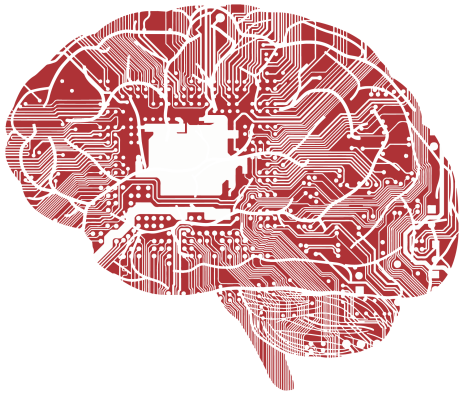
THE STATE SPACE FORMULATION OF ACTIVE INFERENCE

TOWARDS BRAIN-INSPIRED ROBOT CONTROL



SHERIN GRIMBERGEN

TU Delft Delft University of Technology

Delft Center for Systems and Control

Cognitive Robotics

COLOPHON
Online version typeset with LaTeX *classicthesis* template, developed by André Miede. Available at:

https://bitbucket.org/amiede/classicthesis/

The State Space Formulation of Active Inference

Towards Brain-Inspired Robot Control

Master of Science Thesis

For the (double) degree of Master of Science in

Systems and Control
&
Mechanical Engineering

at Delft University of Technology

Sherin Grimbergen

October 2017 - January 2019

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Delft University of Technology

Departments
Delft Center for Systems and Control
and
Biomechanical Engineering

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) to accept a thesis entitled

The State Space Formulation of Active Inference
by
Sherin Grimbergen

in partial fulfillment of the requirements for the degrees of

Master of Science Systems and Control
Master of Science Mechanical Engineering

Dated: January 29, 2019

Supervisor(s):

Prof. M. Wisse

Dr. P. Mohajerin Esfahani

Reader(s):

Dr. S. Baldi

Dr. F.A. Oliehoek

"The improvement of understanding is for two ends: first, our own increase of knowledge; secondly, to enable us to deliver that knowledge to others."

— *John Locke* —

I dedicate this work to the world,
secretly envisioning it as the start of a revolution.

ABSTRACT

This thesis provides an exposition of the theory of Active Inference in a control theoretic context. Active Inference is a remarkably powerful neuroscientific theory that unifies many characteristics of the biological brain. As such, Active Inference provides a valid inspiration in search of improvements in bio-inspired robot control algorithms. The literature on Active Inference however is narrow and complex. The goal of this thesis is to open the door research of Active Inference in robotics, by applying the theory to linear state space systems and exposing the relations and differences with established engineering paradigms.

We provide a detailed account of several critical details, mainly the concept of generalized motions, that are commonly not understood from the scientific literature. A start in the performance analysis of the algorithm is made, by studying the effect of changes in several tuning parameters. Additionally, with Active Inference reformulated as a state space control system, it is shown that standard behavior such as stabilization and tracking can be achieved.

ACKNOWLEDGEMENTS

First of all, I would like to praise my supervisors for pushing me to get the most out of this work. I had never expected to understand this matter in so much detail when I started. Besides, through this project I have attained skills of great personal value.

I'm grateful to my family for being proud of the work that I do, this gives me the motivation and courage that I need in life to pursue my dreams.

Thanks Anoek, Gokul, Stephan and all my climbing friends, who kept reminding me that this work needed to be finished at some point and who provided me an outlet for my daily celebrations as well as frustrations.

And thank you Rianne, for inspiring me with your discipline and perseverance. You enormously helped me get through the home stretch of this endeavour.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

LIST OF SYMBOLS

The list below describes symbols that are frequently used within the body of this document. Subscripts will be used to indicate specific cases of each symbol. As scientific literature on the Free Energy Principle and Active Inference is inconsistent, this list is also intended to set a standard for future work:

Agent Components:

κ, ρ	Learning rates
$\mathcal{D}$	Motion shift matrix
$\mathcal{F}$	Free energy
μ	Mean of Gaussian; expectation of $\tilde{x}$
Π	Precision matrix, i. e., Σ^{-1}
Σ	Covariance matrix
σ^2	Noise variance
$\tilde{K}$	Pole-placement term in ξ
ε	Prediction error
ξ	Prior variable in generative model
ζ	Set of sufficient statistics of recognition density $q(\vartheta)$
G	Forward dynamic model
m	Model that an agent employs (of its world)
p	Embedding order of generative model
$p(\tilde{x}, \tilde{y})$	Generative model
$q(\vartheta; \zeta)$	Recognition density with sufficient statistics ζ
$V(\gamma)$	Temporal variance matrix

Generative Process:

γ	Roughness parameter of noise
λ	The set of hyperparameters, $\{\gamma, \Sigma\}$
ω	(non-Markovian) noise on states x
$\rho(t)$	Noise correlation function

θ	Set of parameters of generative process
$\tilde{x}$	Generalized state
ϑ	Union of $\tilde{x}, \theta$ and λ; the causes
A	State matrix
B	Input matrix
C	Output matrix
$f(\cdot)$	Mapping from states x to their derivatives $\dot{x}$
$g(\cdot)$	Mapping from true states x to sensory data s
$h(\cdot)$	Mapping from states v to their derivatives $\dot{v}$
t	Time
u	Action of agent; control signal
v	Autonomously evolving states; disturbances
x	State of world; hidden state
y	Observation received by agent
z	(non-Markovian) noise on observation y

ACRONYMS

DEM	*Dynamic Expectation Maximization*
EM	*Expectation Maximization*
FEM	*Free Energy Minimization*
FEP	*Free Energy Principle*
GF	*Generalized Filtering*
KL	*Kullback-Leilber*
LMI	*Linear Matrix Inequality*
LTI	*Linear-Time-Invariant*
LQR	*Linear Quadratic Regulator*
MFA	*Mean-Field Approximation*
LQG	*Linear Quadratic Gaussian control*
POMDP	*Partially Observable Markov Decision Process*
SDP	*Semi-Definite Programming*
SPM	*Statistical Parametric Mapping*
VBI	*Variational (Bayesian) Inference*
VF	*Variational Filtering*

1 INTRODUCTION

1.1 MOTIVATION

Research on intelligent robotic is becoming increasingly important in recent decades, and it seems likely for this trend to continue until robotic intelligence exceeds that of humans.

However, state of the art robot intelligence is still nowhere near human-level [59]. Human intelligence still prevails in terms of complex inference, data-efficient learning, motor capabilities (in terms of robustness and generality) and many more aspects. In search of improvements, it is useful to study how intelligence in biological systems is created. This biologically inspired approach has yielded great improvements before, such as artificial neural networks [51], reinforcement learning [72], and most recently deep learning [68].

A very powerful theory about biological behavior is emerging in neuroscience recently: Active Inference. Active Inference is a result of the *Free Energy Principle* (FEP). The FEP provides a very principled account of biological life, or even bolder, the behavior of any self-organizing system. The theory, which is relatively new – the first papers appeared around 2003 [31–33]) – has proven to be the most unifying theory about the brain currently known. Its explanatory power exceeds any previous account of the brain's (of the neocortex' to be precise) information processing mechanism.

Essentially, the main premise of Active Inference is that the main mechanism behind biological behavior (action, perception and learning) is prediction error minimization. It could be possible to mimic biological intelligence in robots through this mechanism. For this reason it is very interesting to study applications of Active Inference in robotics. This thesis provides a starting point for that endeavour.

1.2 PROBLEM STATEMENT

To date, real world robotic applications of Active Inference do not exist ([63] is a first attempt in a simulated environment). This could be due to the fact that the scientific literature on the free energy principle and related concepts is relatively narrow and very complex. Nonetheless, the framework is mathematically well defined, which greatly facilitates the transfer to engineering context.

This thesis is meant to increase understanding of Active Inference from an engineering perspective and through this discover its interesting aspects. This is achieved by providing a detailed and comprehen-

sible exposition of the theory, with application on linear state space systems, since these are the simplest dynamic systems (i. e. "robots") that can be considered. An exposition with such detailed and comprehensible derivations as presented in this thesis has not been published before in the scientific literature. As such, this thesis and resulting paper [50] provide a starting point for more engineers to study Active Inference. It will open the door for future and more elaborate research on robotic applications. The research goal of this thesis is as follows:

Provide a detailed exposition of Active Inference in engineering context, to open the door for research of Active Inference in robotics.

To achieve this goal, three research questions have been composed, which are answered in the three main parts of this thesis:

1.2.1 *To which existing paradigms is Active Inference related?*

This question is answered to discover what the interesting aspects of Active Inference in an engineering context are. Throughout the derivations in Part I of this thesis, several differences between Active Inference and conventional control and filtering algorithms will come to light. These differences provide inspiration to develop novel algorithms that possibly improve current methods.

In Part I we also mention the paradigms and models to which Active Inference is related. Understanding these relations at a mathematical level first requires a deep understanding of Active Inference itself, which is what this thesis provides. Future studies could elaborate more on the exact technicalities of the relation that Active Inference bears to other paradigms.

1.2.2 *What is the state space formulation of Active Inference?*

To understand Active Inference in detail, this thesis will formulate the theory in the context of *Linear-Time-Invariant* (LTI) state space models, in Part II of this thesis. Namely, these are the simplest systems that Active Inference can be applied to. Moreover, a lot of control theoretic tools for in-depth analysis are available for linear state space systems: Bode plots, eigenvalue analysis, Lyapunov theory, robustness analysis etc. This thesis will allow future studies to make use of these tools.

The application to LTI state space systems will (a) increase understanding of several esoteric concepts inherent in the Free Energy Principle and (b) expose differences with existing control methods, further answering the first research question.

1.2.3 *How does Active Inference perform?*

In Part III of this thesis, we make a start in analyzing the performance of Active Inference. This thesis studies the influence of tuning parameters and provides a primer on the relation of Active Inference with *Linear Quadratic Gaussian control* (LQG) (thus partially answering the first research question). We also provide an approach to design a stabilizing controller using a *Linear Matrix Inequality* (LMI) and verify that Active Inference can perform tracking tasks.

Naturally, this performance question is a very broad one. This thesis only makes a start and provides recommendations for further analyses. With the state space Active Inference algorithm it is possible to perform experiments that analyze performance in any way possible as with standard state space controllers: Robustness, sensitivity, stability (or stabilization), optimality etc. can all be analyzed.

1.3 REPORT OVERVIEW

This thesis is comprised of five parts. For readers that are unfamiliar with the free energy principle and Active Inference, Part I provides a contextual and theoretical background. This part also mentions the existing paradigms to which Active Inference is related, to answer the first research question.

In Part II, the state space formulation of Active Inference is derived and explained, answering the second research question. Also, a detailed example is provided to clarify any remaining ambiguities.

After the theoretical second part, a start with the performance analysis of the resulting algorithm will be presented in Part III, to partially answer the third research question. This part also provides several hints on interesting but unsolved questions and problems. Finally, Part IV summarizes and concludes the thesis. In addition, several recommendations for future research directions are provided.

The appendices in Part V contain a few detailed mathematical derivations and examples of MATLAB code corresponding to simulations presented in this thesis.

1.4 TERMINOLOGY AND NOTATIONS

In this thesis a control system is designed through Active Inference. This control system consists, as conventional, of a filter and a controller. We will refer to these to components together as "*the agent*". Anything that is not part of the agent (i. e., filter or controller) is part of the environment. The ability of an agent to autonomously act on its environment is referred to as *agency*.

Furthermore, several notations specific to this thesis are employed to keep equations cleaner and more readable:

$\partial_x f(x, \ldots)$	Partial derivative of $f(x, \ldots)$ with respect to x
$*^{(n)}$ vs $*^n$	n-th derivative vs. exponent (rarely used).
$\tilde{*}$	Indicates generalized form (see Chapter 3).
$q(x; \zeta)$	Semicolon indicates $q(x)$ is parameterized by ζ.
$\mathcal{N}(\mu, \Sigma)$	Gaussian with mean μ and covariance matrix Σ.
$E[\cdot]$	Expectation operator.
I_n	Identity matrix of size $\mathbb{R}^{n \times n}$
$\otimes$	Kronecker tensor product.
$\mathbb{R}^n$	The space of all real n-dimensional vectors.
$\mathbb{R}^+$	The set of all positive real numbers.
$\mathbb{Z}^+$	The set of all positive integers.

Part I

BACKGROUND

This part answers the first research question: *To which existing paradigms is Active Inference related?* To this end, we will first briefly review the theoretical aspects of the free energy principle that are crucial for understanding of derivations later on. Chapter 2 provides the high level context behind this work. Next, Chapter 3 is dedicated to a discussion of generalized motions, since that is a crucial but convoluted concept. Finally, this part culminates in the introduction of Active Inference in Chapter 4.

2

THE FREE ENERGY PRINCIPLE

The holy grail of neuroscience is a single theory that explains all about the brain. Although it is questionable whether such a theory exists, the Free Energy Principle *(FEP) is as of today closest to achieving a unified account of the brain, with explanations covering fundamental accounts of life [40] – as explained below – to extremely detailed explanations of neural activity [4]. This chapter explains the FEP – the overarching theory of Active Inference – as developed by Karl Friston et al.*

Section 2.1 will provide the fundamental arguments that define the FEP. Subsequently Section 2.2 will provide a mathematical formulation of perception by defining the free energy. The three proposed filtering algorithms to minimize the free energy are briefly discussed in Section 2.3.

> **Remark:** The FEP is defined using a generalized form for dynamic variables (i.e. $\tilde{x}, \tilde{y}$). This is an esoteric concept to which Chapter 3 is dedicated. The current chapter uses standard notation for the sake of clarity, but it should be noted that this is a simplification.

2.1 WHAT IS LIFE?

2.1.1 *Resisting the Tendency to Disorder*

The FEP applies to self-organizing systems (i. e., agents) defined by a state x subject to dynamics: $\dot{x} = f(x, \theta) + \omega$, where $f(x, \theta)$ is some function parameterized by θ. ω is a random fluctuation parameterized by λ. The union $\vartheta = \{x, \theta, \lambda\}$ is *the causes*, as it contains all variables causing a system's existence. Furthermore, an agent's existence is defined by the fact that some set of states x is within a certain range. This implies that a small set of (internal) states is more likely than most others [41]. Hence, a probability density $p(x)$ on an agent's states can be defined. For example, an animal's body temperature fluctuates around some mean value in a narrow range.

The state of an agent is defined as the collection of all dynamic variables of interest that are external as viewed from "the brain". Thus, in a biological sense, the state includes measures of both the internal milieu (e.g. body temperature, muscle tension) and measures of the external milieu. This partitioning seems to distinguish biological life from other self-organizing systems [36].

The states of passive systems will eventually undergo some phase transition that implies the end of their existence, due to external influences [43]. This argument follows from the second law of thermo-

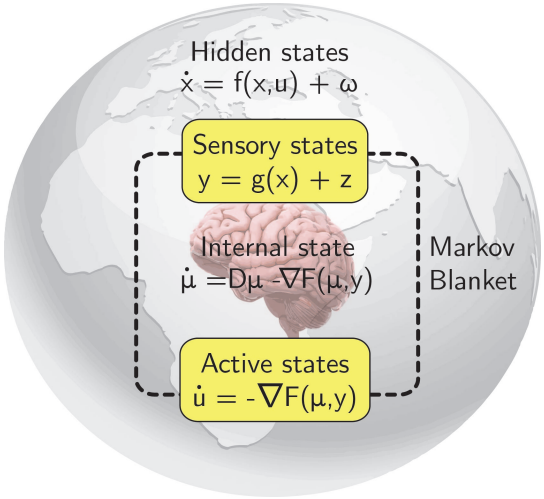


Figure 1: A Markov blanket consists of four sets of states, denoted here by x, y, μ and u. The internal states μ cannot directly interact with the hidden states x. Interaction between the world and brain is mediated through sensory states y (to receive data) and active states u (to act on the world). The equations are covered in Chapter 4.

dynamics, stating that entropy can naturally only increase over time. Self-organizing systems can counteract this natural tendency to disorder. This was first observed by Schrödinger [75].

Summarizing, the main premise of the FEP is that biological (living) systems maintain their existence by counteracting entropy increase of their states. In fact, the principle applies to any ergodic system with a Markov blanket [40] (see Figure 1 for an example of the Markov blanket). Note that it is required that the system has agency. *Agency* is the ability of an agent to autonomously act on its environment.

2.1.2 *Minimizing Sensory Surprisal*

As mentioned before, an agent has a small set of likely states. Thus $p(x)$ is a narrow distribution, which means that it has a low (differential) *entropy*, which is defined as [70]:

$$H(X) = -\int_{-\infty}^{\infty} p(x) \ln p(x) dx \tag{1}$$

The capital X is used because $H(X)$ defines the entropy of $p(x)$ i.e. the entropy over all $x \in X$. We could also have written $H(p(x))$, but the notation above is used in literature. Technically, $p(x)$ is a conditional density, with respect to the model m that an agent employs; $p(x|m)$. This is omitted to keep notation cleaner, however one should realize that $p(x)$ is not a physically defined quantity, but rather a belief following from an agent's model of the world.

The fundamental notion of the FEP is that biological systems should minimize $H(X)$. Namely, due to disturbances entropy naturally in-

creases over time, as previously discussed. An agent must counteract this dispersion at all times to survive. This relates to the first of two crucial assumptions underlying the FEP:

Assumption 1. Biological systems are (locally) ergodic: As $p(x)$ is fixed[1], it is natural to assume that the probability of a state x equals the amount of time the agent will spend in that state (as time of existence goes to infinity), also known as *ergodicity*. Instead of finding $H(X)$ by averaging over all $x \in X$ as in Equation 1, we can also find $H(X)$ by integrating over one lifetime trajectory $x(t)$:

$$H(X) = \lim_{T\to\infty} \frac{1}{T} \int_0^T -\ln p(x(t))dt \tag{2}$$

where $x(t)$ is the lifetime trajectory.

The quantity $-\ln p(x(t))$ is named *surprisal* or self-information in information theory [70].

From Equation 2 we can conclude that minimizing $H(x)$ can be achieved by $p(x)$ minimizing the surprisal $-\ln p(x(t))$ at all times. This is the important corollary of the ergodic assumption: Biological agents avoid high surprisal (i.e. surprising states) at all times. Note that since $p(x(t)) \in [0, 1]$ we have that $-\ln p(x(t)) \in [0, \infty]$; surprisal ranges from "not surprised at all" to "infinitely surprised".

The problem is that the state x is never observed directly. For example, visual information (light) is transformed to electrical signals in nerves. This introduces noise and also leaves some information unobserved (e.g. human eyes only see a very small portion of the EM-spectrum). Hence, an agent cannot minimize the surprisal directly, as defined in the second crucial assumption of the FEP:

Assumption 2. All biological systems posses a Markov blanket: The FEP assumes that a *Markov blanket* separates an agent from the environment. A subset of x is measured through sensory channels:

$$y = g(x, \theta) + z$$

where y is the sensory data, i. e., observation the agent receives, $g(x, \theta)$ is some function (also parameterized by θ) and z is noise (also parameterized by λ). x is referred to as the *hidden state*, as it is hidden from direct interaction with the agent's internal state. Figure 1 clarifies the structure of the Markov blanket. Due to the noise z, we have that

$$H(Y) \geqslant H(X)$$

1 It would change if the agents model m changes.

by definition [70]. Thus, by minimizing $H(Y)$ we approximately minimize $H(X)$ since $H(Y)$, the sensory entropy, is an *upper bound* on $H(X)$. Finally, from Equation 2 it can be concluded that an agent should minimize $-\ln p(y(t))$ (the *sensory surprisal*) at all times.

At this point we conclude that an agent can minimize $-\ln p(y(t))$ since it provides an upper bound on $H(X)$, which was the original measure to be minimized. Nevertheless, there is still a problem with that conclusion. Namely, to evaluate the sensory surprisal, the agent requires a model $p(y)$. Constructing this model would entail integrating $p(y, \vartheta)$ over ϑ, which is generally intractable [10, 25]. This is where the free energy provides a solution, as explained next.

2.2 THE FREE ENERGY

To recapitulate the previous section: An agent can minimize its sensory surprisal in order to approximately minimize the entropy of its states, but an agent cannot evaluate the sensory surprisal due to computational problems.

A solution to this problem is to create a (variational) upper bound, a concept derived from statistical physics [21] and machine learning [60]. The free energy is such an upper bound on the sensory surprisal and it can be evaluated by an ergodic agent with a Markov blanket. This section will explain how the free energy is defined. Time dependency will be omitted in this section to keep notation clean.

A straightforward derivation of the free energy as upper bound on the sensory surprisal does not exist. It is more an "accidental" consequence of an explanation for the concept of perception, i. e., the idea that an agent wants to understand the state of the world through observations. Mathematically, an agent tries to find $p(\vartheta|y)$, the *posterior* (which ϑ is most likely given observation y). Also, we assume that an agent models the world using a *generative model* $p(\vartheta, y)$ which can be factorized into a *prior* $p(\vartheta)$ (belief of the causes) and a *likelihood* $p(y|\vartheta)$ (belief of world structure, how ϑ causes y and vice versa):

$$p(\vartheta, y) = p(y|\vartheta)p(\vartheta) \tag{3}$$

More about generative models is explained in Section 4.1.2. We could invert Equation 3 (using $p(\vartheta, y) = p(\vartheta|y)p(y)$) to find the posterior:

$$p(\vartheta|y) = \frac{p(y|\vartheta)p(\vartheta)}{p(y)} = \frac{p(y|\vartheta)p(\vartheta)}{\int p(\vartheta, y)d\vartheta}$$

However, the integral is generally intractable (again, this is equivalent to modeling $p(y)$) and hence this approach fails in many cases. The solution is to invoke an auxiliary density $q(\vartheta; \zeta)$ to approximate

$p(\vartheta|y)$, as proposed by Hinton et al. [15]. ζ is the set of *sufficient statistics* of $q(\vartheta)$. For a Gaussian these would be the mean and covariance matrix: $\zeta = \{\mu, \Sigma\}$.

$q(\vartheta;\zeta)$ is often called the *ensemble density* or *recognition density*. The former name is derived from statistical physics, indicating that we consider a distribution over all states a system could be in (the ensemble). The latter name is used because $q(\vartheta;\zeta)$ "recognizes" which ϑ most likely caused observation y – remember that $q(\vartheta;\zeta) \approx p(\vartheta|y)$. Thus, perception is to determine ζ such that $q(\vartheta;\zeta)$ and $p(\vartheta|y)$ are most similar. The similarity can be measured through the (reverse) *Kullback-Leilber* (KL) divergence:

$$D_{KL}(q(\vartheta;\zeta) \parallel p(\vartheta|y)) = \int q(\vartheta;\zeta) \ln \frac{q(\vartheta;\zeta)}{p(\vartheta|y)} d\vartheta \tag{4}$$

which is a measure for the similarity between two distributions [56].

The process of approximating a posterior over hidden states with another distribution, is known as *Variational (Bayesian) Inference* (VBI), an approach that originates from machine learning [6, 52]. In a neuroscientific sense, we derived a mathematical formulation for perception. The problem is that a brain cannot evaluate Equation 4 since $p(\vartheta|y)$ is unknown. We can however use the generative model $p(\vartheta, y)$ to define:

$$\mathcal{F}(\zeta, y) = \int q(\vartheta;\zeta) \ln \left(\frac{q(\vartheta;\zeta)}{p(\vartheta, y)} \right) d\vartheta \tag{5}$$

which can be evaluated for any given ζ and y, because both $q(\vartheta;\zeta)$ and $p(\vartheta, y)$ are models in the brain. The functional above is called the *free energy*. Note that $\mathcal{F}(\zeta, y)$ is not a KL-divergence, because it a function of not only ϑ but also y. Namely, after the integration over ϑ, $\mathcal{F}(\zeta, y)$ is still a function of y. Comparing to Equation 4, $p(\vartheta|y)$ is "one dimensional" and $p(\vartheta, y)$ is "two dimensional" (it is a joint density).

Let us see how Equation 5 helps to solve Equation 4. As shown in Section A.1, the free energy can be rewritten as

$$\mathcal{F}(\zeta, y) = -\ln p(y) + D_{KL}\left(q(\vartheta;\zeta) \parallel p(\vartheta|y)\right)$$

Observe that perception i.e. minimizing $\mathcal{F}(\zeta, y)$ with respect to ζ minimizes $D_{KL}\left(q(\vartheta;\zeta) \parallel p(\vartheta|y)\right)$, as desired. So, perception i.e. estimating the posterior $p(\vartheta|y)$ is *Free Energy Minimization* (FEM):

$$\min_{\zeta} \mathcal{F}(\zeta, y) \tag{6}$$

A nice "accidental" consequence is that this also corresponds to making the function $\mathcal{F}(\zeta, y)$ approximate $-\ln p(y)$. Namely, since KL divergence is non-negative, we have that

$$\mathcal{F}(\zeta, y) \geqslant -\ln p(y)$$

and when we have minimized $\mathcal{F}(\zeta, y)$ with respect to ζ

$$\mathcal{F}(\zeta^*, y) \approx -\ln p(y)$$

where ζ^* is the optimal ζ. So, perception provides an upper bound estimate on the sensory surprisal. However, it does not minimize the sensory surprisal. Minimizing $\mathcal{F}(\zeta^*, y)$ with respect to y will actually minimize the sensory surprisal (and as such approximately minimizes $H(X)$). Unfortunately, an agent cannot control y directly. The answer is Active Inference, as explained in Chapter 4.

