic Systems”. In: *AIAA Journal* 3.8, pp. 1445–1450 (cit. on p. 50).
- Anderson, Brian D O and John B Moore (2005). *Optimal Filtering*. Mineola, NY, USA: Dover Publications (cit. on pp. 50, 54, 63).
- Doucet, Arnaud, Simon J Godsill, and Christophe Andrieu (2000). “On Sequential Monte Carlo Sampling Methods for Bayesian Filtering”. In: *Statistics and Computing* 10, pp. 197–208 (cit. on pp. 50, 53, 57).
- Särkkä, Simo (2008). “Unscented Rauch-Tung-Striebel Smoother”. In: *IEEE Transactions on Automatic Control* 53.3, pp. 845–849 (cit. on pp. 51, 52).
- Gelb, Arthur (1974). *Applied Optimal Estimation*. M.I.T. Press, p. 374 (cit. on p. 51).
- Bell, B.M. and F.W. Cathey (1993). “The Iterated Kalman Filter Update as a Gauss-Newton method”. In: *IEEE Transactions on Automatic Control* 38.2, pp. 294–297 (cit. on p. 51).
- Raitoharju, Matti, Lennart Svensson, Ángel F García-Fernández, and Robert Piché (2018). “Damped Posterior Linearization Filter”. In: *IEEE Signal Processing Letters* 25.4, pp. 536–540. arXiv: [arXiv:1704.01113v2](https://arxiv.org/abs/1704.01113v2) (cit. on pp. 51, 56).
- Gultekin, San and John Paisley (2017). “Nonlinear Kalman Filtering With Divergence Minimization”. In: *IEEE Transactions on Signal Processing* 65.23, pp. 1–13. arXiv: [1705.00722](https://arxiv.org/abs/1705.00722) (cit. on pp. 51, 52).
- Thrun, Sebastian, Wolfram Burgard, and Dieter Fox (2005). *Probabilistic Robotics*. Cambridge, MA, USA: The MIT Press (cit. on p. 51).
- Minka, Thomas P (2001b). “Expectation Propagation for Approximate Bayesian Inference”. In: *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence*.

- Ed. by J S Breese and D Koller. Seattle, WA, USA: Morgan Kaufman Publishers, pp. 362–369 (cit. on pp. 51, 57, 61).
- Opper, Manfred and Ole Winther (2001). “Tractable Approximations for Probabilistic Models: The Adaptive TAP Mean Field Approach”. In: *Physical Review Letters* 86.17, p. 5. arXiv: 0102274v1 [arXiv:cond-mat] (cit. on p. 51).
- Heskes, Tom and Onno Zoeter (2002). “Expectation Propagation for Approximate Inference in Dynamic Bayesian Networks”. In: *Proceedings of the International Conference on Uncertainty in Artificial Intelligence*. Ed. by A Darwiche and N Friedman, pp. 216–233 (cit. on p. 51).
- Dehaene, Guillaume P and Simon Barthelmé (2016). “Bounding errors of Expectation-Propagation”. In: *Advances in Neural Information Processing Systems*. arXiv: 1601.02387 (cit. on p. 51).
- Heskes, Tom and Onno Zoeter (2003). *Extended Version: Expectation Propagation for Approximate Inference in Dynamic Bayesian Networks*. Tech. rep. Nijmegen: University of Nijmegen, p. 28 (cit. on pp. 52, 53, 70).
- Zhu, Huaiyu and Richard Rohwer (1995). *Information Geometric Measurements of Generalisation*. Tech. rep. Birmingham: Aston University (cit. on pp. 52, 71).
- Minka, Tom Thomas (2005). *Divergence Measures and Message Passing*. Tech. rep. Microsoft Research (cit. on pp. 52, 59, 71, 72, 75).
- Piché, Robert, Simo Särkkä, and Jouni Hartikainen (2012). “Recursive Outlier-Robust Filtering and Smoothing for Nonlinear Systems using the Multivariate Student-t Distribution”. In: *IEEE International Workshop on Machine Learning for Signal Processing, MLSP*. IEEE, pp. 1–6 (cit. on p. 52).
- Jaakkola, Tommi S (2000). “Tutorial on Variational Approximation Methods”. In: *In Advanced Mean Field Methods: Theory and Practice* (cit. on p. 52).
- Deisenroth, Marc P (2009). “Efficient Reinforcement Learning using Gaussian Processes”. PhD thesis. Karlsruhe Institute of Technology (cit. on p. 54).
