BAUDOUIN BOSC

Geometric Dynamical Systems for Obstacle Avoidance in Manipulator Configuration Space

Supervised by Prof. A. Billard and Dr. B. Fichera
Submitted on July 12, 2024

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Motivation

Robotic manipulators are integral components of contemporary automation processes. They facilitate tasks such as lifting and moving objects for assembly processes or pick-and-place operations. With the diversification of the applications in which they are deployed, the algorithms used to program these robots must evolve to keep up with the increasing complexity and efficiency requirements.

Primarily, the role of these algorithms is to generate a trajectory for the manipulator to follow, enabling it to reach a specific pose to fullfill its task. Needless to say, most environments contain many different objects other than the manipulator itself. We often find other agents, humans or structures that the trajectory should take into account. Unwanted collisions with the manipulator may result in a safety, material, or a performance failure. While few applications only require that the tip of the manipulator avoids the obstacles, in general the whole body should be capable of doing so. With the increasing complexity of the operations performed by those robots, the motion strategies cannot limit themselves to preventing only the end-effector from undesired collisions, and must explore full-arm obstacle avoidance, where the whole structure of the robot can move in accordance with the environment's constraints.

At first glance, it should be possible to register the position of all the objects in the environment before the run, and compute the optimal trajectory offline. This method is often the simplest because there are no constraints on the computational time required by this approach. However, in reality, the environment is rarely static: other agents could be moving at the same time, or other unforeseen events could occur, making the surroundings unpredictable. In such dynamic situations, the motion planned offline becomes unreliable because the position of the obstacles may not be the same anymore. Furthermore, trying to update the motion at runtime with new obstacle positions is not an option due to the extensive computation time required: reacting to external disturbances takes too long. [1].

More reactive approaches are achieved with Artificial Potential Fields [1] or modulated Dynamical Systems (DS) [2], but they do not always guarantee convergence for some kind of obstacles (typically concave ones), and may be vulnerable to local minima [3]. Furthermore, modulated DS focuses only on task space obstacle avoidance. Model Predictive Control (MPC)-based approaches are able to compute collision-free trajectories, but the associated computational cost is often too heavy and limits either the reactiveness of the manipulator, or the quality of the generated trajectory ([3]). Despite the progress in computational hardware, methods relying on the latter might suffer heavy online computations due to the high complexity of the environments. Moreover, some hazardous situations reduce hardware availability, and

thereby the scope of usable methods [4]. While some approaches utilize a combination of MPC with DS or neural networks [3], [5] to mitigate the shortcomings of one method with the advantages of the other, there still remains a computational limit for prediction-based techniques, and an expressivity bottleneck for purely dynamic ones.

Contribution

In this work, we aim at designing a full-body obstacle avoidance purely based on a dynamical system, that overcomes the expressivity issues by utlizing differential geometry concepts. Using the underlying geometric structures of the environment, we build a geometric dynamical system (GDS) that avoids obstacles without being susceptible to local minima, while maintaining stability.

Challenges

Given the complex geometry of nowadays's robots, whole-arm obstacle avoidance presents several challenges. Multiple moving parts, and high number of degrees of freedom, results in a high-dimensional joint space. Firstly high-dimensionality implies a greater computational cost, making real-time adaptation more difficult. Secondly, the information about the workspace of the manipulator are usually not expressed in terms of joint configurations, except for joint limits. Typically, the position of the obstacles are given in Cartesian coordinates, and establishing a bridge between the task space and the configuration space is not trivial. In obstacle avoidance scenarios, a function providing a sense of closeness to an obstacle with respect to the joint configuration is required. It is impractical to compute an distance for every point in the manipulator's mesh given the high number of points shaping a manipulator's surface. Hence, a different method to represent the surface of the manipulator should be used, that allows one to measure the risk of collision depending on a joint configuration. Previous works already explored some modelling functions and proposed various solutions, like neural network-based implicit distance functions [5], bounding volumes [6], or gaussian mixture models (GMM) [7].

Once the first challenge is solved, a new description of the joint space will be available, containing the usual joint angles, in addition to a metric evaluating the closeness betwen the manipulator and the obstacles. Ultimately, the challenge emerging from this method is the formulation of a geometric dynamical system (GDS) that will be able to take advantage of this enhanced description. Our formulation, based on the encoding of non-linearity within space curvature [8] relies on the existence of a latent curved manifold representing the original space — the manipulator's configuration space. It requires a second-order differentiation of the function describing the curvature, and implementing this differentiation in a high-dimensional space while maintaining a fast time cycle is not trivial.

Chapter 2

Background

Our work relies on differential geometry concepts, that allows one to describe the dynamics of a system evolving on a Riemannian manifold. More precisely, we use the notion of geodesic to navigate in curved spaces, where the amount of curvature is decided by how close a joint configuration brings the manipulator to an obstacle. To understand this process, we develop in this chapter the notion of non-linearity and explain, with some approximations, the origin of geodesics. Throughout this development, we will use Einstein summation convention, and adopt the contracted notation $\partial_i \phi = \frac{\partial \phi}{\partial x^i}$.

2.1 Non-linear spaces

In contrast to linear spaces like $\mathbb{R}^2$, $\mathbb{R}^3$, or any $\mathbb{R}^n$ where $n \in \mathbb{N}^*$, a non-linear space, sometimes said to be non-Euclidean, is a space with a non-zero curvature. In these structures, the principles of linearity do not hold. For example, let us consider two points, $\mathbf{x}$ and $\mathbf{y}$ laying in $\mathbb{R}^3$: one of the properties of Euclidean space is that the distance $d \in \mathbb{R}$ between two points is defined as:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(x^1 - y^1)^2 - (x^2 - y^2)^2 - (x^3 - y^3)^2}$$

where the x^i and the y^j are the i-th and j-th coordinates of $\mathbf{x}$ and $\mathbf{y}$ respectively.

While this formula can be extended for any Euclidean space, it cannot be applied to any manifold. Typically, let us consider a simple two-dimensional Riemannian manifold like a sphere. On figure 2.1, both the plane $\mathcal{P}$ — assumed flat — and the sphere $\mathcal{S}$ are two-dimensional manifolds (i.e surfaces). Computing the distance between the point P and Q using the same distance formula as in the plane $\mathcal{P}$ would not give a correct result: if we draw a straight-line made of points which are on the sphere between P and Q, we follow the red line and not a usual *straight line* as we would have between A and B for example. This situation asks for an extension of the concept of distances and straight lines to Riemannian manifolds.

Taking a closer look to Riemannian manifolds: their specificities are that they are smooth, and equipped with a *Riemannian metric* g. A Riemannian metric yields positive-definite inner-product as a generalization of the classical inner product we find in Euclidean spaces between two vectors $\mathbf{u}$ and $\mathbf{v}$, $\mathbf{u} \cdot \mathbf{v} = u_1 v_1 + \cdots + u_n v_n$. This Riemannian metric, sometimes called metric tensor, is of utmost importance since it makes defining the length of a vector or the angle between two vectors possible. Recall that a Riemannian manifold is locally Euclidean. If the metric tensor allows us to define distances locally, then the length of a curve linking two points on a Riemannian manifold is the sum of the lengths of the infinitesimal curves that make up for the total curve.

(a) Plane (b) Sphere

Figure 2.1: Visualization of flat and curved surfaces

Formally, we consider a d-dimensional Riemannian manifold $\mathcal{M}$ equipped with a (positive-definite) metric tensor g. For a point $p \in \mathcal{M}$, a position $\mathbf{x}$ in the local Euclidean representation $\mathbb{R}^d$ of $\mathcal{M}$ is associated. Let's also denote by γ a curve on $\mathcal{M}$, that maps an interval $I \in \mathbb{R}$ to $\mathcal{M}$ such that: $\gamma : I \to \mathcal{M}$.

The length of γ is:

$$l = \int_a^b \sqrt{g_{\gamma(t)}(v_\gamma, v_\gamma)}\, dt$$

where v_γ is a vector field $v_\gamma : t \in I \to \mathcal{T}_p\mathcal{M}$.

$\mathcal{T}_p\mathcal{M}$ refers to the tangent space at $\gamma(t)$. This mapping provides the velocity of a particle moving along γ.

2.2 Geodesics

A geodesics is a curve representing the trajectory of a free particle on a manifold, i.e a particle on which no external forces apply. It means that a curve γ is a geodesic if at each point along it the acceleration is zero:

$$\nabla_{v_\gamma} v_\gamma = \mathbf{0} \tag{2.1}$$

This definition is equivalent to saying that a geodesic is a curve made of stationary points. That implies that the Euler-Lagrange equation applies:

$$\frac{\partial \mathcal{L}}{\partial x^\lambda} - \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{x}^\lambda}\right) = 0 \tag{2.2}$$

where:

- $\mathcal{L}$ is the Lagrangian of the system, typically defined as $\mathcal{L} = T - V = T$, with T being the kinetic energy and V the potential energy, which is zero in this situation since no external forces are present,
- x^λ are the generalized coordinates,
- $\dot{x}^\lambda$ are the generalized velocities, i.e., the derivatives of the generalized coordinates x^λ with respect to t.

The Lagrangian of the system is defined:

$$\mathcal{L} = T = \frac{1}{2} g_{\mu\nu} \dot{x}^\mu \dot{x}^\nu \tag{2.3}$$

Focusing on the first term of the Euler-Lagrange equation:

$$\frac{\partial \mathcal{L}}{\partial x^\lambda} = \frac{\partial}{\partial x^\lambda}\frac{1}{2}g_{\mu\nu}\dot{x}^\mu\dot{x}^\nu = \frac{1}{2}\partial_\lambda g_{\mu\nu}\dot{x}^\mu\dot{x}^\nu \tag{2.4}$$

Then, we can decompose the second term:

$$\frac{\partial \mathcal{L}}{\partial \dot{x}^\lambda} = g_{\mu\lambda}\dot{x}^\mu$$

$$\frac{d}{dt}g_{\mu\lambda}\dot{x}^\mu = \partial_\nu g_{\mu\lambda}\dot{x}^\mu\dot{x}^\nu + g_{\mu\nu}\ddot{x}^\mu = \frac{1}{2}\partial_\nu g_{\mu\lambda}\dot{x}^\mu\dot{x}^\nu + \frac{1}{2}\partial_\mu g_{\nu\lambda}\dot{x}^\mu\dot{x}^\nu + g_{\mu\nu}\ddot{x}^\mu \tag{2.5}$$

where the last equality is obtain by properties of the symmetry of g. Putting together equations 2.4 and 2.5, we obtain:

$$\begin{aligned}
\frac{1}{2}\partial_\lambda g_{\mu\nu}\dot{x}^\mu\dot{x}^\nu &= \frac{1}{2}\partial_\nu g_{\mu\lambda}\dot{x}^\mu\dot{x}^\nu + \frac{1}{2}\partial_\mu g_{\nu\lambda}\dot{x}^\mu\dot{x}^\nu + g_{\mu\nu}\ddot{x}^\mu \\
g_{\mu\nu}\ddot{x}^\mu &= -\frac{1}{2}(\partial_\nu g_{\mu\lambda} + \partial_\mu g_{\nu\lambda} - \partial_\lambda g_{\mu\nu})\dot{x}^\mu\dot{x}^\nu \\
g_{\mu\nu}\ddot{x}^\mu &= -\Gamma_{\lambda\mu\nu}\dot{x}^\mu\dot{x}^\nu
\end{aligned} \tag{2.6}$$

with $\Gamma_{\lambda\mu\nu}$ the Christoffel symbols of the first kind:

$$\Gamma_{\lambda\mu\nu} = \frac{1}{2}\left(\frac{\partial g_{\mu\lambda}}{\partial x^\nu} + \frac{\partial g_{\nu\lambda}}{\partial x^\mu} - \frac{\partial g_{\mu\nu}}{\partial x^\lambda}\right) \tag{2.7}$$

