Robot manipulator control under the Active Inference framework

A. Cibiach Mercadé

Master of Science Thesis

Delft University of Technology

Delft Center for Systems and Control

Robot manipulator control under the Active Inference framework

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

A. Cibiach Mercadé

November 29, 2018

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

Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

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

Robot manipulator control under the Active Inference framework

by

A. Cibiach Mercadé

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: November 29, 2018

Supervisor(s):

dr.ir. C. Hernández Corbato

prof.dr.ir. M.Wisse

Reader(s):

dr.ir. A.J.J. van den Boom

dr.ir. C. Hernández Corbato

prof.dr.ir. M.Wisse

Abstract

The Active Inference framework is a neuroscience theory based on the Free Energy Principle by Karl Friston that has gained considerable prominence as a general theory to explain action and perception. Despite its promising future and use in different fields, its applicability for robot control has been barely investigated in the literature so far. This thesis aims at discerning if the current Active Inference implementation can be used for the control of robot manipulators and which are the requirements to do so. Firstly, an analysis of the current implementation of robot control under the Active Inference is made to understand its accomplishments and limitations. Secondly, both offline and online control schemes under the Active Inference framework are proposed. To continue with, a low-level controller is designed to be used in the Active Inference control scheme as a prior of the robot behaviour. The performance of the designed controller and the Active Inference scheme using this controller as a prior is compared to a benchmark controller. This comparison shows that the proposed implementation of the Active Inference scheme does not perform better than the current methods, but with further research on the how to implement this scheme using a real process it may become a valid alternative to these methods. Finally, recommendations on future work are given to bring this framework to online implementations and to verify the results obtained regarding the applicability of the Active Inference framework to robot manipulator control.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my supervisors dr.ir. C. Hernández Corbato and prof.dr.ir. M.Wisse for their assistance during the writing of this thesis and the support and help provided throughout the last year. Also I would like to thank them for their implication at the end of the process, giving me meaningful feedback to improve the final result of this document.

I want to thank my family that I have felt close during these two years and our weekly Skype calls that have helped me to get away during the challenging moments.

I would also like to thank the students, professors and staff I have met during these two years that have made my experience at this university better. Specially, Carlo and Tori for all the fun moments we had during our first year of master; and Ben and Folkert, who have been very helpful during the past year while working on the thesis.

I want to thank the IO students that I have met, for adopting me as one of them during these 2 years and specially to Alice, Magda and Ninad for all the fun we have had together.

Last but not least, I want to thank Núria for sharing this amazing experience in a foreign country with me. Without her love, her support and all the experiences we have lived, this two-years journey would not have been the same.

Delft, University of Technology
November 29, 2018

A. Cibiach Mercadé

Chapter 1

Introduction

Society is evolving towards a future where robots are becoming more important and their presence is being increased in different fields. Therefore, robot control is a topic that is in permanent evolution and new methods to obtain more autonomous behaviour are constantly researched.

Some of the current methods for robot control, such as optimal control approaches, are efficient and widely used. However, these methods usually require an inverse model to be computed. When dealing with a specific type of robots like the robot manipulators (with redundant joints), this inverse model becomes non trivial and its computation requires knowledge about the dynamic properties of the system.

The recent appearance of a biologically inspired method known as Active Inference, which dispenses with the inverse model as the inversion is done internally, has grabbed the engineering community attention. This framework has been applied in theoretical simulations to solve typical control problems such as the mountain car problem [2]. Thus, the Active Inference framework seems a promising theory that could be applied to robot control and demands further research to investigate its possibilities in this field.

The Active Inference framework [3–5] has acquired significant prominence in computational and systems neuroscience. This framework can be considered a general theory of brain and behaviour that tries to explain action and perception and considers the brain as a predictive processing machine.

In line with the aforementioned, in [1], the first attempt to implement the Active Inference scheme for robot control of a 7 degrees of freedom manipulator is done. It is shown in this paper how Active Inference applied to a robot arm can deal with noise present in the signals. However, from an initial analysis, it has been seen that this implementation may have limitations present in its formulation from a low-level controller perspective. Therefore, a revision of the implementation for robot control seems necessary in order to give a clear picture about the possibilities of this scheme for robot control.

1-1 Thesis objective and contribution

Based on what has been mentioned in the previous section, the thesis objective can be defined. The main research question that this thesis is built around is:

> *Can the current Active Inference framework formulation be used for controlling a robot manipulator in a reaching task?*

Thus, this thesis aims at discerning whether the Active Inference scheme can be used for robot manipulator control. In this thesis, a low-level controller (joint torque control) is intended since when applying this type of control there is no need to rely on an internal controller of the robot manipulator. To achieve such control in Active Inference, how a designed controller can be embedded in this scheme and the resulting control loop needs to be defined. Furthermore, a comparison between the designed controller and a current method available is needed to determine how the proposed implementation performs with respect to the state of the art controller.

In this document, an analysis of the current implementation of robot control for Active Inference [1] is used to understand and to define a proper Active Inference scheme for robot control. The offline control loop of this scheme is achieved in this thesis and is used to test the performance of the controllers. In contrast to what is done in [1], where a low-level position control embedded in the robot manipulator is required, a low-level joint torque control is used in the implementation proposed. This offline control loop also permits a feed-forward control using the Active Inference scheme to test with the real process. Moreover, an online implementation and its requirements are defined.

1-2 Document outline

This document is structured as follows:

- In Chapter 2, the Active Inference framework and the Free Energy Principle are described,
- In Chapter 3, the background of robot control and its application using Active Inference is presented,
- Chapter 4 presents the robot manipulator used in this project and its kinematic and dynamic formulations. The setup that is used for the experiments is also described,
- In Chapter 5, the state of the art and proposed controllers are presented and the control topologies are described,
- In Chapter 6, the different controllers are tested and the simulation results are discussed,
- Finally, Chapter 7 presents the conclusions derived from this thesis and the recommendations for future work.

Chapter 2

Active Inference framework

In the recent years, many theories trying to explain the brain cognitive process have appeared. Moreover, inference-based theories, and more specifically Bayesian inference ones, have served as a starting point for theories in different fields such as decision-making and planning [6]. One of the theories that has brought more attention is Active Inference, based on the Free Energy Principle (FEP), which has gained considerable prominence in computational and systems neuroscience and philosophy as a general theory of brain and behaviour [4].

In this chapter, the FEP is described to understand its main idea; after that, the Active Inference framework is outlined. Finally, a view of Active Inference from a control perspective is given. This chapter aims at giving an overview of the Active Inference framework and the FEP, in which is based, together with the concepts that become important when Active Inference is intended for control.

2-1 The Free Energy Principle

Biological systems are thermodynamically open as they are constantly exchanging energy and entropy with their environment showing a self-organizing behaviour. However, the organisms interact with the environment to endure over periods of time by changing their position within it or their relation to it. This interaction is carried in a way that allows the organism to maintain their states within the bounds that do not affect their physical structure [3]. This idea has been captured in a theory called the Free Energy Principle. In this principle, the agent has an internal model of the world which is known as the generative model.

The FEP bases its theory on the minimization of a concept called free energy F, which is an upper bound on the surprisal of the system states μ. The free energy is a scalar function of the ensemble density and the current sensory input $\tilde{\varphi}$. Note that tilde denotes variables in generalised coordinates of motion[1]. The ensemble density $q(\vartheta; \mu)$ can be considered as the

[1]Generalized coordinates specify movements as predicted trajectories, which include speed, acceleration, jerk, etc. [5]

probability density that a specific causal state ϑ is chosen based on the actual system state μ.

In order to allow for tractable computations, the FEP uses variational inference which enables to summarize beliefs about hidden states with a single quantity. In other words, by means of the Laplace assumption, the ensemble densities are parametrised by its mode or expectation and covariance assuming a Gaussian form [3].

$$\begin{aligned} F &= -\int q(\vartheta) \ln \frac{p(\tilde{\varphi}, \vartheta)}{q(\vartheta|\mu)} \mathrm{d}\vartheta \\ &= -\langle \ln p(\tilde{\varphi}, \vartheta|m) \rangle_q + \langle \ln q(\vartheta|\mu) \rangle_q \end{aligned} \tag{2-1}$$

As it can be seen in Equation (2-1) [3], the free energy is defined by two densities; namely, the ensemble density $q(\vartheta|\mu)$ and the generative density $p(\tilde{\varphi}, \vartheta|m)$. The latter factorises into a likelihood and prior density $p(\tilde{\varphi}|\vartheta)p(\vartheta)$ which determines a probabilistic generative model m. Equation (2-1) can be rewritten as follows [3]:

$$\begin{aligned} F &= D(q(\vartheta|\mu)||p(\vartheta|\tilde{\varphi}, m)) - \ln p(\tilde{\varphi}|m) \\ &= D(q(\vartheta|\mu)||p(\vartheta|m)) - \langle \ln p(\tilde{\varphi}|\vartheta, m) \rangle_q \end{aligned} \tag{2-2}$$

From the equation shown above, it can be see that the free energy, F, minimization can be accomplished with two alternative expressions. The first one shows how perception can be used to optimize the predictions and infer the causes of its sensory samples (by changing internal states $q(\vartheta|\mu)$); the second equation shows how action can reduce the free energy (through the effect of actions on hidden states and the consequent sensory input $p(\tilde{\varphi}|\vartheta, m)$) by increasing accuracy, the expected surprise of sensory data under the recognition density (for more detailed mathematical explanation of the theory see [4, 5, 7, 8]).

2-2 Active Inference

The Active Inference framework uses the FEP - surprise minimization - to explain perception and action and is often considered a corollary of the FEP [9] described in the previous section. It is cast in terms of Bayesian inference: the agent has a model of the world which is optimized using the sensory inputs by predicting and explaining its sensations. The possibility to actively modify the inference of the world through action is what gives the name to this framework.

The main idea behind Active Inference is that the brain can manipulate the sensory input and the brain states in order to minimize the prediction error. In this way, as explained in section 2-1, action and perception are used to minimize the free energy, either by changing its sensory input through action or by modifying its beliefs about the states of the world.

The generative model plays an important role in the Active Inference framework as it captures the dynamics of the interaction between the agent and its environment and is the responsible for generating behaviour that minimizes the prediction error. A schematic layout of the generative model of an active inference agent is shown in figure 2-1. The generative model assumed for Active Inference is generally non-linear, dynamic and deep (i.e. hierarchical) [1].

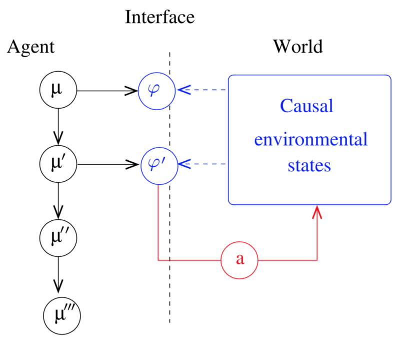


Figure 2-1: Generative model for an active inference agent. Interaction between the generalised brain states ($\mu, \mu', \dots$) and the sensory data ($\varphi, \varphi', \dots$).The agent acts on the world, via variable a (red arrow), to change sensory input φ and minimise free-energy indirectly. Solid arrows represent dependencies within the brain and dashed arrows represent incoming sensory data. Figured reproduced from [8].

Perception and action are defined by the generative model with the following equations [7]:

$$\dot{\mu} = -\mathcal{D}\mu - \partial_\mu F \tag{2-3}$$

$$\dot{a} = -\partial_a F \tag{2-4}$$

where μ are the internal states, a is the action of the agent, F is the free-energy term and $\mathcal{D}$ is a differential matrix operator used to describe the motion of conditional expectations.

The equations shown above are both a gradient descent on the variational free energy F. By using the method *generalised filtering*, the gradients equations can be expressed in terms of prediction errors ε (for a more detailed explanation of the mathematics related to generalised filtering see [10]):

$$\begin{aligned} \tilde{\varepsilon}_x &= \mathcal{D}\tilde{\mu}_x - \tilde{f}(\tilde{\mu}_x, \tilde{\mu}_v) \\ \tilde{\varepsilon}_v &= \tilde{\mu}_v - \tilde{g}(\tilde{\mu}_x, \tilde{\mu}_v) \end{aligned} \tag{2-5}$$

where $f(\cdot)$ and $g(\cdot)$ correspond to the plant dynamics and the sensory mapping of the generative model, respectively.

As mentioned before, the Active Inference scheme does not only have an internal model of the world (generative model) but also represents the real world dynamics by considering them in the generative process. These generative model and generative process are considered two of the main aspects of the Active Inference framework as they define the true and modelled sensory information used in this scheme.

2-2-1 Generative Process

The true dynamics of the system generating the sensory information are defined by the generative process [5]. Since this process corresponds to the dynamics of the real world, the action derived from the Active Inference scheme is part of it. Moreover, this process can be divided in two parts:

- Plant dynamics: $\mathbf{f}(\mathbf{x},\mathbf{v},a)$
- Real sensations mapping: $\mathbf{g}(\mathbf{x},\mathbf{v})$

where $\mathbf{x}$ corresponds to the true hidden states, $\mathbf{v}$ to the true causal states and a to action.

The plant dynamics correspond to the true equations of motion, which are function of the true hidden and causal states and the action; on the other hand, the real sensations correspond to the true sensory mapping, function of the true hidden and causal states.

2-2-2 Generative Model

The generative model comprises the dynamics of the system that model the sensory information [5]. As opposed to the generative process, where the action is part of the true dynamics, the generative model depends only on the hidden and casual states. Similar to the generative process, the generative model can also be divided in two parts:

- Modelled dynamics: $f(\mu_x,\mu_v)$
- Modelled sensations mapping: $g(\mu_x,\mu_v)$

where μ_x represents the hidden states and μ_v the causal states.

The modelled dynamics correspond, in this case, to the equations that model the motion of the system, which are function of the hidden and causal states; on the other hand, the modelled sensations correspond to the sensory mapping modelling sensory information, function of the hidden and causal states.

2-3 Active Inference for control: encoding the prior

As the generative model represents the internal model that the agent has about the environment, this model does not have to represent the real environment but a version of it based on the agent beliefs. Therefore, the modelled dynamics of the system, present in the generative model, can be influenced by prior beliefs that the system has about the causes of the sensory information. This allows for a prior on the behaviour to be encoded in the generative model through the use of these beliefs.

By modifying the defined equations of motion of the generative model, a desired performance of the system can be achieved; the desire to behave in such way can be seen as the influence of the prior beliefs that the system has and that forces it to act in a specific way. This prior belief that acts on the behaviour of the system can be seen as a control signal that drives the motion. Therefore, from a control perspective, a controller can be designed to accomplish a desired motion and the control signal can be used as the prior belief that steers the states of the system towards the desired ones. In this way, the agent believes that it has to behave in a particular way that is specified by these priors.

Chapter 3

Related work

Robot control is a key aspect in obtaining improved performances in robotic systems and, therefore, a lot of development is made to increase the performance and introduce new functionalities. It is important to study the current situation and limitations that the available methods present to obtain a clearer picture of the possible advantages of using the Active Inference scheme for robot control.

This chapter serves as a basis for understanding the current state of robot control and its limitations. Furthermore, it is also discussed how Active Inference can be used for robot manipulator control and the outcomes from the analysis that have helped defining the Active Inference control scheme proposed in this thesis.

3-1 Robot control

There are different controller types that can be used for robot control. These can be divided in two groups:

- Feed-back controller: where the current state of the system is used to compute the control action. In this case, a desired position is compared to the current one and by defining a proper gain the needed controller action is obtained, for instance, the PID controllers. This closed loop control is the most common one because stabilization is easier as the controller is considering the real states of the system and exogenous disturbances can be counteracted.
- Feed-forward controller: where the control action is a function of some parameters that are measured in advance and does not consider the current state of the system. This is type of control is open loop and possible divergences between the model used to obtain the control signal and the real plant are crucial for the stability of the system. However, a variation of this type of controllers can include a feedback loop that considers the current state of the system to adjust the control signal compensating for those divergences.

When dealing with a robot manipulator with the mentioned controllers and intending a low-level torque control, the inverse model is needed to obtain the required torques given joint positions, velocities and accelerations that generate the motion from an initial position to a desired end-effector position.

The inverse dynamics model required to obtain the torques for the robot manipulator is usually hard to compute for manipulators with redundant joints (robot arms with 7 degrees of freedom). Furthermore, all the dynamical parameters of the manipulator, such as the inertia of each joint, need to be known. Therefore, an approach that does not require the aforementioned inverse model and that is easily adaptable to different manipulators would facilitate the control of these manipulators.

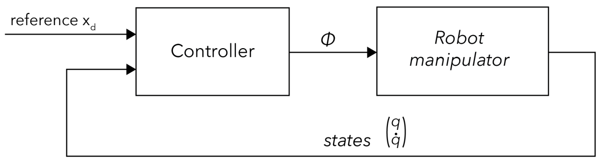
As mentioned in the introduction, this thesis aims at implementing the Active Inference scheme for a reaching task. The reaching task consists on a motion from point A to point B, in which the trajectory between these is not defined. Usually, for a reaching task in a robot manipulator, the variables related to the control are the following:

Concept	**Symbol**	**Description**
reference	x_d	End-effector target position in Cartesian coordinates
control signal	Φ	Actuator signal, for low-level controllers: joint torques
states	$\begin{bmatrix} q \\ \dot{q} \end{bmatrix}$	Joint positions and velocities
pos_{ee}	pos_{ee}	Current end-effector position
error	e	Difference between reference and current end-effector position

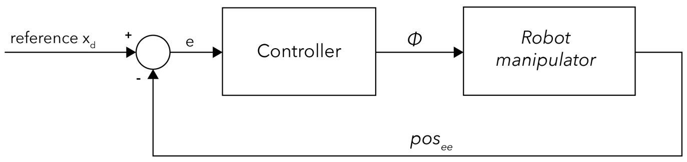
Table 3-1: Variables in the robot control reaching task addressed in this thesis

In figure 3-1, the control scheme using a feedback loop for robot control is depicted. In this case, two different options are shown: first, where the robot states are available (known as joint-space control [11]); and finally, where only the end-effector position is known (known as Cartesian control [12]).

In the first scenario, different controllers can be used: regulation control (PID control), computed torque control (also known as inverse dynamics control). In the second scenario, some of the possible controllers are: Cartesian torque control, joint-space torque control (using inverse kinematics to transform from Cartesian-space to joint-space).



(a) States available



(b) End-effector position available

Figure 3-1: Control scheme for robot manipulator control

3-2 Active Inference robot control

Robot manipulator control has been already addressed in a published paper by Pio-Lopez et al.: "Active inference and robot control: a case study" [1]. A thorough analysis of the paper has been done to extract the interesting contributions and to finesse the control scheme proposed taking into consideration the limitations identified.

3-2-1 Analysis of "Active inference and robot control: a case study"

The authors of this paper present the first implementation of Active Inference for a robot manipulator control. It is applied to a 7 degrees of freedom arm of the PR2 robot, which is simulated using the Robot Operating System (ROS). Their main contribution consists on demonstrating how the Active Inference scheme can control a robot arm under different noise conditions in the sensory mapping (vision) and in the dynamical model (proprioception).

The correspondence between the Active Inference scheme and the robot manipulator variables used in the aforementioned paper [1] is shown in table 3-2.

Generative process

As mentioned in section 2-2, the generative process corresponds to the true dynamics of the real world but in this case the generative process was defined with a dynamical model replacing the actual robot simulation.

Active Inference		**Robot manipulator**	
Concept	**Variable**	**Variable**	**Concept**
Hidden states	$x = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$	$\begin{bmatrix} q \\ \dot{q} \end{bmatrix}$	Joint positions and velocities
Causal states	v	T	Target position
Action	a	τ	Torque signal
Prior belief	Φ	Φ	Controller signal

Table 3-2: Active Inference - robot manipulator correspondence of variables in [1]

As it can be seen in equation 3-1, the sensory mapping function $g(x, v)$ consists on the input states (hidden and causal states) and the end-effector current position. On the other hand, the equations of motion $f(x, v, a)$ comprehend the velocities and accelerations of the joints. The velocities are obtained using the second half of the hidden states corresponding to the velocities $\dot{q}$, while the accelerations are obtained assuming Newtonian dynamics as it has been done for a human arm in [5].

$$g(x, v) = \begin{bmatrix} x \\ v \\ \text{Pos} \end{bmatrix} \qquad f(\tilde{x}, v, a) = \begin{bmatrix} x'_1 \\ x'_2 \\ \vdots \\ x'_7 \\ \frac{(a_1 - k_1 x_1 - \kappa_1 x'_1)}{m_1} \\ \frac{(a_2 - k_1 x_2 - \kappa_1 x'_2)}{m_1} \\ \frac{(a_3 - k_1 x_3 - \kappa_1 x'_3)}{m_1} \\ \frac{(a_4 - k_2 x_4 - \kappa_2 x'_4)}{m_2} \\ \vdots \\ \frac{(a_7 - k_2 x_7 - \kappa_2 x'_7)}{m_2} \end{bmatrix} \tag{3-1}$$

where $x = (x_1, x_2, \ldots, x_7)$ are the angles of the joints (hidden states); $v = (v_1, v_2, v_3)$ is the target position of the end effector; Pos corresponds to the actual position of the end effector; m_i is the mass, a_i corresponds to the action and κ_i and k_i are viscosity and elasticity coefficients.