- Särkkä, Simo, Jouni Hartikainen, Lennart Svensson, and Fredrik Sandblom (2015). “On the Relation between Gaussian Process Quadratures and Sigma-point methods”. ArXiv. arXiv: 1504.05994 (cit. on p. 54).
- Deisenroth, Marc P and Henrik Ohlsson (2011). “A General Perspective on Gaussian Filtering and Smoothing: Explaining Current and Deriving New Algorithms”. In: *Proceedings of the American Control Conference* (cit. on pp. 55, 77, 79).
- Särkkä, Simo (2013). *Bayesian Filtering and Smoothing*. Cambridge University Press, p. 254 (cit. on pp. 55, 78).
- Deisenroth, M.P., R.D. Turner, M.F. Huber, U.D. Hanebeck, and C.E. Rasmussen (2012). “Robust Filtering and Smoothing with Gaussian Processes”. In: *IEEE Transactions on Automatic Control* 57.7, pp. 1865–1871 (cit. on p. 55).

- Arasaratnam, Ienkar and Simon Haykin (2009). “Cubature Kalman Filters”. In: *{IEEE} Transactions on Automatic Control* 54.6, pp. 1254–1269 (cit. on p. 55).
- Senne, K. (1972). “Stochastic Processes and Filtering Theory”. In: *IEEE Transactions on Automatic Control* 17.5, pp. 752–753 (cit. on p. 56).
- Cathey, Frederick W. (1993). “The Iterated Kalman Filter Update as a Gauss—Newton Method”. In: *IEEE Transactions on Automatic Control* 38.2, pp. 294–297 (cit. on p. 56).
- More, J J (1977). “Levenberg. Marquardt Algorithm: Implementation and Theory”. In: *Numerical analysis*. Vol. 630. Lecture notes in mathematics, pp. 105–116 (cit. on p. 56).
- Skoglund, Martin A, Gustaf Hendeby, and Daniel Axehill (2015). “Extended Kalman Filter Modifications based on an Optimization View Point”. In: *Information Fusion (Fusion), 2015 18th International Conference on*, pp. 1856–1861 (cit. on p. 56).
- Havlík, Jindřich and Ondřej Straka (2015). “Performance Evaluation of Iterated Extended Kalman Filter with Variable Step-Length”. In: *Journal of Physics: Conference Series*. Vol. 659. 1 (cit. on p. 56).
- Garcia-Fernandez, Angel F., Lennart Svensson, Mark R. Morelande, and Simo Sarkka (2015). “Posterior Linearization Filter: Principles and Implementation Using Sigma Points”. In: *IEEE Transactions on Signal Processing* 63.20, pp. 5561–5573 (cit. on p. 56).
- Merwe, Rudolph van der and Eric A Wan (2004). “Sigma-Point Kalman Filters for Integrated Navigation”. In: *Proceedings of the 60th Annual Meeting of The Institute of Navigation (ION)*. Dayton, OH, USA (cit. on p. 57).
- Seeger, Matthias W (2005). *Expectation Propagation for Exponential Families*. Tech. rep. University of California Berkeley (cit. on pp. 57, 61, 62, 69, 79, 103).
- Minka, Thomas P (2004). *Power EP*. Tech. rep. MSR-TR-2004-149. Microsoft Research, p. 6 (cit. on pp. 58, 70, 71).
- Särkkä, Simo and Juha Sarmavuori (2013). “Gaussian filtering and smoothing for continuous-discrete dynamic systems”. In: *Signal Processing* 93.2, pp. 500–510 (cit. on p. 58).
- Brandt Petersen Michael Syskind Pedersen, Kaare, Bill Baxter, Brian Templeton, Christian Rishøj, Christian Schröppel, Dan Boley, Douglas L Theobald, Esben Hoegh-Rasmussen, Evripidis Karseras, Georg Martius, Glynne Casteel, Jan Larsen, Jun Bin Gao, Jürgen Struckmeier, Kamil Dedecius, Karim T Abou-Moustafa, Korbinian Strimmer, Lars Christiansen, Lars Kai Hansen, Leland Wilkinson, Liguó He, Loic Thibaut, Markus Froeb, Michael Hubatka, Miguel Barão, Ole Winther, Pavel Sakov, Stephan Hattinger, Troels Pedersen, Vasile Sima, and Vincent Rabaud (2012). *The Matrix Cookbook* (cit. on pp. 60, 68).