2.2.1 *Interim Summary*

To summarize what has been achieved at this point. There is a free energy functional $\mathcal{F}(\zeta, y)$ – defined by two probability densities that are modeled in the brain – that upper bounds the sensory surprisal $-\ln p(y)$. Thus, minimizing the free energy approximately minimizes the sensory surprisal and so the sensory entropy $H(Y)$ at all times. And, because the sensory entropy upper bounds the state entropy $H(X)$, we also approximately minimize $H(X)$ at all times, as was the definition of life.

The free energy solved two main problems. Firstly, state x is hidden and hence we are obligated to work with observations y. Secondly, we changed an intractable integration into a tractable optimization.

2.3 FREE ENERGY MINIMIZATION

It has not yet been discussed how the optimization problem posed in Equation 6 can actually be solved. In fact, there are three methods proposed within the FEP to solve the optimization. All of these methods rely on the definition of generalized motions. This is a rather intricate concept which is elaborated in Chapter 3. The main point is that by estimating the motions of the hidden state x, we obtain a version of VBI that can be applied to dynamical systems. In this sense, FEM is an extension of standard VBI.

2.3.1 *Generalized Filtering*

The most general algorithm to minimize the free energy is *Generalized Filtering* (GF) [46]. This algorithm treats all variables in the hidden state x equally and estimates a posterior over the complete state space of the environment, employing only the Laplace approximation:

> **The Laplace approximation:** Given a unimodal sharply peaked distribution $p(x)$ with $x^* = \arg\max p(x)$. Then also $x^* = \arg\max \ln p(x)$. Now to approximate $p(x)$

let $q(x) = \ln p(x)$. A second order Taylor expansion is used to capture $p(x)$, because it is sharply peaked:

$$q(x) \approx q(x^*) + \partial_x q(x^*)(x - x^*) + \tfrac{1}{2}\partial_x^2 q(x^*)(x - x^*)^2$$

$\partial_x q(x^*) = 0$ because it is at the maximum $x*$ and thus

$$q(x) \approx q(x^*) + \tfrac{1}{2}\partial_x^2 q(x^*)(x - x^*)^2$$

Remember that $q(x) = \ln p(x)$ i.e. $p(x) = \exp(q(x))$. Thus:

$$\begin{aligned} p(x) &= \exp(q(x^*) + \tfrac{1}{2}\partial_x^2 q(x^*)(x - x^*)^2) \\ &= \exp(q(x^*)) \exp(\tfrac{1}{2}\partial_x^2 q(x^*)(x - x^*)^2) \\ &= \exp(q(x^*)) \exp\left(\frac{-(x - x^*)^2}{2[-\partial_x^2 q(x^*)^{-1}]}\right) \end{aligned}$$

which is a Gaussian density with mean x^* and variance $-\partial_x^2 q(x^*)^{-1}$. For further details see [11, 45, 60].

Observe that due to the Laplace approximation the variance is a function of the mean. This means that

$$q(\vartheta; \zeta) \to q(\vartheta; \mu) \sim \mathcal{N}(\mu, \Sigma(\mu))$$

The solution to minimize the free energy is then a generalized gradient descent:

$$\dot{\mu} = \mathcal{D}\mu - \partial_\mu \mathcal{F}(\mu, y)$$

The use of the extra term $\mathcal{D}\mu$ is clarified in Chapter 3.

2.3.2 *Variational Filtering*

Secondly, there is *Variational Filtering* (VF) [24]. In contrast to GF, VF does not use the Laplace approximation but uses the *Mean-Field Approximation* (MFA), which is the standard approximation used in VBI:

The mean-field approximation: It is assumed that several sets of variables in the causes ϑ are conditionally independent so that $q(\vartheta; \zeta)$ can be partitioned [22, 53]:

$$q(\vartheta; \zeta) = \prod_{i=1}^{N} q_i(\vartheta_i; \zeta_i) = q_1(\vartheta_1; \zeta_1) q_2(\vartheta_2; \zeta_2) \dots$$

In the FEP this partitioning is based on a temporal distinction between states in the world [43]. For example dynamic states x change fast independently of physical parameters θ that change very slowly or are constant,

like the hyperparameters λ. So in the context of VF, the partitioning consists of three distributions, assuming that the sets of variables in the causes $\vartheta = \{x, \theta, \lambda\}$ are independent:

$$q(\vartheta; \zeta) = q_x(x; \zeta_x) q_\theta(\theta; \zeta_\theta) q_\lambda(\lambda; \zeta_\lambda) \tag{7}$$

The MFA is not the only possibility through which VBI can be achieved. Other approximations such as the Bethe approximation are studied recently as well [69] and seem to outperform the MFA.

In VF, each of the partitions is tracked by a set of particles – i. e., by a free-form distribution – bearing a lot of resemblance with particle filtering [18].

2.3.3 *Dynamic Expectation Maximization*

Finally, the most simplified form of FEM is *Dynamic Expectation Maximization* (DEM) [28], which employs both the MFA and Laplace approximation. In other words, each of the partitions in Equation 7 is a Gaussian. The result is an algorithm much faster than GF or VF, be it a more specific approximation that might not perform well in complex non-Gaussian environments. The naming is chosen specifically because DEM is an extension of *Expectation Maximization* (EM) [17] that can be applied on dynamic systems. A comparison between VBI and EM can be found in [5]. The main difference is that EM only provides a most likely estimate of parameters whereas VBI estimates a complete distribution over the parameters (the same as the states).

Due to the Laplace approximation, only the mean of all partitions has to be estimated in DEM and as such it is a form of EM extended with an extra step to make it compatible with dynamic systems. This is the D-step and hence the name DEM. So, similar to GF, the sufficient statistics ζ_i of all partitions in the recognition density are only the mean μ_i in DEM. Hence for DEM we have that

$$\mathcal{F}(\zeta, y) = \mathcal{F}(\mu, y)$$

Additionally, the set of causes ϑ and thus μ is partitioned through the MFA in three parts:

$$\mu = \{\mu_x,\ \mu_\theta,\ \mu_\lambda\}$$

where μ_x refers to the dynamic states, μ_θ refers to system parameters and μ_λ refers to hyperparameters, i. e., noise characteristics [28].

2.3.4 *The Active States*

Up to this point we have only discussed minimization of the free energy with respect to ζ, i. e., through perceptual inference. Referring

back to Figure 1, one can see that there is a set of states, the active states, which have not been discussed nor used yet. In fact, extending DEM with the active states i.e. action is what yields Active Inference. Namely, acting on the environment is another way to possibly minimize free energy, as it will change incoming observations y. Chapter 4 will discuss this extension.

To summarize this section, Figure 2 provides an overview of the approximations, assumptions and extensions that bring us from standard VBI to Active Inference. The first step, generalized motions, is the topic of the upcoming chapter. The last step, to arrive at Active Inference, is the topic of Chapter 4.

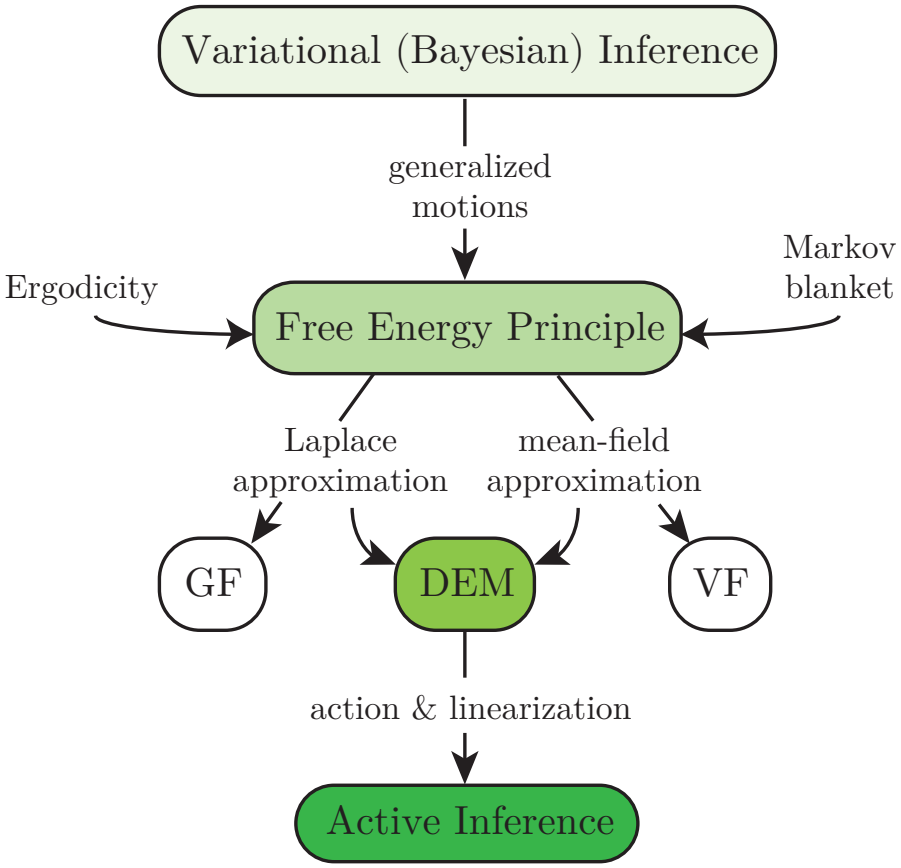


Figure 2: Schematic view of the steps from VBI to Active Inference. This is not technically exact but provides a comprehensible overview of the theory discussed so far. The addition of action and linearization is the topic of Chapter 4 and generalized motions are discussed in Chapter 3.

3

GENERALIZED MOTIONS

Before we can proceed to discuss Active Inference, an intricate though crucial subject has to be covered: generalized motions. As briefly mentioned in the previous chapter, the FEP differs from standard VBI methods in that it is designed for inference on dynamic systems. This feature is enabled by the use of generalized motions, as explained in this chapter.

First, Section 3.1 provides the argumentation as to why the use of generalized motions is appropriate. Following that, Section 3.2 explains the concept of generalized motions and Section 3.3 provides some more detailed explanation to provide a more intuitive insight in the concept.

3.1 DIFFERENTIABLE NOISE

As mentioned the previous chapter, the variational algorithms within the FEP are an extension to standard VBI methods. Namely, the FEP incorporates the use of generalized motions, making the FEP suitable for inference on dynamic systems. This section explains in which case one is allowed to define these generalized motions. For reference, a simple autonomous state space system will be used:

$$\begin{aligned} \dot{x} &= f(x) + \omega \\ y &= g(x) + z \end{aligned} \tag{8}$$

where ω and z are noises.

3.1.1 *Markovian Noise*

The most conventional noise used for ω and z in engineering context is uncorrelated noise, or noise generated from a Wiener process [20] (also called white noise). In mathematical terms: $\omega(t)$ is a real-valued random variable with

$$\rho_\omega(dt) = \frac{E[\omega(t+dt)\omega(t)]}{\sigma_\omega^2} = \begin{cases} 1 & \text{if} \quad dt = 0 \\ 0 & \text{otherwise} \end{cases}$$

where ρ_ω is the *autocorrelation* function and σ_ω^2 is the variance of the noise. In addition, often $\mu_\omega = E[\omega(t)] = 0$ (zero mean white noise).

The exact technicalities of autocorrelation functions are quite involved. The important argument here is that such noises possess the Markov property,i. e., are Markov processes [47]: Each sample of the noise is independent of all others, as $\rho_\omega(dt \neq 0) = 0$. Hence, these noises can be classified as "*Markovian noises*".

Due to this property, Markovian noises cannot be differentiated. Namely, the noise $\omega(t)$ is independent of the noise at each other time $\omega(t^*)$. For the sake of explanation, suppose we would be allowed to differentiate $\omega(t)$. The system state equation would become

$$\ddot{x} = \partial_x f(x)\dot{x} + \dot{\omega}$$

Now, $\dot{\omega}$ will have infinite variance and consequently so will $\ddot{x}$. In other words, the equation contains no information. This is why standard techniques like a Kalman Filter only use the state equation itself. However, if we were able to retrieve information from the higher order derivatives of x, this could improve the filtering performance. This is why the FEP makes a different assumption on the noises.

3.1.2 *Non-Markovian Noise*

In fact, the Markov property is a very specific assumption which is only true in theory. In real world (especially biological) systems, noises are generated by dynamic processes as well [34]. Thus, there will be a correlation between the noise at different times. Intuitively, this will create noises which have a certain amount of smoothness, depending on the width of the correlation function. We can also refer to these noises as *differentiable noises*, because they have an analytic correlation function due to which they can be differentiated.

Any differentiable (auto)correlation function $\rho(t)$ is allowed, but a normalized[1] zero mean Gaussian is usually proposed [28]:

$$\rho(t) = e^{-\frac{\gamma}{4}t^2}. \tag{9}$$

The variance has been substituted as $\sigma^2 = \frac{2}{\gamma}$. γ is the *roughness parameter*, which is inversely proportional to the smoothness of the noise. Appendix B contains a detailed explanation regarding the generation of smooth noises for simulation purposes.

Due to the smoothness in the noise, there is now information contained in the higher order derivatives of x. This is exploited in the FEP as explained in the next section.

3.2 GENERALIZED MOTIONS

If the noises ω, z are analytic, we are allowed to extend the description of the system Equation 8 to:

$$\begin{aligned}
\dot{x} &= f(x) + \omega & y &= g(x) + z \\
\ddot{x} &= \partial_x f(x)\dot{x} + \dot{\omega} & \dot{y} &= \partial_x g(x)\dot{x} + \dot{z} \\
\dddot{x} &= \partial_x f(x)\ddot{x} + \ddot{\omega} & \ddot{y} &= \partial_x g(x)\ddot{x} + \ddot{z} \\
&\vdots & &\vdots
\end{aligned}$$

1 The Gaussian has to be normalized so that $\rho(0) = 1$.

where nonlinear terms were omitted i.e. a linearization is made [34].

The crucial observation is that if the state x and the noises ω, z are known, all information is in the first equations. However, an agent in an uncertain dynamic system does not know the noises ánd neither x. Thus, the estimation of x can be improved by taking into account beliefs about higher order derivatives. To indicate the change to beliefs, the notation is changed to:

$$\begin{aligned} x' &= f(x) + \omega & y &= g(x) + z \\ x'' &= \partial_x f(x)x' + \omega' & y' &= \partial_x g(x)x' + z' \\ x''' &= \partial_x f(x)x'' + \omega'' & y'' &= \partial_x g(x)x'' + z'' \\ &\vdots & &\vdots \end{aligned} \tag{10}$$

At this point, there is not yet an obvious difference between the dots and primes. The difference appears when we move to (posterior) beliefs and write down a filtering algorithm. Another way to see it: A dot represents a temporal relation between x and $\dot{x}$. A prime indicates that x' is a variable calculated from x, representing the belief about $\dot{x}$, but it is not actually $\dot{x}$. To distinguish between the two notations, dots are referred to as "derivatives" and primes as "motions".

Equation 10 can be written more compactly by introducing a generalized form of the state, the *generalized state*:

$$\tilde{x} = \begin{pmatrix} x & x' & x'' & \dots \end{pmatrix}^\top$$

and similar for $\tilde{y}, \tilde{\omega}$ and $\tilde{z}$. This representation is up to infinite order and thus represents the complete current motion of a variable. In addition we define

$$\tilde{f}(\tilde{x}) = \begin{pmatrix} f(x) & \partial_x f(x)x' & \partial_x f(x)x'' & \dots \end{pmatrix}$$

and similar for $\tilde{g}(\tilde{x})$. Finally we introduce a motion-shift operator:

$$\mathcal{D} = \begin{pmatrix} 0 & 1 & & \\ & 0 & \ddots & \\ & & \ddots & 1 \\ & & & 0 \end{pmatrix} \otimes I_n$$

where n is the state dimension, such that $\mathcal{D}\tilde{x} = \tilde{x}'$. In other words, all motions are shifted up one element by multiplication with $\mathcal{D}$.

Now Equation 10 can be reformulated as

$$\mathcal{D}\tilde{x} = \tilde{f}(\tilde{x}) + \tilde{\omega} \qquad \tilde{y} = \tilde{g}(\tilde{x}) + \tilde{z} \tag{11}$$

Observe that the state equation is not a differential equation anymore; there are no dots! The dynamics are now captured by relating the motions at different orders. Equation 11 represents an instantaneous

probabilistic belief about the motions of the state x. This can be seen when rewriting the equations as

$$\mathcal{D}\tilde{x} - \tilde{f}(\tilde{x}) = \tilde{\omega} \qquad \tilde{y} - \tilde{g}(\tilde{x}) = \tilde{z}$$

Since $\tilde{\omega}$ and $\tilde{z}$ are processes defined by a distribution $p(\tilde{\omega})$ or $p(\tilde{z})$, so are the left hand sides of the equations i.e.:

$$\mathcal{D}\tilde{x} - \tilde{f}(\tilde{x}) \sim p(\tilde{\omega}) \qquad \tilde{y} - \tilde{g}(\tilde{x}) \sim p(\tilde{z})$$

The dynamics to update $\tilde{x}$ (or actually its expectation $\mu_{\tilde{x}}$) will be put back later when we define a filtering scheme through FEM.

Finally, assuming zero mean Gaussian noises[2], the noise densities are $p(\tilde{\omega}) \sim \mathcal{N}(0, \Sigma_{\tilde{\omega}})$ and $p(\tilde{z}) \sim \mathcal{N}(0, \Sigma_{\tilde{z}})$. The covariance matrix of the generalized noise $\tilde{\omega}$ is defined as:

$$\Sigma_{\tilde{\omega}} = V(\gamma) \otimes \Sigma_{\omega}$$

where Σ_{ω} is the covariance matrix for ω. Similarly for $\Sigma_{\tilde{z}}$. The matrix $V(\gamma)$ is a *temporal variance matrix* defining the correlations between noises at different orders of motion. It is defined as:

$$V(\gamma) = \begin{pmatrix} 1 & 0 & \ddot{\rho}(0) & \dots \\ 0 & -\ddot{\rho}(0) & 0 & \\ \ddot{\rho}(0) & 0 & \rho^{(4)}(0) & \\ \vdots & & & \ddots \end{pmatrix}.$$

For the Gaussian correlation function (Equation 9) we get

$$V(\gamma) = \begin{pmatrix} 1 & 0 & -\frac{1}{2}\gamma & \dots \\ 0 & \frac{1}{2}\gamma & 0 & \\ -\frac{1}{2}\gamma & 0 & \frac{3}{4}\gamma^2 & \\ \vdots & & & \ddots \end{pmatrix}$$

A more detailed explanation is provided in Section A.2 or [14].

Since the FEP works with precision (the inverse of variance) we will from now on use

$$\Pi_{\tilde{\omega}} = V^{-1}(\gamma) \otimes \Pi_{\omega} \tag{12}$$

where $\Pi_{\omega} = \Sigma_{\omega}^{-1}$ is the *precision matrix* for ω. $V^{-1}(\gamma)$ is often denoted by $S(\gamma)$ in literature and referred to as the *temporal precision*.

2 Confusion might arise here. This assumption is different from the assumption that the correlation function of the noise is a Gaussian! Namely, a realization of Gaussian noise has samples that are distributed as a Gaussian. This does not specify the correlation between samples. The correlation is specified in $\rho(t)$, which in this case also happens to be a Gaussian, but Gaussian noise is not Markovian by definition. It can be either Markovian or non-Markovian depending on $\rho(t)$.

3.3 THE USE OF GENERALIZED MOTIONS

3.3.1 *Stationarity*

In the previous sections the use of generalized motions has been justified through the assumption of non-Markovian noise. However, this assumption is not the reason *why* one would want would to use generalized motions, it is a necessary assumption. The reason why to be interested in using generalized motions is clarified in this section.

First and foremost, the use of generalized motions allows for estimation of the posterior $p(\tilde{x}|\tilde{y})$ that is stationary. Namely, the distribution is defined in a frame of reference that moves with the current motion of the state [28]. In contrast, assume we would only estimate $p(x|y)$. Since x is a (stochastic) dynamic variable, the distribution $p(x|y)$ will change over time. However, we can capture this change by also including the motions of x in the distribution.

3.3.2 *Dynamic Equilibria*

Secondly, the use of generalized motions allows for tracking a dynamic equilibrium. Remember the filtering equation of GF:

$$\dot{\mu} = \mathcal{D}\mu - \partial_\mu \mathcal{F}(\mu, y)$$

where μ is the expectation of $\tilde{x}$: $E[\tilde{x}] = \mu$. Note that this is where the dynamics for $\tilde{x}$ are put back: The filtering equation describes how to update the expectation of the generalized state. By definition – since $\mathcal{D}\mu = \mu'$ – the filtering equation can be written as

$$\dot{\mu} = \mu' - \partial_\mu \mathcal{F}(\mu, y)$$

which shows that when the free energy is *not* minimal, there is a difference between $\dot{\mu}$ and μ'. As soon as we move to beliefs it makes sense to distinguish motions and derivatives.

Now, using the generative model it is possible to shape the free energy such that it's minimum is at any desired μ^*. When the free energy is minimal the filtering equation becomes

$$\dot{\mu}^* = \mu^{*\prime}$$

Which means that the current expectation of the motion $\mu^{*\prime}$ is used to update thee motion. For example, the first element of $\dot{\mu}^*$ is the change in the belief of x, which will be equal to the first element of $\mu^{*\prime}$ which is the current belief of $\dot{x}$. In other words, the *motion* of $\tilde{x}$ will be constant; a dynamic equilibrium is attained, i. e., $\dot{p}(\tilde{x}|y) = 0$.

3.3.3 *State Estimation*

There is a second way in which the difference between derivatives and motions becomes apparent. Consider the following autonomous second order system of the form $\dot{x} = Ax$:

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ a & b \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

Observe that $\dot{x}_1 = x_2$. Now, when extending this system to generalized form. The first generalization x' will be $x' = \begin{pmatrix} x_1' & x_2' \end{pmatrix}$. Adding this to the state yields

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_1' \\ \dot{x}_2' \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ a & b & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & a & b \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_1' \\ x_2' \end{pmatrix} \tag{13}$$

However, since x_1' represents the velocity and so does x_2, there is a redundant state in this system. But, if we change the original system to a stochastic setting:

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ a & b \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} \omega_1 \\ \omega_2 \end{pmatrix}$$

The redundancy this appears because now $\dot{x}_1 \neq x_2$ and so $x_1' \neq x_2$, due to the noises. Figure 3 shows the difference.

This difference is the reason why adding generalized motions improves the estimate. Namely, x_1 is only updated using the belief (from the A-matrix) that $x_2 = \dot{x}_1$ in e. g. a Kalman filter. In the FEP, the actual current motion of x_1 is also accounted for when updating the estimate. There is information that a Kalman filter does not use when smooth noises act on the system.

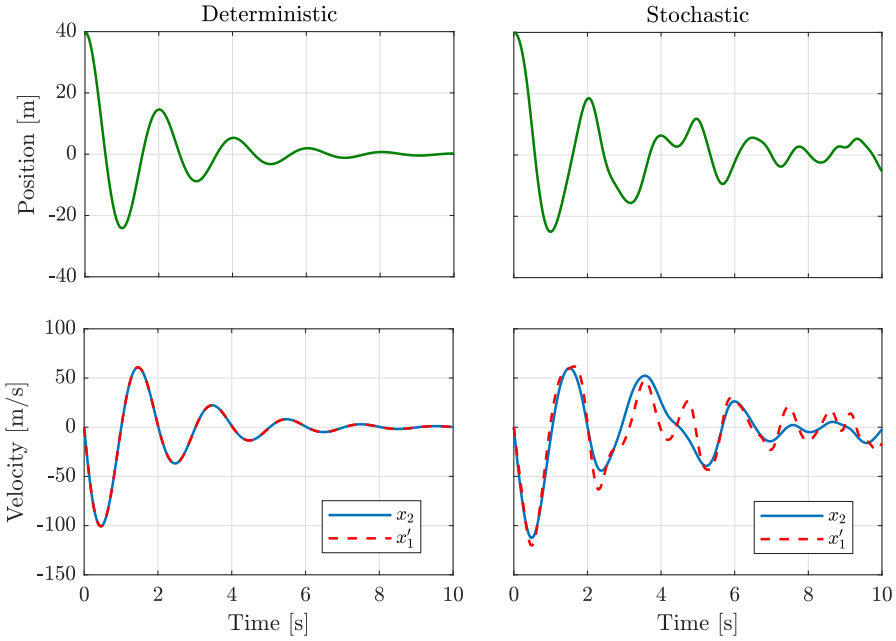


Figure 3: Simulation of Equation 13, without and with noise. Redundancy of generalized motions shows in the deterministic case, because here $x_1' = x_2$. In the stochastic case however (with independent noises on the position and velocity), $x_1' \neq x_2$ and thus the generalized motion x_1' is not redundant. x_1' represents the motion of x_1, $\dot{x}_1$, which need not to be the velocity x_2! Code of the simulation present in Appendix B.

4

ACTIVE INFERENCE

This chapter will present the extension of DEM with the active states of the Markov blanket: Active Inference. Whereas FEM was only performed through perception in Chapter 2, Active Inference adds another mechanism to actively minimize the free energy: action. The integration of perception and action is very unique to Active Inference.