In matrix notation, we can highlight the acceleration of a particle following a geodesic:

$$\mathbf{G}(\mathbf{x})\ddot{\mathbf{x}} = -\bar{\bar{\mathbf{\Xi}}}\dot{\mathbf{x}} \tag{2.8}$$

2.3 The Riemannian metric tensor

As we saw in the previous equations, the metric tensor g of a manifold $\mathcal{M}$ characterizes the geodesic equation. Because g is directly related to the definition of distances on $\mathcal{M}$, one must find the expression of the embedding of $\mathcal{M}$ to formulate g. To do so, we first observe that a curved manifold $\mathcal{M}$ of dimension d possesses an embedded representation in $\mathbb{R}^{d+1}$. Recalling the example of figure 2.1, a sphere is a 2-dimensional manifold, that has an embedded representation in $\mathbb{R}^3$. With the support of [8], we define a smooth embedding $\mathbf{\Psi} : \mathbb{R}^d \to \mathbb{R}^{d+1}$ expressed as:

$$\mathbf{\Psi}(\mathbf{x}) = \begin{bmatrix} x^1 \\ \vdots \\ x^d \\ \psi(\mathbf{x}) \end{bmatrix}$$

where $\psi : \mathbb{R}^d \to \mathbb{R}$ is a smooth C^2 function defining the non-linearity of $\mathcal{M}$. The metric tensor, and thus the Christoffel symbols derive from it Firstly, we can define the embedding's Jacobian:

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \mathbf{I}_{d\times d} \\ \nabla\psi^T \end{bmatrix}$$

with $\nabla\psi = \begin{bmatrix}\partial_1\psi, \ldots, \partial_d\psi\end{bmatrix}^T$ the gradient of ψ. From this, the matrix notation of the metric tensor can be expressed:

$$\mathbf{G}(\mathbf{x}) = \mathbf{J}^T\mathbf{J} = \mathbf{I} + \nabla\psi\nabla\psi^T \tag{2.9}$$

corresponding to

$$g_{ij} = \delta_{ij} + \partial_i\psi\partial_j\psi,\ \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases}$$

2.4 Modellization as a damped harmonic oscillator

As in [8], our DS is modeled as a damped harmonic oscillator. For a point p on $\mathcal{M}$, we therefore have two external forces applying on it. An elastic force attracting the point towards an equilibrium point, and a dissipative force ensuring asymptotical stability around the latter. Following Newton' second law of motion, where the sum of the forces applied on a system amounts to its acceleration, we can enhance equation 2.1:

$$\nabla_{v_\gamma} v_\gamma = -d\phi - D(., v_\gamma) \tag{2.10}$$

where $d\phi$ is the elastic force associated to the potential ϕ and D the dissipative covector field.

Combining equations 2.6 and 2.10 yields the following dynamical system:

$$\begin{aligned} \ddot{x}^\lambda + \Gamma^\lambda_{\mu\nu}\dot{x}^\mu\dot{x}^\nu &= -g^{\lambda a}\partial_a\phi - g^{\lambda a}D_{am}\dot{x}^m \\ \ddot{x}^\lambda + \Gamma^\lambda_{\mu\nu}\dot{x}^\mu\dot{x}^\nu &= -g^{\lambda a}\partial_a\phi - D^\lambda_m\dot{x}^m \end{aligned} \tag{2.11}$$

with $\Gamma^\lambda_{\mu\nu} = \frac{1}{2}g^{\lambda m}\left(\frac{\partial g_{\mu\lambda}}{\partial x^\nu} + \frac{\partial g_{\nu\lambda}}{\partial x^\mu} - \frac{\partial g_{\mu\nu}}{\partial x^\lambda}\right)$ the Christoffel symbols of the second kind and $g^{\lambda a}$ the inverse of $g_{\lambda a}$. In vector notation:

$$\ddot{\mathbf{x}} = -\mathbf{G}(\mathbf{x})^{-1}\nabla\phi - \mathbf{D}\dot{\mathbf{x}} - \mathbf{\Xi}\dot{\mathbf{x}} \tag{2.12}$$

This equation will serve as the base for the development of our obstacle avoidance strategy.

Chapter 3

Representation of the environment

Our stragegy relies on introducing non-linearities in the configuration space of a manipulator, to represent joint angles we would like to avoid. Specifically, the configurations that should be avoided are the ones leading to a collision with an obstacle, or the configurations breaching the joint limits of the manipulator. Therefore, our efforts are firstly focused on providing a measure of closeness between the manipulator and the obstacles with respect to the joint angles.

3.1 Modellizing the robot

The most straightforward way of defining closeness between two objects is to compute an Euclidean distance between them. In this situation, one of the object being the surface of the manipulator, we would need to compute as many distances as there are points on the robot's surface. In practice, it is not achievable because the associated computation time is too high which is not compliant with real-time constraints. Indeed the point cloud representations usually contain hundreds of thousands of points. While it is possible to select a few to represent as closely as possible the surface, this method is still limited because it does not account for the orientations of the manipulator's links.

Instead, our work makes use of gaussian mixture model (GMM) to represent the distribution of the links' points. This method was already proved successfull in representing complex distributions with a relatively small set of parameters, as in [7] or [9], and is advantageous for several reasons. Firstly, GMMs can model a wide variety of shapes and structures by combining multiple Gaussian components. Secondly, the small number of Gaussian components needed to represent a surface reduces the computational requirements compared to using the raw data points directly. Finally, the isolines of the probability density modelled with the GMM gives a smooth representation of the robot's surface. The value of each isoline corresponds to the probability that the manipulator occupies a certain region of the space.

When using GMM to modelize data, several parameters will influence how well the model fits. Mainly:

- the type of covariance:
  - spherical: the covariance is isotropic, and the variance is considered equal in all directions. It is represented as a diagonal matrix, whose diagonal coefficients are equal.
  - diagonal: the directions of variations are independent. This is represented as a diagonal matrix.

(a) Link' surface (b) Pointcloud extracted from the stl

Figure 3.1: Pointcloud extraction of the 5^{th} link of a Franka Research 3 manipulator

  - full: the components may independently adopt any position and shape. It does not assume independence between the directions, often leading to a more accurate representation.

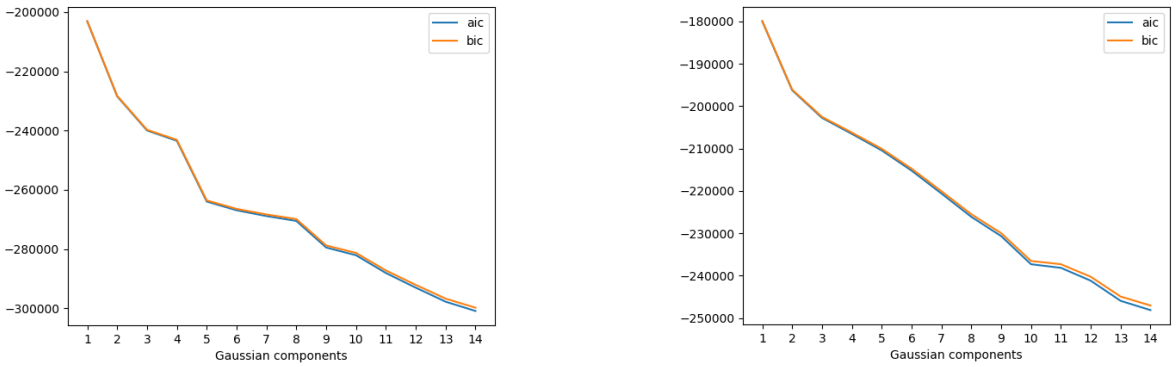
- the number of gaussian components. With more components, the representation grows in complexity at the risk of overfitting.

Choosing the type of covariance depends mostly on the complexity of the shapes one wants to modelize. In this scenario, choosing a *full* type of covariance is logical, given the complexity of the links' surfaces.

Usually, the number of gaussian components could be chosen by using a metric. The akaike information criterion (AIC) and the bayesian information criterion (BIC) are commonly used to evaluate statistical models because they balance model fit and complexity. For example, here is the resulting AIC and BIC scores for the first and sixth link of the Franka Research 3.