- Minka, Thomas P (2008). *EP: A Quick Reference*. Tech. rep. Microsoft Research (cit. on p. 61).
- Deisenroth, Marc Peter and Shakir Mohamed (2012). “Expectation Propagation in Gaussian Process Dynamical Systems”. In: *Advances in Neural Information Processing Systems 2*, pp. 1–9. arXiv: [arXiv:1207.2940v3](#) (cit. on p. 62).
- Amari, Shun-ichi (1985). *Differential-Geometrical Methods in Statistics*. Vol. 28. Lecture Notes in Statistics. New York, NY: Springer New York. arXiv: [arXiv:1011.1669v3](#) (cit. on p. 71).
- Yedidia, J.S., W.T. Freeman, and Y. Weiss (2005). “Constructing Free-Energy Approximations and Generalized Belief Propagation Algorithms”. In: *IEEE Transactions on Information Theory* 51.7, pp. 2282–2312 (cit. on p. 72).
- Hernandez-Lobato, Jose, Yingzhen Li, Mark Rowland, Thang Bui, Daniel Hernandez-Lobato, and Richard Turner (2016). “Black-Box Alpha Divergence Minimization”. In: *Proceedings of The 33rd International Conference on Machine Learning*. Proceedings of Machine Learning Research. PMLR, pp. 1511–1520 (cit. on p. 72).
- Julier, S. (2002). “The Scaled Unscented Transformation”. In: *Proceedings of the 2002 American Control Conference*. Vol. 6, 4555–4559 vol.6 (cit. on pp. 78, 79).
- Elliot, Mark and Josep Domingo-Ferrer (2018). *The Future of Statistical Disclosure Control*. Tech. rep. London, UK: Government Statistical Service. arXiv: [1812.09204](#) (cit. on p. 82).
- Goodfellow, Ian J., Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio (2014). “Generative Adversarial Nets”. In: *Advances in Neural Information Processing Systems*. arXiv: [1406.2661](#) (cit. on p. 82).
- Choi, Edward, Siddharth Biswal, Bradley Malin, Jon Duke, Walter F. Stewart, and Jimeng Sun (2017). “Generating Multi-label Discrete Patient Records using Generative Adversarial Networks”. In: *Machine Learning in Health Care (MLHC)*. arXiv: [1703.06490](#) (cit. on pp. 82, 85, 101).
- Xu, Lei, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni (2019). “Modeling Tabular data using Conditional GAN”. In: *Advances in Neural Information Processing Systems*. arXiv: [1907.00503](#) (cit. on pp. 82, 85, 100, 101).
- Arjovsky, Martin and Léon Bottou (2017). “Towards Principled Methods for Training Generative Adversarial Networks”. In: *International Conference on Learning Representations, ICLR*. arXiv: [1701.04862](#) (cit. on p. 82).
- Wang, Zhengwei, Qi She, and Tomas E Ward (2019). “Generative Adversarial Networks: A Survey and Taxonomy”. In: *arXiv preprint* (cit. on p. 82).
- Kingma, Diederik P and Max Welling (2013). “Auto-Encoding Variational Bayes”. In: *arXiv preprint*. arXiv: [1312.6114](#) (cit. on p. 82).

- Rezende, Danilo Jimenez and Shakir Mohamed (2015). “Variational Inference with Normalizing Flows”. In: *32nd International Conference on Machine Learning, ICML*. arXiv: [1505.05770](#) (cit. on pp. [82](#), [88](#), [89](#)).
- Sklar, M (1959). “Fonctions de Répartition à n Dimensions Et Leurs Marges”. In: *Publications of Institute Statistics University Paris (in French)* 8 (cit. on pp. [83](#), [86](#), [90](#)).
- Nikoloulopoulos, Aristidis K. and Dimitris Karlis (2009). “Modeling Multivariate Count Data Using Copulas”. In: *Communications in Statistics - Simulation and Computation* 1, pp. 172–187 (cit. on p. [83](#)).
- Aas, Kjersti, Claudia Czado, Arnoldo Frigessi, and Henrik Bakken (2009). “Pair-Copula Constructions of Multiple Dependence”. In: *Insurance: Mathematics and Economics* 2, pp. 182–198 (cit. on pp. [83](#), [84](#)).
- Elidan, Gal (2010). “Copula Bayesian Networks”. In: *Advances in Neural Information Processing Systems* (cit. on pp. [83](#), [85](#)).