Generative model

Regarding the generative model, the one proposed in [1] is shown in equation 3-2. The sensory mapping used corresponds to the same as the one from the generative process; while the equations of motion only differ on the acceleration part. The action signal used in the equations of motion from the generative process is substituted here for a controller signal Φ.

$$g(x,v) = \begin{bmatrix} x \\ v \\ \mathrm{Pos} \end{bmatrix} \qquad f(\tilde{x},v) = \begin{bmatrix} x'_1 \\ x'_2 \\ \vdots \\ x'_7 \\ \frac{(\Phi_1 - k_1 x_1 - \kappa_1 x'_1)}{m_1} \\ \frac{(\Phi_2 - k_1 x_2 - \kappa_1 x'_2)}{m_1} \\ \frac{(\Phi_3 - k_1 x_3 - \kappa_1 x'_3)}{m_1} \\ \frac{(\Phi_4 - k_2 x_4 - \kappa_2 x'_4)}{m_2} \\ \vdots \\ \frac{(\Phi_7 - k_2 x_7 - \kappa_2 x'_7)}{m_2} \end{bmatrix} \tag{3-2}$$

where $x = (x_1, x_2, \ldots, x_7)$ are the angles of the joints (hidden states); $v = (v_1, v_2, v_3)$ is the target position of the end effector; Pos corresponds to the actual position of the end effector; m_1 is the mass, Φ_i corresponds to the control signal and κ_i and k_i are viscosity and elasticity coefficients.

Controller

The controller signal (Φ) provides angular increments per joint and is obtained as a sum of two actions α and β:

$$\Phi_i = \alpha_i + \beta_i \tag{3-3}$$

The first action α corresponds to a PI controller that considers the following feedback error:

$$e_i = (\varphi \times \phi_i) \cdot \mathrm{Pos}_i \tag{3-4}$$

where $\varphi = T - J$ describes the shortest path to reach the target being T the target position and J the end effector position, $\phi_i = J_i - J/\|J_i - J\|$ is a unit vector that links each joint position to the end the end effector one, where J_i corresponds to the i^{th} joint position and J to the end effector position, and Pos_i is a unit vector collinear to the rotation axis of the i^{th} joint. The PI controller ensures that:

$$\dot{e}_i = \alpha_i = -p_p e_i + p_i \int_{t=0}^{t=t_0} e_i(t)\mathrm{d}t, \tag{3-5}$$

where t_0 corresponds to the current time and p_i and p_p are two positive gains to adjust the convergence rate.

The second action β is used to predict the influence of the stretched arm singularity. This action is only acting on the elbow and shoulder joints (β_2 and β_4):

$$\left.\begin{aligned} &\beta_1 = \beta_3 = \beta_5 = \beta_6 = \beta_7 = 0 \\ &\beta_2 = \gamma_m \gamma_c p_{p2} \\ &\beta_4 = -\gamma_m \gamma_c p_{p4} \end{aligned}\right\} \tag{3-6}$$

where p_{p2} and p_{p4} are two positive gains used to balance the contribution of the two joints and γ_m and γ_c correspond to:

$$\left.\begin{aligned} \gamma_m &= \left|\frac{\varphi}{\|\varphi\|} \cdot \frac{\phi_2}{\|\phi_2\|}\right| \\ \gamma_c &= \varphi \cdot \phi_2 \end{aligned}\right\} \tag{3-7}$$

The controller used in this implementation has been specifically designed for this robot manipulator considering its kinematics properties. Moreover, 4 different parameters (p_p, p_i, p_{p2} and p_{p4}) need to be tuned for each implementation, which makes this controller less adaptable to different manipulators.

Implementation

Regarding the implementation proposed in this paper shown in figure 3-2, the action obtained from the Active Inference scheme is not used directly on the PR2 robot simulation. In this case, the updated hidden states (joint positions q) are sent to an internal position controller that the simulation includes and that brings the robot arm to the desired joint positions. Furthermore, their implementation consists on an offline scheme as there is no feedback from the simulation in terms of actual joint positions.

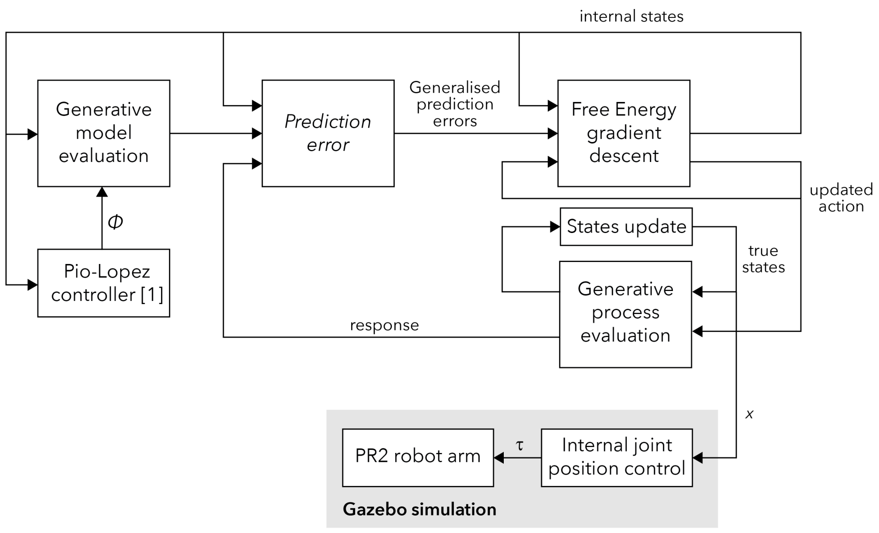

internal states
Generative model evaluation
Prediction error
Generalised prediction errors
Free Energy gradient descent
updated action
Φ
States update
true states
Pio-Lopez controller [1]
Generative process evaluation
response
x
PR2 robot arm
τ
Internal joint position control
Gazebo simulation

Figure 3-2: Active Inference control scheme implementation used in Pio-Lopez et al. [1]

As described in the section above, the controller provides a per-joint angular increment action, but when embedded in the generative model, this action is being used as a force in the Newtonian dynamics model described. Moreover, this dynamic model does not consider gravity, while it is considered in the simulation; however, their simulation still converges thanks to the aforementioned internal position control that computes the necessary torques to keep the robot arm in the desired joint positions.

Additionally, the equations of motion of the generative model proposed in the paper do not correspond to the ones used for their simulations. In these, the elasticity coefficient k_i is set to zero modifying the equations of motion to the following ones:

$$f(\tilde{x}, v) = \begin{bmatrix} x'_1 \\ x'_2 \\ \vdots \\ x'_7 \\ \frac{(\Phi_1 - \kappa_1 x'_1)}{m_1} \\ \frac{(\Phi_2 - \kappa_1 x'_2)}{m_1} \\ \frac{(\Phi_3 - \kappa_1 x'_3)}{m_1} \\ \frac{(\Phi_4 - \kappa_2 x'_4)}{m_2} \\ \vdots \\ \frac{(\Phi_7 - \kappa_2 x'_7)}{m_2} \end{bmatrix} \tag{3-8}$$

3-2-2 Outcomes

After the analysis of the implementation proposed in [1] and the assumptions made by the authors and by replicating their experiments, the interesting outcomes that have been obtained are:

- Robot manipulator implementation: it is the first implementation of the Active Inference scheme for a robot manipulator control and therefore represents a proof of concept.
- Embedding a controller in the generative model: based on the work by Karl Friston in [7], it is shown how using a controller as a prior in the generative model can steer the motion as desired.
- Offline implementation: the authors are not closing the loop in their simulations since only the position command is sent to the robot simulation and there is no data obtained from the real process used in the Active Inference scheme.
- Difference between process and model: even though in the paper the generative model and generative process share the same format of equations of motion, the ones used in the simulations differ one from the other as shown in the section before.

On the other hand, the results obtained are conditioned by the following limitations:

- Dynamic model of a robot manipulator with human-like Newtonian dynamics: the equations of motion of both the generative model and generative process use elasticity and viscosity coefficients to obtain the acceleration of the joints. This dynamic model does not represent the true dynamics of the system that is being controlled and therefore the action computed in the Active Inference scheme does not represent the real action needed to perform the desired motion.

- Position control using internal controller from the simulated robot: in the simulation, the robot motion is controlled by sending a desired vector of joint positions to the simulation and an internal position control computes the required torques to move to the desired position. This limits the application of this scheme only to robot manipulators where the position control is already available.

- Real world dynamics: there is no gravity considered in the dynamic model of the system even though it is considered in the simulation. The results obtained are possible thanks to a joint position control that the simulation includes and that is able to compensate for the gravity internally. Since the model used does not include gravity, the applicability of this scheme is again limited to robot manipulators where the joint position control is available.

- Increments in angular position used as force to compute accelerations: in the equations of motion of the generative model, the acceleration of the joints is obtained using joint angular increments (controller signal) instead of forces. This may not be seen as a limitation but a conceptual error of using angular increments as a force that drives the motion.

- Controller: to obtain a stable system, the controller used has been specifically designed and tuned based on the characteristics of the robot manipulator used in the simulations, including not only the error between the desired and actual position but a component to predict the stretched arm singularity. This particular design becomes a limitation when trying to generalise the scheme to different manipulators.

Chapter 4

Robot manipulator

In this chapter, the robot manipulator chosen for the experiments is depicted. The robot manipulator used is the KUKA LBR iiwa robot arm. In order to design controllers for a given robot manipulator, the kinematics and dynamics of this manipulator need to be studied. Therefore, the forward kinematics and dynamics of this robot manipulator are described, together with their implementation. Finally, the experimental setup used in this thesis is reported.

4-1 KUKA LBR iiwa

The KUKA LBR iiwa is a lightweight robot manipulator with 7 degrees of freedom widely used in research. This ensures the existence of a 3D robot model that can be used for simulating the experiments. Besides, the availability of this robot in the DCSC lab has influenced the decision as an implementation in the real robot was initially intended.

Figure 4-1: KUKA LBR iiwa robot manipulator (Image: KUKA)

4-2 Kinematics and dynamics of the robot manipulator

The kinematics and dynamics of the KUKA LBR iiwa robot manipulator are needed in order to control the system. The controller that is described in section 5-2 requires the forward kinematics and the Jacobian matrix, while the Active Inference scheme requires the forward dynamics model to obtain a part of the robot states as depicted in section 5-3.

As mentioned before, the KUKA LBR iiwa robot manipulator has 7 controlled axes that for this thesis they have been defined as shown in figure 4-2 to obtain the end-effector position using the forward kinematics.

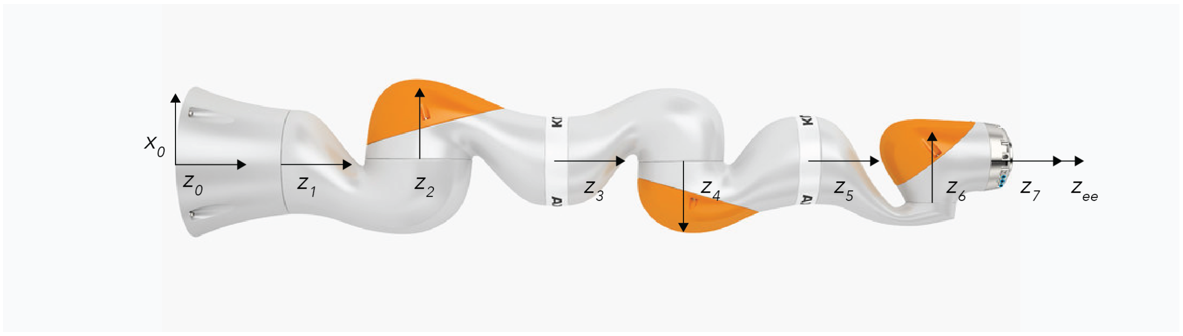


Figure 4-2: KUKA LBR iiwa robot manipulator with the defined axes for each joint

Furthermore, for the forward kinematics problem the lengths of the links are also indispensable. These lengths are shown in table 4-1 referring to the d_i parameters appearing in figure 4-3.

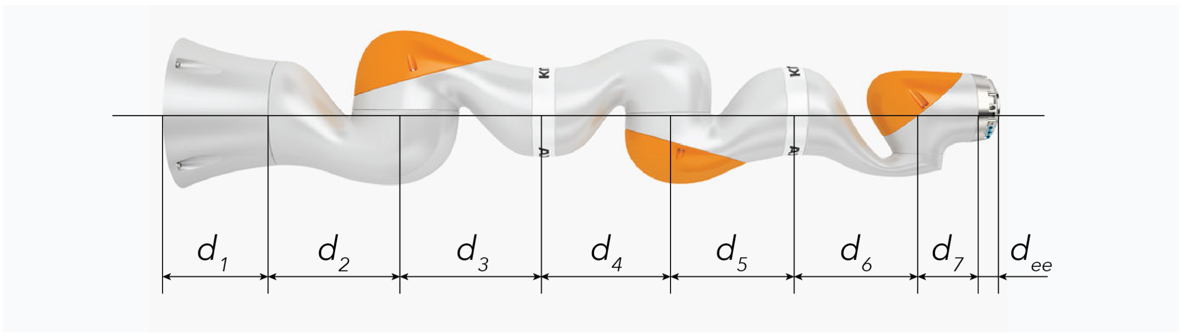


Figure 4-3: KUKA LBR iiwa robot manipulator with distances of the links

d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_{ee}
0.1575	0.2025	0.2045	0.2155	0.1845	0.2155	0.0810	0.0450

Table 4-1: Length of the robot manipulator links in meters as seen in figure 4-3

4-2-1 Forward kinematics

The forward kinematics are used to obtain the end-effector position and orientation of the robot manipulator knowing the joint angles using kinematic equations of the robot manipulator. In other words, it is a transformation from the joint space (q_i) to the cartesian space ($x, y, z, \Phi, \Theta, \Psi$). In this case, only the position (x, y, z) of the end-effector is needed and it is obtained using the Denavit-Hartenberg convention. To do so, the Denavit-Hartenberg (D-H) link parameters are needed. These parameters correspond for each link i to:

- d_i: distance along z_{i-1} to the common normal
- r_i: distance along x_i to the common normal
- α_i: angle between z_{i-1} and z_i measured about x_i
- θ_i: angle between x_{i-1} and x_i measured about z_{i-1}

Given the chosen configuration of the joint axis shown in figure 4-2, the D-H parameters to compute the forward kinematics are shown in table 4-2.

Joint	d	r	α	θ
1	d_1	0	$-\pi/2$	$q(1)$
2	0	$-d_2$	$\pi/2$	$q(2)$
3	d_3	0	$\pi/2$	$q(3)$
4	0	$-d_4$	$-\pi/2$	$q(4)$
5	d_5	0	$-\pi/2$	$q(5)$
6	0	$-d_6$	$\pi/2$	$q(6)$
7	d_7	0	0	$q(7)$
End-effector	d_{ee}	0	0	0

Table 4-2: Denavit-Hartenberg parameters

Once the D-H parameters are defined, the end effector position can be obtained by computing the transformation matrix:

$$[T] = {}^{0}T_n = \prod_{i=1}^{n} {}^{i-1}T_i(\theta_i) \tag{4-1}$$

where

$$ {}^{n-1}T_n = \left[\begin{array}{ccc|c} \cos\theta_n & -\sin\theta_n\cos\alpha_n & \sin\theta_n\sin\alpha_n & r_n\cos\theta_n \\ \sin\theta_n & \cos\theta_n\cos\alpha_n & -\cos\theta_n\sin\alpha_n & r_n\sin\theta_n \\ 0 & \sin\alpha_n & \cos\alpha_n & d_n \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

However, as mentioned before, only the position of the end-effector is required; this corresponds to the top 3 rows of the last column of the transformation matrix $[T]$:

$$[T] = \left[\begin{array}{ccc|c} & \textit{Rotation} & & \textbf{Position} \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

The implementation of the forward kinematics method in Matlab corresponds to the file `IIWA_FW_kinematics.m` and can be seen in appendix A-1. However, the `getTransform` method included in the *robotics* toolbox of Matlab can also be used to obtain the end-effector position. This is possible as the robot chosen, KUKA LBR iiwa, is included in the toolbox as one of the available robots. The `getTransform` method computes the transformation matrix $[T]$ given the joint positions (q) and the name of the joint for which the transformation is required (*iiwa_link_ee* corresponding to the end-effector link in this case).

These two methods have been compared to verify if the `IIWA_FW_kinematics.m` function created provides the same results as the `getTransform` method. This comparison is shown in appendix B-1.

4-2-2 Jacobian

The Jacobian corresponds to the dynamic relationship between two different representations of a system. In this case, the Jacobian matrix (J) relates the velocities in the joint space $(\dot{q})$ to Cartesian space $(\dot{x})$:

$$\dot{x} = J\dot{q} \tag{4-2}$$

Furthermore, the Principle of the Virtual Work [13] demonstrates that the Jacobian matrix also relates the join torques (τ) and the force or torque applied by the end-effector (F):

$$F = J\tau \tag{4-3}$$

this relationship is needed to implement the controller since a force field needs to be converted into a torque field.

For the computation of the Jacobian, the symbolic expression of the end-effector position is obtained by slightly modifying the forward kinematics method described before. The position vector $[x; y; z]$ is an expression which consists on *sin* and *cos* functions that depend on the joint positions q. The Jacobian matrix (J) is then obtained by computing the partial derivatives of each component of the position vector with respect to each of the joint positions q:

$$J = \begin{bmatrix} \frac{\partial x}{\partial q_1} & \frac{\partial x}{\partial q_2} & \cdots & \frac{\partial x}{\partial q_7} \\ \frac{\partial y}{\partial q_1} & \frac{\partial y}{\partial q_2} & \cdots & \frac{\partial y}{\partial q_7} \\ \frac{\partial z}{\partial q_1} & \frac{\partial z}{\partial q_2} & \cdots & \frac{\partial z}{\partial q_7} \end{bmatrix} \tag{4-4}$$

The implementation of the Jacobian matrix function in Matlab is divided in two files to improve the performance of the algorithm. One file corresponds to the symbolic computation of the Jacobian matrix that is time consuming (`IIWA_Jacobian.m`) and therefore only executed once and the other corresponds to the substitution of the joint positions (`IIWA_Jacobian_sub.m`); both files can be seen in appendix A-1.

4-2-3 Forward dynamics

The forward dynamics is a mathematical model that describes how the motion of rigid body, a robot manipulator in this case, changes due to applied torques. This relation becomes essential when the joints accelerations are required, as it will be shown in section 5-3. The dynamics that represent the aforementioned relation is:

$$\ddot{q} = [M(q)]^{-1}\tau + C(q, \dot{q}) + G(q) \tag{4-5}$$

where $q, \dot{q}, \ddot{q}$ represent joint positions, velocities and accelerations, τ the applied torque, $M(q)$ the mass matrix, $C(q, \dot{q})$ the Coriolis and centrifugal forces and $G(q)$ the gravitational force.

However, all the dynamic parameters (inertia, mass, ...) need to be known to analytically compute this relation. As this is not the case, the `forwardDynamics` method included in the *robotics* toolbox of MATLAB is used. This is possible as the robot chosen, KUKA LBR iiwa, is included in the toolbox as one of the available robots. The `forwardDynamics` method computes the joint accelerations ($\ddot{q}$) using the Composite Rigid Body Algorithm [14] given the joint positions (q) and velocities ($\dot{q}$) and the torque applied (τ).

4-2-4 Gravity compensation

As introduced in section 5-2, a gravity compensation torque needs to be computed. In order to obtain the needed torque to compensate the gravity forces existing on the simulation environment, the `gravity_torque` method included in the *robotics* toolbox of MATLAB is used. This is, again, possible as the robot chosen, KUKA LBR iiwa, is included in the toolbox as one of the available robots. The `gravity_torque` method computes the joint torques (τ_g) needed to keep the robot manipulator in a desired position determined by the joint positions (q) by using the inverse dynamics model.

4-3 Experimental setup

In order to simulate the experiments, two main components are needed, namely the 3d model simulation and the platform to run the control algorithm. In this case, the 3d model simulation is provided by the Gazebo® simulator engine, while the platform used to run the control algorithm is MATLAB.

Initially, an offline configuration of the setup shown in figure 4-4 is used to test the experiments, where the 3d model simulation is not used during the execution of the algorithm and a mathematical model is used to represent the robot manipulator in MATLAB. However, the 3d model is used once the necessary commands are obtained from the control loop to verify if the system behaves as predicted.

On the other hand, the online configuration of the setup shown in figure 4-5 uses the 3d model simulation in real time together with the control algorithm, as this uses the sensory information from the Gazebo simulation.