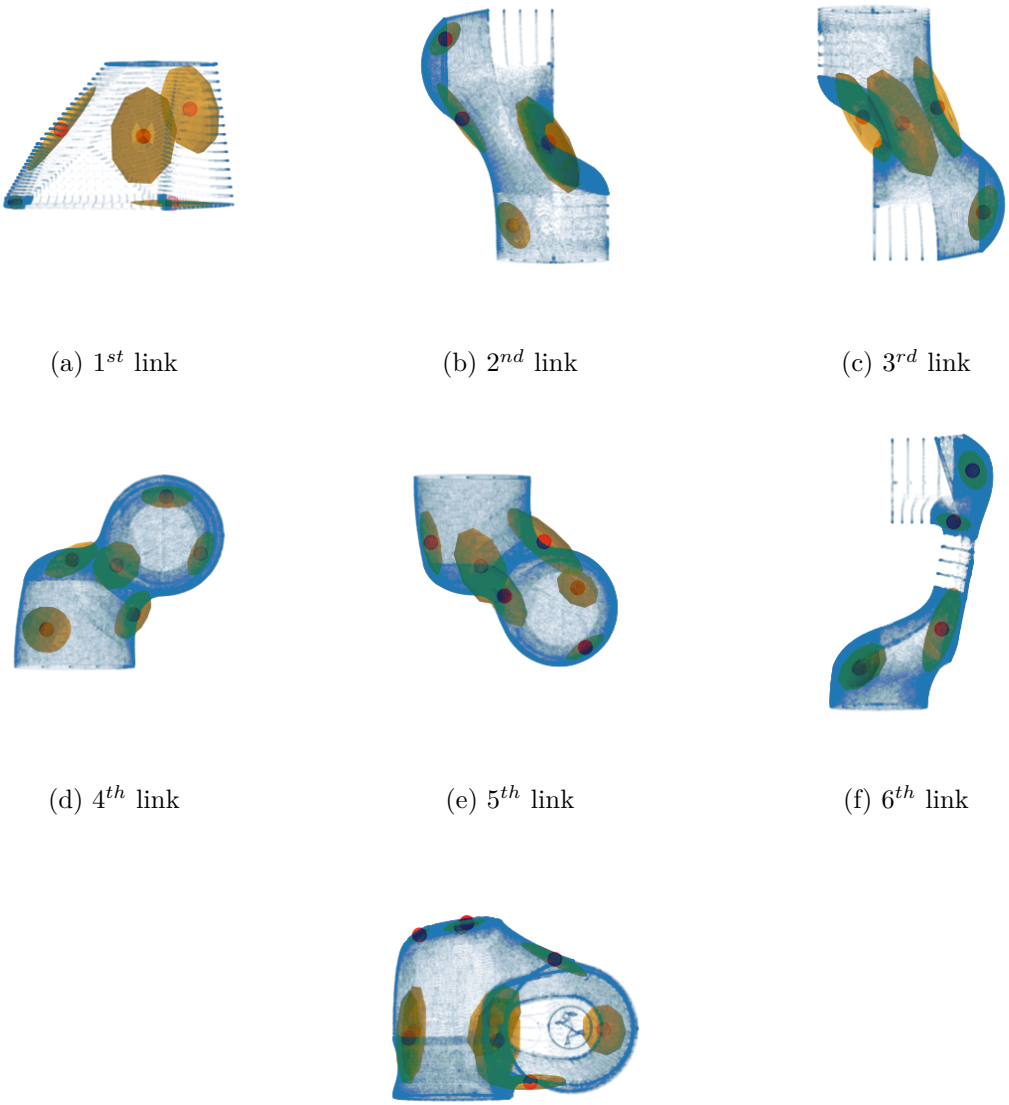
In our situation we want to favour a small complexity over accuracy, because one of the main points of this approach is the computation cost. Furthermore, there are some links that require a very small number of Gaussian components, because their influence in the avoidance task is very limited. This especially applies to the base link of a manipulator, since it is fixed. In conclusion, the optimal number of Gaussian components depends on the task and environment of the manipulator, and on the link itself. A situation where the complexity of the workspace is high requires a higher number of components. On the other hand, for simple scenarios we can minimize the computation cost by using a light model.

An example of GMM representation for a Franka Research 3 is shown in figure 3.3.



(a) 1^{st} link (b) 6^{th} link

Figure 3.2: AIC and BIC scores of GMM models after 10-fold cross-validation



(a) 1^{st} link (b) 2^{nd} link (c) 3^{rd} link

(d) 4^{th} link (e) 5^{th} link (f) 6^{th} link

(g) 7^{th} link

Figure 3.3: GMM representation of the Franka Research 3 manipulator

3.2 Proximity with an obstacle

In the following developments, we will consider a manipulator with d controllable, independent, revolute joints and K links. The joint state of the robot, or its configuration, is defined as the vector containing the angles of each joints $\mathbf{q} = \begin{bmatrix} q_1, \dots, q_d \end{bmatrix} \in \mathbb{R}^d$. The configuration space $\mathcal{C} \subset \mathbb{R}^d$ denotes every possible configuration of the manipulator within the joint limits $\mathbf{q}_{\text{min}}$ and $\mathbf{q}_{\text{max}}$. The task space $\mathcal{T} \subset \mathbb{R}^3$ defines the space in which the robot's task is conducted, i.e the space where the end-effector evolves. The position of obstacles will be defined as a point $\mathbf{x} \in \mathcal{T}$ or a set of points $\mathcal{O} \subset \mathcal{T}$.

A GMM is initially an unsupervised clustering technique used to determine the probability that a data point belongs to a cluster. In our situation, the clusters are the links of the manip-

Figure 3.4: Representation of a task space with a Franka Research 3 manipulator and an obstacle

ulator, and they are approximated by one or more Gaussians. By computing the probabilities that a point $\mathbf{x}$ belongs to the clusters, we get a measure of how close an obstacle is from each link. The higher the probability the closer it is.

To illustrate, we will consider a simple example, featuring a Franka Research 3 manipulator at a given joint configuration $\mathbf{q} = [q_1, \ldots, q_7]^T \in \mathbb{R}^7$ and an obstacle at a know position $\mathbf{x} = [x, y, z]^T \in \mathbb{R}^3$, see figure 3.4.

The robot is represented with one point cloud per link, and an obstacle (the black sphere) is represented as a single point. The collision probability between the i-th link and the obstacle is given by the value of the GMM-modelled probability density:

$$p_i(\mathbf{x}) = \sum_{k=1}^{K_i} \pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{3.1}$$

where $\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ is the multivariate Gaussian distribution:

$$\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{(2\pi)^{3/2}|\boldsymbol{\Sigma}_k|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right) \tag{3.2}$$

- $\mathbf{x}$ is the n-dimensional data vector.
- K is the number of Gaussian components representing the i-th link
- π_k is the mixing coefficient for the k-th Gaussian component, with $\sum_{k=1}^{K} \pi_k = 1$ and $0 \leq \pi_k \leq 1$.
- $\boldsymbol{\mu}_k$ is the mean vector of the k-th Gaussian component.
- $\boldsymbol{\Sigma}_k$ is the covariance matrix of the k-th Gaussian component.
- $|\boldsymbol{\Sigma}_k|$ denotes the determinant of the covariance matrix $\boldsymbol{\Sigma}_k$.

For a manipulator with several links, the total collision probability (unnormalized) is the summation of the individual collision probabilities of the links:

$$p(\mathbf{x}) = \sum_{i=1}^{d} p_i(\mathbf{x}) \tag{3.3}$$

If a link gets close to the obstacle, the distance between the position $\mathbf{x}$ of the obstacle and the means of the link's GMM $\boldsymbol{\mu}$ decrease and $(\mathbf{x} - \boldsymbol{\mu})$ decreases as well. Consequently, $p_i(\mathbf{x})$ increases.

3.3 Variation of the collision probability

What we seek to achieve now is to distinguish the joint configurations that are likely to end up in a collision, and those which are not. It is clear that the collision probability is not the same from one state to another, but this relation is not obvious in equation 3.1.

Because we assume that the position of the obstacles are not influenced by the manipulator, the only elements of equation 3.1 holding a dependency on $\mathbf{q}$ are the means $\boldsymbol{\mu}$ and the covariances $\boldsymbol{\Sigma}$. Indeed, when the configuration $\mathbf{q}$ of the manipulator changes, the pose of the links changes too. Consequently, the means and the covariances representing them change accordingly.

Hence, we can re-write the expression of the collision probability to highlight a dependency of $p(\mathbf{x})$ on $\mathbf{q}$.

$$p(\mathbf{x}, \mathbf{q}) = \sum_{i=1}^{d} \sum_{k=1}^{K_i} \pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k(\mathbf{q}), \boldsymbol{\Sigma}_k(\mathbf{q})) \tag{3.4}$$

To identify the full expression of p, we need to find the expressions of $\boldsymbol{\mu}(\mathbf{q})$ and $\boldsymbol{\Sigma}(\mathbf{q})$.

3.3.1 Expression of $\boldsymbol{\mu}(\mathbf{q})$

The means $\boldsymbol{\mu}$ are concretely the center of the Gaussian component, and consist in a position in the workspace of the robot. For a given joint configuration $\mathbf{q}$, the position of a mean belonging to a Gaussian component representing the i-th link is expressed in the workspace of the robot as:

(a) Mean variation

(b) Covariance variation

Figure 3.5: Visualization of a Gaussian component's variation w.r.t the joint configuration

$$\boldsymbol{\mu}_i = \mathbf{x}_{i-1} + \mathbf{R}_i \boldsymbol{\mu}_{0,i} \tag{3.5}$$

with $\mathbf{x}_{i-1}$ is the position of the i-th link's origin, $\mathbf{R}_i$ the rotation of the i-th link and $\boldsymbol{\mu}_{0,i}$ the position of the mean, defined with respect to the link's origin, when the link is in a neutral configuration (i.e $\mathbf{R}_i = \mathbf{0}$).

Hence, the equation defining the position of the mean of a Gaussian component representing the i-th link with respect to $\mathbf{q}$ is:

$$\boldsymbol{\mu}_i(\mathbf{q}) = \mathbf{x}_{i-1}(\mathbf{q}) + \mathbf{R}_i(\mathbf{q}) \boldsymbol{\mu}_{0,i} \tag{3.6}$$

$\mathbf{x}_{i-1}$ is result of the forward kinematics applied to a structure whose end-effector is the $(i-1)$-th link. $\mathbf{R}_i$ is discussed below.

3.3.2 Expression of $\boldsymbol{\Sigma}(\mathbf{q})$

The covariances $\boldsymbol{\Sigma}$ gives the information of how much the data (in this case the point cloud representing an object's body) are spread, and in which direction. Intuitively, we understand that this information can be represented by a set of vectors, whose origin is the mean of the Gaussian component they belong to. Hence, from one joint configuration to another, the covariances are rotated only. This is highlighted in figure 3.5. Thus, the expression of the covariances with respect to $\mathbf{q}$ is given by rotating the covariant matrix by the same angle as the link's.

$$\boldsymbol{\Sigma}_i(\mathbf{q}) = \mathbf{R}_i(\mathbf{q}) \boldsymbol{\Sigma}_{0,i} \mathbf{R}_i(\mathbf{q})^T \tag{3.7}$$

Again, $\mathbf{R}_i$ is the rotation the i-th link is subjected to, and $\boldsymbol{\Sigma}_{0,i}$ is the covariance matrix of the i-th link under no rotation.

3.3.3 Rotation of a link

In the subsections above, we denoted by $\mathbf{R}_i$ the rotation of the i-th link. Here is a clarification.

Let us consider a manipulator-like structure, with 3 links and 3 joints. A configuration of this manipulator, composed of its joint angles is written $\mathbf{q} = [q_1, q_2, q_3]^T \in \mathbb{R}^3$.

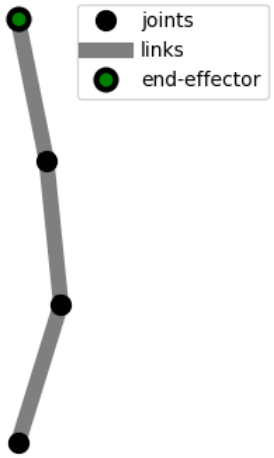


Figure 3.6: Visualization of a 3-dof manipulator

The position of the base is fixed, and the joints, which are considered revolute and independent, are designated as q_1, q_2 and q_3 from bottom to top. The i-th link corresponds to the link located between q_i and q_{i+1}.