- Lopez-paz, David, Jose M. Hernández-lobato, and Zoubin Ghahramani (2013). “Gaussian Process Vine Copulas for Multivariate Dependence”. In: *30th International Conference on Machine Learning, ICML* (cit. on p. [83](#)).
- Czado, Claudia (2010). “Pair-Copula Constructions of Multivariate Copulas BT - Copula Theory and Its Applications”. In: *Copula Theory and Its Applications*. Springer, pp. 93–109 (cit. on p. [83](#)).
- Chang, Bo, Shenyi Pan, and Harry Joe (2019). “Vine Copula Structure Learning via Monte Carlo Tree Search”. In: *Proceedings of Machine Learning Research* (cit. on pp. [83](#), [85](#)).
- Durkan, Conor, Artur Bekasov, Iain Murray, and George Papamakarios (2019). “Neural Spline Flows”. In: *Advances in Neural Information Processing Systems*. Vancouver. arXiv: [1906.04032](#) (cit. on pp. [83](#), [85](#), [90](#), [94](#), [100](#), [105](#), [108](#)).
- Tagasovska, Natasa, Damien Ackerer, and Thibault Vatter (2019). “Copulas as High-Dimensional Generative Models: Vine Copula Autoencoders”. In: arXiv: [1906.05423](#) (cit. on p. [85](#)).
- Wiese, Magnus, Robert Knobloch, and Ralf Korn (2019). “Copula & Marginal Flows: Disentangling the Marginal from its Joint”. In: arXiv: [1907.03361](#) (cit. on p. [85](#)).
- Park, Noseong, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim (2018). “Data Synthesis based on Generative Adversarial Networks”. In: *Proceedings of the VLDB Endowment*. arXiv: [1806.03384](#) (cit. on pp. [85](#), [101](#)).
- Jordon, James, Jinsung Yoon, and Mihaela Van Der Schaar (2019). “PATE-GaN: Generating Synthetic Data with Differential Privacy Guarantees”. In: *7th International Conference on Learning Representations, ICLR* (cit. on pp. [85](#), [100](#)).

- Che, Zhengping, Yu Cheng, Shuangfei Zhai, Zhaonan Sun, and Yan Liu (2017). “Boosting Deep Learning Risk Prediction with Generative Adversarial Networks for Electronic Health Records”. In: *Proceedings - IEEE International Conference on Data Mining, ICDM*. arXiv: [1709.01648](#) (cit. on pp. [85](#), [101](#)).
- Papamakarios, George, Theo Pavlakou, and Iain Murray (2017). “Masked Autoregressive Flow for Density Estimation”. In: *Advances in Neural Information Processing Systems*. arXiv: [1705.07057](#) (cit. on pp. [85](#), [94](#), [99](#), [100](#)).
- Müller, Thomas, Brian McWilliams, Fabrice Rousselle, Markus Gross, and Jan Novák (2019). “Neural Importance Sampling”. In: *ACM Transactions on Graphics* 5, pp. 1–19. arXiv: [1808.03856](#) (cit. on p. [85](#)).
- Nash, Charlie and Conor Durkan (2019). “Autoregressive Energy Machines”. In: *ArXiv Preprint*. arXiv: [1904.05626](#) (cit. on p. [85](#)).
- Dinh, Laurent, David Krueger, and Yoshua Bengio (2015). “NICE: Non-linear Independent Components Estimation”. In: *3rd International Conference on Learning Representations, ICLR 2015 - Workshop Track Proceedings*. arXiv: [1410.8516](#) (cit. on p. [85](#)).
- Dinh, Laurent, Jascha Sohl-Dickstein, and Samy Bengio (2019). “Density Estimation using Real NVP”. In: *5th International Conference on Learning Representations, ICLR*. arXiv: [1605.08803](#) (cit. on p. [85](#)).
- Kingma, Diederik P., Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling (2016). *Improved Variational Inference with Inverse Autoregressive Flow Diederik* (cit. on pp. [85](#), [94](#)).
- Oliva, Junier B., Avinava Dubey, Manzil Zaheer, Barnabás Póczos, Ruslan Salakhutdinov, Eric P. Xing, and Jeff Schneider (2018). “Transformation Autoregressive Networks”. In: *35th International Conference on Machine Learning, ICML*. arXiv: [1801.09819](#) (cit. on pp. [85](#), [99](#)).
- Huang, Chin Wei, David Krueger, Alexandre Lacoste, and Aaron Courville (2018). “Neural Autoregressive Flows”. In: *35th International Conference on Machine Learning, ICML*. arXiv: [1804.00779](#) (cit. on p. [85](#)).