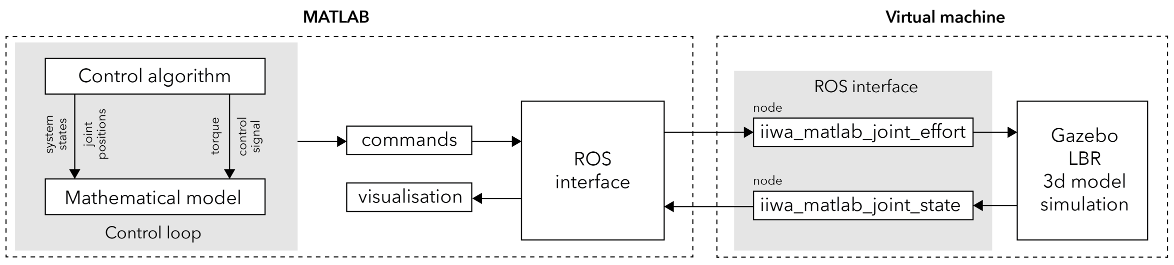


Figure 4-4: Offline configuration of the setup

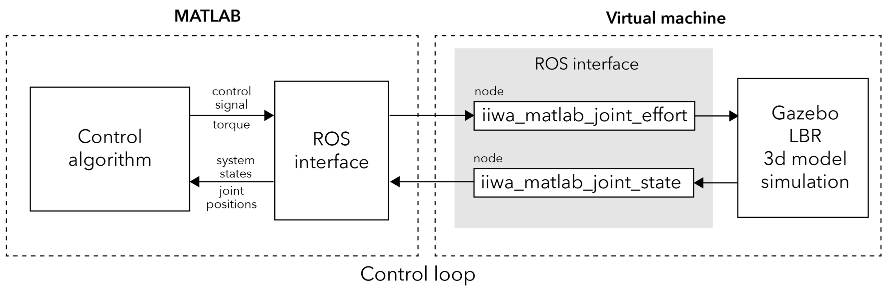


Figure 4-5: Online configuration of the setup

4-3-1 Gazebo simulation

The Gazebo simulator can be used in a virtual machine together with the Robot Operating System (ROS)packages to provide the tools to connect MATLAB to Gazebo through the ROS interface. As mentioned before, since the KUKA LBR iiwa is often used for research, the model of this robot manipulator is available in the virtual machine based on Ubuntu® Linux® that already includes the necessary packages for the connection with MATLAB[1].

The KUKA LBR iiwa robot manipulator that is available in the Gazebo simulator is shown in figure 4-6. The interface that is included with this robot manipulator in the simulation engine provides torque control for each joint. Furthermore, it also allows to modify the physics applied to the simulated world, for instance, the gravity. This is useful when testing simulations as it permits different environments to verify the results.

[1]http://www.mathworks.com/robotics/v3/ros_vm_install

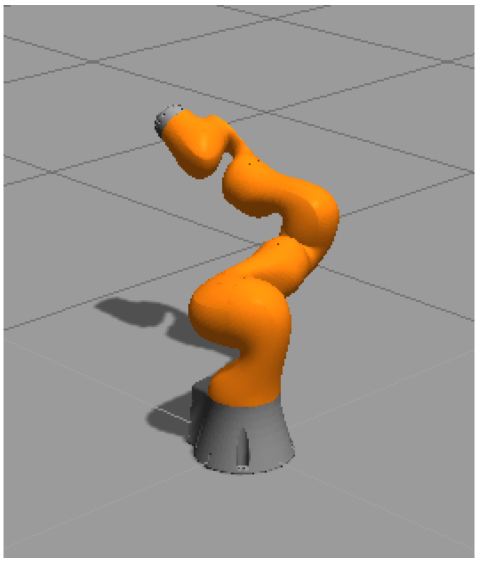


Figure 4-6: KUKA LBR iiwa robot manipulator in the Gazebo simulation engine

Chapter 5

Controller design

Once the current background on robot manipulator control in Active Inference is known, the Active Inference scheme can be designed. This implies the design of a controller to be used as a prior in this scheme. In contrast to what is done in [1], a low-level controller is intended for the reaching task. To demonstrate the possible benefits of this method, the performance of this controller needs to be compared with the current methods available; therefore, a state of the art controller is also needed.

In this chapter, a state of the art controller using an inverse model to obtain the position and velocities used as a benchmark is described. The torque controller designed dispensing with the inverse model is shown together with a gravity compensation. After that, the Active Inference control scheme, where the aforementioned controller serves as a prior, is defined and its main parts are outlined. Finally, the control topologies are shown to ease the understanding and observe the similarities and differences between the 3 schemes.

5-1 State of the art controller

The desire to control a robot manipulator at a low-level (without need of any internal controller) implies the use of a controller where the control signal corresponds to the torque. Joint torque controllers allow to control each joint's torque instead of its position or velocity. This type of controller is useful for human interaction since any external forces are considered in the controller [15].

In this experiment, the state of the art controller is defined as the joint torque controller proposed in [16] where two different torques are computed to control the motion of a robot arm:

- The first torque component (τ_{ff}) corresponds to a feed-forward torque pre-computed using inverse dynamics to obtain the required torque for a desired motion (angular position q_d, velocity $\dot{q_d}$ and acceleration $\ddot{q_d}$).

- The second component (τ_{PD}) corresponds to a PD torque controller to compensate for the errors in joint positions and velocities with respect to the desired ones [17].

Finally, the command that is used to control the robot corresponds to the sum of the torques:

$$\tau = \tau_{ff} + \tau_{PD} \tag{5-1}$$

where,

$$\begin{aligned} \tau_{ff} &= f(q_d, \dot{q}_d, \ddot{q}_d) \\ \tau_{PD} &= k_P(q_d - q) + k_D(\dot{q}_d - \dot{q}) \end{aligned} \tag{5-2}$$

However, based on the availability and use of the signals that the rest of controllers have, the PD torque controller to compensate for errors is applied only to the joint positions (q):

$$\tau_{PD} = k_P(q_d - q), \tag{5-3}$$

and the feed-forward torque τ_{ff} is only computed using the desired position and velocity $(q_d, \dot{q}_d)$.

A schematic of the control loop used for the state of the art controller is shown in Figure 5-1.

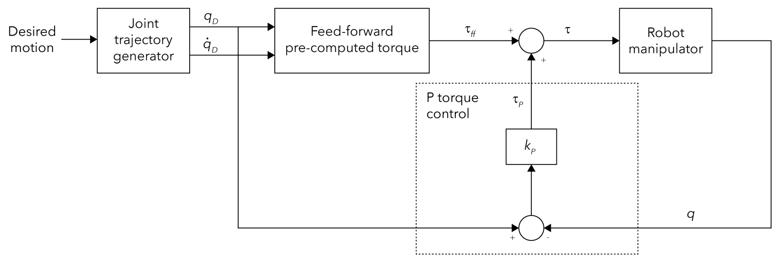


Figure 5-1: Control loop scheme for the joint torque controller with $q_d, \dot{q}_d, \ddot{q}_d$ vectors describing desired motion of each joint, kP and kD being the proportional and derivative gains of the PD controller respectively and $q, \dot{q}$ vectors representing the sensed angular positions and velocities of each joint of the robot arm.

As it can be seen in figure 5-1, the set-point consists on a desired motion used to compute the trajectory. This desired motion is specified with two vectors: one describing the desired joint positions in this motion, while the other one defines the time at which this joint positions should be accomplished. Once the trajectory is obtained using the Piecewise Cubic Hermite Interpolating Polynomial [18], the torque needed to generate this motion is obtained using the inverse dynamics model.

5-2 Torque controller

In section 3-1, the relevance of the inverse model has been pointed out. The obtention of this model is not trivial and, therefore, dispensing with it reduces the difficulty to obtain a robot manipulator controller. Based on this, the designed controller described here does not require an inverse model to compute the control signal required to accomplish the reaching task.

As mentioned before, a low-level controller is intended and since the robot interface provided by the Gazebo simulation offers the possibility of a low-level torque control, the controller that will generate the desired behaviour consists on a torque controller.

The controller used for this experiment is based on the idea proposed by Mohan and Morasso [19]. They proposed the Passive Motion Paradigm (PMP) as an alternative to optimal control. This paradigm makes use of force fields in the context of motor control to attract the end-effector position to a desired position. As it can be seen in figure 5-2, the PMP consists on a closed loop where the force field is mapped into a torque field to obtain joint velocities and then, using the Jacobian and an integrator, mapped into the end effector position.

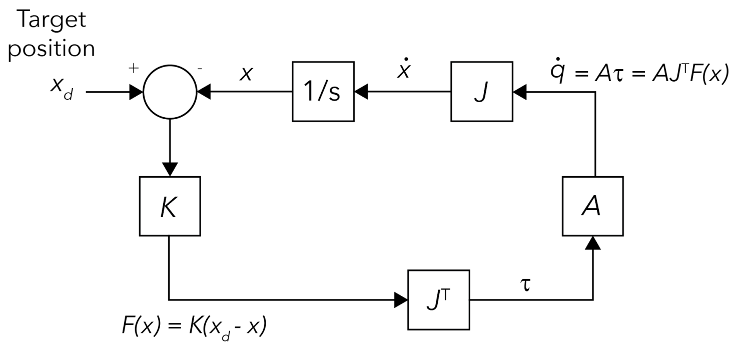


Figure 5-2: Passive Motion Paradigm scheme where x_d corresponds to the desired position, K is a matrix that determines the shape and intensity of the force field F, A is a matrix that transforms the torque field to the degree of participation of each joint and J is the Jacobian matrix of the kinematic transformation.

However, for the implementation proposed in this thesis, a torque controller is intended and, therefore, only the part of the original PMP scheme where the torque is obtained given the actual end-effector position and the desired one is used. The mathematical formulation of the controller designed is:

- First, a virtual attractive force field F to the target is defined:

$$F(x) = K(x_d - pos_{ee}), \tag{5-4}$$

  where pos_{ee} corresponds to the vector that identifies the pose of the end-effector and x_d the target position, both, in Cartesian coordinates; and K a matrix that determines the shape and intensity of the force field.

- Second, the force vector F is mapped into an equivalent torque vector τ_c:

$$\tau_c = J^T F, \tag{5-5}$$

  where J is the Jacobian matrix of the kinematic transformation.

The designed controller computes the needed torque τ_c obtained from the equation above and this torque is added to the torque τ_g which corresponds to the needed torque to compensate for gravity based on the joint positions q. As described in the mathematical formulation, the end-effector position is needed and, therefore, the forward kinematics method described in section 4-2 is used to compute this position given the joint positions q. The control loop of the designed controller is shown in figure 5-3.

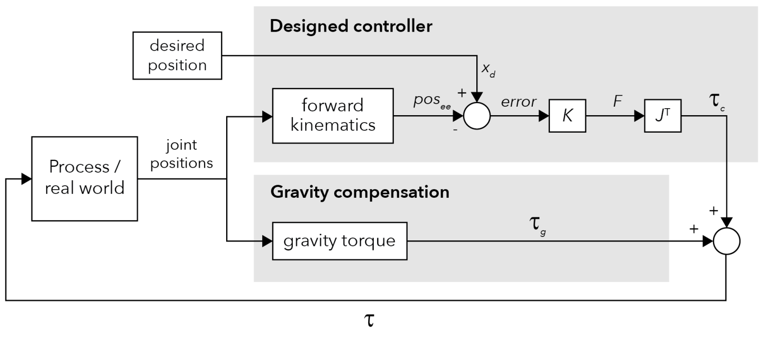


Figure 5-3: Control loop of the torque controller designed including gravity compensation

5-2-1 Gravity compensation

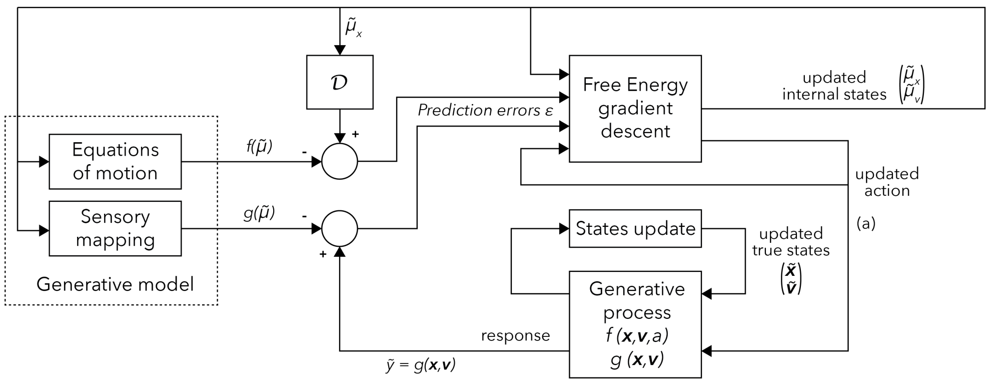
When the controller wants to be tested in the simulation, to bring this simulation closer to the real world robot, the gravity needs to be considered. This leads to the need of a compensation torque to be added to the control signal obtained from the controller that is defined in section 4-2-4.

The gravity compensation is required since the designed controller and the PMP in which it is based do not consider external forces, such as gravity. However, these can be added in addition to the torque obtained as mentioned in [20]; this is shown in figure 5-3.

5-3 Active Inference scheme

In order to understand how the Active Inference scheme needs to be implemented, an overview of the process is needed. The Active Inference loop consists on a prediction error minimization iterative process as shown in figure 5-4. This loop consists on the offline implementation of the Active Inference process from the `spm_ADEM.m` file contained in the Statistical Parametric Mapping (SPM)[1] suite for MATLAB originally developed by Karl Friston.

[1]More information on the SPM software: https://www.fil.ion.ucl.ac.uk/spm/



$\tilde{\mu}_x$
$\mathcal{D}$
Free Energy gradient descent
updated internal states $\begin{pmatrix}\tilde{\mu}_x\\ \tilde{\mu}_v\end{pmatrix}$
Prediction errors ε
Equations of motion
$f(\tilde{\mu})$
Sensory mapping
$g(\tilde{\mu})$
Generative model
updated action
(a)
States update
updated true states $\begin{pmatrix}\tilde{\mathbf{x}}\\ \tilde{\mathbf{v}}\end{pmatrix}$
Generative process $f(\mathbf{x},\mathbf{v},a)$ $g(\mathbf{x},\mathbf{v})$
response
$\tilde{y} = g(\mathbf{x},\mathbf{v})$

Figure 5-4: Active Inference scheme: prediction error minimisation based on `spm_ADEM.m`

The following table shows the relation between the Active Inference scheme variables from figure 5-4 and the robot manipulator variables that are used throughout the document:

Active Inference		**Robot manipulator**	
Concept	**Variable**	**Variable**	**Concept**
hidden states	$\mathbf{x} = \begin{bmatrix}\mathbf{x}\\ \dot{\mathbf{x}}\end{bmatrix}$	$\begin{bmatrix}\mathbf{q}\\ \dot{\mathbf{q}}\end{bmatrix}$	true joint positions and velocities
internal hidden states	$\mu_x = \begin{bmatrix}\mu_x\\ \dot{\mu}_x\end{bmatrix}$	$\begin{bmatrix}q\\ \dot{q}\end{bmatrix}$	modelled joint positions and velocities
causal states	v	x_d	target position
internal causal states	μ_v	x_d	target position
action	a	τ	torque signal
prior belief	Φ	Φ	controller signal

Table 5-1: Active Inference - robot manipulator correspondence of variables

As it can be seen in figure 5-4 and the figures in the coming sections, the target position x_d does not appear as part of the scheme. This is because this variable is only used to initialise μ_v at the beginning of the execution.

Now that the controller that will be used as a prior has been defined and the Active Inference loop has been shown, the Active Inference scheme that will make use of this controller can be studied. To do so, three important aspects have to be defined, these aspects needed in order to apply the Active Inference scheme are: the control loop, the generative model and the generative process.

5-3-1 Control loop

In terms of control, the scheme of the control loop is a fundamental aspect for understanding what is happening. This control loop consists of a schematic visualisation of the process by which the control signal is generated and how it is used as a feedback for the system that needs to be controlled. In this case, two different options for the control loop are possible when using the Active Inference scheme: offline and online. These two schemes are detailed in what follows.

Offline control loop

The offline configuration of the scheme does not consider a real world process, but an internal simulation of it (generative process). In this case, the control loop includes an evaluation of the generative process simulated in order to obtain the causal states and the evolution of the true hidden states derivatives. The causal states from the generative process ($\tilde{y}$, used as the true response from the system) and the internal states $\tilde{\mu}$ are used to compute the prediction error. These internal states are also used in the generative model evaluation, which includes the controller signal Φ as the prior belief.

The prediction errors, the internal states are updated through a gradient descent on the Free Energy. Since the real process is part of the scheme, the true states $\tilde{x}$ are also updated by integrating them. Furthermore, the updated action that needs to be applied to the generative process is also obtained through a gradient descent on the Free Energy. The schematic of this control loop is shown in figure 5-5.

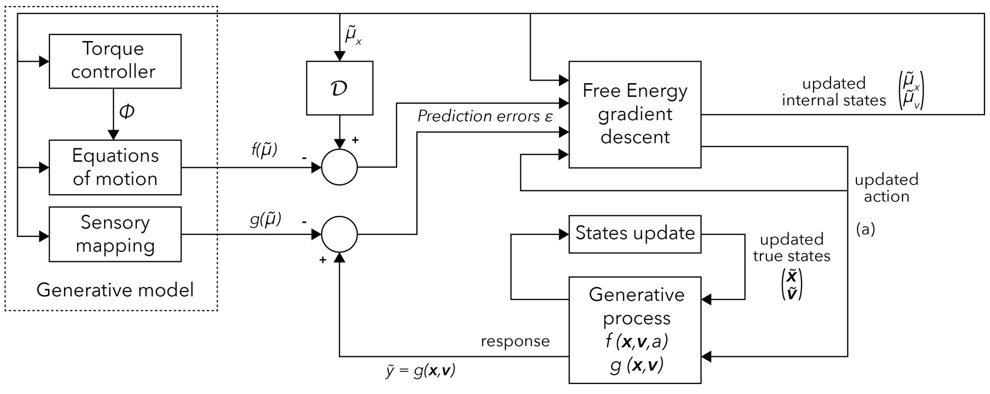

Torque controller
Φ
Equations of motion
Sensory mapping
Generative model
$\tilde{\mu}_x$
$\mathcal{D}$
$f(\tilde{\mu})$
$g(\tilde{\mu})$
Prediction errors ε
Free Energy gradient descent
updated internal states $\begin{pmatrix}\tilde{\mu}_x\\ \tilde{\mu}_v\end{pmatrix}$
updated action
(a)
States update
updated true states $\begin{pmatrix}\tilde{\mathbf{x}}\\ \tilde{\mathbf{v}}\end{pmatrix}$
Generative process $f(\mathbf{x},\mathbf{v},a)$ $g(\mathbf{x},\mathbf{v})$
response
$\tilde{y} = g(\mathbf{x},\mathbf{v})$

Figure 5-5: Proposed offline Active Inference scheme control loop

Using this control loop, when the Active Inference iterations are completed, the actions obtained can be used as a feed-forward torque similar to what is done in the state of the art controller.

Online control loop

On the other hand, the online configuration uses the real world process. In this case, the generative process is no longer part of the control system. Since it is a real world process, the hidden states of this process are unknown from the perspective of the control system and the evolution of them depends on the action received from the control system and it is not used for the control loop.

Similar to the offline control loop, only the response y, which corresponds to the sensory mapping, is used for the prediction errors computation. However, the generalised response $\tilde{y}$ is needed for this computation. As it can be seen in the schematic of this control loop in figure 5-6, the rest of the loop remains the same.

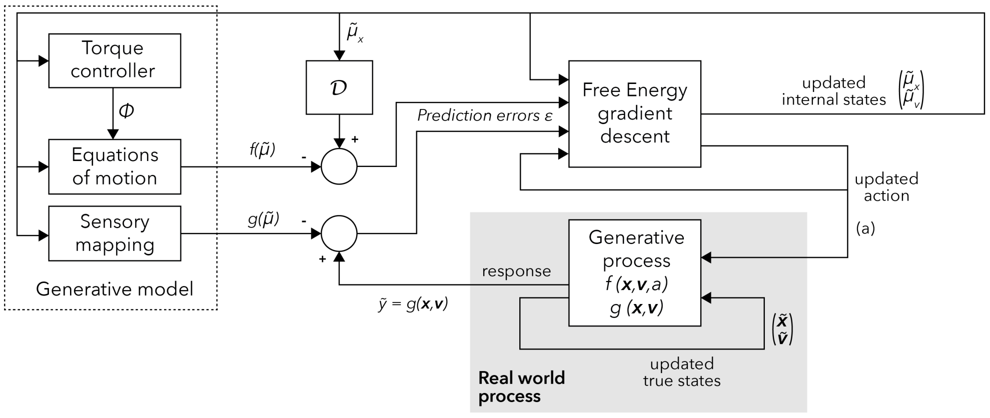

Torque controller
Φ
Equations of motion
Sensory mapping
Generative model
$\tilde{\mu}_x$
$\mathcal{D}$
$f(\tilde{\mu})$
$g(\tilde{\mu})$
-
+
Prediction errors ε
Free Energy gradient descent
updated internal states $\begin{pmatrix}\tilde{\mu}_x\\\tilde{\mu}_v\end{pmatrix}$
updated action
(a)
Generative process
$f\,(\mathbf{x},\mathbf{v},a)$
$g\,(\mathbf{x},\mathbf{v})$
response
$\tilde{y} = g(\mathbf{x},\mathbf{v})$
$\begin{pmatrix}\tilde{\mathbf{x}}\\\tilde{\mathbf{v}}\end{pmatrix}$
updated true states
Real world process

Figure 5-6: Proposed online Active Inference scheme control loop

5-3-2 Generative process

The generative process corresponds to the real world process. In the offline situation, it corresponds to a model of the real world (which should be the closest to reality), while in the online configuration, this process remains outside the control loop as shown in 5-6. When defining this process for the offline configuration, an important part of it is the true plant dynamics function f that represents the equations of motion and that is defined as:

$$\mathbf{f}(\mathbf{x},\mathbf{v},a) = \begin{bmatrix}\dot{\mathbf{q}}\\ \dot{\mathbf{q}}'\end{bmatrix} = \begin{bmatrix}\mathbf{q}'\\ f_1(\mathbf{q},\mathbf{q}',a)\end{bmatrix} = \begin{bmatrix}\mathbf{q}'_1\\ \mathbf{q}'_2\\ \vdots\\ \mathbf{q}'_7\\ f_1(\mathbf{q_1},\mathbf{q_1'},a_1)\\ f_1(\mathbf{q_2},\mathbf{q_2'},a_2)\\ \vdots\\ f_1(\mathbf{q_7},\mathbf{q_7'},a_7)\end{bmatrix} \tag{5-6}$$

where $\mathbf{q}'$ corresponds to the true velocities of the joints and f_1 represents a forward dynamics model function that returns the true accelerations $\dot{\mathbf{q}}'$ given the true hidden states $(\mathbf{q}, \mathbf{q}')$ and the computed action a.