Section 4.1 will explain the two main components beside the free energy that are required to set up Active Inference, namely the world in which Active Inference is performed and the model of this world that is used by the agent. Due to the MFA and Laplace approximation, a simplified formulation of the free energy can be derived, as presented in Section 4.2. Finally, Section 4.3 presents how FEM is achieved using the simplified form of the free energy. This section also connects Active Inference with existing paradigms.

4.1 TWO SIDES OF THE MARKOV BLANKET

4.1.1 *Outside the Blanket: Generative Process*

Active Inference is designed to explain behavior of biological systems. Referring back to Figure 1, this means that the hidden state x is the state of a dynamic and uncertain system: the *generative process*. The naming refers to the fact that this process is the one generating the observations received by the agent. The generative process represents everything dynamic that is not internal to the agent. More precisely, everything external to the agent's brain is in x. The dynamics of x are described by a state space system:

$$\begin{aligned} \dot{x} &= f(x, u, v, \theta) + \omega \\ y &= g(x, v, \theta) + z \\ \dot{v} &= h(v, \theta) + \eta \end{aligned} \tag{14}$$

where x is the hidden state, y is the observation and v are disturbances. Furthermore, u is the control state i.e. the active states of the Markov blanket, θ is the set of parameters defining the mappings $f(\cdot), g(\cdot)$ and $h(\cdot)$, and ω, z, η are non-Markovian noises.

In a biological sense, only neural activity in the brain is within the Markov blanket. Thus, states like joint angles, body temperature and external object motions are all in x or v. Some of these states can be controlled by the agent (e.g. joint angles and body temperature) and are thus in x. Uncontrollable/autonomous states are in v. Thesis will not yet consider disturbances so v is omitted from this point[1].

1 They have been included in Equation 14 for completeness

4.1.2 *Inside the Blanket: Generative Model*

As required by both the free energy principle and the good regulator theorem [13], the agent is required to model the world it inhabits i.e. the generative process. This model is referred to as the *generative model*, representing the agent's belief about the generative process.

There is actually quite some theory involved in the naming, which is outside the scope of this thesis. In short, generative models are able to explain how data is generated [62]. Generative models are becoming more prominent in machine learning in recent years. Many different generative models exist [48, 49], but to start understanding Active Inference it is best to start simple. Therefore, the generative model can be thought of as a state space model:

$$\begin{aligned} \dot{x} &= f_m(x, \theta) + \omega \\ y &= g_m(x, \theta) + z \end{aligned} \tag{15}$$

where the subscript m denotes that this is the model m of the agent. Referring back to Chapter 2, the state space model is what represents $p(x, y)$. In fact writing the model as in Equation 15 is only showing the formal homology between the process and model. The true generative model is a probabilistic mapping – it is $p(x, y)$ – due to the presence of the noises.

Observe that the model is agency free (no modeling of the control u). The reason for this is as follows: Equation 15 and Equation 14 need not to be the same. In fact, a difference is crucial. Namely, control tries to cancel this discrepancy [29] and is thus a signal acting on the generative process such that its behavior conforms to the generative model. This concept is a fundamental difference with respect to conventional approaches in control. Usually, system identification attempts to replicate the world with some model. Afterwards a controller employs that model to determine the control signal that will achieve some goal (defined by a reference signal or cost function). In Active Inference, the desired dynamics or goal is embedded in the generative model. Action merely enforces those desires on the world.

Due to the differentiable nature of the noises, generative model can be formulated in generalized form. The generalized form of Equation 15 is

$$\begin{aligned} \mathcal{D}\tilde{x} &= \tilde{f}_m(\tilde{x}, \theta) + \tilde{\omega} \\ \tilde{y} &= \tilde{g}_m(\tilde{x}, \theta) + \tilde{z} \end{aligned} \tag{16}$$

Remember that these equations are not differential equations, but probabilistic mappings, which is more obvious if we rewrite the equations as follows:

$$\begin{aligned} \mathcal{D}\tilde{x} - \tilde{f}_m(\tilde{x}, \theta) &= \tilde{\omega} \\ \tilde{y} - \tilde{g}_m(\tilde{x}, \theta) &= \tilde{z} \end{aligned}$$

It appears that the motions obey certain probabilistic behavior defined by the generalized noises. Active Inference assumes that the noises are zero mean Gaussian processes [29, 34] and thus

$$\begin{aligned} \mathcal{D}\tilde{x} - \tilde{f}_m(\tilde{x}, \theta) &\sim \mathcal{N}(0, \Sigma_{\tilde{\omega}}) \\ \tilde{y} - \tilde{g}_m(\tilde{x}, \theta) &\sim \mathcal{N}(0, \Sigma_{\tilde{z}}) \end{aligned} \tag{17}$$

This result will be used in the next section, as these probabilistic mappings allow us to rewrite the free energy.

4.2 THE LAPLACE ENCODED FREE ENERGY

Remember that the free energy is defined as

$$\mathcal{F}(\zeta, \tilde{y}) = \int q(\vartheta; \zeta) \ln \left(\frac{q(\vartheta; \zeta)}{p(\vartheta, \tilde{y})} \right) dx$$

where ϑ represents all variables outside the Markov blanket: $\vartheta = \{\tilde{x}, \theta, \lambda\}$. Employing the Laplace approximation and the MFA we get

$$q(\vartheta; \zeta) \to q(\vartheta; \mu) \to q(\tilde{x}, \theta, \lambda; \mu) = q(\tilde{x}; \mu_{\tilde{x}}) q(\theta; \mu_\theta) q(\lambda; \mu_\lambda)$$

where all partitions and thus also the complete density are Gaussian in which the variance is a function of the mean. In this thesis parameters and hyperparameters are considered as given and hence

$$q(\vartheta; \mu) = q(\tilde{x}; \mu_{\tilde{x}} | \theta, \lambda) \sim \mathcal{N}(\mu_{\tilde{x}}, \Sigma(\mu_{\tilde{x}})) \tag{18}$$

This choice is made to keep derivations further on comprehensible. Substituting the above form of $q(\vartheta; \mu)$ in the free energy and making some further assumptions allows to simplify the equation (lengthy derivation, elaborated in [10]):

$$\begin{aligned} \mathcal{F}(\mu_{\tilde{x}}, \tilde{y}) &= -\ln p(\mu_{\tilde{x}}, \tilde{y}) - \tfrac{1}{2} \ln (2\pi\Sigma(\mu_{\tilde{x}})) \\ &= -\ln (p(\tilde{y}|\mu_{\tilde{x}}) p(\mu_{\tilde{x}})) - \tfrac{1}{2} \ln (2\pi\Sigma(\mu_{\tilde{x}})) \\ &= -\ln p(\tilde{y}|\mu_{\tilde{x}}) - \ln p(\mu_{\tilde{x}}) - \tfrac{1}{2} \ln (2\pi\Sigma(\mu_{\tilde{x}})) \end{aligned} \tag{19}$$

Note that, as the mean $\mu_{\tilde{x}}$ is yet to be optimized, it makes sense to write $p(\mu_{\tilde{x}})$. The covariance term does not change the optimum and can be omitted [10]. Only the mean of the recognition density, the generative model and the observation are now required to evaluate the free energy, which is important to realize. The generative model Equation 17 defines $p(\mu_{\tilde{x}}, \tilde{y})$ as follows:

$$\begin{aligned} p(\mu_{\tilde{x}}) &\sim \mathcal{N}(\mathcal{D}\mu_{\tilde{x}} - \tilde{f}_m(\mu_{\tilde{x}}, \theta), \Sigma_{\tilde{\omega}}) \\ p(\tilde{y}|\mu_{\tilde{x}}) &\sim \mathcal{N}(\tilde{y} - \tilde{g}_m(\mu_{\tilde{x}}, \theta), \Sigma_{\tilde{z}}) \end{aligned}$$

Substitution in Equation 19 (omitting the variance term) yields the Laplace encoded free energy [10]:

$$\mathcal{F}(\mu_{\tilde{x}}, \tilde{y}) = \frac{\varepsilon_{\tilde{x}}^\top \varepsilon_{\tilde{x}}}{2\Sigma_{\tilde{\omega}}} + \frac{\varepsilon_{\tilde{y}}^\top \varepsilon_{\tilde{y}}}{2\Sigma_{\tilde{z}}} - \tfrac{1}{2} \ln (\Sigma_{\tilde{\omega}}^{-1} \Sigma_{\tilde{z}}^{-1})$$

where $\varepsilon_{\tilde{x}} = \mathcal{D}\mu_{\tilde{x}} - \tilde{f}_m(\mu_{\tilde{x}}, \theta)$ and $\varepsilon_{\tilde{y}} = \tilde{y} - \tilde{g}_m(\mu_{\tilde{x}}, \theta)$.

In a more compact form:

$$\mathcal{F}(\mu_{\tilde{x}}, \tilde{y}) = \frac{1}{2}\varepsilon^\top \Pi \varepsilon - \frac{1}{2}\ln|\Pi| \tag{20}$$

where

$$\varepsilon = \begin{pmatrix} \varepsilon_{\tilde{x}} \\ \varepsilon_{\tilde{y}} \end{pmatrix} = \begin{pmatrix} \mathcal{D}\mu_{\tilde{x}} - \tilde{f}_m(\mu_{\tilde{x}}, \theta) \\ \tilde{y} - \tilde{g}_m(\mu_{\tilde{x}}, \theta) \end{pmatrix} \tag{21}$$

and Π is the inverse covariance matrix (precision matrix) of the noises:

$$\Pi = \begin{pmatrix} \Sigma_{\tilde{\omega}} & 0 \\ 0 & \Sigma_{\tilde{z}} \end{pmatrix}^{-1}$$

assuming ω and z are independent. $|\Pi|$ denotes the determinant of Π. Again, the precision term can be omitted for minimization of the free energy because it does not change the optimum; Π is not a function of $\mu_{\tilde{x}}$ nor $\tilde{y}$. Π is a function of the hyperparameters Σ_ω, Σ_z and, as shown in Chapter 3, γ.

4.2.1 *The Prediction Errors*

We will inspect ε in more detail here. Remember that ε contains two vectors: $\varepsilon_{\tilde{x}}$ and $\varepsilon_{\tilde{y}}$. The latter represents the sensory prediction error. Namely $\tilde{y}$ is the true (generalized) observation and $\tilde{g}_m(\mu_{\tilde{x}}, \theta)$ is the observation expected under the generative model (16) given the current belief $\mu_{\tilde{x}}$.

$\varepsilon_{\tilde{x}}$ also contains an error, which is more intricate. To understand this, remember that $\mu_{\tilde{x}} = E[\tilde{x}]$ because it was the mean of the Gaussian (see Equation 18). Defining $\mu_{x^{(i)}} = E[x^{(i)}]$, $\mu_{\tilde{x}}$ can be written as

$$\mu_{\tilde{x}} = \begin{pmatrix} \mu_x & \mu_{x'} & \mu_{x''} & \dots \end{pmatrix}^\top.$$

Using this notation we can see that

$$\mathcal{D}\begin{pmatrix} \mu_x \\ \mu_{x'} \\ \mu_{x''} \\ \vdots \end{pmatrix} = \begin{pmatrix} \mu_{x'} \\ \mu_{x''} \\ \mu_{x'''} \\ \vdots \end{pmatrix}$$

Let us now inspect $\varepsilon_{\tilde{x}}$ in more detail. We can expand the equation to see the underlying structure:

$$\varepsilon_{\tilde{x}} = \mathcal{D}\mu_{\tilde{x}} - \tilde{f}_m(\mu_{\tilde{x}}, \theta) = \begin{pmatrix} \mu_{x'} \\ \mu_{x''} \\ \mu_{x'''} \\ \vdots \end{pmatrix} - \begin{pmatrix} f_m(\mu_x, \theta) \\ \partial_x f_m(\mu_x, \theta)\mu_{x'} \\ \partial_x f_m(\mu_x, \theta)\mu_{x''} \\ \vdots \end{pmatrix}$$

Now it has become visible what $\varepsilon_{\tilde{x}}$ represents. Taking the first row as an example: μ_x is the expectation of x, so $\mu_{x'}$ is the current expectation of the (first order) motion of x; $\mu_{x'} = E[x']$. The other term $f_m(\mu_x, \theta)$ is the motion of the expectation $\mu_x = E[x]$ expected under the generative model. The difference represents how the motion of the current expectation of the generalized state $\mathcal{D}\mu_{\tilde{x}}$ differs from the current expectation of the motion of the generalized state as expected under the generative model $\tilde{f}_m(\mu_{\tilde{x}}, \theta)$.

Naturally it can be that $E[x'] = f_m(E[x], \theta)$. This happens when the current belief $\mu_{\tilde{x}}$ is consistent with the generative model; the motion of the belief is equal to the motion predicted by the generative model.

4.2.2 *Convexity*

A final note on the structure of the free energy: Referring to (20), the free energy is nothing more than a squared sum of prediction errors, weighted by the precision of the noises. An important observation is the following: Since Π is the inverse of a covariance matrix, it is positive semi-definite [2]. So, the free energy – without the constant term $-\frac{1}{2}\ln|\Pi|$ – is quadratic product around a positive semi-definite matrix. Hence, $\mathcal{F}(\mu_{\tilde{x}}, \tilde{y}) \geqslant 0$ and it is convex. This allows for the use of very efficient optimization algorithms [7].

It should be note that the convexity only applies to the optimization over $\tilde{x}$. If one is also interested in optimizing θ and γ the convexity does not apply.

4.3 ACTIVE INFERENCE: PREDICTION ERROR MINIMIZATION

This section will explain the equations that an Active Inference agent uses to actively optimize the free energy. This solves the last problem open question posed in Chapter 2: How to minimize $\mathcal{F}(\mu, \tilde{y})$ with respect to $\tilde{y}$?

Before going into this, take note of the following: Active Inference is an extension of DEM and is thus able to estimate a Gaussian density (with mean and variance) over the (generalized) hidden state $\tilde{x}$, parameters (θ) and hyperparameters (γ). In this thesis we will only consider state estimation, as explained in Equation 18. Moreover, the variance matrix is a function of the mean (due to the Laplace approximation, see Section 2.3.1) and thus only the mean $\mu_{\tilde{x}}$ has to be estimated. So, *Active Inference* in this case reduces to

$$\min_{u, \mu_{\tilde{x}}} \mathcal{F}(\mu_{\tilde{x}}, \tilde{y})$$

subject to Equation 14 and a *forward model*. Namely, to perform the minimization with respect to u, the agent requires knowledge of how the observations relate to the control. This information is *not* in the

generative model since that is agency free. Rather, it is defined by a forward model, which is the topic of Section 4.3.1.

The minimization with respect to $\mu_{\tilde{x}}$ represents perceptual inference (i. e.perception) and the minimization with respect to u represents action (i. e.the active part of Active Inference). The control signal u does not appear in generalized form as it is a deterministic signal over which the agent has control. Moreover, u does not appear in the generative model (which is the place where generalized motions are used). Finally, since the free energy is a sum of prediction errors, Active Inference is a form of *prediction error minimization*.

As Active Inference is designed to be biologically plausible, the minimization of $\mathcal{F}(\mu_{\tilde{x}}, \tilde{y})$ is performed through a (generalized) gradient descent. It is believed that this is biologically plausible [33]. The gradient descent equations for (state restricted) Active Inference are:

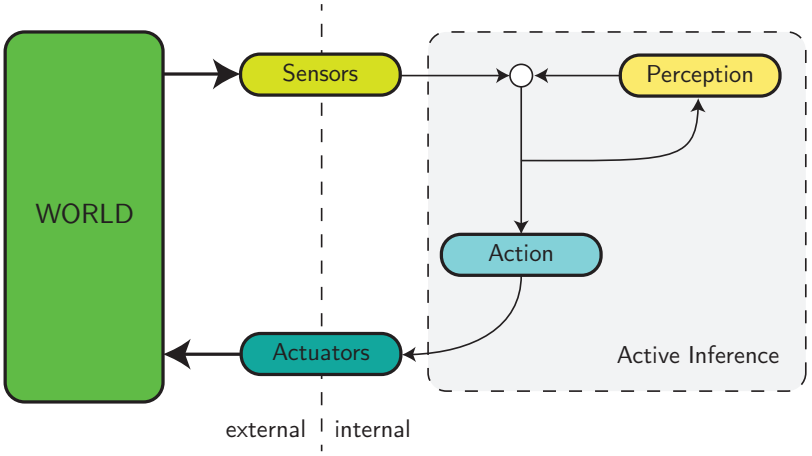
$$\dot{\mu}_{\tilde{x}} = \mathcal{D}\mu_{\tilde{x}} - \partial_{\mu_{\tilde{x}}}\mathcal{F}(\mu_{\tilde{x}}, \tilde{y}) \tag{22}$$

$$\dot{u} = -\partial_{u}\mathcal{F}(\mu_{\tilde{x}}, \tilde{y}) \tag{23}$$

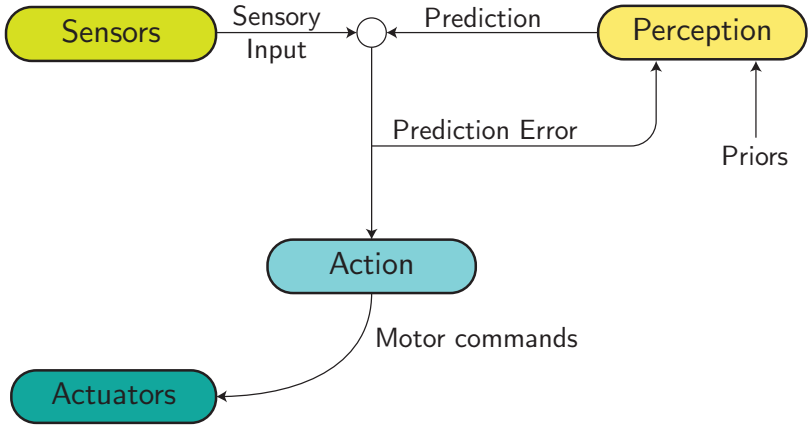
where again the gradient descent for the control u is ill-defined as $\mathcal{F}(\mu_{\tilde{x}}, \tilde{y})$ is not a function of u. More detail is provided in the next section. Observe that Active Inference has an integrating action embedded by default, due to the differential equation that determines the control signal. Equations for unrestricted Active Inference can be found in [28, 29].

To intuitively interpret the gradient descents, remember that $\mu_{\tilde{x}}$ represents neural activity in the brain, the current belief about hidden states. As such, perception corresponds to changing the current predictions – through gradient descent – about hidden states $\mu_{\tilde{x}}$ such that it better explains the observations. Action on the other hand corresponds to changing the observation such that it better explains the predictions. This tight integration of perception and action is something unique in Active Inference [37].

Figure 4 provides a high level overview of Active Inference. In conventional approaches the careful design of a cost function is crucial for good performance. This cost function is the free energy in Active Inference, but the "design part" is actually in the priors. The mathematical form of these prior has not been discussed yet. However, when introducing the generative model the state mappings were given an additional subscript (f_m and g_m) to indicate that these models might differ from the generative process. In fact, this difference is exactly determined by the priors. As such, the careful design part of Active Inference is to find suitable priors. An example of this will be provided in the upcoming part of this thesis.



(a) Closed loop block diagram of Active Inference.



(b) Blow-up view of the internal structure.

Figure 4: Block diagrams for Active Inference. (a) The closed loop diagram of Active Inference contains two loops. The internal loop optimizes predictions through perception and the loop including the world minimizes prediction errors by changing the world state (hidden state). (b) Blow-up of the internal structure of Active Inference, highlighting how prediction (error) signals flow.

4.3.1 *Forward Model*

As mentioned before, Equation 23 is technically incorrect as $\mathcal{F}(\mu_{\tilde{x}}, \tilde{y})$ is not a function of u, while it is differentiated with respect to it. However, realize that $\tilde{y}$ is indirectly a function of u through the generative process dynamics. By the chain rule, Equation 23 can be rewritten as

$$\dot{u} = -\partial_u \tilde{y} \; \partial_{\tilde{y}} \mathcal{F}(\mu_{\tilde{x}}, \tilde{y}).$$

The problem with this result is that the first partial derivative is a very complex object. Physically, it represents the derivative of a *forward dynamic model* $\tilde{y}(u)$ that relates control action to output. In a biological sense, this mapping is represented by simple reflex arcs [1, 37, 39].

For state space systems however, $\tilde{y}(u)$ is not a simple one-to-one mapping. Namely, both u and $\tilde{y}$ are a function (of time) and $\partial_u \tilde{y}$ is thus a functional derivative. It is impossible to determine $\partial_{u(t)} \tilde{y}(t)$ only considering both signals at time t. One has to take into account the complete history of $u(t)$. Namely, remember that the generative process is – without disturbances and noises

$$\dot{x} = f(x, u, \theta)$$
$$y = g(x, \theta)$$

Assuming that a closed-form solution for $x(t)$ given initial condition $x(0) = x_0$ and signal $u(t)$ for $t \in [0, t]$ exists, then we obtain

$$x(t) = \int_0^t f(x(0), u(t), \theta))dt$$
$$y(t) = g(x(t), \theta)$$

Hence $y(t)$ depends on the whole trajectory of the control up to t (and the initial conditions of x). The problem becomes even more complicated once introducing time varying models (i. e., $\theta \to \theta(t)$). A solution will be proposed in Chapter 6. The advantage of the forward model is that there is no need for an inverse model. Inverse models are often harder to derive than forward models, for example because they are anti-causal and can be non-injective and/or bijective. This is a common problem in robotics: the inverse kinematics problem [71].

4.4 A CONCEPTUAL OVERVIEW

All theory constituting Active Inference has been discussed now, with the exception of some details that are not relevant for this thesis. One main construct that has not passed the revue in the exposition presented in this thesis is hierarchical models [4, 34]. This concept uses the fact that, much like in a deep neural network, the cortex consists of several layers. It is proposed that each layer is involved in prediction error minimization. Applications of this concept in engineering would be most interesting in the context of neural networks, where it could provide complex inference capabilities. However, to understand the hierarchical version it is first required to understand what happens in one layer, which is what this thesis focuses on.

The upcoming two sections expose the interesting aspects of Active Inference and briefly sum up the existing paradigms in engineering to which Active Inference is related.

4.4.1 *Interesting Aspects*

This section will provide a brief overview of the interesting aspects of Active Inference for engineering purposes. Thus far, Active Inference has not had significant impact yet in engineering. It is however

becoming increasingly influential in brain sciences, for having great explanatory power and unifying strength of the brain's functioning [1, 26, 33, 35, 44]. If Active Inference indeed captures some crucial aspect of human intelligence, this might well be transferable to machine learning and robotics to create smarter systems.

There are several interesting aspects of Active Inference for engineering sciences. First, at the highest level, Active Inference is minimization of prediction errors in the brain by adjusting internal belief (neural activity) and by executing actions. Using the prediction errors as basis of neural activity is the field of predictive coding [12, 42], which Active Inference is thus closely related to. Using prediction error minimization as basis for neural networks or other control systems might provide interesting developments.

Secondly, note that the Active Inference algorithm is unsupervised, as all unknown parameters are learned through gradient descent. The only supervisory part is that the form of the generative model must be pre-defined. This could be problematic in complex systems, where a proper model is not known. However, in this case Active Inference provides a way to develop unsupervised free-form-model learning algorithms using e.g. neural networks or variational auto-encoders.

Thirdly, in Active Inference there is a deep functional integration of perception and action, which is something missing in previous neurological theories and also engineering models. Many robotics systems have separate systems for these two tasks, whereas Active Inference proposes a way to merge the two systems. Robots often have very separate vision and motion planning systems. Or in control science there is often a separate filter and controller. Moreover, from a control theoretic perspective it is very interesting that the reference (prior) is supplied directly to the filter. Due to this it seems that the separation principle [57] does not hold when applying Active Inference on linear state space models.

Several more detailed concepts are also interesting. Firstly, an interesting phenomenon is the replacement of a cost function with priors. This is claimed to solve the problem of redundant degrees of freedom [37]. Secondly, Active Inference changed a difficult inference problem into an easier optimization. This obviated the need for an inverse model [25, 37], and made the problem computationally tractable. Finally, the use of generalized motions could provide more accurate filtering algorithms [28].

4.4.2 *Relation to Existing Paradigms*

Beside the inspiration that Active Inference provides to develop improved or novel algorithms, Active Inference is remarkably well able to connect many different existing paradigms in engineering. As clear from Chapter 2, Active Inference is a form of variational inference (i. e.

VBI), which is becoming common in machine learning [67]. Recent work also started to combine Active Inference with advancements in variational inference such as variational auto-encoders [73].

Active Inference is claimed to be a form of optimal control [27]. Moreover, it is argued that DEM is more accurate than (extended) Kalman filtering or equivalent for Gaussian noises [28]. A theoretical proof of this has not yet been published, but it could mean that Active Inference is equivalent to LQG. A second interesting proof – if it exists – is the proof that the free energy is a Lyapunov function for the closed loop system. The reason to believe such a proof might exist is because the dynamics of the agent are gradient descents on the free energy, thus always lowering the free energy (except for random fluctuations). Finally, note that the free energy is a function of time since $\mu_{\tilde{x}}$ is a function of time. This implies that minimizing the free energy at all times equates to minimizing its integral over time. As the time-integral of the free energy is action, FEM adheres to the principle of least action [38, 65].

More on the machine learning side of engineering, the relation between Active Inference and reinforcement learning has been examined [25] and implementations of the perceptual inference part of Active Inference in neural networks have been presented [16, 66].