- Nelsen, Roger B. (2006). *An Introduction to Copulas*. New York, NY: Springer New York, pp. 1–249 (cit. on pp. [87](#), [94](#), [95](#)).
- Tabak, Esteban G. and Eric Vanden-Eijnden (2010). “Density Estimation by Dual Ascent of the Log-likelihood”. In: *Communications in Mathematical Sciences* (cit. on p. [88](#)).
- Papamakarios, George, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan (2019). “Normalizing Flows for Probabilistic Modeling and Inference”. In: *ArXiv Preprint*. arXiv: [1912.02762](#) (cit. on pp. [88](#), [94](#)).
- Gregory, J. A. and R. Delbourgo (1982). “Piecewise Rational Quadratic Interpolation to Monotonic Data”. In: *IMA Journal of Numerical Analysis* 2, pp. 123–130 (cit. on pp. [90](#), [94](#), [96](#), [105](#), [106](#)).

- Onken, Arno and Stefano Panzeri (2016). “Mixed Vine Copulas as Joint Models of Spike Counts and Local Field Potentials”. In: *Advances in Neural Information Processing Systems* (cit. on pp. 90, 98).
- Ziegler, Zachary M. and Alexander M. Rush (2019). “Latent Normalizing Flows for Discrete Sequences”. In: *36th International Conference on Machine Learning, ICML*. arXiv: 1901.10548 (cit. on p. 90).
- Tran, Dustin, Keyon Vafa, Kumar Krishna Agrawal, Laurent Dinh, and Ben Poole (2019). “Discrete flows: Invertible Generative Models of Discrete data”. In: *Deep Generative Models for Highly Structured Data, DGS@ICLR 2019 Workshop*. arXiv: 1905.10347 (cit. on p. 90).
- Salimans, Tim, Andrej Karpathy, Xi Chen, and Diederik P. Kingma (2019). “PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications”. In: *5th International Conference on Learning Representations, ICLR*. arXiv: 1701.05517 (cit. on p. 92).
- Dillon, Joshua V., Ian Langmore, Dustin Tran, Eugene Brevdo, Srinivas Vasudevan, Dave Moore, Brian Patton, Alex Alemi, Matt Hoffman, and Rif A. Saurous (2017). “TensorFlow Distributions”. In: *ArXiv Preprint*. arXiv: 1711.10604 (cit. on p. 92).
- Hansen, Bruce E. (1994). “Autoregressive Conditional Density Estimation”. In: *International Economic Review* 3, p. 705 (cit. on p. 92).
- Germain, Mathieu, Karol Gregor, Iain Murray, and Hugo Larochelle (2015). “MADE: Masked Autoencoder for Distribution Estimation”. In: *32nd International Conference on Machine Learning, ICML*. arXiv: 1502.03509 (cit. on pp. 93, 94).
- Rüschendorf, Ludger (2009). “On the Distributional Transform, Sklar’s Theorem, and the Empirical Copula Process”. In: *Journal of Statistical Planning and Inference* (cit. on p. 95).
- Hyvärinen, Aapo and Petteri Pajunen (1999). “Nonlinear Independent Component Analysis: Existence and Uniqueness Results”. In: *Neural Networks* 3, pp. 429–439 (cit. on pp. 96, 97).
- Theis, Lucas, Aäron Van Den Oord, and Matthias Bethge (2016). “A Note on the Evaluation of Generative Models”. In: *4th International Conference on Learning Representations, ICLR*. arXiv: 1511.01844 (cit. on p. 100).
- Zhang, Jun, Graham Cormode, Cecilia M. Procopiuc, Divesh Srivastava, and Xiaokui Xiao (2017). “Priv bayes: Private Data Release via Bayesian Networks”. In: *ACM Transactions on Database Systems* (cit. on p. 101).
- Srivastava, Akash, Lazar Valkov, Chris Russell, Michael U. Gutmann, and Charles Sutton (2017). “VEEGAN: Reducing Mode Collapse in GANs using Implicit Variational Learning”. In: *Advances in Neural Information Processing Systems*. arXiv: 1705.07761 (cit. on p. 101).

- Li, Haoran, Li Xiong, and Xiaoqian Jiang (2014). “Differentially Private Synthesization of Multi-Dimensional Data using Copula Functions”. In: *Advances in Database Technology - EDBT 2014: 17th International Conference on Extending Database Technology, Proceedings* (cit. on p. 101).