The other important part of the generative process is the function g that corresponds to the sensory mapping that returns the true position of the end-effector:

$$\mathbf{g}(\mathbf{x}, \mathbf{v}) = \begin{bmatrix} \mathbf{x} \\ \mathbf{v} \\ \mathbf{pos_{ee}} \end{bmatrix} \tag{5-7}$$

where $\mathbf{pos_{ee}}$ corresponds to the true position of the end-effector of the robot.

5-3-3 Generative model

On the other hand, the generative model corresponds the internal model of the process which is does not need to necessarily be equal to the generative process. In the same way, the plant dynamics function f represents the modeled equations of motion and it is defined as:

$$f(x, v) = \begin{bmatrix} \dot{q} \\ \dot{q}' \end{bmatrix} = \begin{bmatrix} q' \\ f_1(q, q', \tau) \end{bmatrix} = \begin{bmatrix} q_1' \\ q_2' \\ \vdots \\ q_7' \\ f_1(q_1, q_1', \tau_1) \\ f_1(q_2, q_2', \tau_2) \\ \vdots \\ f_1(q_7, q_7', \tau_7) \end{bmatrix} \tag{5-8}$$

where q' corresponds to the velocities of the joints (part of the hidden states) and f_1 represents a forward dynamics model function that returns the accelerations given the modelled positions q and velocities q' and the computed torque τ (controller signal).

The function g corresponds to the sensory mapping that returns the modelled end effector position of the robot arm:

$$g(\mu_x, \mu_v) = \begin{bmatrix} \mu_x \\ \mu_v \\ pos_{ee} \end{bmatrix} \tag{5-9}$$

where pos_{ee} corresponds to the end-effector position obtained through the forward kinematic model given the angular positions of the joints (hidden states q).

5-4 Active Inference scheme models

As mentioned in previous chapters, the generative model and generative process play an important role in the Active Inference framework. Both of them are composed by equations of motion that describe the dynamics of the system and sensory mappings that relate the hidden states with the sensations, which in this case are the end-effector position.

The choice of these two aspects, equations of motion and sensory mapping, becomes a crucial aspect of this implementation. Based on the related work by Pio-Lopez in [1] and the methods given in section 4-2, there are different options available when choosing the aforementioned aspects.

5-4-1 Equations of motion

For the equations of motion of the generative model and generative process, these two options are considered:

1. Forward dynamics model (`forwardDynamics` method) from the robotics toolbox (see section 4-2-3)
2. Newtonian dynamics model from the paper by Pio-Lopez [1] (see section 3-2)

For the generative process, the forward dynamics model (option 1) seems to be the most appropriate as it corresponds to the model which would be closer to the true dynamics of the model as it is based on the `urdf` file which includes all the mechanical information of the robot manipulator used in the simulation.

On the other hand, for the generative model, any of the options available can be accepted as the internal model of the robot could differ from the real one. However, when using option 2, the gravity cannot be considered as this model does not include the effect of gravity on the acceleration.

Based on the theoretical definition of Active Inference, using the same option for the generative process and the generative model is supposed to be the optimal scenario. This is the case because minimizing the prediction error when using the same model implies that the difference between the true and internal states is also minimized.

5-4-2 Sensory mapping

For the sensory mapping of the generative process and generative model, which consists on the forward kinematics returning the end-effector position pos_{ee} given the joint positions q, two options are available:

1. Forward kinematics model (`getTransform` method to obtain the transformation matrix) from the robotics toolbox (see section 4-2-1)
2. Forward kinematics function created based on the robot information (see section 4-2-1)

As mentioned in section 4-2-1, both options are equivalent. Therefore, any possible combination between these two options will result in the same response as shown in appendix B-1. However, the ease of changing parameters in the function created (option 2) allowing to modify the sensory mapping, has been considered to use this option as the one for the generative model.

On the other hand, the generative process sensory mapping uses the forward kinematics model (option 1) so that the changes applied to the function created do not affect the true sensory mapping. This will allow for future work studying the influence of model deviations in the sensory mapping.

5-4-3 Testing and conclusion

In this section, the simulations carried out to verify what has been mentioned earlier in this chapter are shown and the conclusions regarding the optimal options are described.

Equations of motion

Regarding the equations of motion, it has been mentioned that the optimal option would require the generative model and the generative process to have the same equations of motion. However, the four different combinations are tested to verify this initial hypothesis.

The following simulations to chose the optimal equations of motion model for the generative process and generative model are done:

#	**Generative model**	**Generative process**
1	`forwardDynamics` model	`forwardDynamics` model
2	Newtonian dynamics (Pio-Lopez)	Newtonian dynamics (Pio-Lopez)
3	`forwardDynamics` model	Newtonian dynamics (Pio-Lopez)
4	Newtonian dynamics (Pio-Lopez)	`forwardDynamics` model

Table 5-2: Active Inference scheme simulations for equations of motion choice

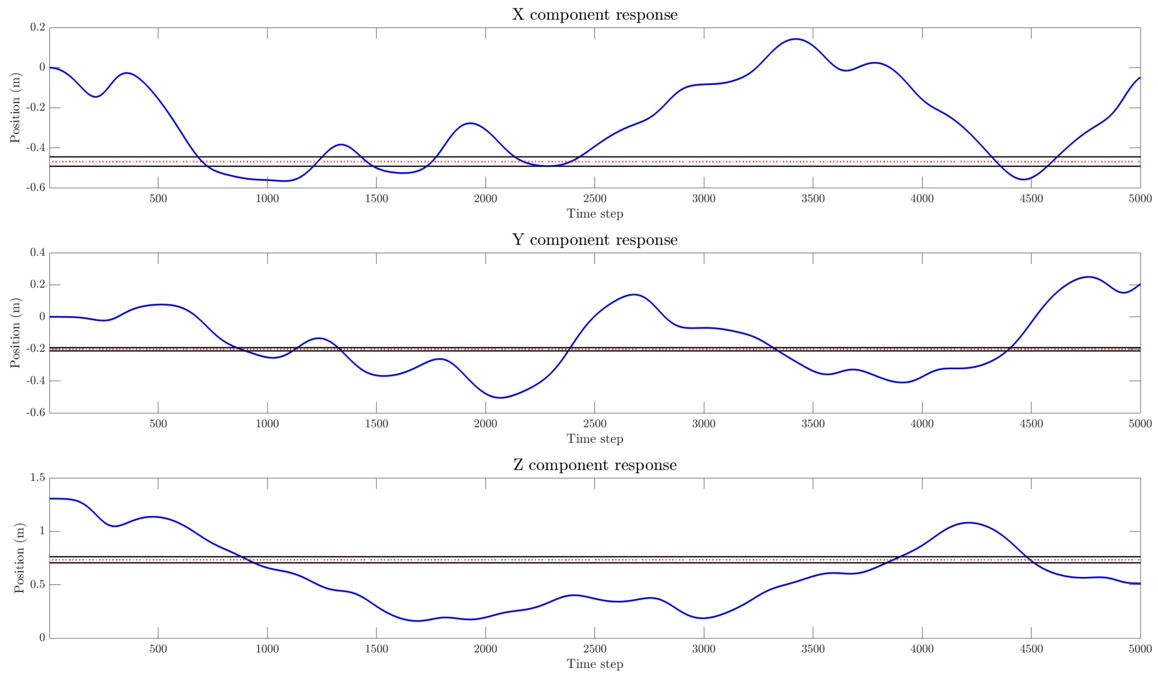


Figure 5-7: Active Inference scheme response for equations of motion simulation 1; response: blue solid, target position: red dotted, 5%: black solid

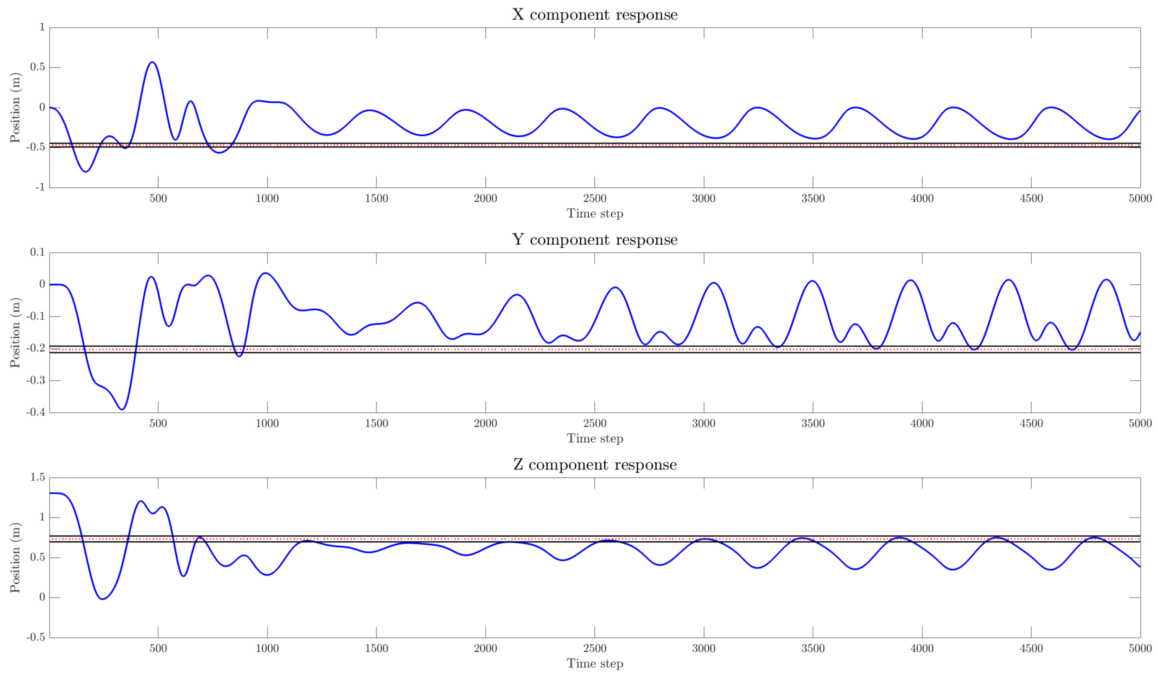


Figure 5-8: Active Inference scheme response for equations of motion simulation 2; response: blue solid, target position: red dotted, 5%: black solid

From simulations number 1 and 2 (figures 5-7 and 5-8) in which the generative model and the generative process are the same, which is supposed to be optimal, it can be seen that the system is not converging towards the desired position.

Simulation 1 (figure 5-7) has a response that is becoming unstable and does not converge at all. This is an unexpected behaviour as the initial hypothesis implied that this case would be the optimal. Many simulations have been carried out, checking the model and the signals, modifying the initial conditions and the parameters of the simulation, trying to figure out the cause of this behaviour. However, the reason has not been found during the process of this thesis. Further research on the implementation of the Active Inference framework by the SPM suite, the forward dynamics model used and the foundational basis of this framework is required to discern whether this is a particular case.

In simulation 2 (figure 5-8), the robot manipulator shows an oscillatory behaviour tending to instability around a point that is not close to the desired goal. The reason why this is happening comes from the form of the equations of motion given in 3-2:

$$\ddot{q}_i = \frac{(\Phi_i - k_i x_i - \kappa_i x_i')}{m_i}.$$

In these equation, where the acceleration is computed, even if the torque is not equal to zero, the acceleration can be zero and then the agent believes it does not have to move. This is possible given a combination of joint positions x_i which, when multiplied by k_i, is equal to the computed torque Φ_i and therefore, even if the control signal is different from 0 because the end-effector is not at the desired position, the acceleration that drives the motion can be

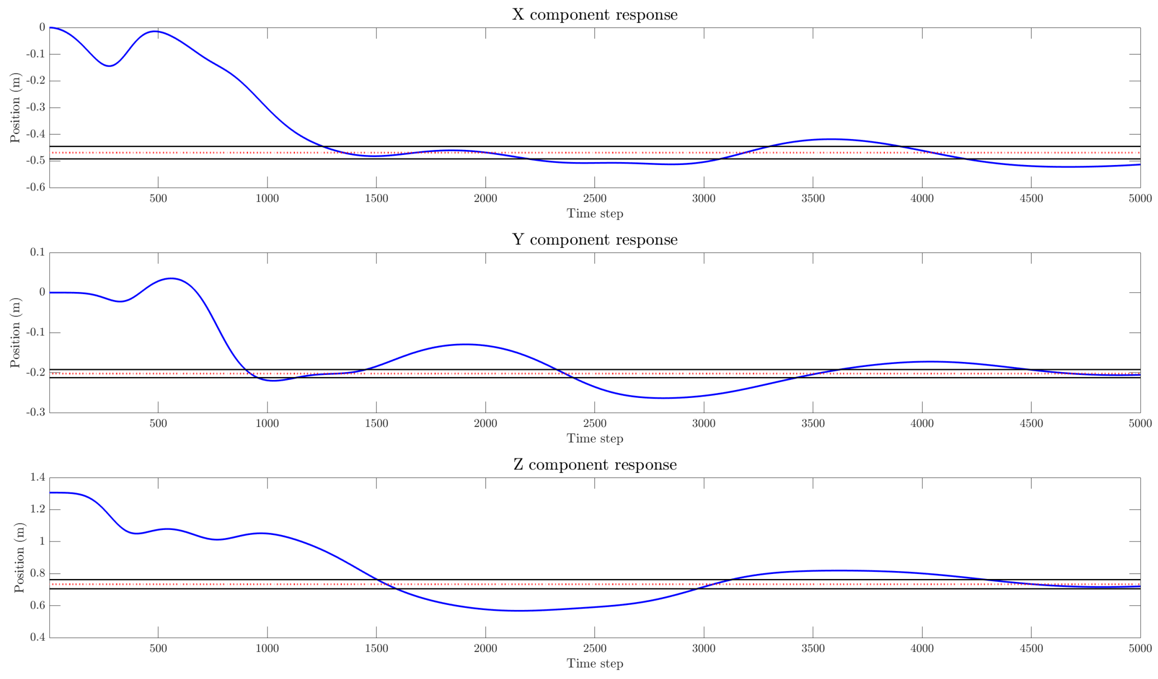


Figure 5-9: Active Inference scheme response for equations of motion simulation 3; response: blue solid, target position: red dotted, 5%: black solid

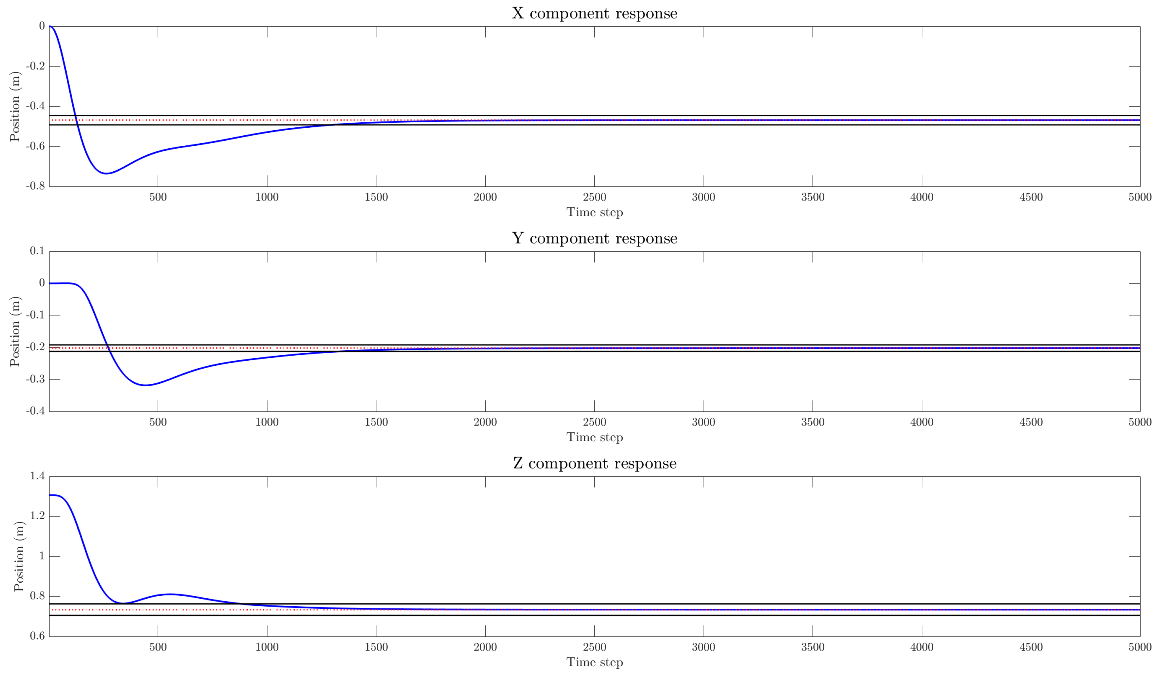


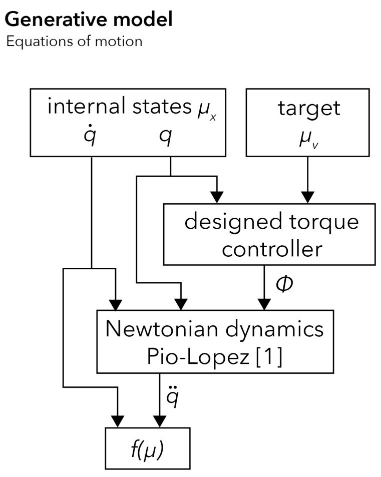
Figure 5-10: Active Inference scheme response for equations of motion simulation 4; response: blue solid, target position: red dotted, 5%: black solid

equal to 0. This is also verified by changing the parameters of this equation and therefore the converging point varies, but never reaching the desired one.

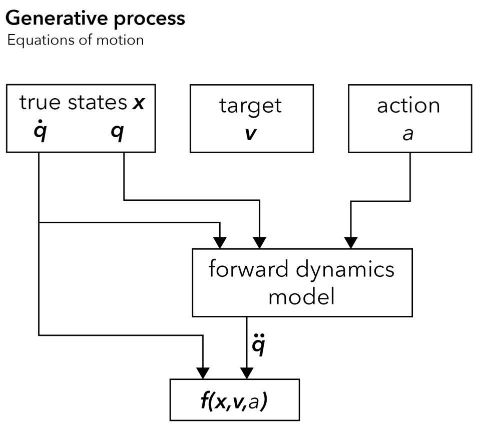
On the other hand, simulations 3 and 4 (figures 5-9 and 5-10), where the models are combinations of the two options available, show a convergence towards the desired end-effector position. However, simulation 3 has a slower response with respect to simulation 4 because the dynamics used in generative process of the former correspond to the Newtonian dynamics one, while the latter uses the real forward dynamics model.

Furthermore, as mentioned before, the most appropriate option would be using the forward dynamics model from the MATLAB *robotics* toolbox for the generative process (which corresponds to simulation 4). This choice is made because when the offline Active Inference scheme is used as a feed-forward controller, in the case of simulation 3, the command signals obtained from this scheme correspond to the signals necessary to control a process described by the Newtonian dynamics of the paper by Pio-Lopez [1]. These signals are not sufficient to control the real process of the simulation which is determined by the the `forwardDynamics` function from the *robotics* toolbox.

Therefore, the schemes corresponding to the definitive equations of motion functions for the generative model and the generative process are shown in figure 5-11. The MATLAB functions corresponding to both equations of motion (`IIWA_spm_fx_robot_dem_reach_offline` and `IIWA_spm_fx_robot_adem_reach_offline`) can be found in appendix A-2-1.



(a) Generative model



(b) Generative process

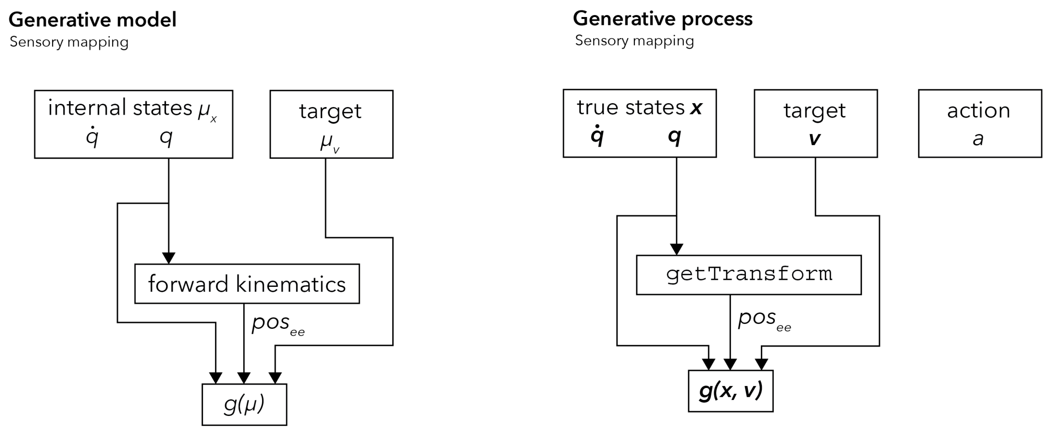
Figure 5-11: Equations of motion schemes

Once these equations have been defined, the controller gain K is optimized through an iterative process of simulations. These simulations and the final choice for this gain that is used in the simulations for comparison are shown in appendix B-2.