The rotation matrix associated to an angle θ depends on which axis the rotation is about. For the three principal axes of $\mathbb{R}^3$, they are:

- Rotation matrix about the X-axis:

$$\mathbf{R}_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

- Rotation matrix about the Y-axis:

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

- Rotation matrix about the Z-axis:

$$\mathbf{R}_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

For the sake of this example, we will assume that every joint rotation is about the x-axis. Since the first link's rotation depends only on q_1, the rotation of the first link is written:

$$\mathbf{R}_1 = \mathbf{R}_x(q_1) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos q_1 & -\sin q_1 \\ 0 & \sin q_1 & \cos q_1 \end{bmatrix}$$

The second link's rotation however, depends on both q_1 and q_2. As such,$\mathbf{R}_2 = \mathbf{R}_x(q_2)\mathbf{R}_x(q_1)$ and in a similar fashion, $\mathbf{R}_3 = \mathbf{R}_x(q_3)\mathbf{R}_x(q_2)\mathbf{R}_x(q_1)$.

Extending this to a more general case of a manipulator with d revolute joints and k links, the rotation of the i-th link is written:

$$\mathbf{R}_i = \prod_{n=i}^{0} \mathbf{R}_{\gamma_n}(q_n) \tag{3.8}$$

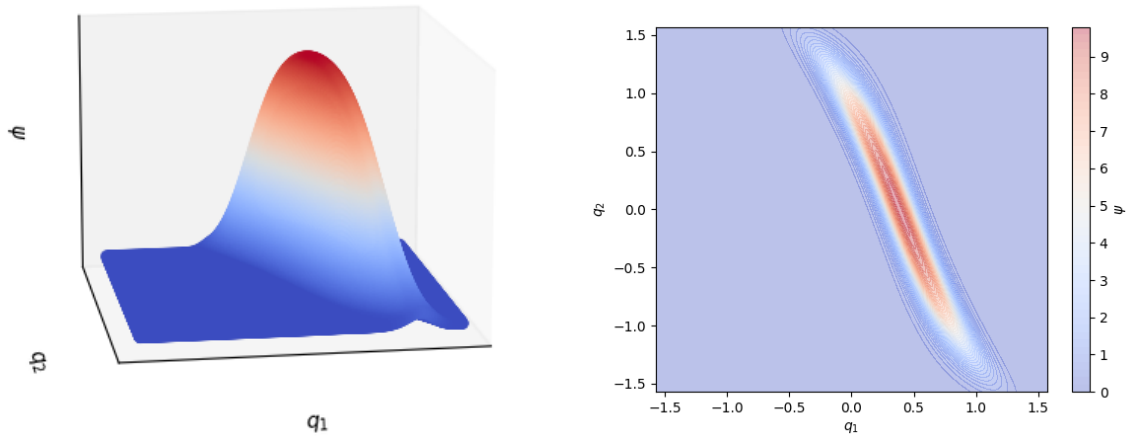
where γ_n refers to the rotation axis of the n-th joint, and q_n the associated rotation angle.

3.3.4 Embedding of the configuration space

By putting together equations 3.1, 3.6 and 3.7, $p(\mathbf{x}, \mathbf{q})$ can be directly computed. To visualize this probability, let us consider a simple scenario featuring a manipulator with 2 joints, hence a 2-dimensional configuration space. From the task space of the robot, a location $\mathbf{x}$ of an obstacle is known. Should we compute $p(\mathbf{x}, \mathbf{q})$ for every joint configuration $\mathbf{q} \in \mathbb{R}^2$, we end up with a mapping of the configuration space, where each joint configuration $\mathbf{q} = (q_1, q_2)$ is associated with a probability $p(\mathbf{x}, \mathbf{q})$. This mapping is a surface, corresponding to an $\mathbb{R}^3$ Euclidean space. It is presented in figure 3.7. Figure 3.7a is the embedded representation of the 2-dimensional joint space. The higher the collision probability, the higher the associated curvature (figure 3.7b).

As in chapter-2, we can build an embedding $\mathbf{\Psi}$ of the manipulator's configuration space and derive from it the metric tensor and the Christoffel symbols that are part of our DS.

$$\mathbf{\Psi}(\mathbf{x}) = \begin{bmatrix} q^1 \\ q^2 \\ \psi(\mathbf{x}, \mathbf{q}) \end{bmatrix} \equiv \begin{bmatrix} q^1 \\ q^2 \\ p(\mathbf{x}, \mathbf{q}) \end{bmatrix}$$



(a) Collision probability over $\mathcal{C}$

(b) Contour plot

Figure 3.7: Representation of the task space in the configuration space of a 2D manipulator

Chapter 4

Analytical differentiations of ψ

In this chapter, we derive from the embedding $\mathbf{\Psi}$ the expression of the metric tensor $\mathbf{G}$ and the Christoffel symbols $\mathbf{\Xi}$ that will be necessary for the formulation of our Dynamical System. We will follow the formulas introduced in chapter-2.

4.1 Notations

In the following section, the space in which the end-effector of a give manipulator evolves will be described as its cartesian space. Any point $\mathbf{x} \in \mathbb{R}^n$ in this cartesian space is of dimension n:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

At a given time t, the state of a manipulator with d joints can be represented by a vector $\mathbf{q} \in \mathbb{R}^d$:

$$\mathbf{q} = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_d \end{bmatrix}$$

where every q_i defines the angle of the joint i. The configuration space thus lies in a d-dimensional space.

4.2 Metric tensor

The metric tensor, denoted by $\mathbf{G}$ and introduced in equation 2.9 is defined as:

$$\mathbf{G}(\mathbf{q}) = \mathbf{J}^T\mathbf{J} = \mathbf{I}_{d\times d} + \nabla\psi(\cdot, \mathbf{q})\nabla\psi(\cdot, \mathbf{q})^T \tag{4.1}$$

with $\mathbf{J}$ the jacobian of the embedding, $\mathbf{I}_{d\times d}$ the $d \times d$ identity matrix, and ψ a function characterizing the non-linearity of the configuration space.

In this context, $\psi(\cdot, \mathbf{q})$ is a probability derived from the forward kinematics of the robot. Indeed, it is the output of the GMM associated to the links of the robot, see section-3.3.4.

The probability density function of a multivariate GMM is given by

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

where

$$\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}_k|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right)$$

As explained before, each link is represented by one or more Gaussians, and its associated centroids $\boldsymbol{\mu}_k$ and covariances $\boldsymbol{\Sigma}_k$ will change according to the configuration of the robot. Therefore, the equation for the embedding of the cartesian space, for an obstacle located in $\mathbf{x}$ is written as:

$$\psi(\mathbf{x}, \mathbf{q}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k(\mathbf{q}), \boldsymbol{\Sigma}_k(\mathbf{q})) \tag{4.2}$$

To compute the metric tensor, the gradient of the embedding is needed, and is defined as such:

$$\nabla\psi(\cdot, \mathbf{q}) = \begin{bmatrix} \frac{\partial\psi}{\partial q_1} \\ \vdots \\ \frac{\partial\psi}{\partial q_n} \end{bmatrix}$$

To compute those derivatives, we can write, according to the chain rule and assuming a fixed obstacle:

$$\frac{\partial\psi}{\partial\mathbf{q}} = \frac{\partial\psi}{\partial\boldsymbol{\mu}} \cdot \frac{\partial\boldsymbol{\mu}}{\partial\mathbf{q}} + \frac{\partial\psi}{\partial\boldsymbol{\Sigma}} \cdot \frac{\partial\boldsymbol{\Sigma}}{\partial\mathbf{q}} \tag{4.3}$$

In practice, this derivative is computed for each gaussian, and the summation of those contributions makes up for the total derivative of the embedding with respect to $\mathbf{q}$.

4.2.1 First order derivation of $\boldsymbol{\mu}$

The centroid of a Gaussian is a point lying in $\mathbb{R}^n$. As explained in section-3.3.1, the transformation from one configuration to another between the positions of the centroid of a link i, μ_i is:

$$\boldsymbol{\mu}_i = \mathbf{R}_i(\mathbf{q})\boldsymbol{\mu}_{0,i} + \mathbf{x}_{i-1} \tag{4.4}$$

$\mathbf{R}_i$ refers to the rotation of the i-th link, $\boldsymbol{\mu}_{0,i}$ the neutral position of the centroid (relative to the origin of the link) and $\mathbf{x}_i$ th position of the origin of the link.

The derivation of $\boldsymbol{\mu}_i$ with respect to q is therefore written:

$$\frac{\partial\boldsymbol{\mu}_i}{\partial\mathbf{q}} = \frac{\partial\boldsymbol{R}_i}{\partial\mathbf{q}}\boldsymbol{\mu}_{i,0} + \frac{\partial\mathbf{x}_{i-1}}{\partial\mathbf{q}} \tag{4.5}$$

Taking a closer look to the last term, we can link it to the Jacobian of the manipulator, also according to the chain rule.

Considering the translational Jacobian $\mathbf{J}_v$ of the link $i-1$, this relation applies:

$$\frac{\partial\mathbf{x}_{i-1}}{\partial t} = \mathbf{J}_v\frac{\partial\boldsymbol{q}}{\partial t} \Leftrightarrow \frac{\partial\mathbf{x}_{i-1}}{\partial\mathbf{q}} = \mathbf{J}_v$$

4.2.2 First order derivation of $\mathbf{R}_i$

The first term of equation 4.5, i.e the derivative of the rotation with respect to the joint configuration of the manipulator is defined as follow.

The rotation of the i-th link is defined as the product of the rotations of the previous links.

$$\mathbf{R}_i = \prod_{n=i}^{0} \mathbf{R}_{\gamma_n,n}$$

where γ_n refers to the axis of rotation associated to the link's parent joint, and i to the index of the link, see section-3.3.3.

The derivative of the rotation at the i-th link is thus:

$$\frac{\partial \mathbf{R}_i}{\partial \mathbf{q}} = \frac{\partial \boldsymbol{R}_{\gamma_i,i}}{\partial \mathbf{q}} \prod_{n=i-1}^{0} \mathbf{R}_{\gamma_n,n} + \mathbf{R}_{\gamma_i,i} \frac{\partial \boldsymbol{R}_{\gamma_{i-1},i-1}}{\partial \mathbf{q}} \prod_{n=i-2}^{0} \mathbf{R}_{\gamma_n,n} \ldots + \prod_{n=i}^{1} \mathbf{R}_{\gamma_n,n} \frac{\partial \boldsymbol{R}_{\gamma_0,0}}{\partial \mathbf{q}}$$

However, because each rotation $\mathbf{R}_{\gamma,n}$ depends only on the angle of its associated joint, we have:

$$\frac{\partial \mathbf{R}_{\gamma_i,i}}{\partial \mathbf{q_j}} = \begin{cases} \frac{\partial \mathbf{R}_{\gamma_i,i}}{\partial \mathbf{q}_j} & \text{if } i = j \\ \mathbf{0}_{n\times n} & \text{otherwise.} \end{cases}$$

The derivative of a rotation matrix with respect to its rotation angle is the element-by-element derivative. For example, considering a rotation about the Z-axis:

$$\frac{\partial \mathbf{R}_z(\theta)}{\partial \theta} = \begin{bmatrix} -\sin\theta & -\cos\theta & 0 \\ \cos\theta & -\sin\theta & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

To generalize this expression, we can use the properties of the rotation matrices.