Of course there is no free lunch, and there have also been critics on Active Inference, mostly regarding the computational complexity [12, 58] and the fact that replacing the cost function with priors does not solve any problem, but merely moves it. The difficulty with Active Inference identified in this thesis is twofold. Firstly, it was unclear how to incorporate the concept of generalized motions into control theoretic models (such as state space models). As literature on this concept is scarce, it is hard to gain insight in this concept. Consequently it cannot yet be determined if there is use for generalized motions in engineering, this thesis provides a means to start answering this question. Secondly, it is unclear how to remove the concept of generalized motions from Active Inference, which complicates the process of connecting the theory to existing paradigms. It is claimed that DEM without generalized motions is equal to a Kalman-Bucy filter [28], but the theoretical proof is hard to find due to aforementioned reasons. A primer is provided in Chapter 10.

Finally, there is only one toolbox[2] (for MATLAB) available which implements Active Inference. The code in this toolbox is unfortunately very complex and it is thus hard to extract the essence.

2 https://www.fil.ion.ucl.ac.uk/spm/

Part II

LTI STATE SPACE FORMULATION

To analyze Active Inference with proper methodologies as used in control science, it is required to formulate the equations of Active Inference in an appropriate manner. In this part the application of Active Inference on *Linear-Time-Invariant* (LTI) state space systems will be elaborated, answering the second research question: *What is the LTI state space formulation of Active Inference?* Besides enabling detailed performance analysis and rigorous qualitative comparisons, this will increase our understanding of Active Inference. Chapter 5 explains the model used in the filtering part of the algorithm. The controller requires a second model which is derived in Chapter 6. Equipped with these two models, the control system representing Active Inference is set up in Chapter 7. Finally, Chapter 8 presents an example in which all details are clarified.

5 GENERATIVE MODEL

This chapter will set up the first of the two models required for an agent to perform Active Inference on an LTI state space system. First, Section 5.1 will define the exact equations describing the environment/system that is to be controlled. In order to control the system, an agent requires a generative model, which is set up in Section 5.2. Finally, Section 5.3 elaborates in more detail on a particular element of the model which defines the agent's behavior.

5.1 THE GENERATIVE PROCESS

The *generative process* in all further discussions is of the form:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) + \omega(t) \\ y(t) &= Cx(t) + z(t) \end{aligned} \tag{24}$$

with $x, \omega \in \mathbb{R}^n$, $u \in \mathbb{R}^m$ and $y, z \in \mathbb{R}^q$. x is the process state, subject to noise ω. u is the control input. y is the measurement received by the agent, subject to noise z. Time is omitted for a cleaner notation.

The agent's goal is to control x to some desired reference x_{eq}. Note that not all choices for x_{eq} are necessarily valid. Namely, there must exist an input u that is able to keep x in equilibrium at x_{eq}:

$$\dot{x} = 0 = Ax + Bu \quad \rightarrow \quad \exists u : Bu = -Ax$$

This corresponds to verifying that x_{eq} is in the row space spanned by the controllability matrix [74] of the generative process. Additionally, the equilibrium must be uniquely determined by the output:

$$\exists! y : y = Cx_{eq}$$

These two requirements guarantee that (a) there is a control signal u that can steer the system towards x_{eq} plus keep it there and (b) this can be achieved through observing y.

5.2 THE GENERATIVE MODEL

In contrast to e.g. a Kalman filter, Active Inference does not use an exact copy of Equation 24 as model. The model used in Active Inference is different from the process in two ways:

1. The state equation is modeled in generalized form. As such, the model is not a dynamic model but an instantaneous probabilistic mapping (see Chapter 3).

2. The term Bu is replaced by a *prior variable*, denoted with ξ. This difference is crucial, it drives an agent's behavior.

These two differences will be clarified in this section. There will also be a difference between the original theory of Active Inference and the model presented here. Namely, in the original theory of Active Inference the output equation is also modeled in generalized form [29], implying that the agent requires not only the measurement y but also its higher order motions. We choose not to do this as to keep the resulting controller coherent with control engineering standards (where only the measurement y is available to the agent). Besides, in discrete time – where the algorithm will eventually work in robotic applications – this means that the agent has a certain delay depending on the embedding order. Appendix A clarifies this and shows how one could generate $\tilde{y}$ from a discrete sequence of data-points.

As mentioned before, we will assume that the parameters and hyperparameters are known, such that only the state partition of the posterior approximation $q(x;\mu)$ needs to be inferred. In other words, the model matrices (A, B, C) are known by the agent. Extrapolating from Chapter 3, we can write Equation 24 as

$$\begin{aligned} x' &= f(x,u) + \omega & y &= g(x) + z \\ x'' &= \partial_x f(x,u)x' + \partial_u f(x,u)u' + \omega' \\ x''' &= \partial_x f(x,u)x'' + \partial_u f(x,u)u'' + \omega'' \\ &\vdots \end{aligned}$$

where generalized output equations have been omitted, as argued above. For the LTI state space system we have that $f(x,u) = Ax + Bu$ and $g(x) = Cx$. The above then becomes

$$\begin{aligned} x' &= Ax + Bu + \omega & y &= Cx + z \\ x'' &= Ax' + Bu' + \omega' \\ x''' &= Ax'' + Bu'' + \omega'' \\ &\vdots \end{aligned} \tag{25}$$

Theoretically, this goes up to infinite order. For practical reasons however, an agent only models p orders of state equations. The *embedding order* $p \in \mathbb{Z}^+$ is the number of state equations used in the generative model. A proper choice for p proposed by Friston [24] is 6.

> **The special case that $p = 1$:** Choosing $p = 1$ (required when the noise *is* Markovian) requires a slightly different approach, as explained in Section A.5.

Using embedding order p, Equation 25 can be written compactly as

$$\mathcal{D}\tilde{x} = \tilde{A}\tilde{x} + \tilde{B}\tilde{u} + \tilde{\omega} \qquad y = \tilde{C}\tilde{x} + z \tag{26}$$

with

$$\mathcal{D} = \begin{pmatrix} 0 & 1 & & \\ & 0 & \ddots & \\ & & \ddots & 1 \\ & & & 0 \end{pmatrix} \otimes I_n$$

where the first matrix is of size $p \times p$ and $I_n \in \mathbb{R}^{n\times n}$. $\otimes$ is the kronecker tensor product. Hence, $\mathcal{D} \in \mathbb{R}^{np\times np}$. Also,

$$\tilde{A} = I_p \otimes A, \qquad \tilde{B} = I_p \otimes B, \qquad \tilde{C} = \begin{pmatrix} C & 0 & \dots & 0 \end{pmatrix}.$$

So $\tilde{A} \in \mathbb{R}^{np\times np}$, $\tilde{B} \in \mathbb{R}^{np\times nm}$ and $\tilde{C} \in \mathbb{R}^{q\times np}$.

Let us rearrange Equation 26 to arrive at:

$$\mathcal{D}\tilde{x} - \tilde{A}\tilde{x} - \tilde{B}\tilde{u} = \tilde{\omega} \qquad y - \tilde{C}\tilde{x} = z \tag{27}$$

Under the assumption that the noises are Gaussian, $p(\tilde{\omega}) \sim \mathcal{N}(0, \Sigma_{\tilde{\omega}})$, $p(z) \sim \mathcal{N}(0, \Sigma_z)$. Also, $p(\tilde{x}) = p(\tilde{\omega})$ and $p(y|\tilde{x}) = p(z)$ as from Equation 27. The mean of both densities is zero, but naturally it is also the expectation of Equation 27. So, the generative model is found by taking the expectation over Equation 27:

$$\mathcal{D}\mu - \tilde{A}\mu + \xi = 0 \qquad y - \tilde{C}\mu = 0 \tag{28}$$

where the first equation has a covariance $\Sigma_{\tilde{\omega}}$ and the second a covariance Σ_z. We have replaced $E[\tilde{B}\tilde{u}]$ by ξ, which introduces the second of two differences with conventional filtering as mentioned before. $\xi \in \mathbb{R}^{np}$ represents a prior expectation over the control signal, or a prior expectation over the generalized state. The next section elaborates on this.

5.3 THE CONTROL PRIOR

In this section we will present a standard form for the prior ξ and show that it is able to drive the state estimation towards the desired value. One is free to define anything for ξ, but in this setting it makes most sense to use

$$\xi = \mathcal{D}\mu_{eq} - (\tilde{A} + \tilde{K})\mu_{eq} + K\mu \tag{29}$$

where $\mu_{eq} = \begin{pmatrix} x_{eq} & \dot{x}_{eq} & \dots \end{pmatrix}^\top$. This prior allows for both loop shaping – through $\tilde{K}$ – and reference tracking – through μ_{eq}. Namely, consider the state equation of the generative model: $\mathcal{D}\mu - \tilde{A}\mu - \xi = 0$. Substituting Equation 29 for ξ and rearranging yields:

$$\mathcal{D}(\mu - \mu_{eq}) - (\tilde{A} + \tilde{K})(\mu - \mu_{eq}) = 0$$

which is simply a mapping for the expectation of the generalized tracking error $(\tilde{x} - \tilde{x}_{eq})$. This equation is only true when $\mu = \mu_{eq}$ (unless $\mathcal{D} = (\tilde{A} + \tilde{K})$) and thus the generative model will drive the belief μ towards μ_{eq} as long as the filter Equation 35 is stable. At this point we have not introduced the filter yet, so for now assume the filter using Equation 29 for ξ can be written as

$$\dot{\mu} = E\mu + Fy + H\mu_{eq}$$

where

$$E = \mathcal{D} - \kappa\tilde{C}^{\top}\Pi_z\tilde{C} - \kappa(\mathcal{D} - \tilde{A} - \tilde{K})\Pi_{\tilde{\omega}}(\mathcal{D} - \tilde{A} - \tilde{K})$$

The filter converges to μ_{eq} when E is stable. The controller will then make sure that $y = C\mu_{eq}$, resulting in a closed loop that tracks the reference.

A final remark on the prior variable: Remember from Chapter 2 that the generative model is

$$p(\tilde{x}, y) = p(y|\tilde{x})p(\tilde{x})$$

The output equation in Equation 28 corresponds to $p(y|\tilde{x})$, as it explains how data y depends on the current generalized state $\tilde{x}$. The state equation corresponds to $p(\tilde{x})$ as it models the belief about the behavior of $\tilde{x}$. Crucially, this belief does not need to be an exact copy of the world and that is why we allow for an extra variable ξ which allows us to reshape the prior $p(\tilde{x})$ in any way possible. Essentially this is also what a control signal does to the true state x and hence it is not a coincidence that ξ can also be seen as the expectation of the control signal. It is however just as valid to think of it as a prior over the states as is evident from Equation 29.

Conventionally, a reference signal is supplied to the controller, but in Active Inference the reference is embedded in the internal model using ξ. This difference is also evident from Figure 5.

6 FORWARD MODEL

This chapter will discuss the second of two models required by an Active Inference agent: the forward model. Namely, besides a model of how the hidden states and observations interact, the agent must also model how its actions affect the observations. This is captured in a forward model, as will be explained in this chapter. Two different derivations are provided, one from a standard control theoretic perspective and one using generalized motions. As both result in the same forward model, the result is more credible.

6.1 DERIVATION FROM STANDARD PROCESS

We can derive a forward model $y(u)$ from the (deterministic) generative process equations, omitting the noise since we are only looking for a relation between u and y:

$$\begin{aligned}\dot{x} &= Ax + Bu\\ y &= Cx\end{aligned}$$

The solution for $x(t)$ given some input signal $u(\tau), \tau \in [0,\ t]$ is

$$x(t) = e^{At}x(0) + \int_0^t e^{A(t-\tau)}Bu(\tau)d\tau$$

The solution for $y(t)$ is then rather simple to determine:

$$y(t) = Ce^{At}x(0) + C\int_0^t e^{A(t-\tau)}Bu(\tau)d\tau\ .$$

Since $u(\tau)$ is some arbitrary signal we cannot further simplify the integral and the partial derivative $\partial_u y$ has to be evaluated using variational calculus. This does however require that the function $u(\tau)$ is known and differentiable, which need not to be the case.

To avoid this problem, consider steady-state behavior i.e assume that $u(\tau) = u$ is constant, A is Hurwitz and t is sufficiently large so that the transient response has vanished ($e^{At} \approx 0$). In this case $u(\tau)$ can be pulled out of the integral so that the partial derivative becomes

$$\begin{aligned}\partial_u y(t) &= \partial_u \left[Ce^{At}x(0) + C\int_0^t e^{A(t-\tau)}Bd\tau\ u\right]\\ &= C\int_0^t e^{A(t-\tau)}Bd\tau\\ &= CA^{-1}e^{At}B - CA^{-1}B\\ &\approx -CA^{-1}B\end{aligned}$$

Note this will be the only place where B appears inside the agent, which represents the agent's belief about how its control affects the process state. The result above is simply the steady state gain matrix. We will define the *forward model* as

$$G = -CA^{-1}B \tag{30}$$

Observe that $\partial_u y = C\partial_u x$. The problem stated above rests in the latter partial derivative. The reason why this partial derivative is problematic is because even if $x(t)$ is known, this does not provide enough information about how $x(t)$ will change with respect to $u(t)$. Namely, it does not provide information about the continuity of $x(t)$, which needs to be ensured. To solve this, the current trajectory of $x(t)$ needs to be known. This trajectory can be found either by evaluating the integral given the history of u, or by assuming it is known. The latter implies knowledge of generalized motions, which should sound familiar. The next section will show how this comes in handy.

6.2 DERIVATION FROM GENERALIZED PROCESS

A second way to find the forward model is to start by writing the deterministic generative process in generalized form:

$$\begin{aligned} \mathcal{D}\tilde{x} &= \tilde{A}\tilde{x} + \tilde{B}\tilde{u} \\ \tilde{y} &= \tilde{C}\tilde{x} \end{aligned}$$

From this it is easy to find that

$$\tilde{y} = \tilde{C}(\mathcal{D} - \tilde{A})\tilde{B}\tilde{u}$$

Remember that the output was chosen not to be modeled in generalized form, so that the above becomes

$$y = \tilde{C}(\mathcal{D} - \tilde{A})^{-1}\tilde{B}\tilde{u} \quad \text{where} \quad \tilde{C} = \begin{pmatrix} C & 0 & 0 & \dots & 0 \end{pmatrix} \tag{31}$$

(and $\tilde{A} = I_p \otimes A$ and $\tilde{B} = I_p \otimes B$). Observe that

$$(\mathcal{D} - \tilde{A}) = \begin{pmatrix} -A & I_n & & \\ & -A & \ddots & \\ & & \ddots & I_n \\ & & & -A \end{pmatrix}$$

which is upper triangular. The inverse is hence easy to evaluate. In fact, due to the structure of $\tilde{C}$ only the first diagonal element is required, which is $-A^{-1}$. Then, expanding the multiplications from Equation 31 yields

$$y = -CA^{-1}Bu$$

and thus

$$\partial_u y = -CA^{-1}B$$

which is the same result as obtained before.

7 FILTER AND CONTROLLER DYNAMICS

The previous two chapters have introduced the two models involved in performing Active Inference on an LTI state space system. This chapter will introduce the exact form of the free energy to be minimized in Section 7.1. The resulting formula is subsequently used to define the filtering and control dynamics in Section 7.2 and Section 7.3 respectively.

7.1 THE REFORMULATED FREE ENERGY

This section will introduce the final form of the free energy that will be used to perform Active Inference on LTI state space systems. All simplifications to arrive at this formulation will also be recapitulated.

To arrive at the free energy formula, we will simply substitute the generative model Equation 28 in the prediction errors Equation 21:

$$\varepsilon = \begin{pmatrix} \varepsilon_x \\ \varepsilon_y \end{pmatrix} = \begin{pmatrix} \mathcal{D}\mu - \tilde{A}\mu - \xi \\ y - \tilde{C}\mu \end{pmatrix}$$

with the corresponding minimizable free energy (omitting constant terms irrelevant for the optimization):

$$\mathcal{F}(\mu, y) = \tfrac{1}{2}\varepsilon^\top \Pi \varepsilon$$

where Π is block diagonal, so the two prediction errors are independent. Then, for completeness, the free energy can be written as:

$$\begin{aligned} \mathcal{F}(\mu, y) = {} & \tfrac{1}{2}(y - \tilde{C}\mu)^\top \Pi_z (y - \tilde{C}\mu) \\ & + \tfrac{1}{2}(\mathcal{D}\mu - \tilde{A}\mu - \xi)^\top \Pi_{\tilde{w}} (\mathcal{D}\mu - \tilde{A}\mu - \xi). \end{aligned} \tag{32}$$

In fact Equation 32 is not close to the original free energy anymore, only a small part of it is left. We will still refer to it as simply the free energy to avoid confusion. Now, the free energy is all that is required to set up Active Inference as explained in the next two sections. These sections will derive Equation 22 and Equation 23 using the free energy as defined above.

7.2 THE FILTERING DYNAMICS

The first gradient descent is for the state estimation i.e. filtering:

$$\dot{\mu} = \mathcal{D}\mu - \kappa \partial_\mu \mathcal{F}(\mu, y) \tag{33}$$

where, with respect to Equation 22, a learning rate $\kappa \in \mathbb{R}^+$ was added, as is standard use in gradient descents. It allows to tune the behavior of the filter. The partial derivative of the free energy is easy to evaluate, it is

$$\partial_\mu \mathcal{F}(\mu, y) = -\tilde{C}^\top \Pi_z \varepsilon_y + (\mathcal{D} - \tilde{A} - \partial_\mu \xi)^\top \Pi_{\tilde{\omega}} \varepsilon_\mu. \tag{34}$$

Since one is free to define the prior, the partial derivative $\partial_\mu \xi$ cannot be specified analytically.

Now, substituting Equation 34 in Equation 33, the dynamics for μ become

$$\begin{aligned} \dot{\mu} &= \mathcal{D}\mu - \kappa \left(-\tilde{C}^\top \Pi_z \varepsilon_y + (\mathcal{D} - \tilde{A} - \partial_\mu \xi)^\top \Pi_{\tilde{\omega}} \varepsilon_\mu \right) \\ &= \mathcal{D}\mu + \kappa \left[\tilde{C}^\top \Pi_z (y - \tilde{C}\mu) - (\mathcal{D} - \tilde{A} - \partial_\mu \xi)^\top \Pi_{\tilde{\omega}} (\mathcal{D}\mu - \tilde{A}\mu - \xi) \right] \end{aligned} \tag{35}$$

As mentioned before gradient descent might not be the most efficient optimization available, but it is chosen due to the biological plausibility [29]. It should also be noted that due to the structure of $V(\gamma)$, prediction errors in higher order motions of μ quickly contribute less. This is because the precision of noises in $\tilde{\omega}$ quickly decreases at higher order. For this reason it has been proposed that $p = 6$ is sufficient in most cases [28].

As also mentioned earlier, the generative model is only a probabilistic mapping, which is not dynamic. The filtering equation provided here is where the dynamics are put back into the state estimation μ. The filtering equation is a form of state estimation much like a Kalman-Bucy filter does [9, 54]. This will be elaborated some more in Chapter 10.

7.3 THE CONTROL DYNAMICS

As explained in Section 4.3, the second gradient descent determines the control dynamics:

$$\dot{u} = -\rho \partial_u y^\top \partial_y \mathcal{F}(\mu, y)$$

where $\rho \in \mathbb{R}^+$ is a learning rate, added to tune the behavior of the controller. The transpose is required since we are working in the matrix domain [3]. From Equation 32 it is easily determined that

$$\partial_y \mathcal{F}(\mu, y) = \Pi_z \varepsilon_y$$

The other partial derivative is the forward model $G = -CA^{-1}B$. Combining the two yields that

$$\dot{u} = -\rho G^\top \Pi_z (y - \tilde{C}\mu) \tag{36}$$

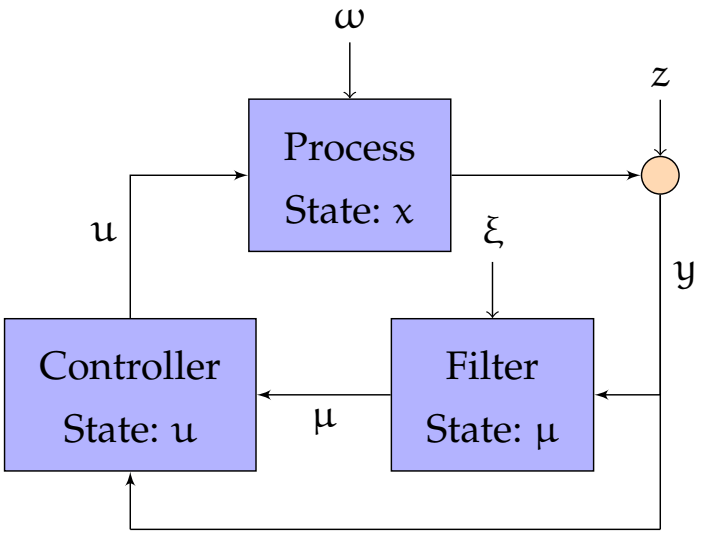


Figure 5: Closed loop scheme of Active Inference. An agent (filter + controller) controls some process with state x which is subject to dynamics, and noise ω. The observation is also subject to noise z. Internally, the agent keeps track of a state estimate μ and of the control signal u, used to control the process. ξ is the prior belief defining desired process behavior.

7.4 CLOSED LOOP MODEL

By connecting the agent's gradient descent equations with the generative process, a closed loop system description can be obtained. This is useful for simulation purposes[1]. Figure 5 shows a block scheme of all signals in the closed loop. From the closed loop perspective, ω, z and ξ are inputs and x, μ, u are the state space states.

The first step in describing the closed loop equation is to write a state space model of the agent. From Equation 35 and Equation 36 we know that

$$\dot{\mu} = \mathcal{D}\mu + \kappa\left(\tilde{C}^\top \Pi_z(y - \tilde{C}\mu) - (\mathcal{D} - \tilde{A})^\top \Pi_{\tilde{\omega}}(\mathcal{D}\mu - \tilde{A}\mu - \xi)\right) \tag{37}$$

$$\dot{u} = -\rho G^\top \Pi_z(y - \tilde{C}\mu) \tag{38}$$

where it was used that $\partial_\mu \xi = 0$, i. e., assuming no pole placement. These two equations can be written as a state space model. As can be seen from Figure 5, y and ξ are inputs to the agent and u is the output. The two variables subject to dynamics (μ and u) are the state x_a of the agent state space model. Therefore the model is:

$$\dot{x}_a = A_a x_a + B_a \begin{pmatrix} y \\ \xi \end{pmatrix}$$

$$u = C_a x_a$$

1 We will use MATLAB's `lsim` function.

where $x_a = \begin{pmatrix} \mu & u \end{pmatrix}^\top$ and from Equation 37 and Equation 38:

$$A_a = \begin{pmatrix} \mathcal{M} & 0 \\ \rho G^\top \Pi_z \tilde{C} & 0 \end{pmatrix},$$
$$B_a = \begin{pmatrix} \kappa \tilde{C}^\top \Pi_z & \kappa(\mathcal{D} - A)^\top \Pi_{\tilde{\omega}} \\ -\rho G^\top \Pi_z & 0 \end{pmatrix},$$
$$C_a = \begin{pmatrix} 0 & I_m \end{pmatrix}$$

where we have defined

$$\mathcal{M} = \mathcal{D} - \kappa(\mathcal{D} - \tilde{A})^\top \Pi_{\tilde{\omega}} (\mathcal{D} - \tilde{A}) - \kappa \tilde{C}^\top \Pi_z \tilde{C}$$

to avoid cluttering.

At this point there are two state space models, described by the following four equations:

$$\dot{x} = Ax + Bu + \omega$$
$$y = Cx + z$$
$$\dot{x}_a = A_a x_a + B_a \begin{pmatrix} y \\ \xi \end{pmatrix}$$
$$u = C_a x_a$$

The output equations can be eliminated by substituting them in the state equations:

$$\dot{x} = Ax + BC_a x_a + \omega$$
$$\dot{x}_a = A_a x_a + B_a \begin{pmatrix} Cx + z \\ \xi \end{pmatrix}$$

This is equivalent to closing the loop, since we have now connected the in- and outputs of the two systems. We can write the closed loop as an augmented state space model

$$\dot{x}_{cl} = A_{cl} x_{cl} + B_{cl} u_{cl}$$

where $x_{cl} = \begin{pmatrix} x & x_a \end{pmatrix}^\top = \begin{pmatrix} x & \mu & u \end{pmatrix}^\top$. The signals ω, z and ξ are inputs: $u_{cl} = \begin{pmatrix} \xi & \omega & z \end{pmatrix}^\top$. Note that only ξ is controllable. ω and z are disturbances. With some bookkeeping it is easy to derive that

$$A_{cl} = \begin{pmatrix} A & 0 & B \\ \kappa \tilde{C}^\top \Pi_z C & \mathcal{M} & 0 \\ -\rho G^\top \Pi_z C & \rho G^\top \Pi_z \tilde{C} & 0 \end{pmatrix},$$
$$B_{cl} = \begin{pmatrix} 0 & I_n & 0 \\ \kappa(\mathcal{D} - \tilde{A})^\top \Pi_{\tilde{\omega}} & 0 & \kappa \tilde{C}^\top \Pi_z \\ 0 & 0 & -\rho G^\top \Pi_z \end{pmatrix}.$$

7.5 SUMMARY

This section will compactly summarize all essential equations for Active Inference on stochastic LTI state space systems, as derived in Chapter 5, Chapter 6 and the current chapter.