The controller gain K is set to: $\begin{bmatrix} 40 & 0 & 0 \\ 0 & 40 & 0 \\ 0 & 0 & 40 \end{bmatrix}$.

Sensory mapping

As shown in appendix B-1 and described in 5-4-2, the schemes corresponding to the definitive sensory mapping functions for the generative model and the generative process are shown in figure 5-12. The MATLAB functions created corresponding to these sensory mappings (`IIWA_spm_gx_robot_dem_reach_offline` and `IIWA_spm_gx_robot_adem_reach_offline`) can be found in appendix A-2-2.



(a) Generative model **(b)** Generative process

Figure 5-12: Sensory mapping schemes

Conclusion

Once the Active Inference scheme models have been chosen the simulations to compare this method with the other controllers proposed in chapter 5 can be executed. The final Active Inference scheme including the controller and the final equations of motion and sensory mappings can be seen in figure 5-13.

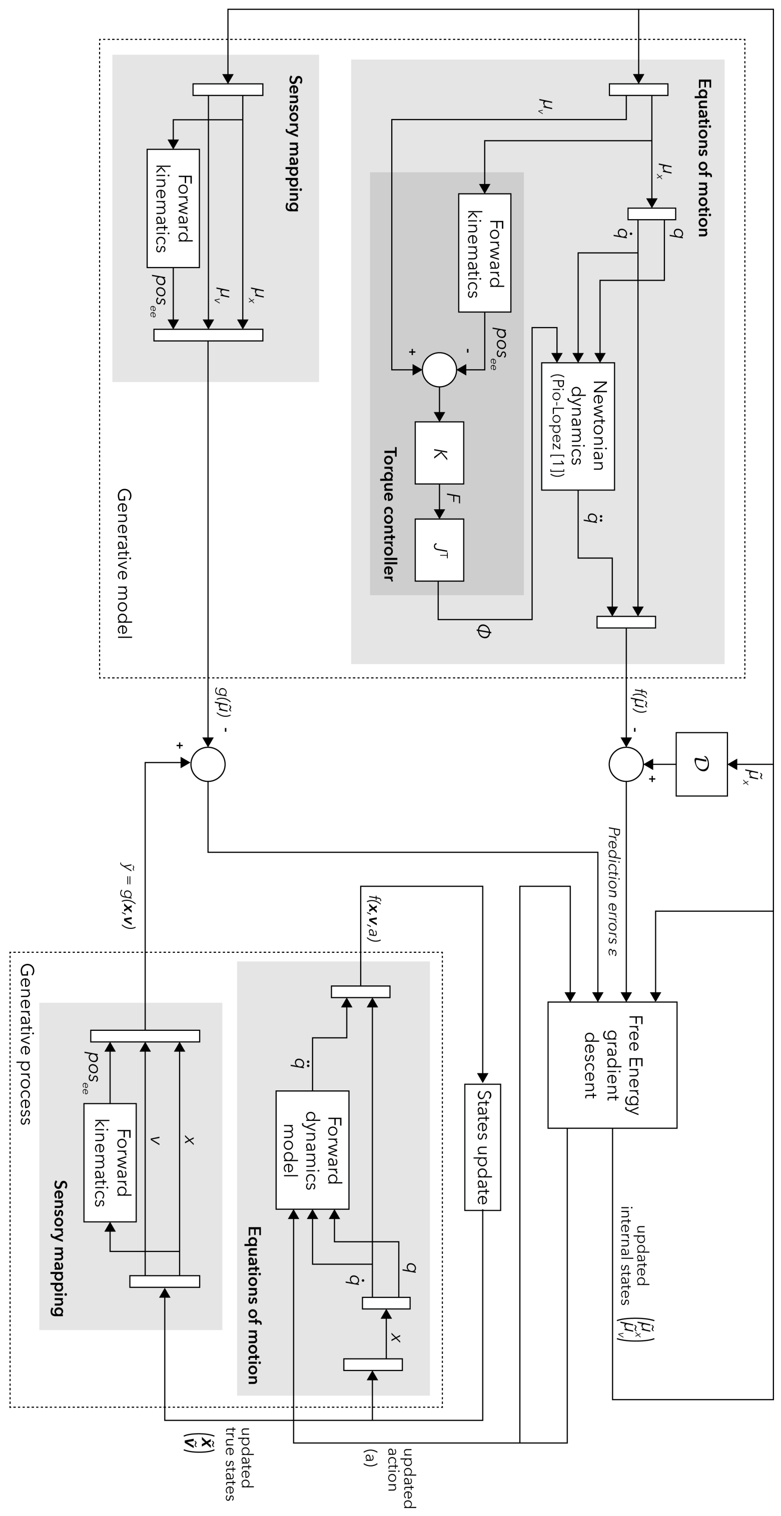


Figure 5-13: Definitive offline Active Inference scheme for the simulations

5-5 Control topologies

In order to obtain a better picture of the controllers, the three control topologies used are shown here. This allows the reader to understand better the comparison that is described in the following chapter as all the variables and parameters can be related through the following figures.

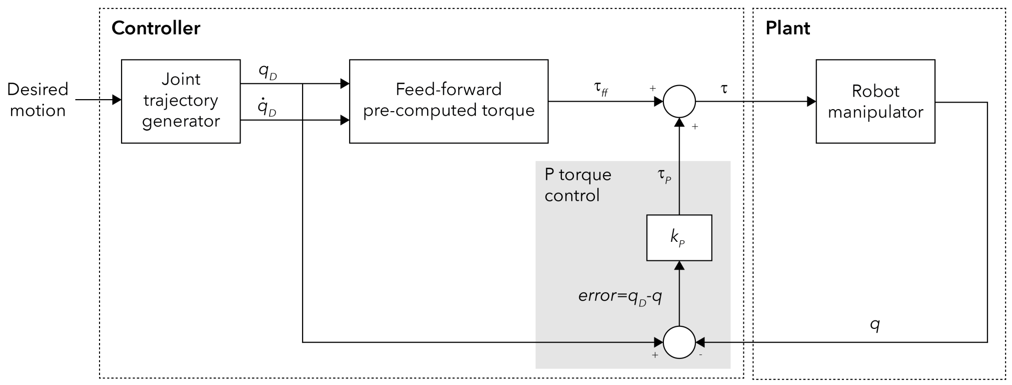


Figure 5-14: State of the art controller topology

In figure 5-14, the state of the art controller topology is shown. In this figure, the desired motion is specified as a vector of desired positions at a certain time; using this, the trajectory is generated $(q_D, \dot{q}_D)$. The feed-forward torque is pre-computed to satisfy the desired trajectory using the trajectory generated. During the simulation, these torques are the control signal sent to the robot manipulator adding an extra compensation torque that regulates possible deviations between the desired and actual trajectory.

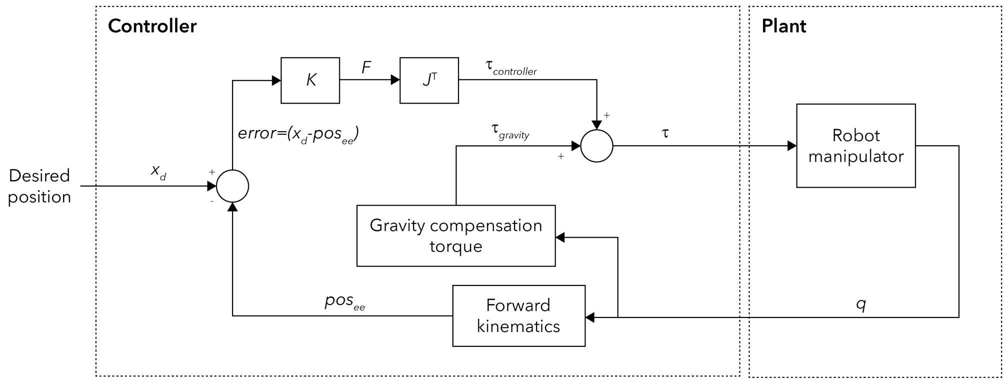


Figure 5-15: Torque controller designed topology

In figure 5-14, the control topology corresponding to the designed torque controller is shown. In this case, the controller computes a torque $\tau_{controller}$ based on the difference between the current position pos_{ee}, obtained using a forward kinematics model, and the desired position x_d by using the Jacobian matrix J. However, the gravity compensation torque $\tau_{gravity}$ is added to the controller one to obtain the control signal sent to the robot manipulator.

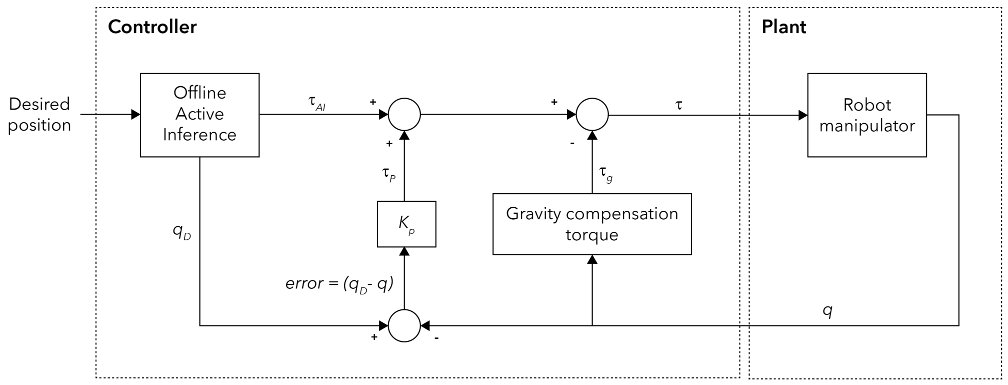


Figure 5-16: Offline Active Inference scheme control topology

In figure 5-16, the offline Active Inference scheme topology is shown. In this figure, the offline Active Inference block corresponds to the process shown in figure 5-13, where, after all the iterations are completed, the action signals (torques) are used as a feed-forward control signal (similar to the benchmark topology, figure 5-14). Inspired by the benchmark controller, a compensation torque to regulate possible deviations is also included in this scheme. Finally, the gravity compensation torque is also added to the control signal.

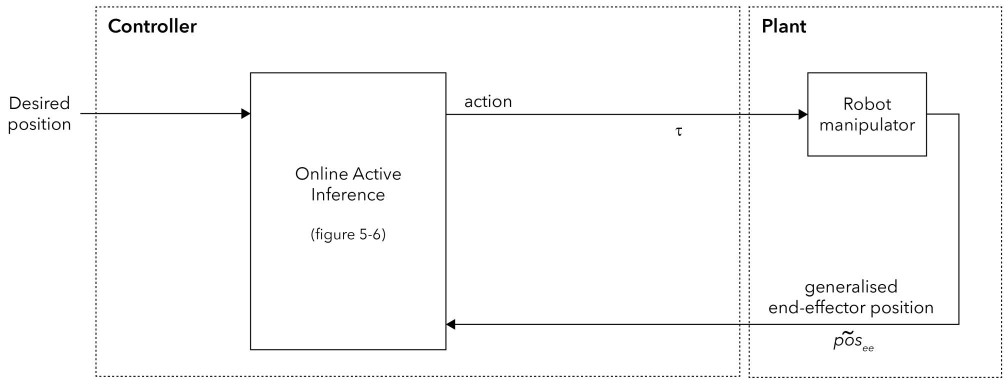


Figure 5-17: Online Active Inference scheme control topology

In figure 5-17, the control topology corresponding to the online Active Inference scheme is shown. In this case, the online Active Inference block corresponds to the process shown in figure 5-6 where the end-effector position corresponds to the input together with the desired position and the action signal (torque) is used as the control signal sent to the robot manipulator.

As it can be seen in figures 5-14, 5-15, 5-16, the controller blocks share the same inputs and outputs, as well as the real world process (plant). The controllers proposed use the joint positions obtained from the robot manipulator simulation to compute the torque (control action) that needs to be sent to the simulation in order to satisfy the desired task (a reaching task in this case). However, the state of the art controller requires the desired motion to generate the trajectory to reach a target position instead of the desired end-effector position, this implies that the manipulator motion can be defined but requiring the inverse dynamics model unnecessary in the two other controllers. On the other hand, the online Active Inference scheme (figure 5-17), which has been discarded (see section 6-0-2), should only have access the end-effector position from the robot manipulator, modifying, in this case, the input signal used in the controller.

Chapter 6

Simulations and results

In this chapter, the simulations necessary to verify if the Active Inference scheme can be used for robot control are done. Together with the simulations needed to compare the Active Inference scheme with the designed controller as a prior to the designed controller itself and the benchmark controller shown in chapter 5.

6-0-1 Preliminary considerations

Some aspects concerning the simulations need to be considered beforehand. While the controller gains are optimized throughout an iterative process of simulations, there are constants such as the sampling time that need to be defined. In this case, the fast dynamics of the system that is being controlled (robot manipulator) require a low value for this parameter. After simulations with the state of the art controller and considering that a robot manipulator presents fast dynamics, it has been seen that a sampling time of 0.01 seconds is sufficient to capture the dynamics of the system to be able to control it.

Besides, the Active Inference scheme process involves the configuration of different parameters that are outlined in what follows.

- The embedding order parameters corresponds to the amount of derivatives of the state included in the generalized coordinates vector. Based on Karl Friston's recommendations for the hidden states, a value between 4 and 6 is sufficient to capture the necessary dynamics of a trajectory. For the causal states, as these are defined constant, a standard value is used.

- Error precisions correspond to the inverse covariance of the noise considered in the signals. Values equal or higher than $\exp^{16}$ correspond to a no-noise scenario. In this thesis, the no-noise scenario is chosen to prove the validity of the implementation proposed with ideal conditions.

- Integration time used to update the states in the Active Inference loop which has to be the same as the sample time in order to obtain states and actions for each of the time steps; as mentioned before, due to the fast dynamics a low value is chosen.

The following table lists the final values chosen for these parameters that remain invariant in all the simulations carried out:

Parameter	**Value**
Embedding order of hidden states	4
Embedding order of causal states	2
Error precision of hidden states	$\exp^{16}$
Error precision of sensations	$\exp^{16}$
Integration time (sample time)	0.01

Table 6-1: Active Inference parameters for simulations

6-0-2 Control loop considerations

Regarding the control loop, only the offline implementation of the Active Inference scheme has been considered, this implementation corresponds to the one shown in figure 5-5. The online implementation initially considered has been discarded due to:

- Computational time: each iteration of the `spm_ADEM` implementation of the Active Inference scheme where the states are updated takes about 1 second to complete. Therefore, for a system with fast dynamics like the robot manipulator in which a low sample time is required, the implementation in real time is currently not possible.
- Current Matlab implementation: the current Active Inference scheme implementation in Matlab, included in the Statistical Parametric Mapping (SPM) suite, is not prepared for an online implementation as it was originally developed for offline simulations. This results from the true hidden states $\mathbf{x}$ being part of the scheme, when in an online implementation, these states are unknown from the Active Inference scheme perspective.
- Generalised coordinates in the system response: the Active Inference scheme requires not only the response of the system y but the generalised coordinates of this variable $\tilde{y}$ as shown in 5-6. Generally, the derivatives included in the generalised coordinates are not available directly as outputs of the real process.

Thus, all the Active Inference scheme simulations that are shown in this chapter correspond to the offline Active Inference scheme control topology shown in figure 5-16.

6-0-3 Simulation settings and considerations

All simulations carried out in this chapter consist on a reaching task performed by the robot manipulator. This reaching task consists on the robot manipulator moving from point A to B, in which A corresponds to the *home* configuration of the robot manipulator, which means that the joint positions are equal to zero and corresponds to the upright configuration; and B, the final position, consists on the joint positions that represent a randomly chosen end-effector position.

As it will be shown in the coming sections, the chosen trajectory (point A to point B) does not present any singularities during the motion obtained when using the state of the art controller and the designed torque one. However, it is still possible that some of the results may be biased by this specific trajectory used. Originally, different random desired end-effector positions were considered, however, due to the computational time required for each simulation of the Active Inference scheme only the following desired end-effector position has been used for the comparison:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -0.4689 \\ -0.2024 \\ 0.7339 \end{bmatrix}.$$

Apart from that, the simulations are going to be compared according to the amount of iterations needed to settle within a 5% of the desired position (settling time), the overshoot present in their responses and the time required for the response to rise from 10% to 90% of the target position (rise time). Furthermore, the time required to run each simulation has also been considered for the comparison of the controllers.

6-1 Zero-gravity simulations

Once the generative process and generative model have been defined for the Active Inference scheme, the simulations of the three proposed controllers, in order to compare them, has to be carried out. The first comparison is made for a situation with zero-gravity conditions. The three controllers are compared using the same sample time $dt = 0.01$ and for a number of samples/iterations $n = 5000$ which corresponds to 50 seconds.

6-1-1 Benchmark simulation

The state of the art controller described in section 5-1 and that corresponds to the control topology shown in figure 5-14 is tested in this section. The only variable of the controller corresponding to the gain k_p is kept as defined in [16]. Furthermore, as this simulation corresponds to the non-gravity ones, the inverse dynamics model is configured with gravity equal to zero.

One of the advantages that this controller presents is the possibility to define not only the initial and end position but the trajectory between these points thanks to the use of the inverse dynamics model. Therefore, the other parameters that need to be set in this controller correspond to the desired trajectory. In this case, the trajectory is only defined in two points, the initial position and the desired one. This is done to bring this simulation closer to the other controllers where only these two positions are defined. Each point of the trajectory also requires the time in which these positions should be achieved. For the simulations, the time where the target position should be achieved has been chosen considering the dynamics of the system and to obtain a smooth response with a low settling time and overshoot. This has been done through an iterative process observing the response of the system when modifying the target position timing. Finally, it has been seen that with 2 seconds as the time between the initial and final position, a response with low overshoot and with a reasonable settling time is obtained.

The response obtained using the benchmark controller is shown in figure 6-1. From this simulation it can be seen that the response obtained with this controller is smooth, this is because of the generated trajectory using the inverse model. In this case, the time needed for this controller to settle within the boundaries of the 5% margin is close to 4 seconds. The transient state of the response does not present overshoot in the y and z components while, in the x component it is 5.16%. Moreover, the rise time of the 3 components is: $[1.55; 1.93; 2.02]$ seconds . Furthermore, the steady state error after 50 seconds is: $1\text{e}{-4} \times \begin{bmatrix} 0.3817 \\ -0.2851 \\ 0.4462 \end{bmatrix}$.

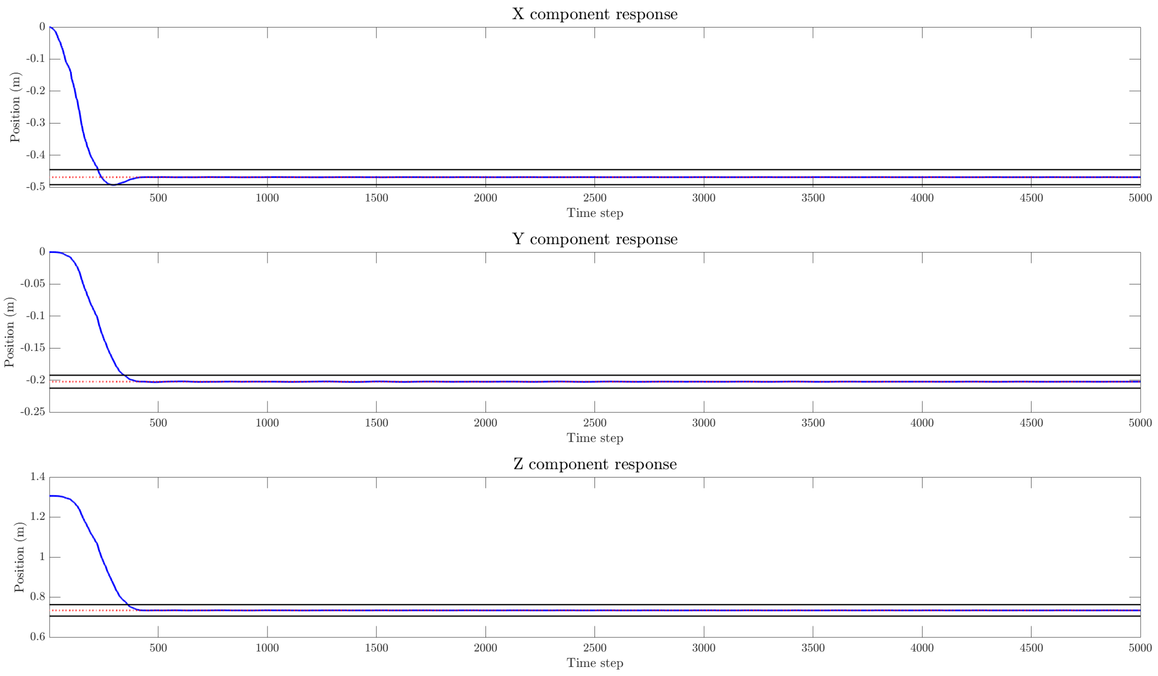


Figure 6-1: Benchmark controller response of the x,y,z components in Cartesian coordinates considering zero gravity; response: blue solid, target position: dotted red, 5% margin settling: black solid

6-1-2 Torque controller simulation

In this section, the torque controller designed in section 5-2 and with the control topology shown in figure 5-15 is tested. The input of this controller consists on the desired end position defined in section 6-0-3 and the only variable that needs to be defined is the controller gain K. Since this simulation corresponds to the zero-gravity section, the gravity compensation torque block is not needed as the resultant torque would be zero.