A rotation matrix $\mathbf{R} \in SO(3)$ is orthogonal, and thus respects the following relation:

$$\mathbf{R}\mathbf{R}^T = \mathbf{I}$$

Using the standard derivation rules for matrices, we follow up with:

$$\frac{\partial \mathbf{R}}{\partial \theta}\mathbf{R}^T + \mathbf{R}\frac{\partial \mathbf{R}^T}{\partial \theta} = \mathbf{0} \Leftrightarrow \mathbf{S} + \mathbf{S}^T = \mathbf{0}$$

with $\mathbf{S} := \frac{\partial \mathbf{R}}{\partial \theta}\mathbf{R}^T$ a skew-symmetric matrix by definition.

By multiplying both sides of the definition of $\mathbf{S}$ by $\mathbf{R}$, we ultimately get:

$$\mathbf{R}\mathbf{S} = \frac{\partial \mathbf{R}}{\partial \theta}$$

For the three main axes, the skew-symmetric matrices can be easily computed:

- Skew-symmetric matrix for the X-axis ($\hat{\boldsymbol{\imath}}$):

$$\hat{\boldsymbol{\imath}}^{\times} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$$

- Skew-symmetric matrix for the Y-axis ($\hat{\boldsymbol{\jmath}}$):

$$\hat{\boldsymbol{\jmath}}^{\times} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}$$

- Skew-symmetric matrix for the Z-axis ($\hat{\mathbf{k}}$):

$$\hat{\mathbf{k}}^{\times} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

In general, given a rotation vector $\boldsymbol{\omega} = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^T$, the associated skew-symmetric matrix is given by:

$$\mathbf{S}(\boldsymbol{\omega}) = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

Therefore, we can complete the formula for the derivative of the rotation as:

$$\frac{\partial \boldsymbol{R}_i}{\partial \mathbf{q}} = \begin{bmatrix} \mathbf{S}(\boldsymbol{\omega_i})\mathbf{R}_{\gamma_i,i} \prod_{n=i-1}^{0} \mathbf{R}_{\gamma_n,n} & \dots & \prod_{n=i}^{1} \mathbf{R}_{\gamma_n,n} \mathbf{S}(\boldsymbol{\omega_0})\mathbf{R}_{\gamma_0,0} \end{bmatrix} \tag{4.6}$$

4.2.3 First order derivation of $\boldsymbol{\Sigma}$

Following section-3.3.2, the covariances $\boldsymbol{\Sigma}_i$ of the i-th link of a manipulator are actually the representation of n vectors, which are only subject to a rotation from one joint configuration to another, i.e:

$$\boldsymbol{\Sigma}_i = \boldsymbol{R}_i(\mathbf{q})\boldsymbol{\Sigma}_{i,0}\boldsymbol{R}_i^T(\mathbf{q}) \tag{4.7}$$

where $\boldsymbol{\Sigma}_{i,0}$ is the previous covariance matrix.

The derivative of the covariance matrix with respect to the joint configuration is the derivative of a product of matrices, i.e:

$$\frac{\partial \boldsymbol{\Sigma}_i}{\partial \mathbf{q}} = \frac{\partial \boldsymbol{R}_i}{\partial \mathbf{q}} \boldsymbol{\Sigma}_{i,0}\boldsymbol{R}_i^T(\mathbf{q}) + \boldsymbol{R}_i(\mathbf{q})\boldsymbol{\Sigma}_{i,0}\left(\frac{\partial \boldsymbol{R}_i}{\partial \mathbf{q}}\right)^T \tag{4.8}$$

and $\frac{\partial \boldsymbol{R}_i}{\partial \mathbf{q}}$ is the one defined in 4.6.

4.2.4 Expression of $\frac{\partial \psi}{\partial \boldsymbol{\mu}}$

Let us recall the full expression of the embedding ψ:

$$\psi(\mathbf{x}, \mathbf{q}) = \sum_{k=1}^{K} \pi_k \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}_k|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right)$$

Using basic linear algebra, we can develop:

$$\begin{aligned} \frac{\partial \psi}{\partial \boldsymbol{\mu}} &= \sum_{k=1}^{K} \pi_k \frac{\partial}{\partial \boldsymbol{\mu}} \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \\ &= \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \frac{1}{2}\left(\boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k) + (\mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}\right) \\ &= \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\left(\boldsymbol{\Sigma}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right) \\ &\text{since } \boldsymbol{\Sigma}_k \text{ is positive semi-definite (PSD)} \end{aligned} \tag{4.9}$$

4.2.5 Expression of $\frac{\partial \psi}{\partial \mathbf{\Sigma}}$

Similarily to equation 4.9, we can write:

$$\begin{aligned}
\frac{\partial \psi}{\partial \mathbf{\Sigma}} &= \sum_{k=1}^{K} \pi_k \frac{\partial}{\partial \mathbf{\Sigma}} \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \mathbf{\Sigma}_k) \\
&= \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \mathbf{\Sigma}_k) \frac{1}{2} \left(\mathbf{\Sigma}_k^{-T} (\mathbf{x} - \boldsymbol{\mu}_k)(\mathbf{x} - \boldsymbol{\mu}_k)^T \mathbf{\Sigma}_k^{-T} \right) \\
&= \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \mathbf{\Sigma}_k) \frac{1}{2} \left(\mathbf{\Sigma}_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k)(\mathbf{x} - \boldsymbol{\mu}_k)^T \mathbf{\Sigma}_k^{-1} \right) \text{since } \mathbf{\Sigma}_k \text{ is PSD}
\end{aligned} \tag{4.10}$$

With all the necessary components computed, the metric tensor can be formed according to equation 4.1, and the harmonic part is now complete.

4.3 Christoffel symbols

The main terms of the geodesic part of the dynamical system are the Christoffel symbols. Let us recall their general formula:

$$\Gamma_{ij}^k = \frac{1}{2} \left(\frac{\partial g_{il}}{\partial q^j} + \frac{\partial g_{jl}}{\partial q^i} - \frac{\partial g_{ij}}{\partial q^l} \right)$$

where:

- g_{ij} is the metric tensor.
- $\frac{\partial g_{il}}{\partial q^j}$ denotes the partial derivative of g_{il} with respect to q^j.

The metric tensor was calculated above, so the missing term to compute those symbols is the derivative of the metric tensor with respect to the joint configuration $\mathbf{q}$.

Let us decompose the derivative of the metric tensor as such:

$$\begin{aligned}
\frac{\partial \mathbf{G}}{\partial \mathbf{q}} &= \frac{\partial}{\partial \mathbf{q}} (\mathbf{I}_{d \times d} + \nabla \psi(\mathbf{q}) \nabla \psi(\mathbf{q})^T) \\
&= \frac{\partial \nabla \psi}{\partial \mathbf{q}} \nabla \psi(\mathbf{q})^T + \nabla \psi (\frac{\partial \nabla \psi}{\partial \mathbf{q}})^T
\end{aligned} \tag{4.11}$$

Hence, the only needed term to compute the Christoffel symbols is $\frac{\partial \nabla \psi}{\partial \mathbf{q}}$.

Similarly as in equation 4.3, we can use the chain rule to write:

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{q}} \nabla \psi &= \frac{\partial}{\partial \mathbf{q}} \left(\frac{\partial \psi}{\partial \boldsymbol{\mu}} \cdot \frac{\partial \boldsymbol{\mu}}{\partial \mathbf{q}} + \frac{\partial \psi}{\partial \mathbf{\Sigma}} \cdot \frac{\partial \mathbf{\Sigma}}{\partial \mathbf{q}} \right) \\
&= \frac{\partial^2 \psi}{\partial \mathbf{q} \partial \boldsymbol{\mu}} \frac{\partial \boldsymbol{\mu}}{\partial \mathbf{q}} + \frac{\partial \psi}{\partial \boldsymbol{\mu}} \frac{\partial^2 \boldsymbol{\mu}}{\partial \mathbf{q}^2} + \frac{\partial^2 \psi}{\partial \mathbf{q} \partial \mathbf{\Sigma}} \frac{\partial \mathbf{\Sigma}}{\partial \mathbf{q}} + \frac{\partial \psi}{\partial \mathbf{\Sigma}} \frac{\partial^2 \mathbf{\Sigma}}{\partial \mathbf{q}^2}
\end{aligned} \tag{4.12}$$

4.3.1 Expression of $\frac{\partial^2 \psi}{\partial \mathbf{q} \partial \mu}$

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{q}} \frac{\partial \psi}{\partial \boldsymbol{\mu}} &= \sum_{k=1}^{K} \pi_k \frac{\partial}{\partial \mathbf{q}} \left(\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \left(\boldsymbol{\Sigma}_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k) \right) \right) \\
&= \sum_{k=1}^{K} \pi_k \frac{\partial}{\partial \mathbf{q}} (\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)) \left(\boldsymbol{\Sigma}_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k) \right) \\
&\qquad + \pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \frac{\partial}{\partial \mathbf{q}} \left(\boldsymbol{\Sigma}_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k) \right) \\
&= \sum_{k=1}^{K} \pi_k \frac{\partial}{\partial \mathbf{q}} (\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)) \left(\boldsymbol{\Sigma}_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k) \right) \\
&\qquad + \pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \left(\frac{\partial \boldsymbol{\Sigma}_k^{-1}}{\partial \mathbf{q}} (\mathbf{x} - \boldsymbol{\mu}_k) - \boldsymbol{\Sigma}_k^{-1} \frac{\partial \boldsymbol{\mu}_k}{\partial \mathbf{q}} \right)
\end{aligned} \tag{4.13}$$

with

$$\frac{\partial \Sigma_k^{-1}}{\partial \mathbf{q}} = -\boldsymbol{\Sigma}_k^{-1} \frac{\partial \boldsymbol{\Sigma}_k}{\partial \mathbf{q}} \boldsymbol{\Sigma}_k^{-1}, \text{ see [10]}$$

and $\frac{\partial}{\partial \mathbf{q}}(\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k))$, $\frac{\partial \boldsymbol{\Sigma}_k}{\partial \mathbf{q}}$, $\frac{\partial \boldsymbol{\mu}_k}{\partial \mathbf{q}}$ which have already been computed in equation 4.3.