All of this thesis so far essentially culminated in four equations, together constituting the generative process and Active Inference agent:

$$\begin{aligned}
\dot{x} &= Ax + Bu + \omega \\
y &= Cx \\
\dot{\mu} &= \mathcal{D}\mu + \kappa[\tilde{C}^\top \Pi_z(y - \tilde{C}\mu) - (\mathcal{D} - \tilde{A} - \partial_\mu \xi)^\top \Pi_{\tilde{\omega}}(\mathcal{D}\mu - \tilde{A}\mu - \xi)] \\
\dot{u} &= -\rho G^\top \Pi_z(y - \tilde{C}\mu)
\end{aligned}$$

which are the generative process (first two equations), the filter and the controller. In these equations we have

$$\begin{aligned}
\tilde{A} &= I_p \otimes A \\
\tilde{C} &= \begin{pmatrix} C & 0 & 0 & \dots & 0 \end{pmatrix} \\
G &= -CA^{-1}B \\
\mathcal{D} &= \begin{pmatrix} 0 & 1 & & \\ & 0 & \ddots & \\ & & \ddots & 1 \\ & & & 0 \end{pmatrix} \otimes I_n \\
\Pi_z &= \Sigma_z^{-1} \\
\Pi_{\tilde{\omega}} &= V(\gamma)^{-1} \otimes \Pi_\omega = V(\gamma)^{-1} \otimes \Sigma_\omega^{-1}
\end{aligned}$$

where for $\mathcal{D}$, the first matrix is of size $p \times p$. Also, $V(\gamma) \in \mathbb{R}^{p \times p}$ with the following internal structure:

$$V(\gamma) = \begin{pmatrix} 1 & 0 & \ddot{\rho}(0) & \dots \\ 0 & -\ddot{\rho}(0) & 0 & \\ \ddot{\rho}(0) & 0 & \rho^{(4)}(0) & \\ \vdots & & & \ddots \end{pmatrix}.$$

in which $\rho(t)$ is the autocorrelation function of the noise, parameterized by the roughness parameter γ.

The (Laplace encoded) free energy itself is superfluous, but for completeness, it is defined as:

$$\begin{aligned}
\mathcal{F}(\mu, y) = &\tfrac{1}{2}(y - \tilde{C}\mu)^\top \Pi_z(y - \tilde{C}\mu) \\
&+ \tfrac{1}{2}(\mathcal{D}\mu - \tilde{A}\mu - \xi)^\top \Pi_{\tilde{\omega}}(\mathcal{D}\mu - \tilde{A}\mu - \xi).
\end{aligned}$$

Parameters that have to be chosen manually (tuning parameters) are summarized in Table 1.

Table 1: Tuning parameters in Active Inference.

PARAMETER NAME	SYMBOL
Embedding order	$p \in \mathbb{R}^+$
Learning rates	$\rho, \kappa \in \mathbb{R}^+$
Control/state prior	$\xi \in \mathbb{R}^{np}$
Roughness parameter	$\gamma \in \mathbb{R}^+$
Noise covariances	$\Sigma_\omega \in \mathbb{R}^{n\times n}, \Sigma_z \in \mathbb{R}^{q\times q}$

Several assumptions and approximations have been employed to arrive at the formulation of Active Inference presented above:

1. Parameters and hyperparameters (θ and $\gamma, \Sigma_\omega, \Sigma_z$) are known.
2. Mean-field approximation: $q(\vartheta;\mu) = q(\tilde{x};\mu_{\tilde{x}})q(\theta;\mu_\theta)q(\gamma;\mu_\gamma)$
3. Laplace approximation $q(\tilde{x};\mu_{\tilde{x}}) \sim \mathcal{N}(0, \Sigma_{\tilde{\omega}})$
4. Noises are non-Markovian and Gaussian
5. Noises have a Gaussian autocorrelation
6. Application to LTI state space systems
7. *No* generalized observation $\tilde{y}$

Having arrived at the formulation of Active Inference that is applicable to LTI state space models, the next step is of course to study the performance and analyze the algorithm mathematically. This is the subject of the next part of this thesis. Before going into this, the next chapter provides a toy example to solve any remaining ambiguities on how to implement the state space Active Inference algorithm.

8 AN EXAMPLE

8.1 THE GENERATIVE PROCESS

The generative process we consider is a simple mass-spring-damper system with mass m [kg] , spring constant k [N/m] and damping constant c [Ns].

$$\begin{pmatrix} \dot{x} \\ \ddot{x} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{pmatrix} \begin{pmatrix} x \\ \dot{x} \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{1}{m} \end{pmatrix} u + \begin{pmatrix} \omega_x \\ \omega_{\dot{x}} \end{pmatrix}$$
$$y = \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ \dot{x} \end{pmatrix} + z$$

We set a constant equilibrium position of α, so $x_{eq} = \begin{pmatrix} \alpha & 0 \end{pmatrix}^\top$. Besides this, several (hyper)parameters have to be chosen. All parameter values are listed in 2. To keep the example compact, $p = 2$ has been chosen. The learning rates κ and ρ have been manually optimized, which is the main difficulty in achieving satisfying performance. The other parameters are arbitrary with the exception that m, k and c are chosen such that the resulting system is stable.

8.2 CONTROL SYSTEM SETUP

This section provides an explicit example of all the components constituting the agent i. e.control system.

Since $p = 2$ we have that

$$\tilde{A} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -\frac{k}{m} & -\frac{c}{m} & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\frac{k}{m} & -\frac{c}{m} \end{pmatrix}$$
$$\tilde{C} = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}$$

Remember that we do not consider generalized measurements and hence the output y is still simply the position. $\tilde{B}$ is not required for

Table 2: Parameters and hyperparameters of mass-spring-damper example.

m	k	c	α	σ_ω^2	σ_z^2	γ	p	κ	ρ
10	100	40	40	10	1	16	2	10	10^4

the filter nor controller, so we need not to define it. B will appear in the forward model.

For the precision matrices, firstly we have that (since $p = 2$)

$$V(\gamma) = \begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{2}\gamma \end{pmatrix}.$$

Assuming that the noises in ω are independent:

$$\Pi_\omega = \Sigma_\omega^{-1} = \begin{pmatrix} \sigma^2_{\omega_x} & 0 \\ 0 & \sigma^2_{\omega_{\dot{x}}} \end{pmatrix}^{-1}$$

In this example we give both noises in ω the same variance σ^2_ω, so $\Sigma_\omega = \sigma^2_\omega I_2$. Then, from Equation 12:

$$\Pi_{\tilde{\omega}} = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{2}\gamma \end{bmatrix}^{-1} \otimes \frac{1}{\sigma^2_\omega} I_2$$

Because generalized motions are not used in the measurement noise,

$$\Pi_z = \Sigma_z^{-1} = \frac{1}{\sigma^2_z}.$$

Since $p = 2$ and $n = 2$ (the state dimension) we have that

$$\mathcal{D} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \otimes I_2 = \begin{bmatrix} 0 & I_2 \\ 0 & 0 \end{bmatrix}$$

For the control prior, remember that x_{eq} is constant (and again $p = 2$) so that

$$\mu_{eq} = \begin{bmatrix} x_{eq} & \dot{x}_{eq} \end{bmatrix}^\top = \begin{bmatrix} \alpha & 0 & 0 & 0 \end{bmatrix}^\top$$

In addition, we do not consider a pole placement term $\tilde{K}$, so from Equation 29 then

$$\xi = -\tilde{A}\mu_{eq} = \begin{bmatrix} 0 & \alpha\frac{k}{m} & 0 & 0 \end{bmatrix}^\top$$

Observe that the same can be obtained by setting the input $u_{eq} = \alpha k$, since then $Bu_{eq} = \alpha\frac{k}{m}$. This shows that it does not make much of a difference whether we see xi as a prior on the states $\xi = E[\tilde{A}\tilde{x}]$ or on the control $\xi = E[Bu]$.

Finally, for the forward model we get

$$G = -CA^{-1}B = \frac{1}{k}$$

which is the inverse spring constant. This represents the relation between u (the force) and y (the position) in steady state: $y = u/k$.

8.3 SIMULATION

To simulate the system the MATLAB function `lsim` will be used. This function takes input of the following form:

```
ycl = lsim(system,t,ucl,x0).
```

To be compatible with this function, a closed loop system was defined in Section 7.4. The idea was to connect the control signal of the agent to the generative process and to connect the output of the generative process to the agent. In short, the result is a new state space system with state $x_{cl} = \begin{pmatrix} x & \mu & u \end{pmatrix}^\top$ and input $u_{cl} = \begin{pmatrix} \xi & \omega & z \end{pmatrix}^\top$:

$$\dot{x}_{cl} = A_{cl}x_{cl} + B_{cl}u_{cl} \tag{39}$$

where

$$A_{cl} = \begin{pmatrix} A & 0 & B \\ \kappa\tilde{C}^\top\Pi_z C & \mathcal{M} & 0 \\ -\rho\hat{G}^\top\Pi_z C & \rho\hat{G}^\top\Pi_z\tilde{C} & 0 \end{pmatrix},$$

$$B_{cl} = \begin{pmatrix} 0 & I & 0 \\ \kappa(\mathcal{D}-\tilde{A})^\top\Pi_w & 0 & \kappa\tilde{C}^\top\Pi_z \\ 0 & 0 & -\rho\hat{G}^\top\Pi_z \end{pmatrix},$$

in which for cleanliness we defined

$$\mathcal{M} = \mathcal{D} - \kappa(\mathcal{D}-\tilde{A})^\top\Pi_{\tilde{w}}(\mathcal{D}-\tilde{A}) - \kappa\tilde{C}^\top\Pi_z\tilde{C}$$

For inspection purposes an output equation is added which measures the complete state:

$$y_{cl} = C_{cl}x_{cl}$$

where $C_{cl} = I_{(n+np+m)}$. This is stored in a state space structure as:

```
system = ss(Acl,Bcl,Ccl,0);
```

After defining the input signals in u_{cl}, time vector `t` and initial state $x_{cl}(0)$, the function `lsim` can be called with the state space model from Equation 39. The code for this simulation is listed in Section B.3.

The results of the simulation are shown in Figure 6. Active Inference achieves control as desired. The initial strange behavior in μ_x is due to the trade-off between minimizing ε_y and ε_μ in the filter. Namely, the minimizing ε_y drives μ_x towards x whereas minimizing ε_μ drives μ_x towards the equilibrium value defined in ξ.

The mean of the free energy settles at approximately 1, an explanation is provided in Section A.4. Note that the settling time cannot

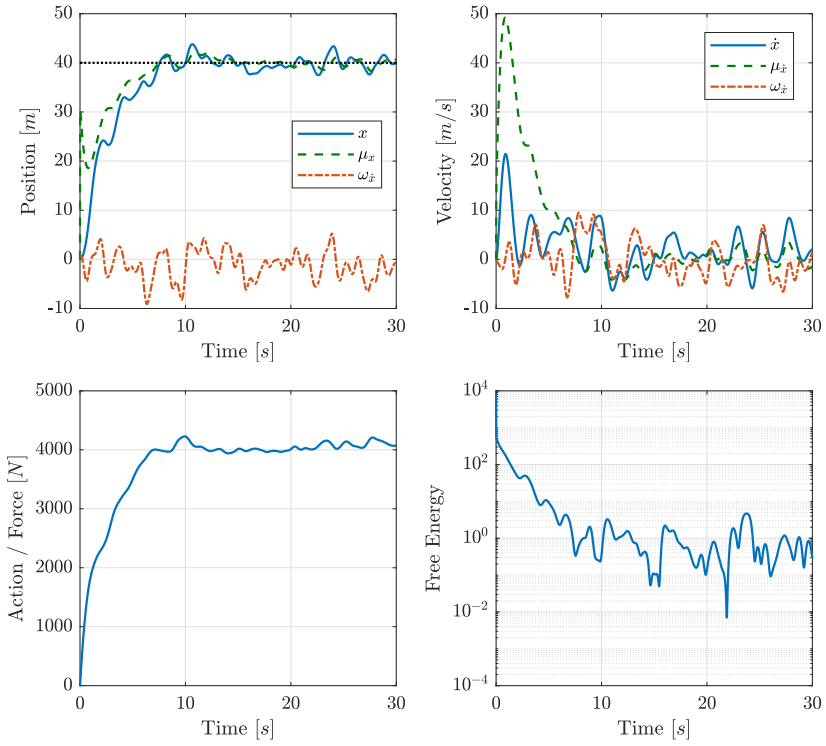


Figure 6: Top row: The state responses and noises of the mass-spring-damper simulation. The system settles in around 10 seconds to the desired equilibrium $x = [40, 0]$. Bottom row: The control signal (i.e. force exerted on the mass) and free energy of the mass-spring-damper simulation. Initial state $x_{cl}(0) = 0$. Time-step used: `dt=0.01`. Parameters listed in Table 2. (To allow for exact replication: The MATLAB (R2017a) random generator has been set to `rng(1)`)

be improved much by tuning the learning rates κ or ρ, as these have been optimized (manually) already. Improvement could be possible by changing several other parameters (p, ξ, and in principle also $\Pi_{\tilde{\omega}}$, Π_z). Then, it is of course interesting to study the effects of the tuning parameters and compare the performance with conventional setups. This is the subject of the upcoming part, where the performance of the algorithm will be further analyzed and several primers for theoretical studies are provided.

Part III

PERFORMANCE ANALYSIS

This part consists of several chapters that make a start in analyzing the performance of Active Inference in the LTI state space setting, to partially answer the third research question: *How does Active Inference perform?* First, the effects of changes in different (hyper)parameters is studied in Chapter 9. This is followed by Chapter 10, which is a primer in the study of the comparison between Active Inference and optimal control. A primer on possible stabilization and tracking methods is presented in Chapter 11.

9 PARAMETER TUNING

Several parameters in the Active Inference agent can still be tuned to obtain satisfactory performance. This chapter studies the effect of the free parameters on performance and stability. The first section, Section 9.1, studies how changes in the learning rates of the gradient descents influence the responses. The section afterwards, Section 9.2, shows how the embedding order changes the behavior of the closed loop system. Finally, Section 9.3 considers what happens when the agent's belief about the noise does not match the true noises, i.e., changes in the precision matrices.

9.1 LEARNING RATES

The first an most obvious tuning freedom in the state space Active Inference algorithm is in the learning rates ρ and κ of the gradient descents. Naturally, changing these parameters affects the rate of descent and as such we can predict that the influence will show mostly in how fast the agent achieves the desired equilibrium.

In all of this chapter we will consider a simple first order system, which can be thought of as a damped mass excited by an external force that is the control of the agent[1]:

$$\begin{aligned} \dot{x} &= -\tfrac{d}{m}x + \tfrac{1}{m}u + \omega \\ y &= x + z \end{aligned} \tag{40}$$

where $d = 800$, $m = 10^3$. The variances of the noises are $\sigma_\omega^2 = \sigma_z^2 = 10$ and the desired equilibrium will be $x_{eq} = 40$. For this section specifically, the smoothness parameter is $\gamma = 200$.

There are a lot of experiments that can be conducted to study the effects of the learning rates. We provide one interesting feature that has been discovered, regarding the ratio between the learning rates:

$$c = \frac{\kappa}{\rho}$$

Figure 7 shows the effect of changes in c for constant ρ. Clearly, decreasing the ratio (decreasing κ for constant ρ) makes the control faster, but also sensitive to the noise. Due to that, when decreasing the ratio too much, the closed loop becomes unstable as shown in Figure 8. This is confirmed by a right-half-plane eigenvalue in A_{cl}. It is unusual that a smaller step-size in a gradient descent makes a system unstable. Only decreasing κ makes the controller "stronger"

1 We will not consider units as it carries no actual meaning for the results

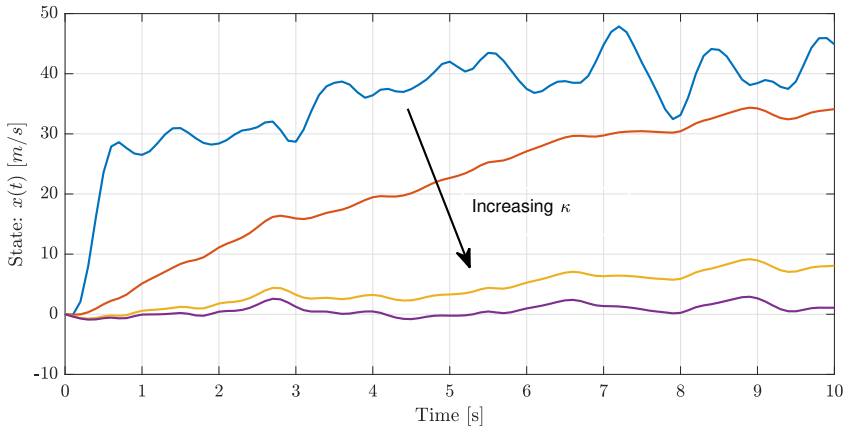


Figure 7: Effect of changes in the ratio between the learning rates on the state response. $\rho = 3 \cdot 10^8$ and c ranges from 10^{-7} to 10^{-1} with steps of 10^2.

with respect to the filter, which would explain an increased sensitivity to the noise ω or z.

Similar experiments have been conducted by first finding an optimal set of learning rates, which – for the system considered here – were $\rho = 3 \cdot 10^8$ and $\kappa = 10^3$, the optimal ratio is thus $c \approx 3 \cdot 10^{-6}$. With this ratio different sets of learning rates have been tested, where ρ ranged from 1 to 10^{16}. Listing all the results would take too much space and since the results are as one would expect, we simply mention the conclusion here: Decreasing the learning rate ρ will never make the system unstable as long as c is kept constant. The closed loop performance merely deteriorates. Increasing ρ – with constant c – will make the system unstable at some point.

A final note on the learning rates with respect to the state dimension of μ: In this thesis we have considered scalar learning rates and one might argue that this could be extended to a matrix form because the state dimension of the gradient descent is mostly larger than one. However, the effect of different learning rates at different orders of motion is already captured in the precision matrices, which contain information about the amount of confidence in each order of motion.

A final important observation is that the learning rates – and so c as well – have to be re-tuned for every system, optimal values depend on the system matrices. Also for changes in the variances of the noises other optimal values arise. This is not convenient, because it means that it is hard to find a standardized approach that works in a wide range of settings.

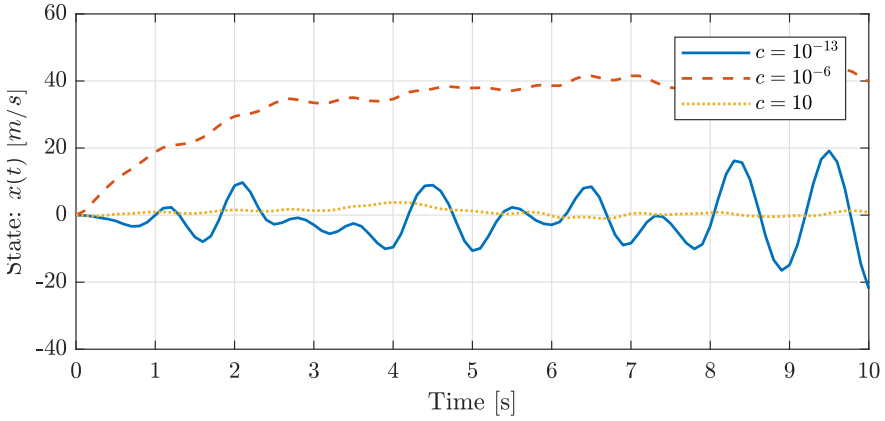


Figure 8: By decreasing the ratio c too much, the filter becomes so weak that the controller is trying to control the noise, making the closed loop unstable. $\rho = 3 \cdot 10^8$ in all cases.

Table 3: Parameters and hyperparameters of the simulation in Figure 9.

σ_ω^2	σ_z^2	γ	p	κ	ρ	x_{eq}	m	d
100	10	32	4	10^3	$8 \cdot 10^8$	40	10^3	800

9.2 EMBEDDING ORDER

The second degree of freedom in the algorithm is in the embedding order p, which is the order of the generative model. We should expect that above $p = 6$ not much changes, as argued in [28].

The first experiment that has been conducted inspects the behavior of the higher order motions in the agent. Parameters as used for this experiment are listed in table Table 3. Figure 9 shows the resulting motions up to 4th order.

As expected, all motions except for the first converge to zero. A second observation is that the amount of noise in the motions seems to decrease at higher orders as well as the amount of "movement" per se. These phenomena are due to the structure of the precision matrix $\Pi_{\tilde{\omega}}$. The precision at higher order motions quickly falls to zero and thus these motions are updated with much smaller steps in the filtering scheme. A final observation lies in the physical interpretation of the motions. Note that the derivative of μ_x is not equal to the motion $\mu_{x'}$. As explained in Chapter 3, this is to be expected.

The effect of changes in the embedding order on the system response has also been studied. A first observation is that the effect of changes also depends on other parameters, such as the smoothness of the noise. This shows in the difference between Figure 10 and Figure 11, between which the only difference is the roughness parameter,

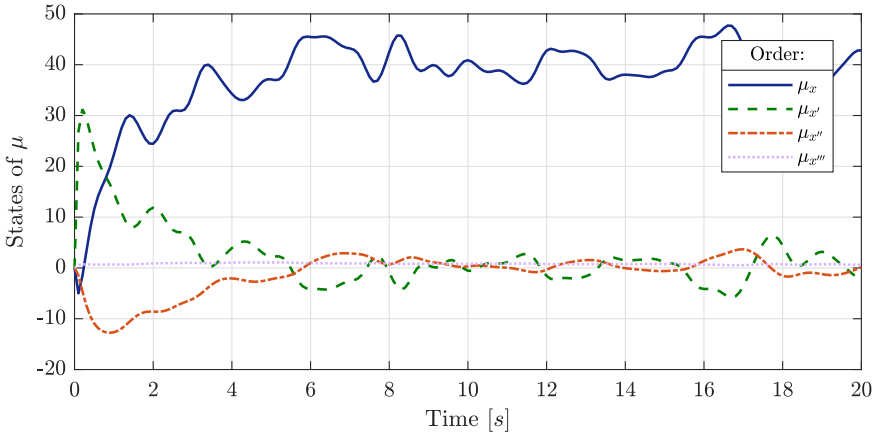


Figure 9: Behavior of higher order motions in the agent applied to Equation 40 with parameters from Table 3. Higher order motions are less excited and contain less noise than lower order motions.

$\gamma = 32$ and $\gamma = 0.89$ respectively. It is remarkable that adding higher order motions creates a smoother response when the noise is smooth. An explanation for this has not yet been discovered. Changes in the variances of the noises and the learning rates have similar effects on the difference between different embedding orders. In general, the difference between $p = 2$ and higher orders is large but above $p = 2$ the difference in responses is much smaller. Also, p can be increased indefinitely and will never make the closed loop unstable.

9.3 PRECISION MATRICES

A final, less obvious parameter set that can be tuned is the set of precision matrices $\Pi_{\tilde{\omega}}$ and Π_z. Of course, the belief of the agent about the noises does not necessarily need to be equal to the true noises and hence one could argue we are free to change $\Pi_{\tilde{\omega}}$ and Π_z.

The effect on the state response for scalar changes in $\Pi_{\tilde{\omega}}$ and Π_z is shown in Figure 12 and Figure 13 respectively. The parameters used for the simulations are listed in Table 3. The effect of changes in either of the matrices is similar, but not equal.

Increasing $\Pi_{\tilde{\omega}}$ makes the response faster but with more oscillations. An explanation for this is that the belief of very precise noises puts more emphasis on the generative model, which contains the tracking error $\tilde{A}\boldsymbol{\mu} - \boldsymbol{\xi} = \tilde{A}\mu - \tilde{A}\boldsymbol{\mu}_{eq}$. The exact cause for the increase in oscillations is unclear at this point, but it is likely due to the fact that noises propagate into the estimation through the observations. Because the gain $\Pi_{\tilde{\omega}}$ is large, these noises are amplified.

Most remarkable is that increasing Π_z only has effect up to a certain point. It is to be expected that decreasing Π_z makes the response

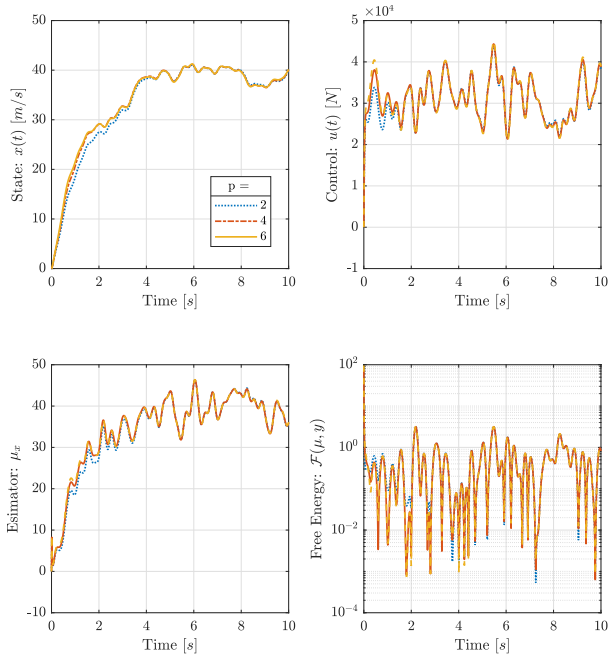


Figure 10: The system response for different embedding orders with parameters from Table 3, except $\sigma_\omega^2 = 10$. Not much difference exists for rough noises, because higher order precisions quickly fall to zero and do not influence the response a lot.