Regarding the controller gain K, this is chosen to be the same as the one mentioned in section 5-4-3 and obtained in appendix B-2. As the value determined in the mentioned section is the controller gain that will be used to test the Active Inference scheme and it is intended to compare the performance of these controllers, the controller gain K is set to:

$$\begin{bmatrix} 40 & 0 & 0 \\ 0 & 40 & 0 \\ 0 & 0 & 40 \end{bmatrix}.$$

The response obtained using the designed torque controller is shown in figure 6-2. From this simulation it can be seen that the time needed for this controller to settle within the boundaries of the 5% margin is around 5.5 seconds. The transient state of the response shows oscillations in the 3 components with remarkable overshoots (specially in the y component); the overshoot for each component is: $[31.29\%; 71.34\%; 68.71\%]$; furthermore, the rise time for this response is: $[0.24; 0.23; 0.18]$ seconds. Nonetheless, after these oscillations, the steady state error after 50 seconds is: $1\mathrm{e}{-7} \times \begin{bmatrix} 0.1211 \\ -0.6205 \\ 0.0516 \end{bmatrix}$.

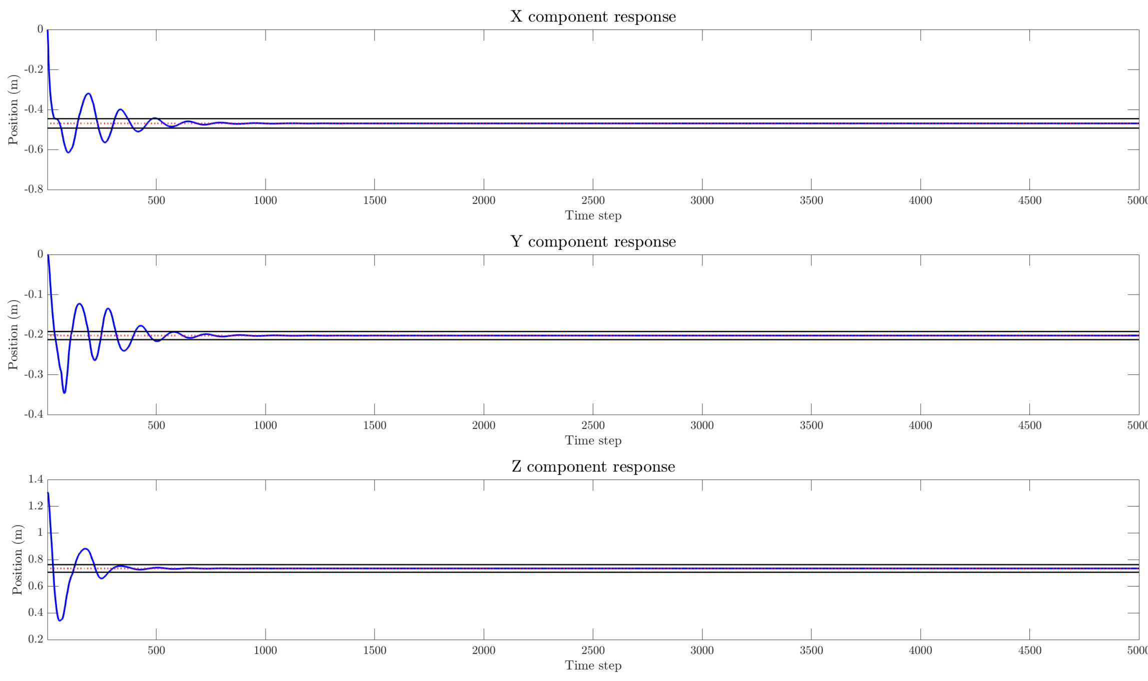


Figure 6-2: Designed torque controller response of the x,y,z components in Cartesian coordinates considering zero gravity; response: blue solid, target position: dotted red, 5% margin settling: black solid

Tuning the controller gain to a lower value, the system has a smoother response while still being fast at settling within the boundaries, however since the same controller wants to be compared the responses are compared using the same gain K.

6-1-3 Active Inference scheme simulation

The Active Inference scheme in closed loop defined in section 5-3 and that corresponds to the topology shown in figure 5-16 is tested in this section. In this case, since the Active Inference scheme is executed offline and the commands are therefore used as a feed-forward gain, two different responses are available. One corresponds to the Active Inference scheme loop and is shown in figure 6-3, while the other one corresponds to the real simulation response (figure 6-4).

In the first response (figure 6-3), one can see a smooth response with some oscillations at the transient state and some remarkable overshoots. However, it can be seen that the time needed for this controller to settle within the boundaries of the 5% margin is around 8 seconds and with a steady state error after 50 seconds of: $1e{-7} \times \begin{bmatrix} -0.5294 \\ -0.2273 \\ 0.8136 \end{bmatrix}$.

Furthermore, the overshoot for this response in each component is: $[64.47\%; 74.31\%; 22.92\%]$; on the other hand, the rise time is: $[0.73; 1.06; 1.14]$ seconds.

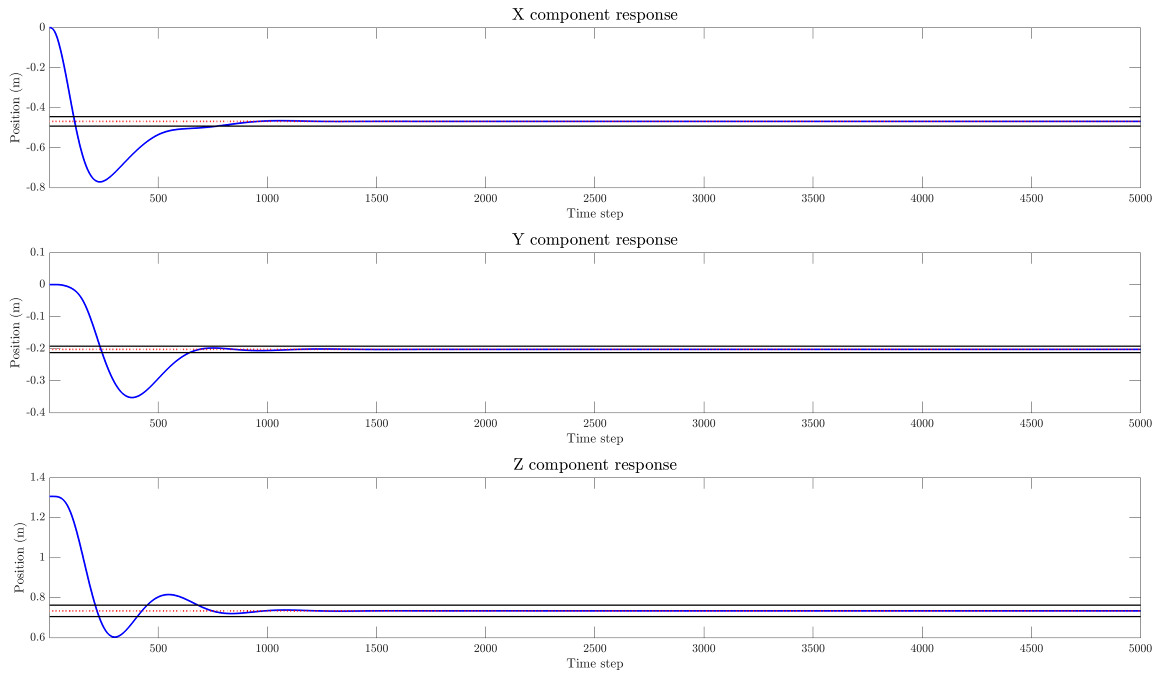


Figure 6-3: Active Inference scheme controller expected response of the x,y,z components in Cartesian coordinates considering zero gravity; response: blue solid, target position: dotted red, 5% margin settling: black solid

On the other hand, when bringing these commands to the real simulation and closing the loop (figure 6-4), the response is less smooth. In this case, there are big oscillations at the transient state, which is far from the desired behaviour and the one obtained in the Active Inference loop. At the beginning of the simulation, the z component (height) has big oscillations that go from the highest point possible for the robot manipulator to the ground, this happens because the initial state is highly unstable when the gravity is considered and the system needs some iterations to recover from it. However, the system is able to stabilize and the the time needed for this controller to settle within the boundaries of the 5% margin is around 11 seconds and with a steady state error after 50 seconds of: $1e{-}6 \times \begin{bmatrix} -0.3240 \\ 0.1228 \\ 0.0014 \end{bmatrix}$.

In this case, the overshoots present are large: $[257.50\%; 549.95\%; 128.11\%]$; while the rise time is influenced by the initial instability: $[0.26; 0.36; 0.29]$ seconds.

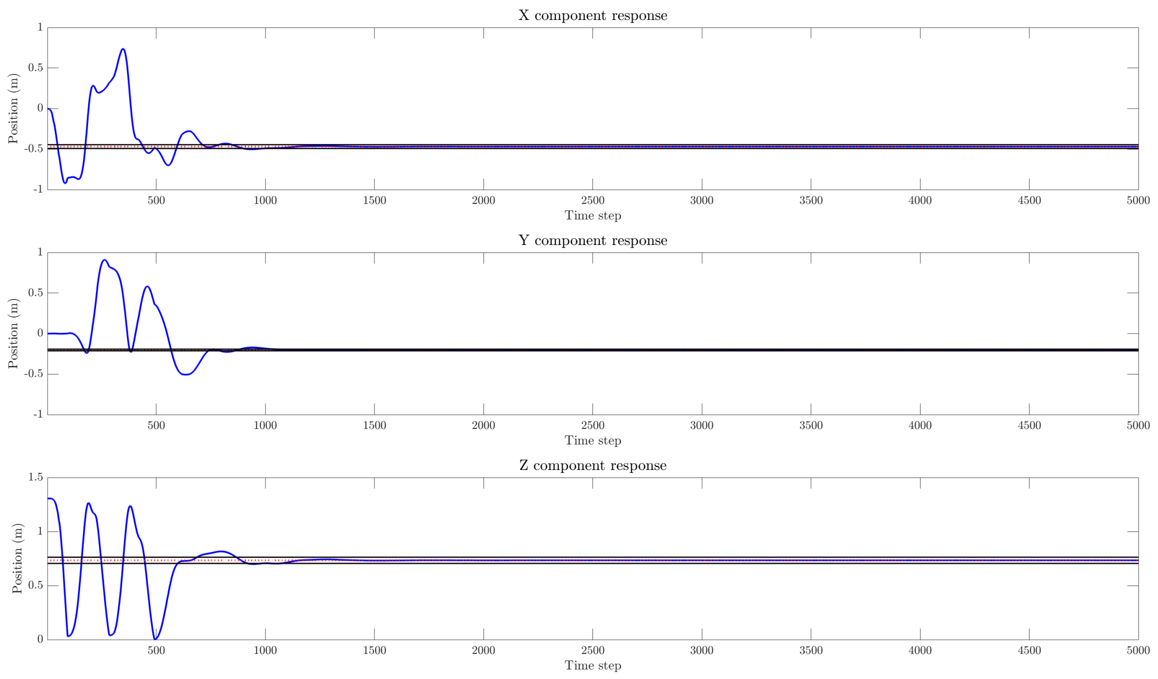


Figure 6-4: Active Inference scheme controller response of the x,y,z components in Cartesian coordinates considering zero gravity; response: blue solid, target position: dotted red, 5% margin settling: black solid

6-1-4 Comparison

Once the three response of the controllers have been shown, the comparison in terms of performance can be discussed. Table 6-2 shows the settling time needed for each of the controllers to be within a 5% margin, the response overshoot and rise time for the (x, y, z) components, as well as the computational time.

Controller	**Settling time** (s)	**Overshoot** (%)	**Rise time** (s)	**Computational time**
State of the art	4	$\begin{bmatrix}5.16\\0\\0\end{bmatrix}$	$\begin{bmatrix}1.55\\1.93\\2.02\end{bmatrix}$	Low
Designed torque controller	5.5	$\begin{bmatrix}31.29\\71.34\\68.71\end{bmatrix}$	$\begin{bmatrix}0.24\\0.23\\0.18\end{bmatrix}$	Real time
Active Inference scheme control	11	$\begin{bmatrix}257.50\\549.95\\128.11\end{bmatrix}$	$\begin{bmatrix}0.26\\0.36\\0.29\end{bmatrix}$	High

Table 6-2: Controllers performance comparison with zero gravity

Regarding the settling time, it can be seen that the state of the art obtains the best performance; however, the designed torque controller is able to settle with a difference of around 1.5 seconds. Considering that the latter does not require an inverse model and the control loop is executed in real time, the designed controller performance is comparable to the one from the state of the art controller.

On the other hand, the Active Inference scheme controller requires 11 seconds to settle. The difference between the behaviour of the designed controller and the Active Inference scheme, where the designed controller is used as a prior, is due to the dynamics used for the control. In the first case, the true dynamics of the simulation are used as this simulation is run online, while for the Active Inference scheme, the dynamics used in the model are different from the true ones (equations of motion by Pio-Lopez [1]).

In terms of the overshoot present in the three controllers, the state of the art controller offers the best performance as its overshoot is almost inexistent except for the x component. This is possible due to the definition of the trajectory in which the time between the initial and final point plays a crucial role. On the other hand, the designed controller presents higher overshoots in the 3 components due to the initial instability of the system caused by the upright initial position. Finally, even thought the expected Active Inference response (6-3) presented overshoots similar to the ones from the designed controller, the closed loop simulation shows huge overshoots for the 3 components. This is, again, due to the initial instability of the system that forces the robot manipulator to oscillate.

Regarding the rise time, both, the designed controller and the Active Inference scheme present fast rise time; however this is also related to the overshoot in these responses. On the other hand, the state of the art controller has a slower rise time ($1.5 - 2$ seconds).

In terms of the computational time, the designed torque controller does not require any previous computation as the control is done in real time, while the state of the art controller requires of the previous computation of all the control signals (torques) need to perform the reaching task. This pre-computation of the torques is executed in around 20 seconds for all the time instances (5000). On the other hand, as mentioned before, the current Active Inference implementation takes about 1 second per iteration; therefore, for a 5000 iterations simulation, the computational time required is about 3500-4500 seconds.

6-2 Earth's gravity simulations

Once the performance of the three controllers has been compared in a zero gravity environment, their performances need to be tested in a environment considering Earth's gravity. This simulations are required to obtain a better understanding of the performances of the controllers in a real world environment where the gravity is present.

This has not been the initial point of the simulations, since the way the system dynamics are defined in the generative model for the Active Inference scheme does not consider gravity. However, in this section the suitability of the gravity compensation block added in the designed torque controller and the Active Inference scheme control topologies is verified.

As mentioned in section 4-3, the simulation environment Gazebo allows different settings of the gravity and, therefore for this simulations, it is set to: $[0; 0; -9.8]$.

6-2-1 Benchmark simulation

This simulation is defined in the same way as the zero-gravity one. The only difference remains in the gravity values used for the inverse dynamics model. When importing the model from the *robotics* toolbox in MATLAB that is used for the inverse dynamics computation, the gravity vector that applies to this model is modified to consider gravity.

The response obtained using the benchmark controller with gravity is shown in figure 6-5. From this simulation it can be seen that the response obtained with this controller is smooth, this is because of the generated trajectory using the inverse model. In this case, the time needed for this controller to settle within the boundaries of the 5% margin is around 4 seconds. The transient state of the response does not present overshoot in the y and z components while, in the x component it is 4.73%. Moreover, the rise time of the 3 components is: $[1.63; 2.15; 2.31]$ seconds. Furthermore, the steady state error after 50 seconds is: $\begin{bmatrix} 0.0011 \\ -0.0006 \\ 0.0014 \end{bmatrix}$.

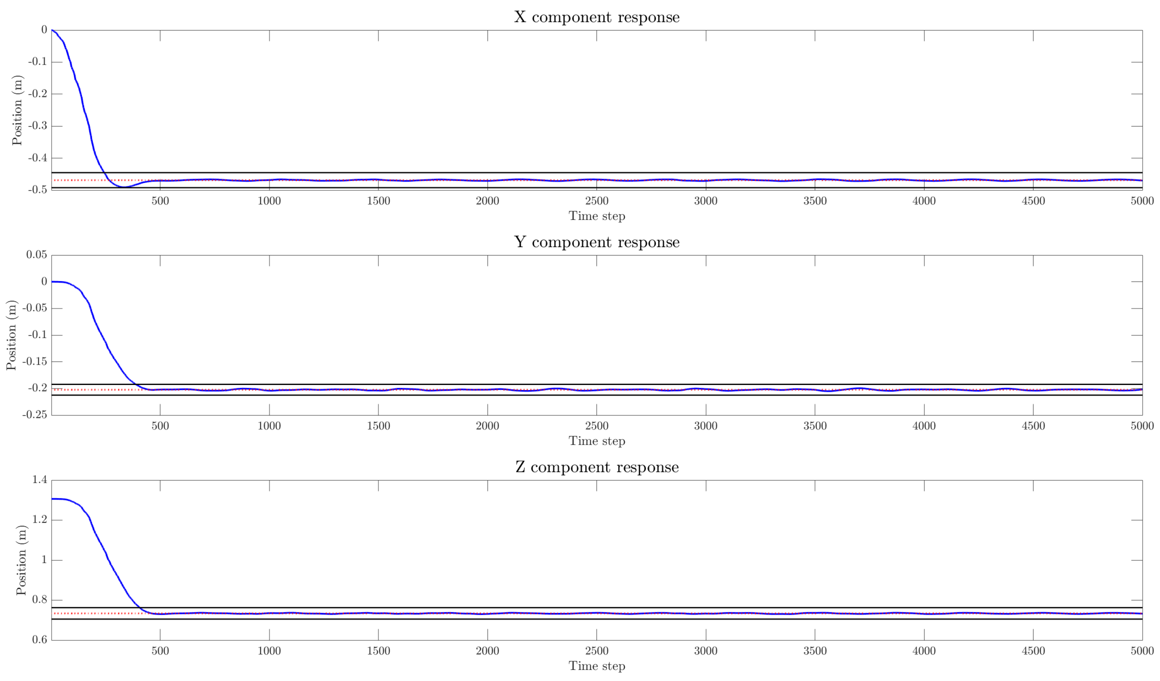


Figure 6-5: Benchmark controller response of the x,y,z components in Cartesian coordinates considering Earth's gravity; response: blue solid, target position: dotted red, 5% margin settling: black solid

6-2-2 Torque controller simulation

Like the benchmark controller, this simulation is defined in the same way as the zero-gravity one. The only difference remains in the gravity compensation torque block which is considered in this simulation. This is done through the adjustment of the gravity vector of the model imported in MATLAB which is used in the gravity compensation block.

The response obtained using the designed torque controller with gravity is shown in figure 6-6. From this simulation it can be seen that the time needed for this controller to settle within the boundaries of the 5% margin is around 6 seconds. The transient state of the response shows oscillations in the 3 components with important overshoots of: $[40.07\%; 73.91\%; 73.50\%]$; furthermore, the rise time of each of the components is: $[0.18; 0.20; 0.14]$ seconds. Nonetheless, after these oscillations, the steady state error after 50 seconds is: $1e{-}5 \times \begin{bmatrix} 0.2757 \\ 0.1151 \\ 0.0091 \end{bmatrix}$.