4.3.2 Expression of $\frac{\partial^2 \psi}{\partial \mathbf{q} \partial \Sigma}$

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{q}} \frac{\partial \psi}{\partial \boldsymbol{\Sigma}} &= \sum_{k=1}^{K} \pi_k \frac{\partial}{\partial \mathbf{q}} \left(\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \frac{1}{2} \left(\boldsymbol{\Sigma}_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k)(\mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} \right) \right) \\
&= \sum_{k=1}^{K} \pi_k \frac{\partial}{\partial \mathbf{q}} \left(\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right) \frac{1}{2} \left(\boldsymbol{\Sigma}_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k)(\mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} \right) \\
&\qquad + \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \frac{1}{2} [\frac{\partial \boldsymbol{\Sigma}_k^{-1}}{\partial \mathbf{q}} (\mathbf{x} - \boldsymbol{\mu}_k)(\mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} \\
&\qquad - \boldsymbol{\Sigma}_k^{-1} \frac{\partial \boldsymbol{\mu}_k}{\partial \mathbf{q}} (\mathbf{x} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} \\
&\qquad - \boldsymbol{\Sigma}_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k) \frac{\partial \boldsymbol{\mu}_k}{\partial \mathbf{q}}^T \boldsymbol{\Sigma}_k^{-1} \\
&\qquad + \boldsymbol{\Sigma}_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k)(\mathbf{x} - \boldsymbol{\mu}_k)^T \frac{\partial \boldsymbol{\Sigma}_k^{-1}}{\partial \mathbf{q}}]
\end{aligned} \tag{4.14}$$

4.3.3 Expression of $\frac{\partial^2 \boldsymbol{\mu}_k}{\partial \mathbf{q}^2}$

The computation of the Christoffel symbols requires the computation of $\frac{\partial^2 \boldsymbol{\mu}_k}{\partial \mathbf{q}^2}$.
Starting from equation 4.5, we can write:

$$\frac{\partial^2 \boldsymbol{\mu}_i}{\partial \mathbf{q}^2} = \frac{\partial^2 \mathbf{R}_i}{\partial \mathbf{q}^2} \boldsymbol{\mu}_{i,0} + \frac{\partial^2 \mathbf{x}_{i-1}}{\partial \mathbf{q}^2} \tag{4.15}$$

The second partial derivative of the rotation is straightforward. Using equation 4.6,

$$\frac{\partial^2 \mathbf{R}_\gamma(\theta)}{\partial \theta^2} = \mathbf{S}(\omega) \frac{\partial \mathbf{R}_\gamma(\theta)}{\partial \theta} = \mathbf{S}(\omega)^2 \mathbf{R}_\gamma(\theta)$$

If $\frac{\partial \boldsymbol{R}_i}{\partial \mathbf{q}}$ could be expressed as a $d \times n \times n$ tensor, then $\frac{\partial^2 \mathbf{R}_i}{\partial \mathbf{q}^2}$ can be expressed as a $d \times d \times n \times n$ tensor where:

$$\begin{aligned}\frac{\partial^2 \mathbf{R}_i}{\partial \mathbf{q}^2}_{pq} &= \frac{\partial}{\partial \mathbf{q}_p}\mathbf{R}_{\gamma_i,i}\dots S(\boldsymbol{\omega}_q)\mathbf{R}_{\gamma_q,q}\dots\mathbf{R}_{\gamma_0,0} \\ &= \begin{cases} \frac{\partial}{\partial \mathbf{q}_p}\mathbf{R}_{\gamma_i,i}\dots S(\boldsymbol{\omega}_p)\mathbf{R}_{\gamma_p,p}\dots S(\boldsymbol{\omega}_q)\mathbf{R}_{\gamma_q,q}\dots\mathbf{R}_{\gamma_0,0} & \text{if } p > q \\ \frac{\partial}{\partial \mathbf{q}_p}\mathbf{R}_{\gamma_i,i}\dots S(\boldsymbol{\omega}_p)^2\mathbf{R}_{\gamma_p,p}\dots\mathbf{R}_{\gamma_0,0} & \text{if } p = q \end{cases}\end{aligned} \tag{4.16}$$

The second term of equation 4.15, $\frac{\partial^2 \mathbf{x}_{i-1}}{\partial \mathbf{q}^2}$ is the second derivative of the considered link's origin. In practice, it is possible to obtain such value with a rigid body dynamics library, e.g [11]. If not available, one can start from this relation and use the chain rule:

$$\frac{\partial \mathbf{x}}{\partial \mathbf{q}} = \mathbf{J} \Leftrightarrow \frac{\partial}{\partial \mathbf{q}}\frac{\partial \mathbf{x}}{\partial \mathbf{q}} = \frac{\partial \mathbf{J}}{\partial \mathbf{q}} \Leftrightarrow \frac{\partial^2 \mathbf{x}}{\partial \mathbf{q}^2}\frac{\partial \mathbf{q}}{\partial t} = \frac{\partial \mathbf{J}}{\partial \mathbf{q}}\frac{\partial \mathbf{q}}{\partial t} \Leftrightarrow \frac{\partial^2 \mathbf{x}}{\partial \mathbf{q}^2}\dot{\mathbf{q}} = \dot{\mathbf{J}}$$

4.3.4 Expression of $\frac{\partial^2 \boldsymbol{\Sigma}_i}{\partial^2 \mathbf{q}}$

Based on equation 4.8, the second order derivative of the covariances with respect to the joint configuration is:

$$\frac{\partial^2 \boldsymbol{\Sigma}_i}{\partial^2 \mathbf{q}} = \frac{\partial^2 \boldsymbol{R}_i}{\partial \mathbf{q}^2}\boldsymbol{\Sigma}_{i,0}\boldsymbol{R}_i^T(\mathbf{q}) + 2\frac{\partial \boldsymbol{R}_i}{\partial \mathbf{q}}\boldsymbol{\Sigma}_{i,0}\left(\frac{\partial \boldsymbol{R}_i}{\partial \mathbf{q}}\right)^T + \boldsymbol{R}_i(\mathbf{q})\boldsymbol{\Sigma}_{i,0}\left(\frac{\partial^2 \boldsymbol{R}_i}{\partial \mathbf{q}^2}\right)^T \tag{4.17}$$

Now that all the required terms have been computed, they can be put together as described in equations 4.12 and 4.11 to build the Christoffel symbols.

Chapter 5

Geometric Dynamical System

Now that the expression of the embedding function ψ, the embedded representation's metric tensor $\mathbf{G}$ and the associated Christoffel symbols $\mathbf{\Xi}$ are formulated, we can finally implement the dynamical system introduced in equation 2.12.

5.1 Basics of dynamical systems

A system is said to be dynamical when it changes through time. At a given time, the system is described by a state, and an associated function will describe the future state that will follow the current one.

For example, let us consider a system described by the following differential equation:

$$\dot{x} = -kx$$

Since it's a first-order differential equation, it corresponds to a first-order dynamical system. If the state at a time t is known, it is possible to compute the system's velocity:

$$\dot{x}_t = -kx_t$$

After a timestep dt, the future state of the system can be found by numerically integrating the speed once, according to the forward Euler method:

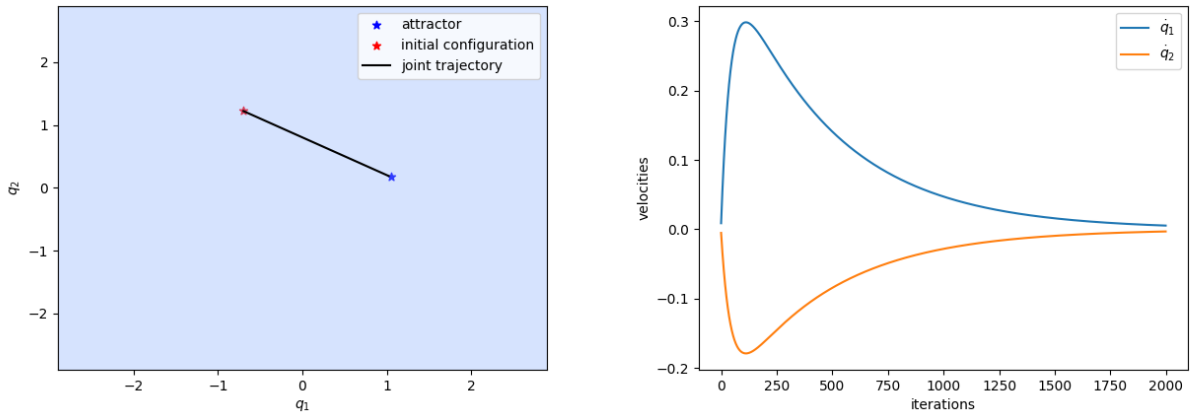
$$x_{t+1} = x_t + \dot{x}_t dt$$

One of the advantages of those systems is their capacity to react quickly to a perturbation. Indeed, because only the next state is computed from one timestep to another, the computational time is rather short and allows for high control frequency.

In our situation, the system we would like to characterize is the joint configuration of a manipulator, denoted with $\mathbf{q} \in \mathbb{R}^d$, where d is the number of joints the manipulator possesses. Eventually, the state of the system should reach a targeted joint configuration, that we will denote by $\mathbf{q}^*$.

We modellize this system as a damped harmonic oscillator: a particle ($\mathbf{q}$) is put far from its equilibrium state ($\mathbf{q}^*$). Upon release, it should try to reach this equilibrium state to achieve the desired task. To prevent oscillation around this point, we also add a dissipation term acting on the velocity of the system. With no other external forces applied, the expression of such system in matrix form is written:

$$\ddot{\mathbf{q}} = -\mathbf{K}(\mathbf{q} - \mathbf{q}^*) - \mathbf{D}\dot{\mathbf{q}} \tag{5.1}$$



(a) Joint trajectory

(b) Joint velocities

Figure 5.1: Trajectory computed from equation 5.1 in a flat space

With $\mathbf{K}$ the stiffness matrix, and $\mathbf{D}$ the damping coefficients. The system is massless, since it is a representation of a joint configuration. By integrating twice the computed joint acceleration, the system would eventually converge towards the target $\mathbf{q}^*$.

$$\begin{cases} \ddot{\mathbf{q}} = -\mathbf{K}(\mathbf{q} - \mathbf{q}^*) - \mathbf{D}\dot{\mathbf{q}} \\ \dot{\mathbf{q}} = \dot{\mathbf{q}} + \ddot{\mathbf{q}}dt \\ \mathbf{q} = \mathbf{q} + \dot{\mathbf{q}}dt \end{cases} .$$

On a flat, Euclidean space, this dynamical system reaches the intended position by following a straight line. This trajectory is shown in figure 5.1. The velocity profiles show the effect of the dissipative term, eventually bringing the velocity to 0 near the attractor.

However, this formulation is not adapted to non-linear spaces, like the distorted configuration space in figure 3.7b. In fact, no terms in this equation are accounting for the deformation of the space. Hence, the trajectory is the same whether an obstacle is present in the workspace of the robot, or not, eventually leading to a collision if the obstacle is on the manipulator's path. Differential geometry provides the tools to remediate this situation.