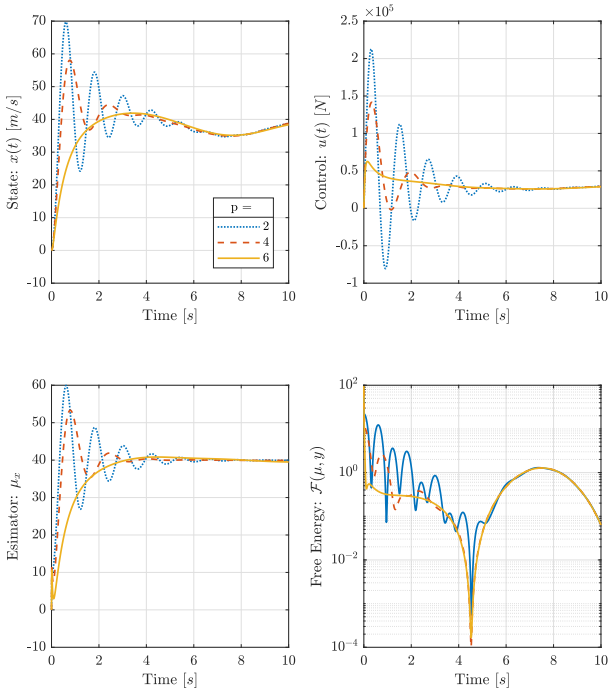


Figure 11: The system response for different embedding orders with parameters from Table 3 except $\sigma_\omega^2 = 10$ and $\gamma = 0.88$ (very smooth noise). Higher orders do contribute significantly due to the low roughness and this causes a lot of difference between embedding orders.

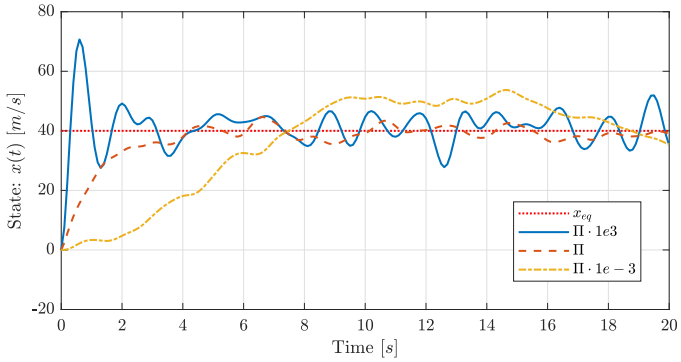


Figure 12: The effect of scalar changes in the belief about the state noise variance, expressed as scaling $\Pi_{\tilde{w}}$, on the state response. With belief of higher precision the system responds faster, with belief of lower precision the system responds slower.

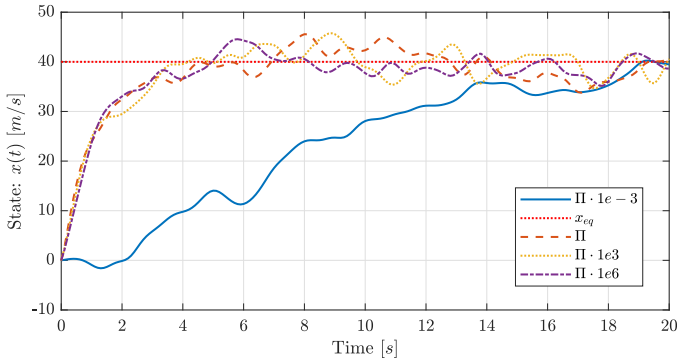


Figure 13: The effect of scalar changes in the belief about the observation noise variance, expressed as scaling Π_z, on the state response. With belief of lower precision the system responds slower. Increasing the belief however has no significant effect.

slower, because Π_z is basically the gain of the controller. However, when increasing Π_z enough the response does not change much. This is most likely due to the slower filter dynamics becoming dominant. The rate at which μ_x converges to x_{eq} is determined by $\Pi_{\tilde{w}}$. The controller can only be as fast as the filter.

There seems to be an equivalence here with the Q and R matrix in optimal control, which can also be used to tune the response of the system in a similar way. Higher Π_z or allows for more control action to be used, yielding faster response. For $\Pi_{\tilde{w}}$ an increase puts more emphasis on the tracking error and thus also makes the system converge faster. This relation is the subject of the upcoming chapter.

10 EQUIVALENCE WITH OPTIMAL CONTROL

The free energy looks a lot like the quadratic cost functions used in optimal control. Also, in the previous chapter we have seen that there is an apparent similarity between the precision matrices and the Q *and* R *matrices used in optimal control. This chapter makes a start in the analysis of the exact relation between Active Inference and* LQG *control.*

10.1 SIMULATION: LQR VS. ACTIVE INFERENCE

It has been argued in [28] that DEM is equivalent to extended Kalman filtering for linear systems and Gaussian (Markovian) noises. Hence, it is natural to wonder whether Active Inference is equivalent to *Linear Quadratic Gaussian control* (LQG), which is a Kalman filter supplemented with an *Linear Quadratic Regulator* (LQR) controller. This section will show that it is possible to design an Active Inference controller (and estimator) that gives the same response as an LQR controller. We will use the same system as before,

$$\begin{aligned}\dot{x} &= -\tfrac{d}{m}x + \tfrac{1}{m}u + \omega\\ y &= x + z\end{aligned}$$

but with parameters as listed in Table 4.

The first important remark is that optimal control theory makes use of (Gaussian) Markovian noises and thus we need to consider Active Inference for $p = 1$ to make a proper mathematical comparison. As this is a special case of which the details are yet unclear, we will use $p = 2$ for simulations. Section A.5 explains the Active Inference filter for this setting and for $p = 1$. Note that in Active Inference the filter cannot be removed since the state estimation is required in the controller:

$$\dot{u} = -\rho G^\top \Pi_z (y - C\hat{x})$$

where we replaced μ with $\hat{x}$ to denote the standard state estimate. Removing the filter also removes the prediction error that the controller minimizes (because it removes the variable $\hat{x}$). Therefore, it seems that the separation principle does not apply to Active Inference.

We assume full state information ($C = I$) so that the Kalman filter is superfluous – as one usually does to transfer from LQG to LQR – which results in

$$\dot{u} = -\rho G^\top \Pi_z (x - \hat{x})$$

Table 4: Parameters and hyperparameters of the simulation in Figure 14.

σ_ω^2	σ_z^2	γ	p	κ	ρ	x_{eq}	m	d	Q	R
10	0	200	2	0.1	$8 \cdot 10^3$	40	10^3	80	$9 \cdot 10^3$	$8 \cdot 10^{-3}$

This controller is Active Inference applied to a system where $z \approx 0$, which corresponds to having full state information since $C = I$.

Next, we consider an LQR controller. More details about LQR control can be found in [76]. In short, the (infinite-horizon) LQR controller to track x_{eq} is

$$u = -Lx + u_{ffw}$$

where $L = R^{-1}B^\top P$ and P the solution of the algebraic ricatti equation

$$A^\top P + PA - PBR^{-1}B^\top P + Q = 0$$

and u_{ffw} is the feedforward control corresponding to x_{eq}:

$$u_{ffw} = (d + K)x_{eq}$$

for this case specifically. Using Q and R as specified in Table 4, the learning rates of Active Inference have been tuned to get the response of Active Inference as close as possible to that of the LQR controller. The result is shown in Figure 14. It is important to take note of the initial values. Namely, the initial control of the LQR system is nonzero: $u(0) = -Kx(0) + u_{ffw}$. However, the controller for Active Inference is an integrator which would usually start at initial zero state. The initial state of the Active Inference controller has been set to the same value as the LQR controller to obtain the responses shown. In this case, clearly the two are equivalent, which is very remarkable. The upcoming section provides a first look into the theoretical comparison of Active Inference and optimal control, i. e., LQG.

10.2 THEORETICAL PROOF: A PRIMER

A very interesting topic for further studies on Active Inference is the proof of equivalence between Active Inference and LQG, of which the existence has been hinted at by Friston [27, 28]. This section does not complete the proof, but provides a compact overview of the equations which must be proven to be equivalent. Also some other general observations are mentioned.

To properly compare the two paradigms we must assume Markovian noise, which is explained in Section A.5 for Active Inference. To improve the comparison, we will replace μ with the conventional estimate $\hat{x}$ and replace $\xi = Bu$ again. Since we are not working in generalized form anymore, this yields no other notational problems.

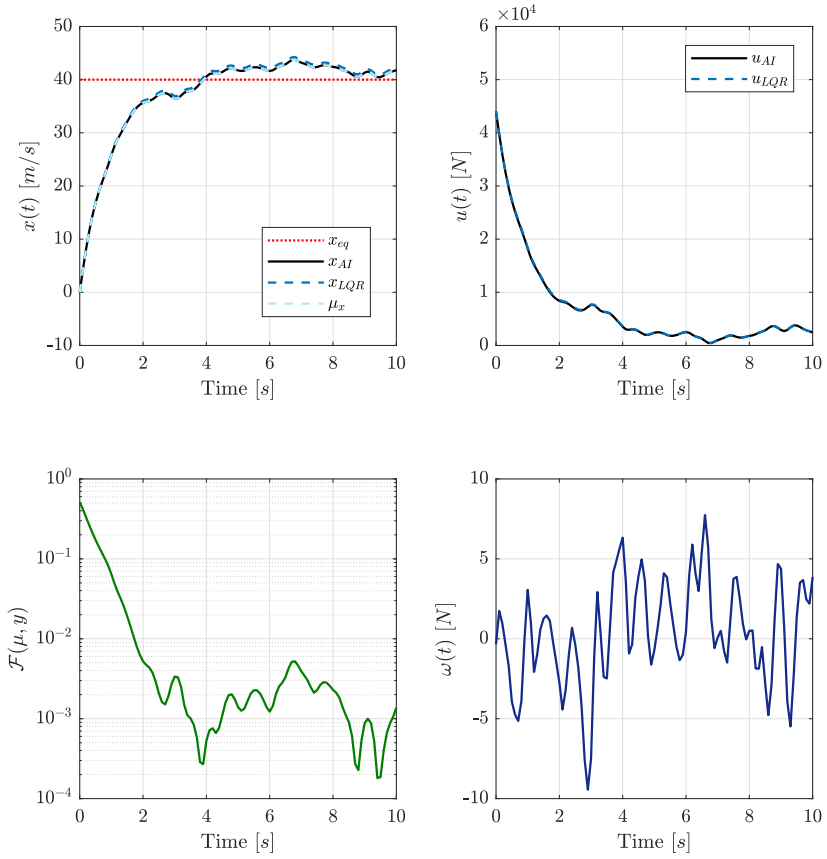


Figure 14: Active Inference tuned to show a response equivalent to that of the LQG controller. Parameters listed in Table 4.

A first observation is the different form of the cost function for LQR control and the free energy:

$$J = \int_0^\infty (x^\top Q x + u^\top R u) dt$$

versus

$$\mathcal{F}(\hat{x}, \hat{x}', x, u) = \frac{1}{2}\varepsilon_y^\top \Pi_z \varepsilon_y + \frac{1}{2}\varepsilon_x^\top \Pi_\omega \varepsilon_x$$

where

$$\varepsilon_y = x - \hat{x}$$
$$\varepsilon_x = \hat{x}' - A\hat{x} - Bu$$

The free energy does not contain an integral and considers errors instead of states and control. Moveover, LQG has a separate cost function for the Kalman filter and the LQR controller, whereas Active Inference only uses a single cost function. This is a fundamental difference that seemingly contradicts the equivalence.

Next, let us consider the filtering part. The Kalman-Bucy filter (continuous time Kalman filter) is [54]:

$$\dot{\hat{x}} = A\hat{x} + Bu + K\varepsilon_y$$

where optimal kalman gain $K = PC^\top R^{-1}$ is the solution of

$$AP + PA^\top - PC^\top R^{-1} CP + Q = 0$$

Observe the equivalence with the solution for the LQR gain, which is explained by the duality of the two problems [57]. For better comparison we substitute the optimal kalman gain:

$$\dot{\hat{x}} = A\hat{x} + Bu + PC^\top R^{-1}\varepsilon_y \tag{41}$$

The filter of Active Inference for $p = 1$ (with $\kappa = 1$) is:

$$\begin{aligned}\dot{\hat{x}} &= \hat{x}' + A^\top \Pi_\omega \varepsilon_x + C^\top \Pi_z \varepsilon_y \\ \dot{\hat{x}}' &= -\Pi_\omega \varepsilon_x\end{aligned}$$

Again, it does not seem that the two are equivalent, although it can be observed that the term in both filters multiplied with ε_y is similar. It looks like $R^{-1} = \Pi_z$ (and then correspondingly $Q^{-1} = \Pi_\omega$).

One possibility to bring us closer to the equivalence is to set $\dot{\hat{x}}' = 0$, which implies that $\varepsilon_x = 0$ i. e. $\hat{x}' = A\hat{x} + Bu$, which of course looks very familiar. Substituting this in the Active Inference filter yields

$$\dot{\hat{x}} = A\hat{x} + Bu + C^\top \Pi_z \varepsilon_y$$

which starts to look very similar to the Kalman filter Equation 41, except that the covariance matrix P does not appear. The equivalence has not been proven yet further than this point.

Considering the controllers, the LQR controller is

$$u = -R^{-1} B^\top P\hat{x} + u_{ffw}$$

and the Active Inference controller is

$$\dot{u} = -\rho G^\top \Pi_z (x - \hat{x})$$

The two do not look similar. The most critical difference is the integral present in the Active Inference controller. Also, Active Inference controls an error whereas LQR is a state feedback.

Concluding this chapter, the equivalence is not clear form a mathematical point of view although simulations suggest it exists, at least for the scalar case. This chapter should not be considered as a concrete proof, but it provides a lead for further studies.

11

STABILITY AND TRACKING

In this chapter we study two of the most basic problems in control theory regarding state space systems: Stability and tracking. Like any other conventional state space controller it would be required for Active Inference to be able to perform these tasks if it is to be considered useful. This chapter does not provide complete solutions to all of the problems posed, but states the answers for as far as these have been worked out at this point.

11.1 STABILIZATION: AN LMI APPROACH

Again, let us consider the generative process:

$$\dot{x} = Ax + Bu + \omega$$
$$y = Cx + z$$

and let us assume that the A matrix of the generative process is unstable, i. e., has eigenvalue(s) in the right half of the complex plane. A conventional approach would be to design a state space controller using state feedback (pole placement or LQR for example [23]) and a filter when the state is not observed directly. The two can be combined based on the separation principle [57].

In Active Inference however, this problem is different. Namely, the controller is predefined, except for the learning rate ρ:

$$\dot{u} = -\rho G^\top \tilde{C}^\top \Pi_z (y - \tilde{C}\mu)$$

All the design possibilities are in the filtering part of the agent, more specifically in the prior variable ξ:

$$\dot{\mu} = \mathcal{D}\mu - \kappa[-\tilde{C}^\top \Pi_z (y - \tilde{C}\mu) + (\mathcal{D} - \tilde{A} - \partial_\mu \xi)^\top \Pi_{\tilde{\omega}} (\mathcal{D}\mu - \tilde{A}\mu - \xi)]$$

This variable provides the flexibility to change the generative model in any way desirable. The question is, how to choose the prior ξ such that we obtain desirable behavior, or, in this case, stabilization? Remember that the full prior was defined as

$$\xi = \mathcal{D}\mu_{eq} - (\tilde{A} + \tilde{K})\mu_{eq} + \tilde{K}\mu$$

In this section we will consider the loop shaping part of the prior. So, much like standard state feedback controllers, we choose

$$\xi = \tilde{K}\mu$$

where similar to $\tilde{A}$, $\tilde{K} = I_p \otimes K$, such that the state equation of the generative model becomes

$$\mathcal{D}\mu = (\tilde{A} + \tilde{K})\mu$$

We provide more freedom by omitting the tensor product requirement for $\tilde{K}$. Namely, the approach presented below is not guaranteed to provide an existing solution for the tensor product form, thus the free form for $\tilde{K}$ will be assumed. The above equation looks a lot like a pole placement controller, but then in a generalized form due to the use of generalized motions.

The following sections provide a possibility to design a stabilizing filter, based on *Linear Matrix Inequality* (LMI) optimization. We start by defining the optimization problem and subsequently solve the feedback matrix $\tilde{K}$ from the optimization result.

11.1.1 *The optimization problem*

Remember the closed loop matrix

$$A_{cl} = \begin{pmatrix} A & 0 & B \\ \kappa\tilde{C}^\top \Pi_z C & \mathcal{M} & 0 \\ -\rho\hat{G}^\top \Pi_z C & \rho\hat{G}^\top \Pi_z \tilde{C} & 0 \end{pmatrix}$$

where $\mathcal{M}$ contains the generative model, which we are free to shape. In other words: we are free to define $\mathcal{M}$. Hence, stabilization is equivalent to finding a matrix $\mathcal{M}$ and P such that

$$A_{cl}(\mathcal{M})^\top P + P A_{cl}(\mathcal{M}) \prec 0$$

where P is some positive definite matrix. This is the Lyapunov inequality [8]. If there exists a positive definite P such that the inequality holds, A_{cl} is stable. However, the problem now is that both A_{cl} and P can be optimized such that the inequality holds. To solve this problem we can turn to a sequential optimization approach. First, we relax the Lyapunov inequality:

$$A_{cl}(\mathcal{M})^\top P + P A_{cl}(\mathcal{M}) \preceq \delta I$$

where $\delta \in \mathbb{R}$ and I is an identity matrix of appropriate size $(n + np + q)$. Now, assuming as initial condition the standard form of $\mathcal{M}$ where $\tilde{K} = 0$, we can check if the closed loop is stable by solving:

$$\min_{\delta, P} A_{cl}(\mathcal{M})^\top P + P A_{cl}(\mathcal{M}) \preceq \delta I$$

This optimization can be performed with software like YALMIP[1] and an *Semi-Definite Programming* (SDP) solver. If there exists a solution for

1 https://yalmip.github.io/

Table 5: Parameters and hyperparameters of the simulation in Figure 15.

σ_ω^2	σ_z^2	γ	p	κ	ρ	x_{eq}	m	d
10	10	50	2	$1 \cdot 10^2$	$2 \cdot 10^8$	40	$1 \cdot 10^3$	800

P where $\delta < 0$, the problem is solved since then the actual Lyapunov inequality also holds. If this is not the case – the optimal δ is positive – we proceed to solve

$$\min_{\delta, \mathcal{M}} A_{cl}(\mathcal{M})^\top P + P A_{cl}(\mathcal{M}) \preceq \delta I$$

using the value of P found in the previous step. Now, if the solution is some $\mathcal{M}$ and $\delta < 0$, the problem is solved. If this is not the case, δ will at least be smaller than before, since we took the same value for P but optimized $\mathcal{M}$ together with δ now. With the current value for $\mathcal{M}$ we can repeat the first optimization, and then the second, until we find negative value for δ. The corresponding value for $\mathcal{M}$ is a stabilizing solution for the closed loop. The final problem is to re-extract the feedback matrix $\tilde{K}$ from $\mathcal{M}$. This is the topic of the next section.

Note that it is in fact possible for Active Inference to stabilize systems. An example is provided in Figure 15, where the system was

$$\dot{x} = \tfrac{d}{m} x + \tfrac{1}{m} u + \omega$$
$$y = x + z$$

and the prior state feedback matrix was $K = -2\frac{d}{m}$ and $\tilde{K} = I_p \otimes K$. Other variables for the simulation are summarized in Table 5. For this system the approach presented above should be guaranteed to provide a stabilizing solution. It could be interesting for future research to prove whether or not Active Inference is able to stabilize any stabilizable system, like pole placement control.

11.1.2 *Solving the state matrix*

When using the state feedback form for ξ we have that

$$\mathcal{M} = \mathcal{D} - \kappa(\mathcal{D} - \tilde{A} - \tilde{K})^\top \Pi_{\tilde{\omega}} (\mathcal{D} - \tilde{A} - \tilde{K}) - \kappa \tilde{C}^\top \Pi_z \tilde{C}$$

where $\mathcal{M}$ is known to stabilize the closed loop, as explained in the previous section. Observe that the only unknown is $\tilde{K}$. First, by rearranging some terms, the equation can be written as

$$\kappa^{-1}(\mathcal{D} - \mathcal{M}) + \tilde{C}^\top \Pi_z \tilde{C} = (\mathcal{D} - \tilde{A} - \tilde{K})^\top \Pi_{\tilde{\omega}} (\mathcal{D} - \tilde{A} - \tilde{K}) \tag{42}$$

Next, since $\Pi_{\tilde{\omega}}$ is an inverse covariance matrix, it is positive (semi)-definite [2]. Let us assume it is positive definite because all noises

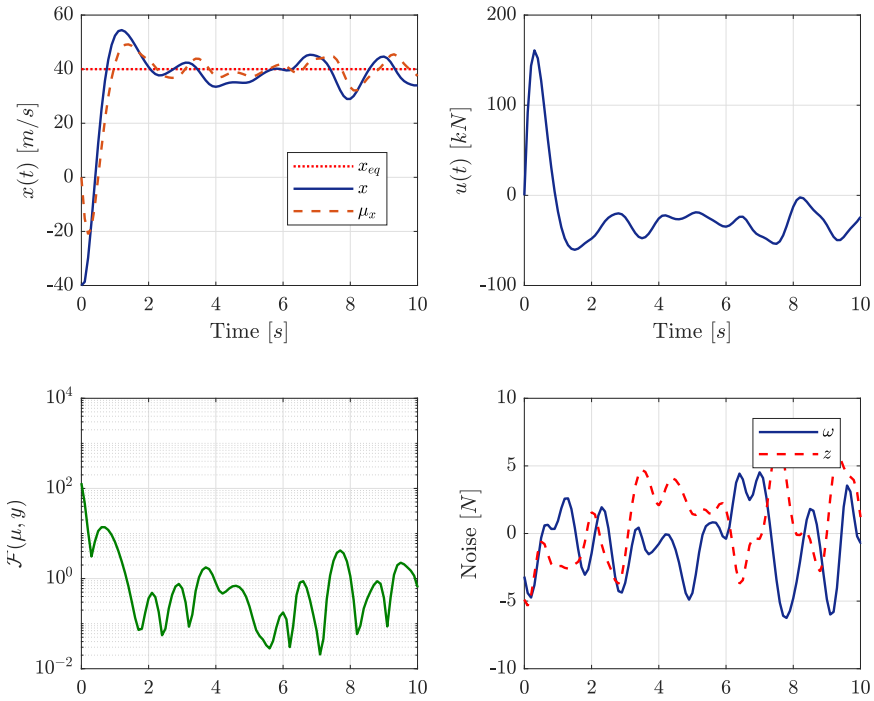


Figure 15: The state response, control, free energy and noises for Active Inference on an unstable system with a stabilizing generative model. The system settles in around 2 seconds to the desired equilibrium. Simulation parameters listed in Table 5.

have at least an infinitely small variance. This implies that the precision matrix can be decomposed as

$$\Pi_{\tilde{\omega}} = Q^\top Q$$

where the simplest option is to take $Q = \Pi_{\tilde{\omega}}^{\frac{1}{2}}$ such that

$$\Pi_{\tilde{\omega}} = \Pi_{\tilde{\omega}}^{\frac{1}{2}\top} \Pi_{\tilde{\omega}}^{\frac{1}{2}}$$

Now since $\Pi_{\tilde{\omega}}^{\frac{1}{2}}$ is the root of a positive definite matrix, it is also positive definite and thus invertible and symmetric [55].

Substituting this form makes the left hand side of Equation 42 quadratic. Thus by taking the root on both sides:

$$\left(\kappa^{-1}(\mathcal{D} - \mathcal{M}) + \tilde{C}^\top \Pi_z \tilde{C}\right)^{\frac{1}{2}} = \Pi_{\tilde{\omega}}^{\frac{1}{2}}(\mathcal{D} - \tilde{A} - \tilde{K})$$

Then

$$\sqrt{\Pi_{\tilde{\omega}}^{-1}\left(\mathcal{D} - \mathcal{M} + \kappa \tilde{C}^\top \Pi_z \tilde{C}\right)} = (\mathcal{D} - \tilde{A} - \tilde{K})$$

from which it follows that

$$\tilde{K} = \mathcal{D} - \tilde{A} - \sqrt{\Pi_{\tilde{\omega}}^{-1}\left(\mathcal{D} - \mathcal{M} + \kappa \tilde{C}^{\top} \Pi_z \tilde{C}\right)} \tag{43}$$

Note that it is not an option in this framework to define $\tilde{K}$ as optimization variable, or K when $\tilde{K} = I_p \otimes K$. Namely, $\mathcal{M}$ is quadratic in $\tilde{K}$ and thus LMI techniques are not sufficient to solve the problem. By the approach proposed here it is not guaranteed that the terms on the left hand side of Equation 43 can be written as $I_p \otimes K$.

11.2 TRACKING

In this section we will consider the performance of Active Inference in a tracking scenario, the second part of the prior beside the loop shaping. The algorithm will be applied to the same system as before, but a stable version. So:

$$\begin{aligned} \dot{x} &= -\tfrac{d}{m}x + \tfrac{1}{m}u + \omega \\ y &= x + z \end{aligned} \tag{44}$$

with parameters and hyperparameters as from Table 6.