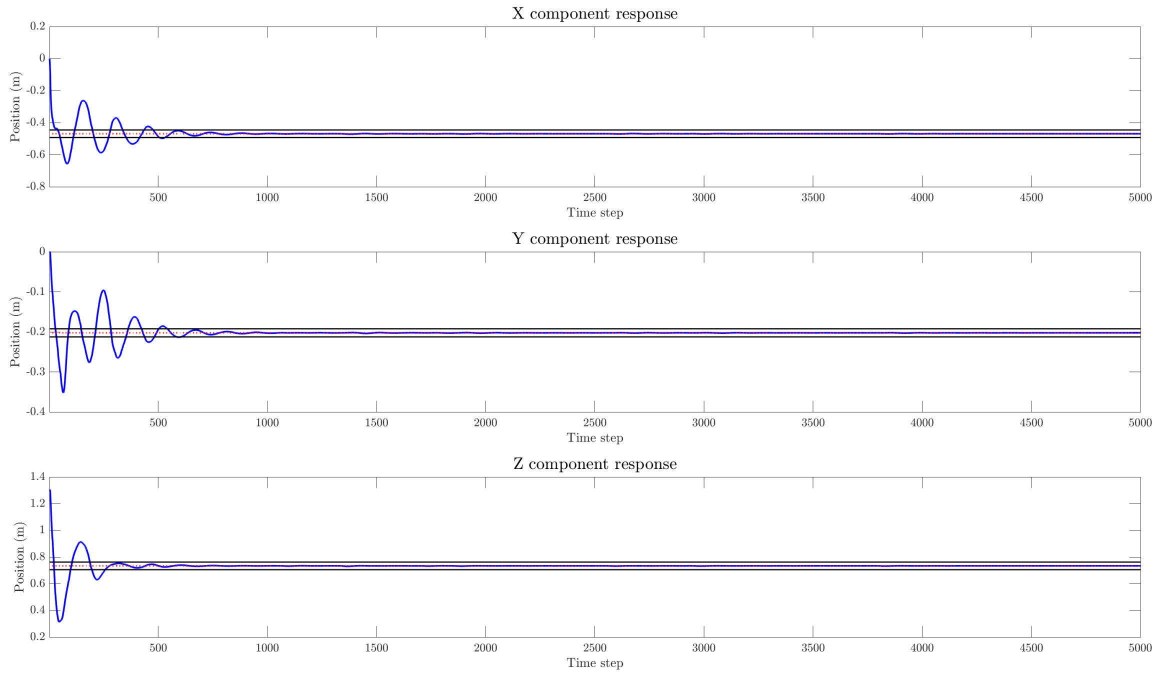


Figure 6-6: Designed torque controller response of the x,y,z components in Cartesian coordinates considering Earth's gravity; response: blue solid, target position: dotted red, 5% margin settling: black solid

6-2-3 Active Inference scheme simulation

Like the two other controllers, this simulation is defined in the same way as the zero-gravity one. In this case, as done in the designed torque controller, the gravity compensation torque block is considered in this simulation. This is also done through the adjustment of the gravity vector of the model imported in MATLAB which is used in the gravity compensation block.

The response obtained using the Active Inference scheme with gravity is shown in figure 6-7. The response obtained in this simulation corresponds to the robot manipulator falling to the ground. This can be seen as the z component drops to almost 0 within the first iterations and stabilises in that point, while the other two components are barely changing. As predicted, the simulation does not converge to the desired position even if the gravity compensation block is added. The fact that the generative model does not represent the true dynamics influences the response obtained and shows that just adding the gravity compensation is not sufficient to generate the desired motion.

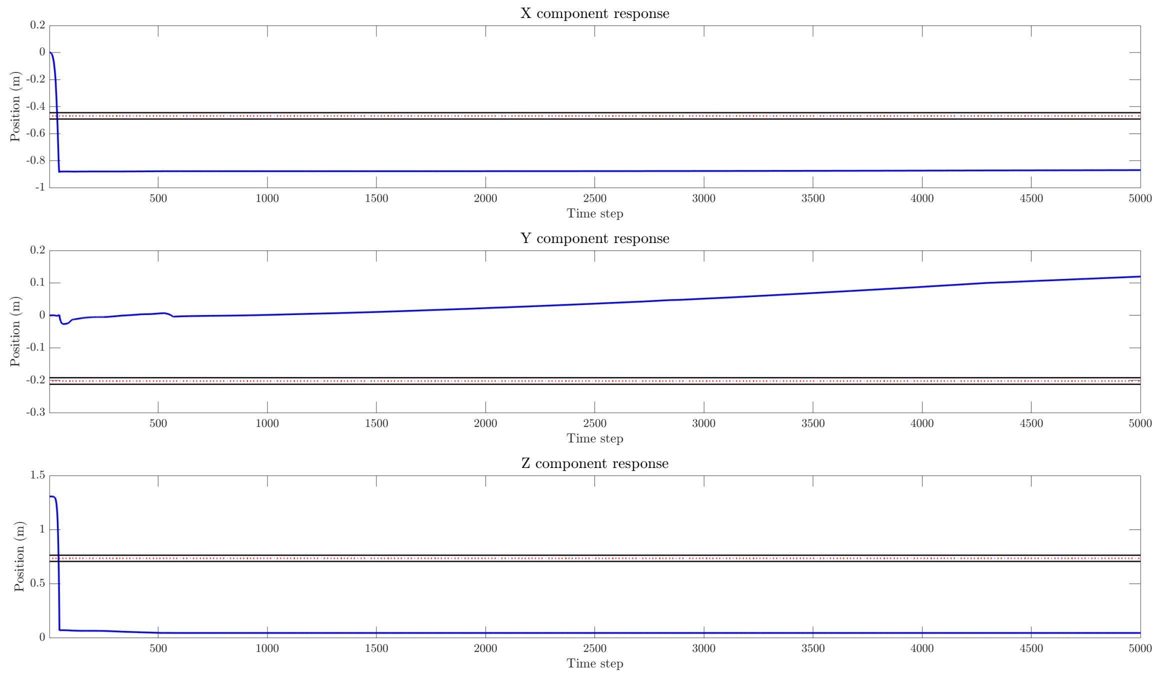


Figure 6-7: Active Inference scheme response of the x,y,z components in Cartesian coordinates considering Earth's gravity; response: blue solid, target position: dotted red, 5% margin settling: black solid

6-2-4 Comparison

Once the three response of the controllers have been shown, the comparison in terms of performance can be discussed. Table 6-3 shows the settling time needed for each of the controllers to be within a 5% margin, the response overshoot and rise time for the (x, y, z) components, as well as the computational time.

As it can be seen in the table 6-3, the Active Inference scheme row is empty because as shown in the previous section, this scheme is not able to obtain a stable response when the gravity is present.

In this section, only a comparison between the state of the art controller and the designed one is possible. In terms of settling time, the results obtained are similar to the zero-gravity simulations, where the benchmark controller was slightly faster; in this case, the difference between these controllers is of 2 seconds.

Regarding the overshoot, like the zero-gravity simulations, the state of the art presents an almost inexistent overshoot, while the designed controller reaches values of approximately 73%. This can be explained, again, because of the initial instability of the system. In the case of the rise time, similar results are obtained with respect to the zero-gravity simulations. Here, the rise time of the designed controller is also lower (approximately 0.20 seconds) but increases the oscillations of the transient state.

Controller	**Settling time** (s)	**Overshoot** (%)	**Rise time** (s)	**Computational time**
State of the art	4	$\begin{bmatrix}4.73\\0\\0\end{bmatrix}$	$\begin{bmatrix}1.63\\2.15\\2.31\end{bmatrix}$	Low
Designed torque controller	6	$\begin{bmatrix}40.07\\73.91\\73.50\end{bmatrix}$	$\begin{bmatrix}0.18\\0.20\\0.14\end{bmatrix}$	Real time
Active Inference scheme control				

Table 6-3: Controllers performance comparison with Earth's gravity

Finally, in terms of computational time, the same as the zero-gravity simulations can be applied. The designed torque control is executed in real time, without any pre-computation needed, while the benchmark controller requires the computation of the torques needed which takes about 20 seconds.

6-3 Conclusions

After having showed the performances of the three controllers in two different scenarios (zero-gravity and Earth's gravity), the conclusions about these performances can be extracted.

From the zero-gravity simulations, one can observe that the three controllers are properly designed as they can stabilise the system within the required time span. However, the optimal performance corresponds to the state of the art controller as its response does not present important overshoots and the settling and rise time are satisfactory.

Furthermore, the simulation has shown that using the offline Active Inference scheme as a feed-forward signal does not result in an optimal behaviour, whereas looking only at the Active Inference scheme loop simulation response (figure 6-3), it presents a better performance. The different behaviour between these may be caused by the difference of the internal model used with respect to the real world process. In the former, consisting on the equations of motion in Pio-Lopez paper [1], the inertial forces are not considered and, therefore, the first iterations present an oscillatory behaviour until these are counteracted with the feed-back loop.

From the simulations that consider gravity, similar conclusions can be extracted regarding the state of the art and designed controllers. The optimal performance corresponds to the benchmark one, but the designed controller is also able to stabilise the system but with higher overshoots. On the other hand, the Active Inference scheme controller is not able to stabilise the system when closing the loop, this is due to the torques obtained in the offline computation that do not include any gravitational force. However, from the offline simulation results, one could say that an online implementation of this scheme should obtain a similar response to the offline one.

In conclusion, the state of the art controller has shown the best performance among the compared controllers in both scenarios. However, the designed torque controller tested is also able to stabilise the system (in both scenarios) in a similar amount of time but including higher overshoots than the benchmark controller. Finally, the Active Inference scheme has shown that it is able to stabilise the system, only for the zero-gravity case, in the offline loop and the simulation response. However, the impossibility of using a dynamic model that considers gravity in the generative model has become crucial in the non-converging behaviour of the response in the simulations that consider gravity.

Chapter 7

Conclusions and Future work

In this chapter, the final conclusions obtained from the work done throughout the thesis and some recommendations on future work are given.

7-1 Conclusions

As mentioned in the Introduction chapter, the purpose of this thesis consists on discerning whether the current Active Inference implementation can be used for robot manipulator control in a reaching task. To do so, the current state of the art has been studied, a low-level controller has been designed to use in the Active Inference scheme and a comparison of this scheme with a benchmark controller is performed.

To begin with, in Chapter 3, a thorough analysis on the related work on robot control using Active Inference was carried out. More specifically, the analysis of the paper by Pio-Lopez [1] brought many outcomes about the interesting work (embedding a controller as a prior, offline implementation) done in the paper and how these could help in the implementation proposed here. But, this analysis also showed some limitations that the implementation proposed in the paper had (internal position control required, dynamic model different from true process). This served as an starting point to understand what had to be done in order to improve the current state of the art.

From the analysis mentioned before, the need of a controller arose. Therefore, a torque controller that does not require an inverse model to control the robot manipulator has been designed inspired in the Passive Motion Paradigm (PMP), which was researched during the literature review prior to this thesis. This low-level controller allows to control the robot manipulator without the need of any internal position controller as was done in the previous implementation [1]. However, in order to give a better understanding of the benefits that the Active Inference scheme can contribute with, a benchmark controller is required.

Once the controllers were designed, the Active Inference scheme has been defined. From the aforementioned analysis, an offline implementation of the Active Inference scheme has been studied and redefined giving a clear picture of the control loop and the evolution of the signals within the scheme. Furthermore, an online implementation in which the process is not part of the algorithm of the Active Inference scheme has also been defined in order to bring this scheme to real world simulations.

To continue with, the robot manipulator that has been used for the simulations has been studied in order to obtain the necessary models to implement the Active Inference scheme described. The forward kinematics, Jacobian, forward dynamics and a gravity compensation torque have been defined either by obtaining them analytically or using the available methods in the MATLAB *robotics* toolbox.

Having defined all the required aspects for the implementation of the Active Inference scheme, the three controllers can be tested and the comparison done. Regarding the simulations, the online implementation of the Active Inference scheme has been discarded, however, a definition of it and recommendations on future work based on the current implementation method have been proposed.

Besides this, the comparison between the controllers has shown that the offline Active Inference scheme is able to control a robot manipulator for a reaching task with a similar behaviour than the designed torque controller. In contrast to what has been done in Pio-Lopez paper, this thesis has designed an implementation in which the control loop is closed and the Active Inference scheme stabilises the system using the pre-computed actions and a feed-back component. However, using the closed loop implementation, the performance is still far from an optimal behaviour compared to the other controllers tested. While the benchmark controller is able to obtain a stable response without overshoot, the designed controller can also obtain a stable response with a satisfactory settling time but with higher overshoots. On the other hand, as mentioned before, the close loop Active Inference scheme response requires twice the time to settle within the defined margins and the overshoots are also higher. The aforementioned simulations correspond to the zero-gravity scenario comparison.

Furthermore, a comparison of the controllers when gravity is considered has also been carried out. In this case, for the benchmark controller, the gravity considered in the model used for the pre-computation of the feed-forward torque is modified. For the two other controllers (designed torque controller and Active Inference scheme), the gravity compensation blocks that compute an extra torque that is added to the one obtained from the controllers is considered. From these simulations, it has been shown that the responses obtained with the benchmark controller and the designed one, do not differ much from the simulations where gravity was not considered in terms of settling time, overshoot and rise time. On the other hand, the Active Inference scheme is not able to stabilise the system when the gravity torque compensation is included. This may be due to the fact that the offline computation of the Active Inference loop used to generate the actions required (torque) does not include the gravity in its generative model.

However, as shown in the designed controller response, an online implementation of the Active Inference scheme should be able to stabilise the system by adding the gravity compensation torque to the action obtained from the Active Inference loop. Further research on this matter is needed when the Active Inference scheme can be implemented online and real time.

Finally, an unexpected outcome obtained during this thesis is related to the dynamic model used in the generative process and generative model. While the initial hypothesis, based on the Active Inference theory, was that the performance would be optimal when the generative process and model had the same equations, the simulations have shown that, in this case, the simulation does not converge to the desired position. During the execution of this thesis, no explanation could be found to describe this behaviour and further research is needed to verify whether this is a problem of the Active Inference implementation proposed.

As a final conclusion, this thesis serves as an initial point for further research on Active Inference for online robot control, while defining the limitations of current implementations and providing the requirements needed for an online implementation. The implementation proposed here generalises the Active Inference scheme loop as it can be easily adapted to other manipulators, in contrast to the available implementations. This adaptability is due to the controller created, in which only the forward kinematics and the Jacobian matrix are needed. Regarding the applicability of Active Inference for robot control, the simulations have shown that the current implementation does not improve the state of the art methods; however, it also demonstrates that the Active Inference framework can be used for robot control and with further research it may be an alternative to the current methods.

7-2 Recommendations and future work

The work done throughout the thesis has brought many interesting insights that need to be researched further.

The online implementation of the Active Inference scheme, which is needed to verify the true advantages of the framework, requires an implementation different from the one that has been used here. The Statistical Parametric Mapping (SPM) suite by Karl Friston of MATLAB is developed for offline loops, where the generative process is not a real world process but a simulation of it. This becomes a problem when trying to use this implementation for an online loop, as the hidden states of the process, which should be unknown from the perspective of the Active Inference scheme, are part of the whole process.

Another problem encountered when defining the online implementation is also related to the other uses of the SPM suite. In this case, the software includes different calculations that are not required for the purpose of implementing Active Inference for robot control. However, the software is programmed in such a way, that deleting these calculations cannot be done and therefore a new implementation of the algorithm is needed. The computational time is one of the limitations of the current implementation as each of the iterations takes about one second, to solve this issue, apart from removing unnecessary calculations, the new implementation should be done in a faster programming language, such as C++.

Furthermore, another issue related to the online implementation of the Active Inference scheme arises when observing at the sensations from the real world process. These sensations, that should be the only input to the Active Inference scheme from the real world process, need to be in generalised coordinates. However, from a real system only the first component of the generalised coordinates of the sensations is usually available. To complete these coordinates, the derivatives of the sensations are needed. Thus, further research on how this issue can be addressed is needed.

Besides, it has been seen that the offline implementation of Active Inference has been accomplished using equations of motion for the generative model that do not correspond to the real dynamic model of the system. Even though, this has given satisfactory results when tested without gravity, the model used does not consider the effect of the gravity in the dynamics of the system. Therefore, further research is needed in this direction, either on obtaining a forward dynamics model that considers gravity or on finding the reason why the Active Inference loop presented does not converge when the generative model and the generative process are the same.

Finally, with a functioning online implementation of Active Inference as described in this thesis, the promising properties of this framework in terms of model deviations and noise rejection [1] could be verified.

Appendix A

Matlab code

A-1 KUKA kinematics methods

A-1-1 Forward kinematics method

```
function [Pos,Rot,Tr] = IIWA_FW_kinematics(q,k)
% @author: Arnau Cibiach
%
% returns the i-joint position and rotation matrix of the iiwa robot
    given
% the joint angles q in [rad]
%
% FORMAT [Pos,Rot,Tr] = IIWA_FW_kinematics(q,k)
%
% x     - hidden states
%   x(1)  - joint angle 'mw_iiwa_link_1_roll'
%   x(2)  - joint angle 'mw_iiwa_link_2_flex'
%   x(3)  - joint angle 'mw_iiwa_link_3_roll'
%   x(4)  - joint angle 'mw_iiwa_link_4_flex'
%   x(5)  - joint angle 'mw_iiwa_link_5_roll'
%   x(6)  - joint angle 'mw_iiwa_link_6_flex'
%   x(7)  - joint angle 'mw_iiwa_link_7_roll'
%
% k   - number of joint
%
% This function solves the problem of the forward kinematics for the
% "KUKA LBR iiwa 14 R820" robot manipulator.
%_______________________________________________________________________

if ~exist('k','var')
    k = 8;
end

```

```
d1 = 0.1575;
d2 = 0.2025;
d3 = 0.2045;
d4 = 0.2155;
d5 = 0.1845;
d6 = 0.2155;
d7 = 0.0810;
de = 0.0450;
DH=[d1        0    -pi/2        q(1);
     0      -d2     pi/2        q(2);
    d3        0     pi/2        q(3);
     0      -d4    -pi/2        q(4);
    d5        0    -pi/2        q(5);
     0      -d6     pi/2        q(6);
    d7        0        0        q(7);
    de        0        0           0];


for i=1:k
    d=DH(i,1);  % d
    r=DH(i,2);  % r
    al=DH(i,3); % alpha
    t=DH(i,4);  % theta

     T(:,:,i)=[cos(t)    -sin(t)*cos(al)    sin(t)*sin(al)    r*cos(al);
               sin(t)     cos(t)*cos(al)   -cos(t)*sin(al)    r*sin(al);
                   0              sin(al)           cos(al)            d;
                   0                    0                 0            1];

end

Tr=eye(4);
for i=1:k
    Tr=Tr*T(:,:,i);
end

Rot = Tr(1:3,1:3);
Pos = Tr(1:3,4);

end
```

A-1-2 Jacobian matrix method

```
function [Jacobian] = IIWA_Jacobian(k)
% @author: Arnau Cibiach
%
% returns the symbolical Jacobian matrix of the iiwa robot to reduce
% computational time
%
% FORMAT [Jac] = IIWA_Jacobian(k)
%
% k   - number of joint
%_______________________________________________________________________
```

```

syms q_1 q_2 q_3 q_4 q_5 q_6 q_7
persistent Jac

T = sym('T', [4 4 7]);
if isempty(Jac)

    % link lengths obtained from the specifications
    d1 = 0.1575;
    d2 = 0.2025;
    d3 = 0.2045;
    d4 = 0.2155;
    d5 = 0.1845;
    d6 = 0.2155;
    d7 = 0.0810;
    de = 0.0450;

    DH=[d1        0     -sym(pi)/2      0;  %q(1)
        0       -d2      sym(pi)/2      0;  %q(2)
        d3        0      sym(pi)/2      0;  %q(3)
        0       -d4    -sym(pi)/2       0;  %q(4)
        d5        0     -sym(pi)/2      0;  %q(5)
        0       -d6      sym(pi)/2      0;  %q(6)
        d7        0         0           0;  %q(7)
        de        0         0           0];

    T_=[q_1, q_2, q_3, q_4, q_5, q_6, q_7, 0];

    for i=1:k
        d=DH(i,1);
        a=DH(i,2);
        al=DH(i,3);
        t=T_(i);

        T(:,:,i)=[cos(t)  -sin(t)*cos(al)   sin(t)*sin(al)  a*cos(al);
                  sin(t)   cos(t)*cos(al)  -cos(t)*sin(al)  a*sin(al);
                     0          sin(al)          cos(al)          d;
                     0             0                0             1];
    end

    Tr=eye(4);
    for i=1:k
        Tr=Tr*T(:,:,i);
    end

    Pos = Tr(1:3,4);
    Jac = [     diff(Pos(1),q_1) diff(Pos(2),q_1) diff(Pos(3),q_1);
                diff(Pos(1),q_2) diff(Pos(2),q_2) diff(Pos(3),q_2);
                diff(Pos(1),q_3) diff(Pos(2),q_3) diff(Pos(3),q_3);
                diff(Pos(1),q_4) diff(Pos(2),q_4) diff(Pos(3),q_4);
                diff(Pos(1),q_5) diff(Pos(2),q_5) diff(Pos(3),q_5);
                diff(Pos(1),q_6) diff(Pos(2),q_6) diff(Pos(3),q_6);
                diff(Pos(1),q_7) diff(Pos(2),q_7) diff(Pos(3),q_7)]';
```

```
end

Jacob(q_1,q_2,q_3,q_4,q_5,q_6,q_7) = Jac;
Jacobian = matlabFunction(Jacob);

end
```

```
function [Jacobian] = IIWA_Jacobian_sub(Jac,q)
% @author: Arnau Cibiach
%
% returns the analytical Jacobian matrix of the iiwa robot given the
% symbolical Jacobian and the joint angles q in [rad]
%
% FORMAT [Jacobian] = IIWA_Jacobian(Jac,q,k)
%
% Jac   - symbolical Jacobian matrix
%
% q     - joint angles
%   q(1)  - joint angle 'mw_iiwa_link_1_roll'
%   q(2)  - joint angle 'mw_iiwa_link_2_flex'
%   q(3)  - joint angle 'mw_iiwa_link_3_roll'
%   q(4)  - joint angle 'mw_iiwa_link_4_flex'
%   q(5)  - joint angle 'mw_iiwa_link_5_roll'
%   q(6)  - joint angle 'mw_iiwa_link_6_flex'
%   q(7)  - joint angle 'mw_iiwa_link_7_roll'
%_____________________________________________________________________

Jacobian=double(Jac(q(1),q(2),q(3),q(4),q(5),q(6),q(7)));


end
```