5.2 Navigating in non-linear spaces

As explained before, the configuration space of our manipulator is extended by another dimension, i.e the collision probability. This extended space, subset of $\mathbb{R}^{d+1}$, is the linear representation of the non-linear d-dimensional joint space where the amount of non-linearity is encoded by the probability. Therefore, the higher the probability the greater the associated curvature. Since the non-linear regions represent the configurations where a collision is likely, the dynamical system we are trying to implement must be capable of navigating the distorted space while avoiding the non-linear regions, and must then be aware of the deformations. The avoidance should be stronger as the deformation grows, too.

This behaviour is similar to spacetime curvature in general relativity. In this context, gravity is not described as a force anymore, but as a result of the curvature of spacetime. This curvature is caused by matter, and the 'heavier' this matter is, the greater the resulting curvature.

For example, this theory is able to explain the trajectory of light around heavy celestial bodies like stars or black holes.

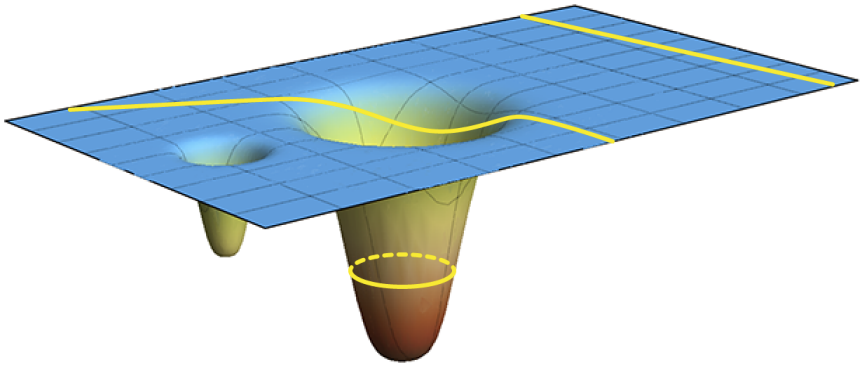


Figure 5.2: Trajectory of light depending on spacetime curvature

Figure 5.2 is an illustration of the path of light for different degrees of spacetime curvature. Of course, this representation is flawed, e.g the 4D manifold is represented as a surface. But the point remains: the curvature affects the trajectory of a particle traveling in its neighbourhood.

Instead of the Newtonnian $\mathbf{F} = m\mathbf{a}$, the theory of general relativity proposes that free-falling particles move along locally straight paths, the so-called **geodesics**. As explained in chapter-2, geodesics generalize the notion of straight lines from Euclidean spaces to Riemannian manifolds, and describe free-falling accelerations, where no external forces are applied on the system/particle.

As stated before, equation 2.1 describes the acceleration of a particle following a curve γ on a Riemannian manifold. When no external forces apply, we can solve this equation:

$$\left(\nabla_{v_\gamma} v_\gamma\right)^k = 0 \Leftrightarrow g_{km}\ddot{x}^k + \Gamma^k_{ij}\dot{x}^i\dot{x}^j = 0 \tag{5.2}$$

and that corresponds to the equation of a geodesic. A particle following it should thus be able to avoid the highly non-linear regions in a way similar to light around heavy celestial bodies.

By integrating this equation to our dynamical system, we generate a trajectory that can avoid obstacles in a similar fashion.

5.3 Geometric Dynamical System

Adapting equation 5.2 for a joint configuration $\mathbf{q}$ yields the following dynamical system:

$$\ddot{q}^k = -g^{km}\Gamma^k_{ij}\dot{q}^i\dot{q}^j$$

or, in matrix notation:

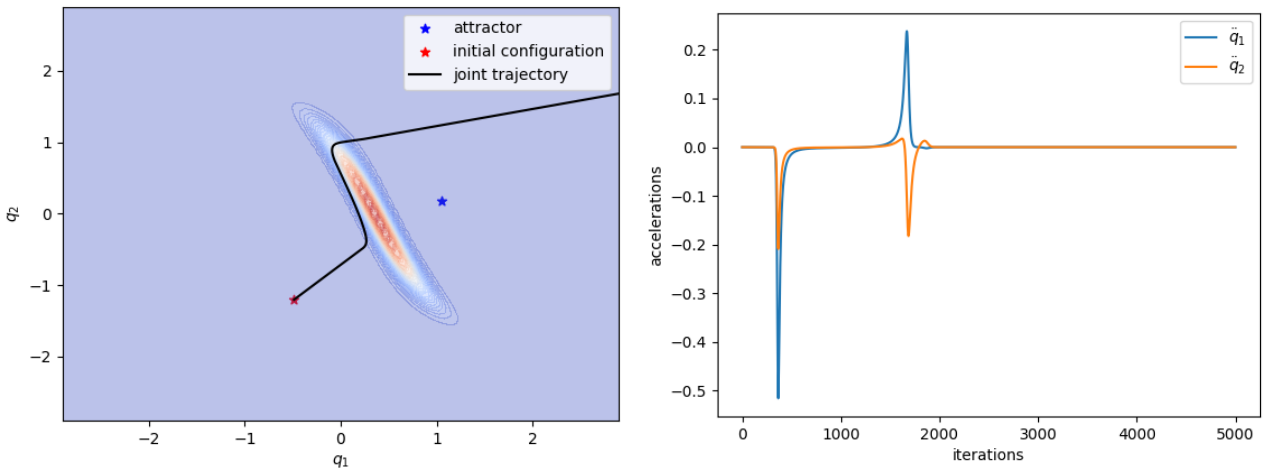
$$\ddot{\mathbf{q}} = -\boldsymbol{\Xi}\dot{\mathbf{q}} \tag{5.3}$$

where the coefficients of $\boldsymbol{\Xi}$ are the $\Gamma^k_{ij}\dot{q}^i$. In a situation similar to figure 3.7, a system following the dynamical system described in equation 5.3 would behave as shown in figure 5.3.

This trajectory is obtained by giving an initial speed to the system in the direction of the attractor.

The main observation here is that the trajectory, as expected, avoids the most non-linear regions of the embedded configuration space. One thing of importance however, is that the

[1] image source: https://writescience.com



(a) Joint trajectory

(b) Joint accelerations

Figure 5.3: Pure geodesic motion (equation 5.3)

attractor (the targetted joint configuration) is not reached. Indeed, no terms in the geodesic equation are providing information regarding a potential target.

Recall that the geodesic is the trajectory of a particle under no other influence than the curvature of the space in which it evolves. Therefore, to make the dynamical system reach a target, the elastic force and the dissipative force should be added to the equation, as per equation 2.12.

Applying equation 2.12 to a joint configuration $\mathbf{q}$, the final DS is:

$$\ddot{\mathbf{q}} = -\underbrace{\mathbf{G}(\mathbf{q})^{-1}\mathbf{K}(\mathbf{q}-\mathbf{q}^*) - \mathbf{D}\dot{\mathbf{q}}}_{\text{harmonic}} \underbrace{-\boldsymbol{\Xi}\dot{\mathbf{q}}}_{\text{geodesic}} \tag{5.4}$$

Implementing the DS as per equation 5.4 yields the trajectory in figure 5.4a. Because the forces of the harmonic part of the dynamical system and the acceleration resulting of the space curvature are conflicting, the trajectory comes to a stop in front of the non-linear region. The former is driving the system towards the attractor regardless of the curvature, while the latter is trying to avoid it. Simply put, the harmonic part is adapted when no curvature is encountered, while the geodesic part is tailored for the non-linear regions. To produce a dynamical system able to both avoid the obstacle and reach the intended target, it is necessary to modulate between the harmonic and the geodesic term, depending on the local curvature.

A simple yet effective switching is the following:

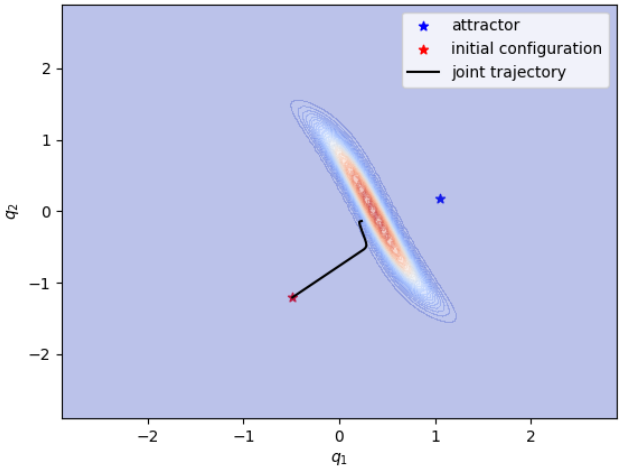
$$\sigma = \begin{cases} 1 & \text{if } \psi \leq \kappa \\ 0 & \text{otherwise.} \end{cases} \tag{5.5}$$

where κ is an arbitrary threshold. Because the probability distribution issued from ψ is unnormalized, this coefficient may depend on the number of gaussian components, or the size of the obstacles. The resulting dynamical system is as follow:

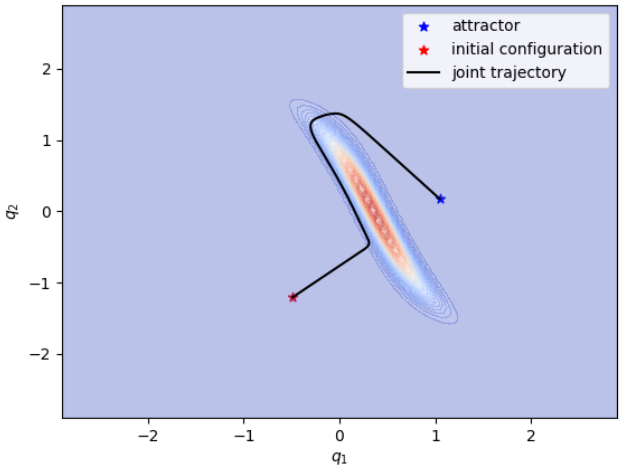
$$\ddot{\mathbf{q}} = -\sigma(\mathbf{G}(\mathbf{q})^{-1}\mathbf{K}(\mathbf{q}-\mathbf{q}^*) + \mathbf{D}\dot{\mathbf{q}}) - (1-\sigma)\boldsymbol{\Xi}\dot{\mathbf{q}} \tag{5.6}$$

and the resulting trajectory is observed in figure 5.4b.

However, κ depends heavily on the scenario: the number of joints, the GMM representation of the manipulator, or the obstacles. As such, it requires tuning, otherwise the motion may result in a collision. Figure 5.5 shows the effect of kappa on the trajectory of the manipulator. Later on, we will see how to mitigate this drawback.