For completeness, the full formulation of the prior was

$$\xi = \mathcal{D}\mu_{eq} - \kappa(\tilde{A} + \tilde{K})\mu_{eq} + \tilde{K}\mu$$

which was said to be able to perform loop shaping (through $\tilde{K}$) and tracking (through μ_{eq}). Loop shaping, or actually stabilization, has been considered in the previous section. Here we will consider only tracking and thus reformulate the prior as:

$$\xi = (\mathcal{D} - \tilde{A})\mu_{eq}$$

To verify that this prior makes the filter converge to the desired reference, a pure generative model filter has been simulated. This means that we update μ as

$$\dot{\mu} = \mathcal{D}\mu - (\mathcal{D} - \tilde{A})^{\top}\Pi_{\tilde{\omega}}(\mathcal{D}\mu - \tilde{A}\mu - \xi)$$

where it is assumed that ξ is not a function of μ. The result for a sinusoid reference $r(t) = 40\sin(3t/2)$ switched on at a random time is shown in Figure 16. Clearly, the filter works as expected.

The next step is to apply this approach in closed loop. The resulting response is shown in Figure 17. Concluding from this, the tracking approach presented above works as desired. Future work could rewrite this scenario to disturbance rejection. Also convergence properties of the filter are interesting, as it is not always stable in closed loop.

Table 6: Parameters and hyperparameters of the simulations in Section 11.2.

σ_ω^2	σ_z^2	γ	p	κ	ρ	m	d
10	10	32	2	10^3	$8 \cdot 10^8$	10^3	800

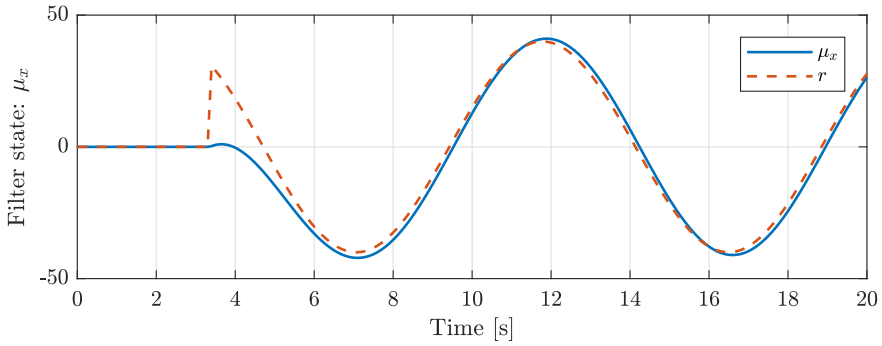


Figure 16: State estimation μ_x for a pure generative model scenario. As required, the filter tracks the reference. Note that there is a small error/lag. This is error is controlled by the learning rate κ of the gradient descent. Parameters used as from Table 6 except $\kappa = 10$.

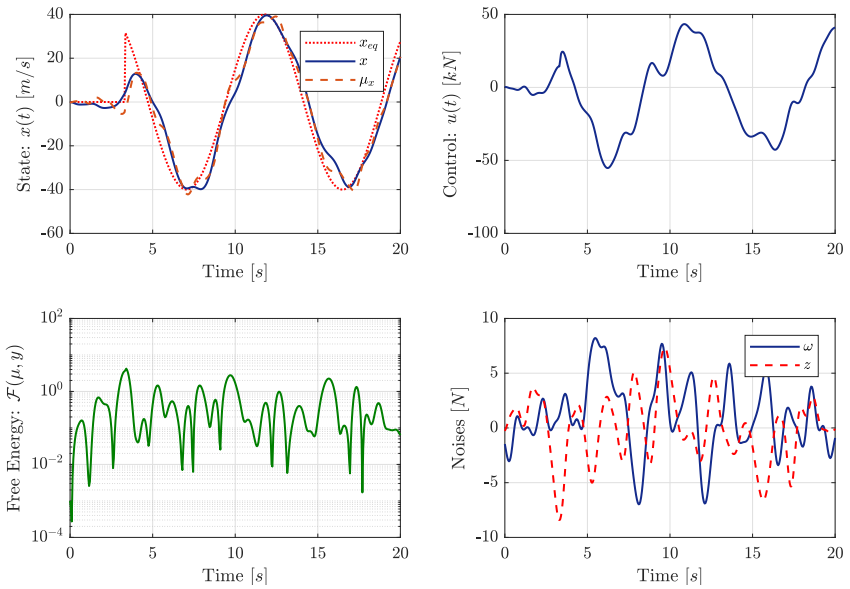


Figure 17: Closed loop tracking performance of Active Inference. The reference is tracked quite well, with a small lag and of course some fluctuating error due to noises. Parameters used as from Table 6.

Part IV

CONCLUSIONS

12 SUMMARY AND CONCLUSIONS

In this chapter we will first review the contents of this thesis from a high level perspective. Subsequently, the research questions as posed in the introduction will be answered, in which we consider a separate view for control theoretic research and robotics research.

12.1 SUMMARY

This thesis started with a part exposing the Free Energy Principle and its considerations, culminating in the formulation of Active Inference. Several esoteric concepts that receive little attention in the scientific literature have been explicitly elaborated: Generalized motions, the temporal variance matrix and the forward dynamic model. Some attention has been spent on mentioning the existing paradigms in engineering to which Active Inference is related. Active Inference relates to many theories; from the principle of least action to reinforcement learning, variational inference and Kalman filtering.

Afterwards, in the second part, Active Inference has been applied on a stochastic LTI state space system, to expose several remarkable differences with conventional control algorithms and to increase our understanding of the algorithm. The scope has been limited to state estimation, but in principle Active Inference provides a means to perform (unsupervised) estimation of states, parameters and hyperparameters. The resulting Active Inference algorithm comprises of two equations, comparable to a filter and controller as usual in control science. Both are a gradient descent on the free energy.

In the third part a start has been made in the analysis of the two equations constituting *state restricted LTI Active Inference* (to indicate the scope limit to state estimation and application to LTI state space systems). Beside studies of the effects that tuning parameters have on the response, a start in examining the relation with LQG has been made. It appears Active Inference can achieve equivalent response, which lead to the question if there is a theoretical proof of this equivalence as well. There should be according to Friston [27, 28]. Also, a possible approach to design a stabilizing controller has been presented and it has been verified that Active Inference is indeed able to stabilize at least one unstable system. Additionally, it has been shown that the tracking performance is as desired.

12.2 CONCLUSIONS

The research goal of this thesis was to provide a detailed exposition of Active Inference in an engineering context, which would be achieved by answering three research questions. This section revisits the research questions and presents the conclusions that we can draw regarding the research questions. Since this thesis is more control theoretic, but the long term goal is an application to robotics, the conclusions are provided from two different perspectives: a control theoretic perspective and a robotics perspective.

To conclude on the research goal: In this thesis, specific attention has been spent on elaborating several intricate concepts in Active Inference that are overlooked in literature. Additionally, these concepts have been clarified through application on simple LTI state space systems. Therefore, this thesis is to date the most detailed and comprehensible elaboration of Active Inference. Thus, we can conclude that the research goal has been achieved.

12.2.1 *Control Theoretic perspective*

The first research question was: *To which existing paradigms is Active related?* It became apparent that Active Inference has relations with many engineering principles (variational inference, filtering, principle of least action, and many more) and several differences with conventional approaches have been identified.

The first main difference between Active Inference and common methods, in the generative process, is that noises are non-Markovian, i. e., do not have the Markov property. This means we can make use of generalized motions to improve the state estimation. A second important contrast is that in Active Inference the reference appears in the filter. The controller is only a simple integrator that minimizes an error. The key in Active Inference is to design the filter such that the agent performs the required task, which can be tracking, regulation or stabilization.

Also interesting is the fact that due to the use of generalized motions, the generative model is a probabilistic instantaneous mapping instead of a dynamic model. The gradient descent on the free energy then yields an unconventional way to perform state estimation. Moreover, the same cost function – the free energy – is used for the controller. It seems that the separation principle does not apply for Active Inference, as the same cost function is used in both the filter and controller.

The second research question was: *What is the LTI state space formulation of Active Inference?* This is essentially answered by two equations: the filtering and controller equation, which are summarized in Section 7.5. In essence, both are a (generalized) gradient descent on the

free energy, which in turn is a squared sum of prediction errors. This implies that the Active Inference algorithm is a form of prediction error minimization.

The last question was to study how the algorithm performs. A start to answer this broad question has been made by examining the effect of tuning parameters. Tuning the learning rates of the gradient descents is the bottleneck in achieving a satisfactory performance. Also, the equivalence with LQG has been hinted at. It seems that Active Inference can achieve similar performance as LQG, if not equivalent. Furthermore, we have proposed a method to stabilize unstable systems using Active Inference and we have shown that Active Inference is able to stabilize at least one unstable system. Finally, we have shown that Active Inference can track a dynamic reference.

12.2.2 *Robotics perspective*

In this thesis, the focus has been on the control theoretic analysis of Active Inference, to increase understanding of the framework from an engineering perspective. This is required before the theory can be applied on more complex systems such as robots. The relevance of Active Inference for robotics can at this point be found in more high level observations. This mainly concludes the first research question: *To which existing paradigms is Active Inference related?* The answer to this question in a robotics context will tell us what Active Inference adds to the current methods that are used to control robots. At this point, two interesting areas can be identified:

Firstly, the integration of action and perception might provide inspiration for robotic systems which use a simpler architecture to close the loop. Whereas now robots use very separated vision and motion planning systems for example, Active Inference could provide a way to integrate these modules in a more efficient scheme.

Secondly, the inference part of Active Inference is an extension on standard variational inference by using generalized motions. Combining this with recent advancements in variational inference research, it could be possible to use Active Inference for designing systems that can perform inference on very complex dynamical systems, i. e., provide robots a "better" understanding of their environment.

13 RECOMMENDATIONS

This chapter provides several recommendations and proposals for further studies of Active Inference in engineering context. In short, it is important to first understand the technical details of Active Inference in order to judge whether or not there are aspects that could yield improvements. Two areas for improvement are most probable: (a) Control systems with more accurate filtering (by using generalized motions) and better performance on complex systems, or (b) advances in (variational) inference systems.

The first step for further studies is to relate the state space formulation of Active Inference to existing paradigms – like LQG, the principle of least action, Lyapunov theory – in detail. The bottleneck in here is to understand how to remove generalized coordinates from the framework. Karl Friston's view on this has been presented in Section A.5. This increases control theoretic understanding, so that it can be judged whether or not Active Inference provides improvement over existing methods. The major remaining question is whether the use of generalized motions yields better performance (more accurate filtering, faster settling times etc.) than conventional controllers. Also the design of the prior variable is an interesting topic.

When the understanding of Active Inference as presented in this thesis is at a sufficient level, several extensions can be made. First is to include the learning of parameters and hyperparameters. Secondly, the algorithm can be applied on nonlinear systems. This will pave the way to proceed to more complex robotic applications, as most robots are nonlinear. Finally, the extension to hierarchical models can be made [34]. This might allow to control even more complex systems. The main difficulty will be to find sensible representations of the generative models at each layer. This is where the structure of neural networks might be useful, as these present an easy way to model complex functions without having to define a model structure beforehand.

From a robotics perspective, the main interest is the inference part of Active Inference. Using the hierarchical models it might be possible to construct inference machines that can "understand" the complex scenes encountered in robotics. There are recent developments in generative models (adversarial networks [64] and variational autoencoders [19, 61]) that could be combined with the framework of Active Inference to produce systems with very advanced inference capabilities. The controller will, as in humans, only contains reflex arcs. All of

the interesting computation takes place in the generative model. The main problem is to design a generative model (mainly the prior part) such that the robot outputs desired behavior.

More on the machine learning side of robotics, deep learning algorithms might improve by applying local prediction error minimization. Active Inference provides a way to develop unsupervised networks using local optimization, which is computationally efficient. Additionally, Active Inference can extend neural networks with the ability to output control signals. To date, neural networks mostly perform inference and miss agency (i. e., open loop). Extra systems are required to add the agency to the system, i. e., close the loop. Active Inference could provide a means to close the loop using only one single simpler structure.

Finally, the application of Active Inference on POMDP's – a discrete process often used in robotics – are also being studied recently (in a neuroscientific setting) [25–27, 30]. This framework relates Active Inference more closely to reinforcement learning algorithms and perhaps provides inspiration for improvements there. This thesis has not considered this version of Active Inference to limit the scope.

Part V

APPENDIX

A

SELECTED PROOFS AND DERIVATIONS

A.1 FREE ENERGY UPPER BOUND PROOF

The free energy is defined as

$$\mathcal{F}(\zeta, \tilde{y}) = \int q(\vartheta;\zeta) \ln \left(\frac{q(\vartheta;\zeta)}{p(\vartheta, \tilde{y})} \right) d\vartheta.$$

Let us now substitute in the equation above that

$$p(\vartheta, \tilde{y}) = p(\vartheta|\tilde{y})p(\tilde{y})$$

to arrive at

$$\mathcal{F}(\zeta, \tilde{y}) = \int q(\vartheta;\zeta) \ln \left(\frac{q(\vartheta;\zeta)}{p(\vartheta|\tilde{y})p(\tilde{y})} \right) d\vartheta$$

Using the logarithm product and quotient rules the integral can be divided in two terms:

$$\begin{aligned}
\mathcal{F}(\zeta, \tilde{y}) &= \int q(\vartheta;\zeta) \ln \left(\frac{q(\vartheta;\zeta)}{p(\vartheta|y)} \right) d\vartheta - \int q(\vartheta;\zeta) \ln p(\tilde{y}) d\vartheta \\
&= \int q(\vartheta;\zeta) \ln \left(\frac{q(\vartheta;\zeta)}{p(\vartheta|\tilde{y})} \right) d\vartheta - \ln p(\tilde{y}) \int q(\vartheta;\zeta) d\vartheta \\
&= \int q(\vartheta;\zeta) \ln \left(\frac{q(\vartheta;\zeta)}{p(\vartheta|\tilde{y})} \right) d\vartheta - \ln p(\tilde{y}) \qquad (45)
\end{aligned}$$

where for the last step we used the property that the integral over a probability density function equals one. The first term in Equation 45 is the Kullback-Leibler divergence between $q(\vartheta)$ and $p(\vartheta|\tilde{y})$. Hence we have that

$$\mathcal{F}(\zeta, \tilde{y}) = -\ln p(\tilde{y}) + D_{KL}\left(q(\vartheta;\zeta) \parallel p(\vartheta|\tilde{y})\right)$$

and since the Kullback-Leibler divergence is non-negative

$$\mathcal{F}(\zeta, \tilde{y}) \geqslant -\ln p(\tilde{y})$$

A.2 THE TEMPORAL VARIANCE MATRIX

The temporal variance matrix $V(\gamma)$ is a matrix describing the correlations of random variables at different orders of motion. This section will derive the covariance matrix of a (continuous) random variable in generalized form, in which $V(\gamma)$ will appear.

Consider continuous stochastic process $x(t)$ with

$$E[x(t)] = 0, \quad Var[x(t)] = \sigma^2, \quad Cov[x(t+dt), x(t)] = \sigma^2 \rho(dt)$$

where $\rho(t)$ is the autocorrelation function (for example Equation 9).

The first assumption we need to make is that $x(t)$ is continuous. This is equivalent to requiring that $\rho(0)$ is continuous [14], i. e., we are dealing with non-Markovian noise. Next, let the derivatives of $x(t)$ be defined using the limit approximation:

$$\begin{aligned}\dot{x}(t) &= \lim_{dt\to 0} \frac{x(t+dt)-x(t)}{dt}\\ \ddot{x}(t) &= \lim_{dt\to 0} \frac{x(t+dt)-2x(t)+x(t-dt)}{dt^2}\\ &\vdots\end{aligned}$$

We can now set up the covariance matrix $\Sigma_{\tilde{x}}$, which has the following internal structure up to the first two derivatives:

$$\Sigma_{\tilde{x}} = \begin{pmatrix} \mathrm{Cov}[x(t),x(t)] & \mathrm{Cov}[x(t),\dot{x}(t)] & \mathrm{Cov}[x(t),\ddot{x}(t)] \\ \mathrm{Cov}[\dot{x}(t),x(t)] & \mathrm{Cov}[\dot{x}(t),\dot{x}(t)] & \mathrm{Cov}[\dot{x}(t),\ddot{x}(t)] \\ \mathrm{Cov}[\ddot{x}(t),x(t)] & \mathrm{Cov}[\ddot{x}(t),\dot{x}(t)] & \mathrm{Cov}[\ddot{x}(t),\ddot{x}(t)] \end{pmatrix}$$

Which is straightforward to extend to higher orders. Since $x(t)$ has expectation zero,

$$\mathrm{Cov}[x(t),x(t)] = \mathrm{E}[x(t)^2] = \mathrm{Var}[x(t)] = \sigma^2$$

From the limit definitions, observe that the expectations of the derivatives are zero. So, for $\dot{x}(t)$ we have that

$$\begin{aligned}\mathrm{Cov}[\dot{x}(t),\dot{x}(t)] &= \mathrm{E}\left[\lim_{dt\to 0}\left(\frac{x(t+dt)-x(t)}{dt}\right)^2\right]\\ &= \lim_{dt\to 0}\frac{\mathrm{E}\left[x(t+dt)^2-2x(t+dt)x(t)+x(t)^2\right]}{dt^2}\\ &= \lim_{dt\to 0}\frac{\sigma^2-2\sigma^2\rho(dt)+\sigma^2}{dt^2}\\ &= \lim_{dt\to 0}\sigma^2\frac{2-2\rho(dt)}{dt^2}\\ &= \lim_{dt\to 0}\sigma^2\frac{2\rho(0)-\rho(dt)-\rho(-dt)}{dt^2}\end{aligned}$$

where for the last step it was used that $\rho(0) = 1$ and $\rho(dt) = \rho(-dt)$, both true by definition. Observe that the last equation contains exactly the (negated) limit approximation for $\ddot{\rho}(0)$ and thus

$$\mathrm{Cov}[\dot{x}(t),\dot{x}(t)] = -\sigma^2\ddot{\rho}(0).$$

Using a similar approach for $\ddot{x}(t)$ yields:

$$\mathrm{Cov}[\ddot{x}(t),\ddot{x}(t)] = -\sigma^2\rho^{(4)}(0)$$

Then for the off-diagonal elements:

$$\begin{aligned}
\mathrm{Cov}[x(t), \dot{x}(t)] &= \mathrm{E}\left[\lim_{dt\to 0} x(t)\frac{x(t+dt)-x(t)}{dt}\right] \\
&= \lim_{dt\to 0}\frac{\mathrm{E}\left[x(t)\left(x(t+dt)-x(t)\right)\right]}{dt} \\
&= \lim_{dt\to 0}\frac{\mathrm{E}\left[x(t)x(t+dt)-x(t)^2\right]}{dt} \\
&= \lim_{dt\to 0}\frac{\mathrm{E}\left[\sigma^2\rho(dt)-\sigma^2\right]}{dt} = 0
\end{aligned}$$

Similar derivations show that for $\mathrm{Cov}[x(t), \ddot{x}(t)]$ the limit approximation appears again and

$$\mathrm{Cov}[x(t), \ddot{x}(t)] = \sigma^2\ddot{\rho}(0)$$

Finally,

$$\mathrm{Cov}[\dot{x}(t), \ddot{x}(t)] = 0$$

Note that by definition, the covariance is commutative and thus using all of the above the covariance matrix can be completed:

$$\Sigma_{\tilde{x}} = \begin{bmatrix} \sigma^2 & 0 & \sigma^2\ddot{\rho}(0) \\ 0 & -\sigma^2\ddot{\rho}(0) & 0 \\ \sigma^2\ddot{\rho}(0) & 0 & \sigma^2\rho^{(4)}(0) \end{bmatrix}$$

This can be rewritten as

$$\Sigma_{\tilde{x}} = V(\gamma)\otimes\sigma^2 \qquad \text{where} \qquad V(\gamma) = \begin{bmatrix} 1 & 0 & \ddot{\rho}(0) \\ 0 & -\ddot{\rho}(0) & 0 \\ \ddot{\rho}(0) & 0 & \rho^{(4)}(0) \end{bmatrix}$$

Here, γ parameterizes $\rho(t)$ – thus technically we should have written $\rho(t,\gamma)$ everywhere. It might not seem that V is a function of γ, however γ will appear in the derivatives of the correlation function.

It is relatively straightforward to extend the derivations above up to higher orders. In fact, the SPM software has some very efficient code (see `spm_DEM_R.m`).

A.3 THE GENERALIZED MEASUREMENT

If Active Inference is to be applied in practical robotic settings, the data sent and received will be in the discrete domain, whereas this thesis has considered the continuous domain only. This was convenient for analyzing the algorithm, but has to be changed for practical applications. Friston has proposed a method through which a sequence of discrete observations can be converted to a generalized

form [28]. This section will shortly present this method and show a critical deficiency in this method.

Assume that a sequence of N data-points is available:

$$y(1:N) = \begin{bmatrix} y(1) & y(2) & \dots & y(N) \end{bmatrix}$$

which we would like to convert to a generalized measurement $\tilde{y}$. This will be done using a Taylor expansion around the center of the data sequence. As such it is required that $p \leqslant N$. (The order of the expansion cannot be higher than the number of samples available.)

We can write the transformation as

$$\tilde{y} = \tilde{E} y(1:N) \tag{46}$$

where dt is the sampling time and

$$\tilde{E} = E \otimes I_q$$

(remember $y \in \mathbb{R}^q$). E is a Taylor expansion matrix:

$$E^{-1}_{ij} = \frac{((i-x)dt)^{(j-1)}}{(j-1)!}$$

where $x = \texttt{round}(N/2)$, rounded up to the nearest integer.

The intuitive interpretation of this transformation is easier to show with the inverse problem:

$$y(1:N) = E^{-1}\tilde{y}$$

For example, assume that we have three one-dimensional samples ($N = 3$, $y \in \mathbb{R}$) and want an embedding order of three ($p = 3$) as well. The matrix E will be

$$E^{-1} = \begin{pmatrix} 1 & -dt & \frac{dt^2}{2} \\ 1 & 0 & 0 \\ 1 & dt & \frac{dt^2}{2} \end{pmatrix}$$

Expanding Equation 46:

$$\begin{aligned} y(1) &= y - y'dt + y''\frac{dt^2}{2} \\ y(2) &= y \\ y(3) &= y + y'dt + y''\frac{dt^2}{2} \end{aligned}$$

Now it should be obvious that $y(2)$ (the center sample) is y, and the sample before and after $y(2)$ are calculated using a Taylor expansion with the coefficients from the generalized measurement $\tilde{y}$.

Naturally, the disadvantage is that there will always be a delay (of `round(N/2)`) between the time of the newest data and the time at which $\tilde{y}$ is known. This is especially problematic when p is large.

A.4 FREE ENERGY EXPECTATION

The reason that the free energy will settle around unity in steady state is as follows. When taking the expectation of the free energy:

$$E[\mathcal{F}(\mu, y)] = E\left[\tfrac{1}{2}\varepsilon_x^\top \Pi_{\tilde{\omega}} \varepsilon_x + \tfrac{1}{2}\varepsilon_y^\top \Pi_z \varepsilon_y\right]$$

remember that the prediction errors are probabilistically equivalent to the noises (see Equation 17). Then the above can be written as:

$$E[\mathcal{F}(\mu, y)] = \tfrac{1}{2}\left(E[\tilde{\omega}^\top \Pi_{\tilde{\omega}} \tilde{\omega}] + E[z^\top \Pi_z z]\right)$$

Next, since $\Pi = \Sigma^{-1}$, the above can be written as

$$E[\mathcal{F}(\mu, y)] = \tfrac{1}{2}\left(E\left[\frac{\tilde{\omega}^\top \tilde{\omega}}{\Sigma_{\tilde{\omega}}}\right] + E\left[\frac{z^\top z}{\Sigma_z}\right]\right)$$

with slight abuse of notation.

Now, the covariance matrices are deterministic and do thus not affect the expectation. In addition, note that $E[\tilde{\omega}^\top \tilde{\omega}] = \Sigma_{\tilde{\omega}}$ by definition. Thus:

$$\begin{aligned} E[\mathcal{F}(\mu, y)] &= \tfrac{1}{2}\left(\frac{E[\tilde{\omega}^\top \tilde{\omega}]}{\Sigma_{\tilde{\omega}}} + \frac{E[z^\top z]}{\Sigma_z}\right) \\ &= \tfrac{1}{2}\left(\frac{\Sigma_{\tilde{\omega}}}{\Sigma_{\tilde{\omega}}} + \frac{\Sigma_z}{\Sigma_z}\right) \\ &= \tfrac{1}{2}(1+1) \\ &= 1 \end{aligned}$$

In reality – in simulations actually – the average value is most of the time somewhat lower. A reason for this has not been identified yet.

A.5 ACTIVE INFERENCE WITH MARKOVIAN NOISE

This section considers the limit case that $p = 1$. In other words, the only equation considered in the generative model for an LTI state space system is:

$$x' = \tilde{A}x + Bu + \boldsymbol{\omega}$$

Any higher order equation has infinite variance, because the noise is Markovian or very rough. The only two variables containing information are $\boldsymbol{x}$ and $\boldsymbol{x}'$.