A-2 Active Inference

A-2-1 Equations of motion

```
function [f]= IIWA_spm_fx_robot_dem_reach_offline(x,v,P)
% @author: Arnau Cibiach
%
% returns the flow for a seven-joint robot manipulator
%
% FORMAT [f]= IIWA_spm_fx_robot_dem_reach(x,v,P)
%
% x     - hidden states
%   x(1) to x(7) - joint angles
%   x(8) to x(14) - joint velocities
%
% v     - causal states
%   v(1) - target location (x)
%   v(2) - target location (y)
%   v(3) - target location (z)
%
% P     - model parameters (not used)
%
% This function corresponds to the equations of motion of the generative
% model
%_________________________________________________________________________

% Robot manipulator parameters (extracted from the specifications)
m1    = 4;
m2    = 4;
m3    = 3;
m4    = 2.7;
m5    = 1.7;
m6    = 1.8;
m7    = 0.3;
k1    = 4;
k2    = 4;
k3    = 3;
k4    = 2.7;
k5    = 1.7;
k6    = 1.8;
k7    = 0.3;
visco = 2;

global lbr_model

x = full(x);
joints = x(1:7);
velocities = x(8:14);

% Target position
Target=[v(1);v(2);v(3)];

% Controller action computed + torque to compensate gravity (if needed)
T = IIWA_Controller(joints,Target)+gravityTorque(lbr_model,joints);

f  = [ velocities(1);      % joint velocities
    velocities(2);
    velocities(3);
    velocities(4);
    velocities(5);
    velocities(6);
    velocities(7);
    (T(1)-visco*k1*x(8))/m1;      % velocities
    (T(2)-visco*k2*x(9))/m2;
    (T(3)-visco*k3*x(10))/m3;
    (T(4)-visco*k4*x(11))/m4;
    (T(5)-visco*k5*x(12))/m5;
    (T(6)-visco*k6*x(13))/m6;
    (T(7)-visco*k7*x(14))/m7;];
```

```matlab
function [f]= IIWA_spm_fx_robot_adem_reach_offline(x,v,a,P)
% @author: Arnau Cibiach
%
% returns the flow for a seven-joints robot manipulator (with action)
%
% FORMAT [f]= IIWA_spm_fx_robot_adem_reach(x,v,a,P)
%
% x    - hidden states
%   x(1) to x(7) - joint angles
%   x(8) to x(14) - joint velocities
%
% v    - causal states
%   v(1) - target location (x)
%   v(2) - target location (y)
%   v(3) - target location (z)
%
% a    - action (forces) (x,y)
%
% P    - model parameters (not used)
%
% This function corresponds to the equations of motion of the generative
% process
%_______________________________________________________________________

global lbr_process;

action = full(a);
x = full(x);

joints = x(1:7);
velocities =  x(8:14);

% Forward dynamics model to obtain the accelerations
accelerations = forwardDynamics(lbr_process, joints, velocities, action);

f  = [ velocities(1);      % joint velocities
    velocities(2);
    velocities(3);
    velocities(4);
    velocities(5);
    velocities(6);
    velocities(7);
    accelerations(1);      % joint accelerations
    accelerations(2);
    accelerations(3);
    accelerations(4);
    accelerations(5);
    accelerations(6);
    accelerations(7);];
```

A-2-2 Sensory mapping

```matlab
function [g]= IIWA_spm_gx_robot_dem_reach_offline(x,v,P)
% @author: Arnau Cibiach
%
% returns the prediction for a seven-joint arm
%
% FORMAT [g] = IIWA_spm_gx_robot_dem_reach(x,v,P)
%
% x     - hidden states
%   x(1) to x(7)  - joint angles
%   x(8) to x(14) - joint velocities
%
% v     - causal states
%   v(1) - target location (x)
%   v(2) - target location (y)
%   v(3) - target location (z)
%
% P     - model parameters (not used)
%
% This function corresponds to the sensory mapping of the generative
% model
%__________________________________________________________________

x = full(x);

% End-effector position computed given the joint positions
Pose = IIWA_FW_kinematics(x(1:7));

% Sensory mapping
g  = [x(1:7); v; Pose(1);Pose(2);Pose(3)];
```

```matlab
function [g] = IIWA_spm_gx_robot_adem_reach_offline(x,v,a,P)
% @author: Arnau Cibiach
%
% returns the prediction for a seven-joint arm (with action)
%
% FORMAT [g] = IIWA_spm_gx_robot_adem_reach(x,v,a,P)
%
% x     - hidden states
%   x(1) to x(7)  - joint angles
%   x(8) to x(14) - joint velocities
%
% v     - causal states
%   v(1) - target location (x)
%   v(2) - target location (y)
%   v(3) - target location (z)
%
% a     - action
%
% P     - model parameters (not used)
%
% This function corresponds to the sensory mapping of the generative
% proces
%_____________________________________________________________________

global lbr_process

joints = x(1:7);

% End-effector position computed given the joint positions
T = getTransform(lbr_process, joints,'iiwa_link_ee');
Pose = T(1:3,end);

% Sensory mapping
g  = [joints; v; Pose(1);Pose(2);Pose(3)];
```

Appendix B

Robot control

B-1 Forward kinematics

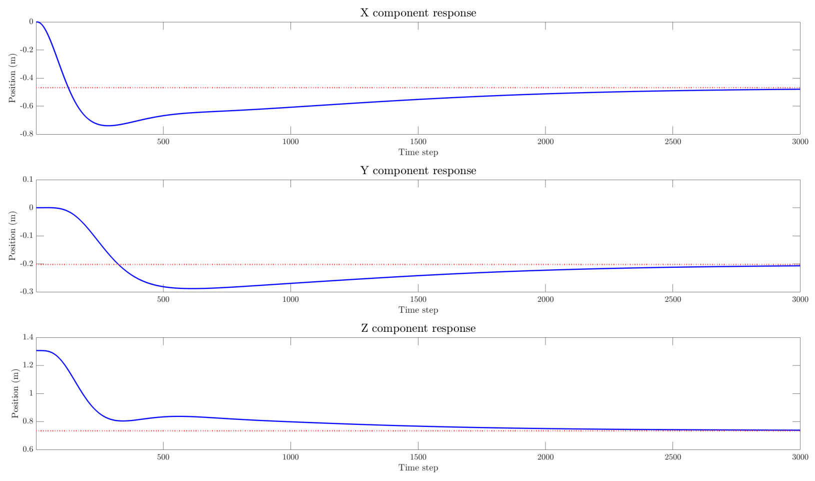
As described in section 4-2-1, two different implementations of the forward kinematics used for the sensory mapping are available. Both approaches are based on the transformation matrix and should provide the same results. To verify so, the following simulations are done:

#	**Generative model**	**Generative process**
1	Forward kinematics `getTransform`	Forward kinematics `getTransform`
2	Forward kinematics `getTransform`	Forward kinematics function
3	Forward kinematics function	Forward kinematics `getTransform`

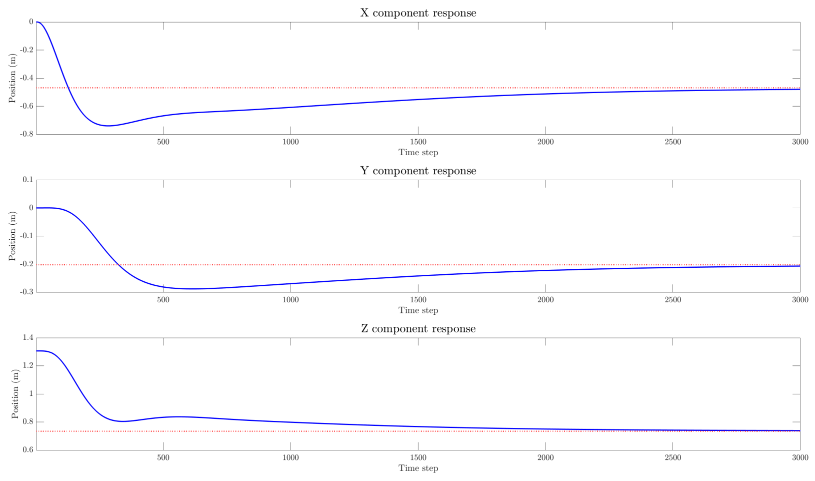
Table B-1: Active Inference scheme simulations for sensory mapping choice

The responses obtained with the simulations shown in table B-1 are shown in figure B-1. The three simulations share the same equations of motion and controller parameters, whereas the sensory mapping is the only aspect modified in each simulation according to the mentioned table.

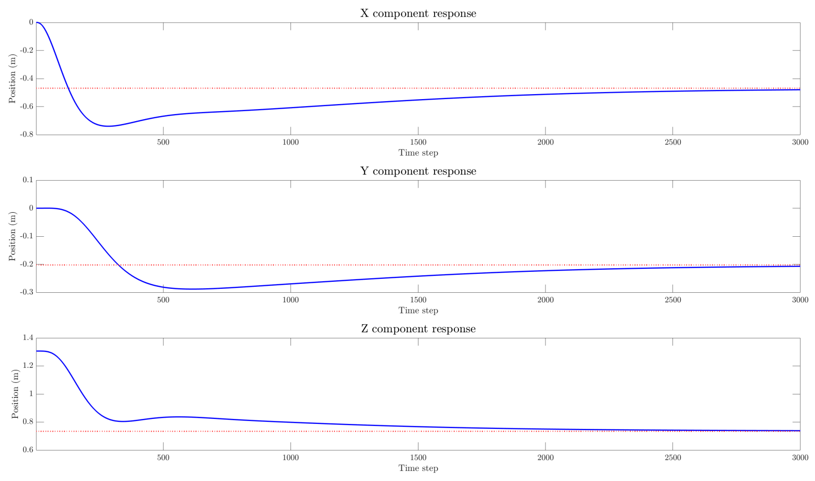
The simulations have shown that no difference can be appreciated between using the two options available for the sensory mapping of the generative process and the generative model. Therefore, the two methods are equivalent and can be used indistinctly for both aspects.



(a) Simulation 1



(b) Simulation 2



(c) Simulation 3

Figure B-1: End-effector position response in (x,y,z) coordinates of the sensory mapping simulations

B-2 Controller gain

Once the definitive Active Inference scheme has been chosen, the controller gains can be tuned in order to obtain a satisfactory behaviour in the Active Inference simulations. To do so, different gains have been tested to observe the response for each of them.

The 3 components of the gain, each corresponding to one of the x, y, z coordinates respectively, have the same weight in all the simulations done. This implies that the importance of each of the components is not higher than the rest. However, different weights in each of the entries of the K matrix could be acceptable as well. Further work could be done in this area, to study the influence of these gains in the final response of the system.

As mentioned, the controller gains consist on an identity matrix of size 3×3 , I_3, multiplied by an integer. In this case, the different controller gains used here are:

$$[3, 5, 10, 20, 30, 40, 50, 60, 75, 100, 400, 500, 1000] \times I_3$$

All simulations have been executed with the same configuration parameters and with a time step $dt = 0.01$ and a number of samples/iterations $n = 5000$. The responses obtained from these simulations are used to create the following figures, where the settling time and the overshoot for each component are depicted.

In the first figure B-2, the number of iterations for each of the simulation to obtain a stable signal within a 5% margin around the desired position is shown. The circles correspond to the measurements corresponding to the simulation, while the solid line represents a spline function that interpolates the points in between the measurements.

In the second figure B-3, the percentage of overshoot (the highest peak with respect to the reference) in each of the components (x, y, z) is shown. Like the previous figure, the circles correspond to the measurements and the solid line, the spline function.

As it can be seen in both images, only the gains until 500 are considered because with a controller gain $K = 1000 \times I_3$, the response becomes unstable and the simulations crashes.

Based on the settling time figure (B-2), it can be seen that there is a minimum on the number of iterations needed given a gain of approximately 40. It can also be seen, how gains lower than 5 are too slow to converge within the given 5000 iterations and on the other side, values higher than 400 present an oscillatory response that is not settled within the margins in less than 5000 iterations.

Regarding the overshoot, it can be seen in figure B-3 that the gains in which a lower settling time was obtained (around 40) present a considerable overshoot compared to values below 20. However, a trade-off between the settling time and the overshoot is present and considering the settling time and the percentage of overshoot obtained, the final controller gain is chosen to be set at:

$$40 \times I_3 = \begin{bmatrix} 40 & 0 & 0 \\ 0 & 40 & 0 \\ 0 & 0 & 40 \end{bmatrix}$$

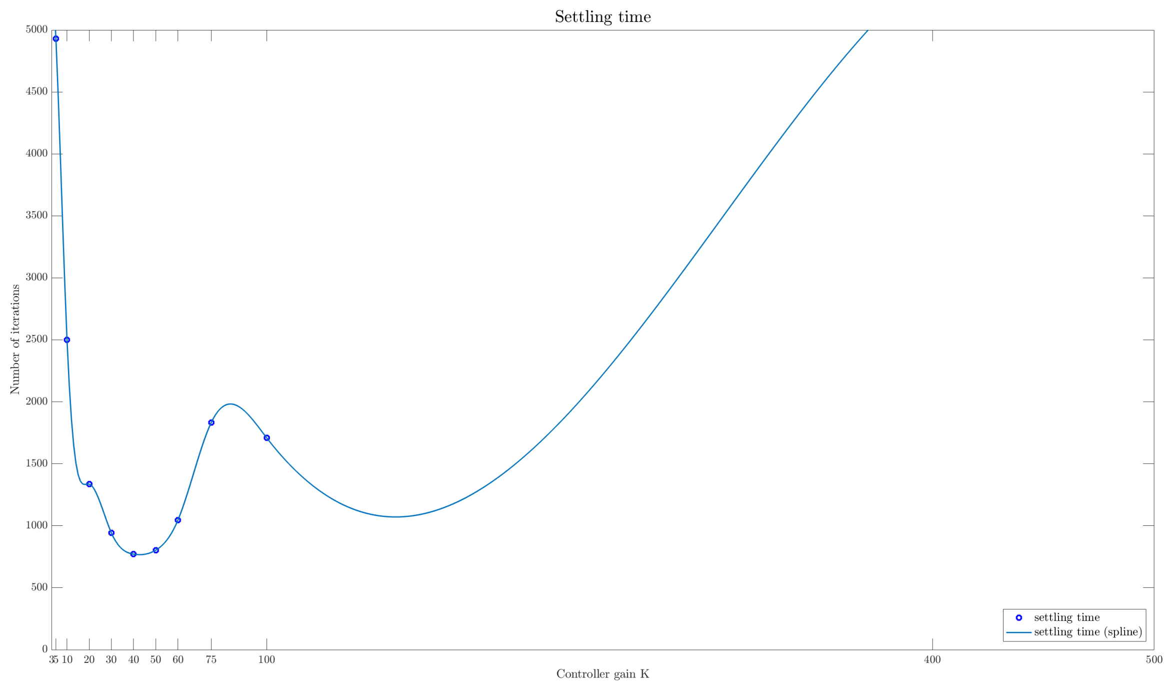


Figure B-2: Settling time (number of iterations) for different controller gains; circles: measurements, line: spline function

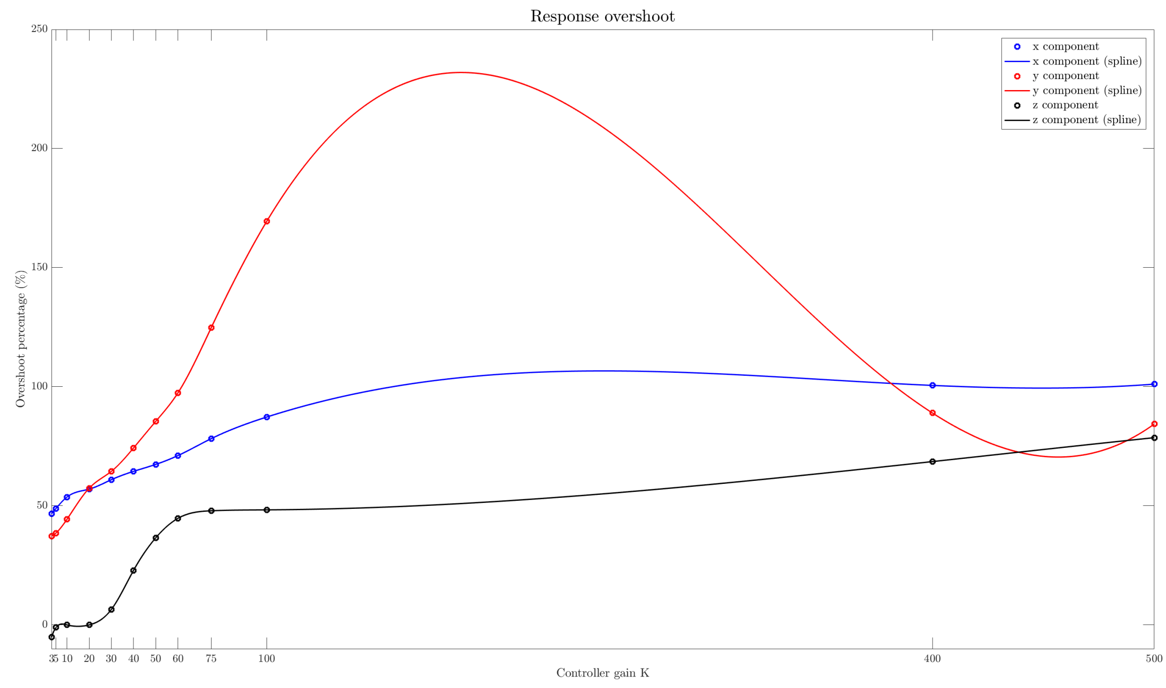


Figure B-3: Overshoot obtained with different controller gains in each component; circles: measurements, line: spline function

Bibliography

[1] L. Pio-Lopez, A. Nizard, K. Friston, and G. Pezzulo, “Active inference and robot control: a case study,” *Journal of The Royal Society Interface*, vol. 13, no. 122, p. 20160616, 2016.

[2] K. J. Friston, J. Daunizeau, and S. J. Kiebel, “Reinforcement learning or active inference?,” *PloS one*, vol. 4, no. 7, p. e6421, 2009.

[3] K. Friston, J. Kilner, and L. Harrison, “A free energy principle for the brain,” *Journal of Physiology-Paris*, vol. 100, no. 1-3, pp. 70–87, 2006.

[4] K. Friston, “The free-energy principle: a unified brain theory?,” *Nature Reviews Neuroscience*, vol. 11, no. 2, p. 127, 2010.

[5] K. J. Friston, J. Daunizeau, J. Kilner, and S. J. Kiebel, “Action and behavior: a free-energy formulation,” *Biological cybernetics*, vol. 102, no. 3, pp. 227–260, 2010.

[6] M. Botvinick and M. Toussaint, “Planning as inference,” *Trends in cognitive sciences*, vol. 16, no. 10, pp. 485–488, 2012.

[7] K. Friston, J. Mattout, and J. Kilner, “Action understanding and active inference,” *Biological cybernetics*, vol. 104, no. 1-2, pp. 137–160, 2011.

[8] C. L. Buckley, C. S. Kim, S. McGregor, and A. K. Seth, “The free energy principle for action and perception: A mathematical review,” *Journal of Mathematical Psychology*, 2017.

[9] K. Friston, S. Samothrakis, and R. Montague, “Active inference and agency: optimal control without cost functions,” *Biological cybernetics*, vol. 106, no. 8-9, pp. 523–541, 2012.

[10] K. Friston, K. Stephan, B. Li, and J. Daunizeau, “Generalised filtering,” *Mathematical Problems in Engineering*, vol. 2010, 2010.

[11] C. C. de Wit, B. Siciliano, and G. Bastin, *Theory of robot control.* Springer Science & Business Media, 2012.

[12] D. M. Dawson, C. T. Abdallah, and F. L. Lewis, *Robot manipulator control: theory and practice*. CRC Press, 2003.

[13] H. Asada, "Introduction to robotics - chapter 6 - statics." https://ocw.mit.edu/courses/mechanical-engineering/2-12-introduction-to-robotics-fall-2005/lecture-notes/chapter6.pdf, Fall 2005.

[14] R. Featherstone, *Forward Dynamics — The Composite-Rigid-Body Method*, pp. 79–88. Boston, MA: Springer US, 1987.

[15] V. Chawda and G. Niemeyer, "Toward torque control of a kuka lbr iiwa for physical human-robot interaction," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6387–6392, Sept 2017.

[16] "Control LBR Manipulator Motion Through Joint Torque Commands." https://nl.mathworks.com/help/robotics/examples/control-lbr-manipulator-motion-through-joint-torque.html. [Online; accessed 6-August-2018].

[17] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[18] F. N. Fritsch and R. E. Carlson, "Monotone piecewise cubic interpolation," *SIAM Journal on Numerical Analysis*, vol. 17, no. 2, pp. 238–246, 1980.

[19] V. Mohan and P. Morasso, "Passive motion paradigm: an alternative to optimal control," *Frontiers in Neurorobotics*, vol. 5, p. 4, 2011.

[20] P. Tommasino and D. Campolo, "An extended passive motion paradigm for human-like posture and movement planning in redundant manipulators," *Frontiers in Neurorobotics*, vol. 11, Nov 2017.

Glossary

List of Acronyms

FEP	Free Energy Principle
PMP	Passive Motion Paradigm
ROS	Robot Operating System
SPM	Statistical Parametric Mapping