Assuming that p is some value larger than one, the p equations in the generative model $\mathcal{D}\mu = \tilde{A}\mu + \xi$ are:

$$\begin{aligned} \mu_{x'} &= A\mu_x + \xi_x \\ \mu_{x''} &= A\mu_{x'} + \xi_{x'} \\ &\vdots \\ 0 &= A\mu_{x^{(p)}} + \xi_{x^{(p)}} \quad (= \mu_{x^{(p+1)}}) \end{aligned}$$

for which $\mu = \begin{pmatrix} \mu_x & \mu_{x'} & \dots & \mu_{x^{(p)}} \end{pmatrix}^\top$. Observe that the highest order motion $\mu_{x^{(p+1)}}$ is not in the generalized state and it is zero. This is equivalent to assuming that the precision of the noise at the highest order is so low that there is no information in $\mu^{(p+1)}$. We can keep removing equations such that p approaches 1. When $p = 2$ the generative model is:

$$\begin{aligned} \mu_{x'} &= A\mu_x + \xi_x \\ 0 &= A\mu_{x'} + \xi_{x'} \end{aligned} \tag{47}$$

But when $p = 1$ we do *not* have that

$$0 = A\mu_x + \xi_x$$

Because in this case there *is* information in x' (as long as $\sigma_\omega^2 \neq \infty$) and we do thus not need to assume that $\mu_{x'} = 0$. In this case the generative model is

$$\mu_{x'} = A\mu_x + \xi_x \tag{48}$$

Note that this model cannot be written in the standard form using $\mathcal{D}$.

The problem now is, we have two formulations by which we can update the first two orders of motion, i. e., for which $\mu = \begin{pmatrix} \mu_x & \mu_{x'} \end{pmatrix}^\top$. The reason to choose either of the two is as follows: When it is assumed that the noise is Markovian, we are not allowed to model higher order equations and thus the model from (48) should be used. If we were to assume that the noise is non-Markovian and choose $p = 1$, we should also use this model, because we know that there are two motions that contain information. The model from Equation 47 is used when choosing $p = 2$ with non-Markovian noise.

For both models, the update equation for μ_x is (omitting ξ):

$$\dot{\mu}_x = \mu_{x'} - C^\top \Pi_z (y - C\mu_x) + A^\top \Pi_\omega (\mu_{x'} - A\mu_x)$$

When $p = 1$ the update for $\mu_{x'}$ is

$$\dot{\mu}_{x'} = -\Pi_\omega (\mu_{x'} - A\mu_x) \tag{49}$$

as can be derived by formulating the free energy for both cases and performing the gradient descent. But when $p = 2$ the update contains an extra term:

$$\dot{\mu}_{x'} = -\Pi_\omega (\mu_{x'} - A\mu_x) - A^\top \Pi_\omega A\mu_{x'}$$

The extra term originates from the second equation in the generative model. The effects of this difference are yet to be studied. We conclude that when the noise is non-Markovian there are two ways to update the mean $\mu = \begin{pmatrix} \mu_x & \mu_{x'} \end{pmatrix}$. When the noise is Markovian one must use Equation 49 for $\mu_{x'}$.

B

MATLAB EXAMPLES

B.1 GENERATING SMOOTH NOISE

To generate smooth Gaussian noise $\omega(t)$ (Gaussian noise with a Gaussian autocorrelation), we can simply convolute Markovian Gaussian noise $w(t)$ with a Gaussian kernel. The kernel is defined as

$$\rho(t) = e^{-\frac{\gamma}{4}t^2}$$

Note that the kernel is a normalized Gausian, so that $\rho(0) = 1$, which is required if it is to be an autocorrelation function. When choosing other kernels, keep in mind that beside this the kernel must be symmetric. Both requirements follow from the definition of autocorrelation.

In a simulation environment, the convolution $\omega(t) = w(t) * \rho(t)$ can be performed by multiplying Gaussian noise vector w with a convolution matrix K:

$$\omega = wK$$

where K is a normalized toeplitz matrix. Listing 1 shows a MATLAB code example of this process. Figure 18 gives an example of the influence of the kernel width γ (the roughness parameter) on the smoothness of the noise.

Listing 1: MATLAB code: smoothing Gaussian noise.

```
% Time definition:
T = 10; dt = 0.1; t = 0:dt:T;
N = numel(t);

% Noise parameters:
gamma = 8;  % roughness
varw  = 10; % variance

% Parameter conversion:
s = sqrt(2/gamma);  % kernel variance

% Temporal convolution matrix:
T  = toeplitz(exp(-t.^2/(2*s^2)));  % Gaussian convolution matrix
K  = diag(1./sqrt(diag(T*T')))*T;   % normalization of T

% Smooth noise construction:
w = varw*randn(1,N)*K;
```

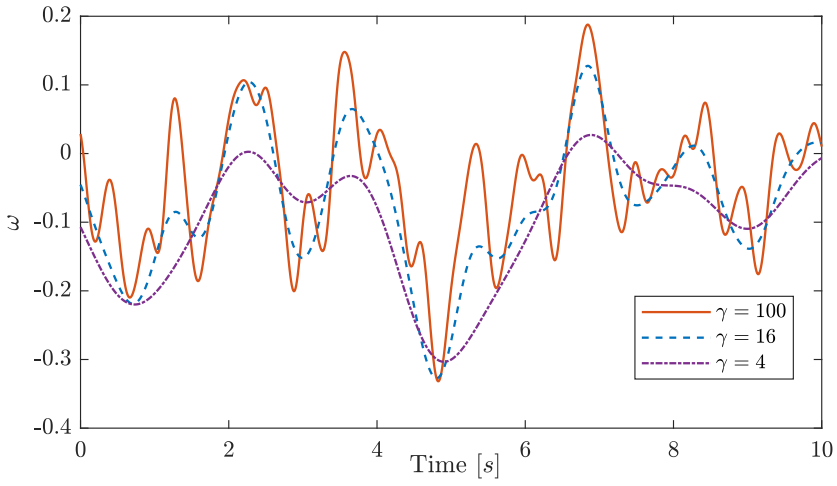


Figure 18: Influence of the roughness parameter γ on the smoothness of the noise. Here, the same white noise sequence (not shown to avoid cluttering) has been smoothed using Gaussian kernels of three different widths. Clearly, the smoothness of the noises is inversely proportional to γ.

B.2 COMPARING PRIMES AND DOTS

The code in the listing below produces a simulation that shows that in a stochastic mass-spring damper system with independent noises on the position and velocity $\dot{x}_1 \neq x_1'$ i. e. the motion of the position is not equal to the velocity in the stochastic case. In the deterministic case these are equal and generalized motions are redundant.

Listing 2: MATLAB code: comparing dots and primes

```
m = 10;k = 1e2; c = 10;
A = [0 1; -k/m -c/m];
B = eye(2);
C = eye(2);
process = ss(A,B,C,0);

%% Preprocessing:
% Time:
T_end = 10;
dt    = 0.05;
t     = 0:dt:T_end;
N     = numel(t);

% Generate analytic state noise:
varw  = 200*[1 1];      s     = 0.15;
[w,~] = makeNoise(process,[],varw,s,t,dt);

% Differentiate the noise:
w1dot = [diff(w(1,:))/dt 0];
w2dot = [diff(w(2,:))/dt 0];
```

```
wdot  = [w1dot ;w2dot];

% Initial conditions:
p0 = 40;   % initial position
v0 = 0;    % initial velocity
x0 = [p0 v0];
dx0 = [v0 -k/m*p0-c/m*v0]; % This is a consistent initial
    condition dx(0) = Ax(0) (velocity and acceleration)

%% Simulate deterministic system:
xd = lsim(process,zeros(2,N),t,x0);   % Standard process
pxd = lsim(process,zeros(2,N),t,dx0); % First generalization

%% Simulate the system with noise input:
x = lsim(process,w,t,x0);      % Standard process
px = lsim(process,wdot,t,dx0); % First generalization

% Plot results:
figure(1);
subplot(221);
plot(t,xd(:,1)); title('Deterministic');
ylabel('Position [m]'); grid on;

subplot(223)
plot(t,xd(:,2)); hold on; grid on;
plot(t,pxd(:,1),'--');
xlabel('Time [s]'); ylabel('Velocity [m/s]');
legend('$x_2$','$x_1''$');

subplot(222);
plot(t,x(:,1)); title('Stochastic');
grid on;
subplot(224);
plot(t,x(:,2)); hold on; grid on;
plot(t,px(:,1),'--');
xlabel('Time [s]');
legend('$x_2$','$x_1''$');
```

B.3 MASS SPRING DAMPER SIMULATION

Below is the MATLAB code to simulate the mass spring damper system from Chapter 8. Running this code will reproduce Figure 6.

Listing 3: MATLAB code: mass spring damper simulation.

```
%% Generative process:
m = 10;k = 1e2; c = 40;
A = [0 1; -k/m -c/m];
B = [0; 1/m];
C = [1 0];
```

```
x_eq = [40;0]; % desired equilibrium (constant)

system = ss(A,B,C,0);

%% Set up agent components:
% Tuning parameters
rho   = 1e4;    % control learning rate (default: 1e4)
kappa = 1e1;    % estimation learning rate (default: 10)
p     = 2;      % number of generalized states (default: 2)
varw = [10 10]; % uncertainty in states (variance)
varz  = 1;      % uncertainty in outputs (variance)
s     = 0.25;   % correlation kernel variance. (gamma = 1/s^2)

% Helper variables:
n = size(A,2);  % state dim
m = size(B,2);  % input dim
q = size(C,1);  % output dim

Ip = eye(p);
In = eye(n);

% Matrices:
Atilde = kron(Ip,A);
Ctilde = [C zeros(q,n*(p-1))];
Ghat = -C*(A\B);

T = toeplitz(zeros(1,p),[0 1 zeros(1,p-2)]);
D = kron(T,In);

S = smoothnessMatrix(p,s); % function: see below!
Piw = diag(1./varw);
Piw_tilde = kron(S,Piw);
Piz = diag(1./varz);

% Prior (for constant x_eq!):
x_gen_eq = [x_eq ;zeros(n*(p-1),1)];   % generalized equilibrium
xi = -Atilde*x_gen_eq;

%% Define closed loop:
M = D - kappa*(D-Atilde).'*Piw_tilde*(D-Atilde) - kappa*Ctilde.'*
    Piz*Ctilde;

% The state is [x mu u], hence the 3x3 partitioning:
Acl = [A                    zeros(n,n*p)         B;...
       kappa*Ctilde'*Piz*C  M                    zeros(n*p,m);...
      -rho*Ghat'*Piz*C      rho*Ghat'*Piz*Ctilde zeros(m,m)];

% The inputs are [xi w z], hence the 3x3 partitioning:
Bcl = [zeros(n,n*p)          eye(n)       zeros(n,q);...
 kappa*(D-Atilde)'*Piw_tilde zeros(n*p,n) kappa*Ctilde'*Piz;...
 zeros(m,n*p)                zeros(m,n)  -rho*Ghat'*Piz];
```

```
% For inspection purposes, we measure all signals:
Ccl = eye(size(Acl));

% There is no direct feedthrough of any signal:
Dcl = 0;

% Put the four matrices in a state space sturcture (continuous):
AIcl = ss(Acl,Bcl,Ccl,Dcl);

%% Simulate:
% Time:
T_end = 30;
dt    = 0.01;
t     = 0:dt:T_end;
N     = numel(t);
% Set random generator:
rng(1);

% Construct input signals:
xi = xi*ones(1,N);           % prior sequence
[w,z] = makeNoise(system,varz,varw,s,t);
ucl = [xi;w;z];

% Simulation:
ycl = lsim(AIcl,ucl,t);  % zero initial state implicit

% Extract states from the results:
x  = ycl(:,1:n).';
y  = C*x;
mu = ycl(:,n+1:n*(p+1)).';
u  = ycl(:,n*(p+1)+1:end).';

% Calculate the Free Energy:
for i = 1:N
    e_mu =(D*mu(:,i) - Atilde*mu(:,i)-xi(:,i));
    e_y  =(y(:,i) - Ctilde*mu(:,i));

    F(i)   = 0.5*e_mu.'*Piw_tilde*e_mu + 0.5*e_y.'*Piz*e_y;
end

%% Plot results: States, estimates, action and free energy:
subplot(221)
plot(t,x(1,:)); hold on; grid on;
plot(t,mu(1,:),'--');
plot(t,w(1,:));
plot(t,x_eq(1)*ones(size(t)));
xlabel('Time $[s]$');
ylabel('Position $[m]$');

subplot(222)
plot(t,x(2,:)); hold on; grid on;
plot(t,mu(2,:),'--');
```

```
plot(t,w(2,:));
xlabel('Time $[s]$');
ylabel('Velocity $[m/s]$');

subplot(223)
plot(t,u); grid on;
xlabel('Time $[s]$');
ylabel('Action / Force $[N]$');

subplot(224);
semilogy(t,F); grid on;
xlabel('Time $[s]$');
ylabel('Free Energy');

% ------------------------------------------------------
% Below are two functions used in the code above:

%% Smoothness matrix construction:
function [S] = smoothnessMatrix(p,s)
    k          = 0:(p - 1);
    r(1 + 2*k) = cumprod(1 - 2*k)./(s.^(2*k));

    % Covariance matrix:
    V     = [];
    for i = 1:p
        V = [V; r([1:p] + i - 1)];
        r = -r;
    end

    % Smoothness matrix:
    S = inv(V);
end
%% Generate smooth noises:
function [w,z] = makeNoise(system,varz,varw,s,t)
    n = size(system.A,2);
    q = size(system.C,1);
    N = numel(t);

    % Temporal convolution matrix:
    T  = toeplitz(exp(-t.^2/(2*s^2)));
    K  = diag(1./sqrt(diag(T*T')))*T;

    % Generate measurement noise:
    P = diag(1./varz);
    z  = sqrtm(inv(P))*randn(q,N)*K;

    % Generate state noise:
    P = diag(1./varw);
    w = sqrtm(inv(P))*randn(n,N)*K;
end
```

BIBLIOGRAPHY

[1] Rick A Adams, Stewart Shipp, and Karl J Friston. "Predictions not commands: active inference in the motor system." In: *Brain Structure and Function* 218.3 (2013), pp. 611–643.

[2] Theodore Wilbur Anderson and et al. *An introduction to multivariate statistical analysis*. Vol. 2. Wiley New York, 1958.

[3] Randal J Barnes. "Matrix differentiation." In: *Springs Journal* (2006), pp. 1–9.

[4] Andre M. Bastos, W. Martin Usrey, Rick A. Adams, George R. Mangun, Pascal Fries, and Karl J. Friston. "Canonical Microcircuits for Predictive Coding." In: *Neuron* 76.4 (2012), pp. 695 – 711. ISSN: 0896-6273.

[5] Matthew James Beal et al. *Variational algorithms for approximate Bayesian inference*. university of London London, 2003.

[6] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. "Variational inference: A review for statisticians." In: *Journal of the American Statistical Association* 112.518 (2017), pp. 859–877.

[7] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[8] Stephen Boyd, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishnan. *Linear matrix inequalities in system and control theory*. Vol. 15. Siam, 1994.

[9] Robert Grover Brown, Patrick YC Hwang, et al. *Introduction to random signals and applied Kalman filtering*. Vol. 3. Wiley New York, 1992.

[10] Christopher L Buckley, Chang Sub Kim, Simon McGregor, and Anil K Seth. "The free energy principle for action and perception: A mathematical review." In: *Journal of Mathematical Psychology* (2017).

[11] Ronald W. Butler. *Saddlepoint approximations with applications*. Vol. 22. Cambridge University Press, 2007.

[12] Andy Clark. "Whatever next? Predictive brains, situated agents, and the future of cognitive science." In: *Behavioral and brain sciences* 36.3 (2013), pp. 181–204.

[13] Roger C Conant and W Ross Ashby. "Every good regulator of a system must be a model of that system." In: *International journal of systems science* 1.2 (1970), pp. 89–97.

[14] D.R. Cox and H.D. Miller. In: *The theory of stochastic processes*. Routledge, 2017. Chap. 7 Stationary Processes: Time Domain, pp. 293–295.

[15] Peter Dayan, Geoffrey E Hinton, Radford M Neal, and Richard S Zemel. "The helmholtz machine." In: *Neural computation* 7.5 (1995), pp. 889–904.

[16] Peter Dayan, Geoffrey E Hinton, Radford M Neal, and Richard S Zemel. "The helmholtz machine." In: *Neural computation* 7.5 (1995), pp. 889–904.

[17] Arthur P Dempster, Nan M Laird, and Donald B Rubin. "Maximum likelihood from incomplete data via the EM algorithm." In: *Journal of the royal statistical society. Series B (methodological)* (1977), pp. 1–38.

[18] Petar M Djuric, Jayesh H Kotecha, Jianqui Zhang, Yufei Huang, Tadesse Ghirmai, Mónica F Bugallo, and Joaquin Miguez. "Particle filtering." In: *IEEE signal processing magazine* 20.5 (2003), pp. 19–38.

[19] Carl Doersch. "Tutorial on variational autoencoders." In: *arXiv preprint arXiv:1606.05908* (2016).

[20] Joseph L Doob and Joseph L Doob. *Stochastic processes*. Vol. 7. 2. Wiley New York, 1953.

[21] Richard P Feynman. *Statistical mechanics: a set of lectures*. 2018.

[22] Charles W Fox and Stephen J Roberts. "A tutorial on variational Bayesian inference." In: *Artificial intelligence review* 38.2 (2012), pp. 85–95.

[23] Bernard Friedland. *Control system design: an introduction to state-space methods*. Courier Corporation, 2012.

[24] Karl J Friston. "Variational filtering." In: *NeuroImage* 41.3 (2008), pp. 747–766.

[25] Karl J Friston, Jean Daunizeau, and Stefan J Kiebel. "Reinforcement learning or active inference?" In: *PloS one* 4.7 (2009), e6421.

[26] Karl J Friston, Thomas Parr, and Bert de Vries. "The graphical brain: belief propagation and active inference." In: *Network Neuroscience* 1.4 (2017), pp. 381–414.

[27] Karl J Friston, Spyridon Samothrakis, and Read Montague. "Active inference and agency: optimal control without cost functions." In: *Biological cybernetics* 106.8-9 (2012), pp. 523–541.

[28] Karl J Friston, N Trujillo-Barreto, and Jean Daunizeau. "DEM: a variational treatment of dynamic systems." In: *Neuroimage* 41.3 (2008), pp. 849–885.

[29] Karl J Friston, Jean Daunizeau, James Kilner, and Stefan J Kiebel. "Action and behavior: a free-energy formulation." In: *Biological cybernetics* 102.3 (2010), pp. 227–260.

[30] Karl J Friston, Richard Rosch, Thomas Parr, Cathy Price, and Howard Bowman. "Deep temporal models and active inference." In: *Neuroscience & Biobehavioral Reviews* 90 (2018), pp. 486–501.

[31] Karl Friston. "Functional integration and inference in the brain." In: *Progress in Neurobiology* 68.2 (2002), pp. 113 –143. ISSN: 0301-0082.

[32] Karl Friston. "Learning and inference in the brain." In: *Neural Networks* 16.9 (2003), pp. 1325–1352.

[33] Karl Friston. "A theory of cortical responses." In: *Philosophical Transactions of the Royal Society of London B: Biological Sciences* 360.1456 (2005), pp. 815–836.

[34] Karl Friston. "Hierarchical models in the brain." In: *PLoS computational biology* 4.11 (2008), e1000211.

[35] Karl Friston. "The free-energy principle: a rough guide to the brain?" In: *Trends in cognitive sciences* 13.7 (2009), pp. 293–301.

[36] Karl Friston. "The free-energy principle: a unified brain theory?" In: *Nature Reviews Neuroscience* 11.2 (2010), p. 127.

[37] Karl Friston. "What Is Optimal about Motor Control?" In: *Neuron* 72.3 (2011), pp. 488 –498. ISSN: 0896-6273.

[38] Karl Friston. "A free energy principle for biological systems." In: *Entropy* 14.11 (2012), pp. 2100–2121.

[39] Karl Friston. "Prediction, perception and agency." In: *International Journal of Psychophysiology* 83.2 (2012), pp. 248–252.

[40] Karl Friston. "Life as we know it." In: *Journal of the Royal Society Interface* 10.86 (2013), p. 20130475.

[41] Karl Friston and Stefan Kiebel. "Predictive coding under the free-energy principle." In: *Philosophical Transactions of the Royal Society of London B: Biological Sciences* 364.1521 (2009), pp. 1211–1221.

[42] Karl Friston and Stefan Kiebel. "Predictive coding under the free-energy principle." In: *Philosophical Transactions of the Royal Society of London B: Biological Sciences* 364.1521 (2009), pp. 1211–1221.

[43] Karl Friston, James Kilner, and Lee Harrison. "A free energy principle for the brain." In: *Journal of Physiology-Paris* 100.1-3 (2006), pp. 70–87.

[44] Karl Friston, Jérémie Mattout, and James Kilner. "Action understanding and active inference." In: *Biological cybernetics* 104.1-2 (2011), pp. 137–160.

[45] Karl Friston, Jérémie Mattout, Nelson Trujillo-Barreto, John Ashburner, and Will Penny. "Variational free energy and the Laplace approximation." In: *Neuroimage* 34.1 (2007), pp. 220–234.

[46] Karl Friston, Klaas Stephan, Baojuan Li, and Jean Daunizeau. "Generalised filtering." In: *Mathematical Problems in Engineering* 2010 (2010).

[47] Crispin Gardiner. *Stochastic methods*. Vol. 4. springer Berlin, 2009.

[48] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[49] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets." In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[50] S.S. Grimbergen, C. van Hoof, and M. Wisse. "Active Inference for State Space Models: A Tutorial." unpublished. 2019.

[51] Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[52] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. "Stochastic variational inference." In: *The Journal of Machine Learning Research* 14.1 (2013), pp. 1303–1347.

[53] T Jaakkola. "Tutorial on variational approximation methods." In: *Advanced mean field methods: theory and practice* (2001), p. 129.

[54] Rudolph E Kalman and Richard S Bucy. "New results in linear filtering and prediction theory." In: *Journal of basic engineering* 83.1 (1961), pp. 95–108.

[55] Martin Koeber and Uwe Schäfer. "The unique square root of a positive semidefinite matrix." In: *International Journal of Mathematical Education in Science and Technology* 37.8 (2006), pp. 990–992.

[56] Solomon Kullback and Richard A Leibler. "On information and sufficiency." In: *The annals of mathematical statistics* 22.1 (1951), pp. 79–86.

[57] Huibert Kwakernaak and Raphael Sivan. *Linear optimal control systems*. Vol. 1. Wiley-Interscience New York, 1972.

[58] Johan Kwisthout. "Minimizing relative entropy in hierarchical predictive coding." In: *European Workshop on Probabilistic Graphical Models*. Springer. 2014, pp. 254–270.

[59] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. "Building machines that learn and think like people." In: *Behavioral and Brain Sciences* 40 (2017).

[60] David JC MacKay. "Free energy minimisation algorithm for decoding and cryptanalysis." In: *Electronics Letters* 31.6 (1995), pp. 446–447.

[61] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. "Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks." In: *arXiv preprint arXiv: 1701.04722* (2017).

[62] Nasser M Nasrabadi. "Pattern recognition and machine learning." In: *Journal of electronic imaging* 16.4 (2007), p. 049901.

[63] Léo Pio-Lopez, Ange Nizard, Karl Friston, and Giovanni Pezzulo. "Active inference and robot control: a case study." In: *Journal of The Royal Society Interface* 13.122 (2016), p. 20160616.

[64] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." In: *arXiv preprint arXiv:1511.06434* (2015).

[65] Maxwell James Désormeau Ramstead, Paul Benjamin Badcock, and Karl John Friston. "Answering Schrödinger's question: a free-energy formulation." In: *Physics of life reviews* 24 (2018), pp. 1–16.

[66] Rajesh PN Rao and Dana H Ballard. "Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects." In: *Nature neuroscience* 2.1 (1999), p. 79.

[67] Christian Robert. *Machine learning, a probabilistic perspective*. 2014.

[68] Jürgen Schmidhuber. "Deep learning in neural networks: An overview." In: *Neural networks* 61 (2015), pp. 85–117.

[69] Sarah Schwöbel, Stefan Kiebel, and Dimitrije Marković. "Active Inference, Belief Propagation, and the Bethe Approximation." In: *Neural Computation* 30.9 (2018), pp. 2530–2567.

[70] Claude Elwood Shannon. "A mathematical theory of communication." In: *ACM SIGMOBILE Mobile Computing and Communications Review* 5.1 (2001), pp. 3–55.

[71] Bruno Siciliano and Oussama Khatib. *Springer handbook of robotics*. Springer, 2016.

[72] Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*. Vol. 135. MIT press Cambridge, 1998.

[73] Kai Ueltzhöffer. "Deep active inference." In: *Biological Cybernetics* 112.6 (2018), pp. 547–573.

[74] Frank Uhlig. "On the matrix equation AX = B with applications to the generators of a controllability matrix." In: *Linear Algebra and its Applications* 85 (1987), pp. 203–209.

[75] J Woodhead-Galloway. "Schrödinger: what is life?" In: *Physics Bulletin* 34.12 (1983), p. 490.

[76] Kemin Zhou, John Comstock Doyle, Keith Glover, et al. *Robust and optimal control*. Vol. 40. Prentice hall New Jersey, 1996.