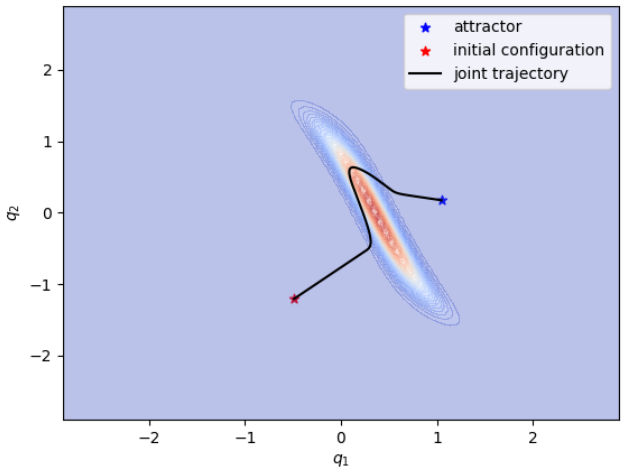


(a) Without switch, see equation 5.4

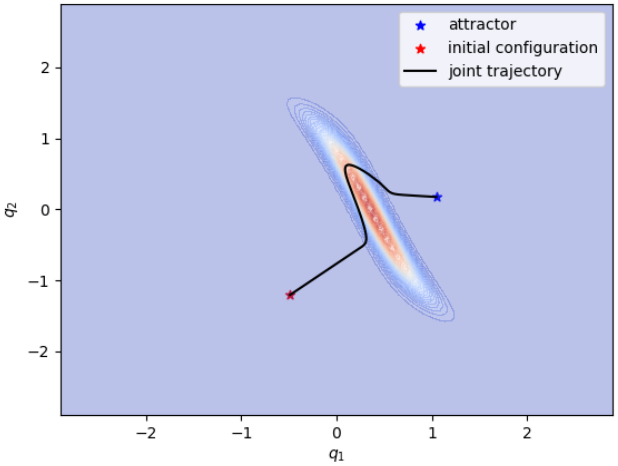


(b) With a basic switch, see equation 5.6

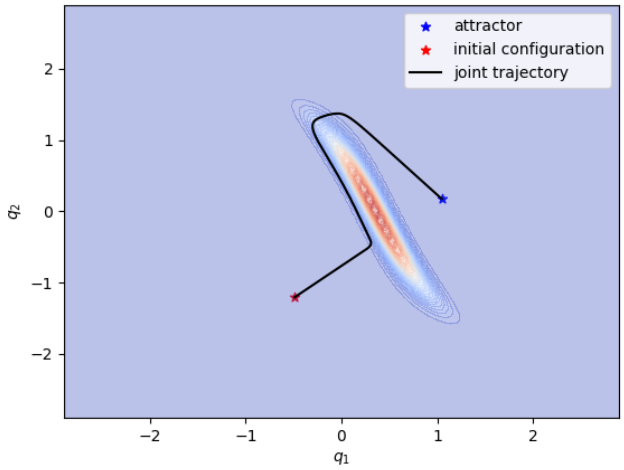
Figure 5.4: Joint trajectory of a 2d manipulator



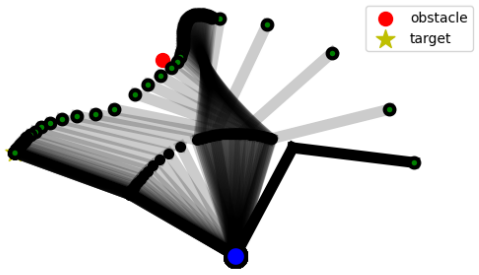
(a) $\kappa = 0.1$



(b) $\kappa = 0.3$



(c) $\kappa = 0.5$



(d) Task space trajectory for $\kappa = 0.5$

Figure 5.5: Effect of the probability threshold κ

5.4 Handling joint limits

Limiting the joint angles of the trajectory is crucial to ensure safe and reliable operation of robotic systems. With this DS, it can be done geometrically by placing potential boundaries in the joint space. The velocity, acceleration and torque limits are handled by the model-based QP controller presented in equation 6.3.

Following the concepts used so far, a joint angle limit consists of preventing the trajectory to reach certain regions within the configuration space. That, in essence, is similar to an obstacle avoidance problem. To do so, we introduce non-linearities along the joint limits and consider these non-linearities as borders the system should not cross.

Formally, we want to maintain the condition:

$$\mathbf{q}^- \leq \mathbf{q} \leq \mathbf{q}^+$$

There are several approaches that come to mind when one need to implement a repulsive barrier:

- kernels: it consists of placing several kernels at the borders, and computing the distance between the system's position and all the placed kernels. The smaller the distance, the greater the repulsive response. In practice, it works well in low dimensional problem (2D or 3D). However, because the spatial complexity grows exponentially, the number of required kernels is too high at higher dimensions (e.g 7) and the resulting computation frequency too low.

- potentials: this method fits particularily well in this framework. The idea behind it is to compute a potential based on the difference between the current configuration and the limit configurations. The lower this difference, the higher the potential and vice versa. That means only one scalar is computed per timestep, and the sole operation is a difference. Although, this difference should be wrapped by an other function to provide a smoother approach that won't be brutal as the system draws near it.

Formally, the potential approach is written as follow. The embedding of the configuration space must take the potentials into account now. The function ψ characterizing it is now modified to:

$$\psi(\mathbf{x}, \mathbf{q}) = p(\mathbf{x}, \mathbf{q}) + \zeta(\mathbf{q})$$

where $\zeta : \mathbb{R}^d \to \mathbb{R}$ will be called the potential function.

The barrier shown in figure 5.6 was generated with a sigmoid-like function of the form:

$$\zeta(\mathbf{q}) = \zeta_+(\mathbf{q}) + \zeta_-(\mathbf{q}) = \frac{1}{1 + e^{\alpha(\mathbf{q}^+ - \mathbf{q})}} + \frac{1}{1 + e^{\beta(\mathbf{q} - \mathbf{q}^-)}} \tag{5.7}$$

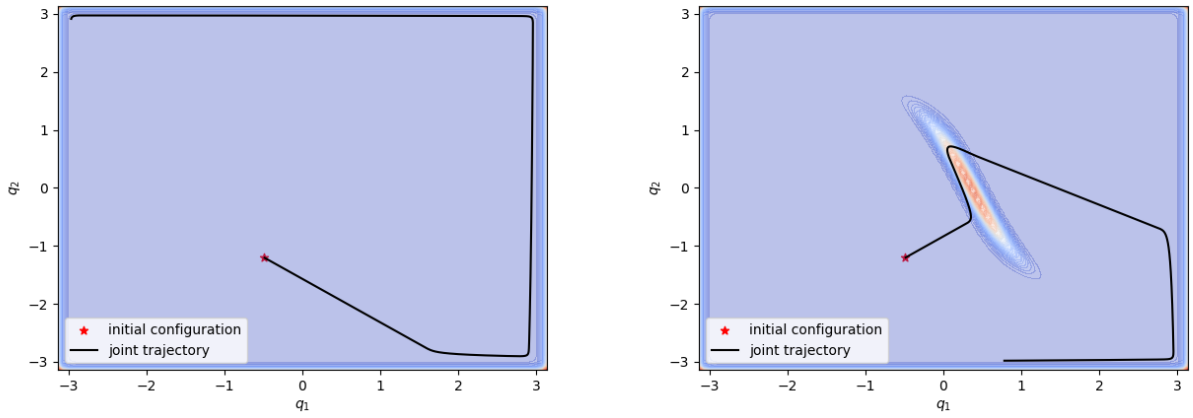
with $\alpha, \beta \in \mathbb{R}$.

Because we introduced a change in ψ that have a dependency on $\mathbf{q}$, the metric tensor and the Christoffel symbols need to change accordingly, through these new expressions.

$$\frac{\partial \psi}{\partial \mathbf{q}} = \frac{\partial p}{\partial \boldsymbol{\mu}} \cdot \frac{\partial \boldsymbol{\mu}}{\partial \mathbf{q}} + \frac{\partial p}{\partial \boldsymbol{\Sigma}} \cdot \frac{\partial \boldsymbol{\Sigma}}{\partial \mathbf{q}} + \frac{\partial \zeta}{\partial \mathbf{q}} \tag{5.8}$$

and

$$\frac{\partial^2 \psi}{\partial \mathbf{q}^2} = \frac{\partial^2 p}{\partial \mathbf{q} \partial \boldsymbol{\mu}} \frac{\partial \boldsymbol{\mu}}{\partial \mathbf{q}} + \frac{\partial p}{\partial \boldsymbol{\mu}} \frac{\partial^2 \boldsymbol{\mu}}{\partial \mathbf{q}^2} + \frac{\partial^2 p}{\partial \mathbf{q} \partial \boldsymbol{\Sigma}} \frac{\partial \boldsymbol{\Sigma}}{\partial \mathbf{q}} + \frac{\partial p}{\partial \boldsymbol{\Sigma}} \frac{\partial^2 \boldsymbol{\Sigma}}{\partial \mathbf{q}^2} + \frac{\partial^2 \zeta}{\partial \mathbf{q}^2} \tag{5.9}$$



(a) Without switch, see equation 5.4

(b) With a basic switch, see equation 5.6

Figure 5.6: Joint trajectory of a 2d manipulator

In the case when ζ is similar to equation 5.7, the first and second order derivative are simple:

$$\frac{\partial \zeta}{\partial \mathbf{q}} = \frac{\alpha e^{\alpha(\mathbf{q}^+ - \mathbf{q})}}{\left(1 + e^{\alpha(\mathbf{q}^+ - \mathbf{q})}\right)^2} - \frac{\beta e^{\beta(\mathbf{q} - \mathbf{q}^-)}}{\left(1 + e^{\beta(\mathbf{q} - \mathbf{q}^-)}\right)^2} \tag{5.10}$$

$$\begin{aligned} \frac{\partial^2 \zeta}{\partial \mathbf{q}^2} = & \frac{-\alpha^2 e^{\alpha(\mathbf{q}^+ - \mathbf{q})}\left(1 + e^{\alpha(\mathbf{q}^+ - \mathbf{q})}\right)^2 + 2\alpha^2 e^{2\alpha(\mathbf{q}^+ - \mathbf{q})}\left(1 + e^{\alpha(\mathbf{q}^+ - \mathbf{q})}\right)^4}{\left(1 + e^{\alpha(\mathbf{q}^+ - \mathbf{q})}\right)} \\ & - \frac{\beta^2 e^{\beta(\mathbf{q} - \mathbf{q}^-)}\left(1 + e^{\beta(\mathbf{q} - \mathbf{q}^+)}\right)^2 - 2\beta^2 e^{2\beta(\mathbf{q} - \mathbf{q}^-)}\left(1 + e^{\beta(\mathbf{q} - \mathbf{q}^-)}\right)}{\left(1 + e^{\beta(\mathbf{q} - \mathbf{q}^-)}\right)^4} \end{aligned} \tag{5.11}$$

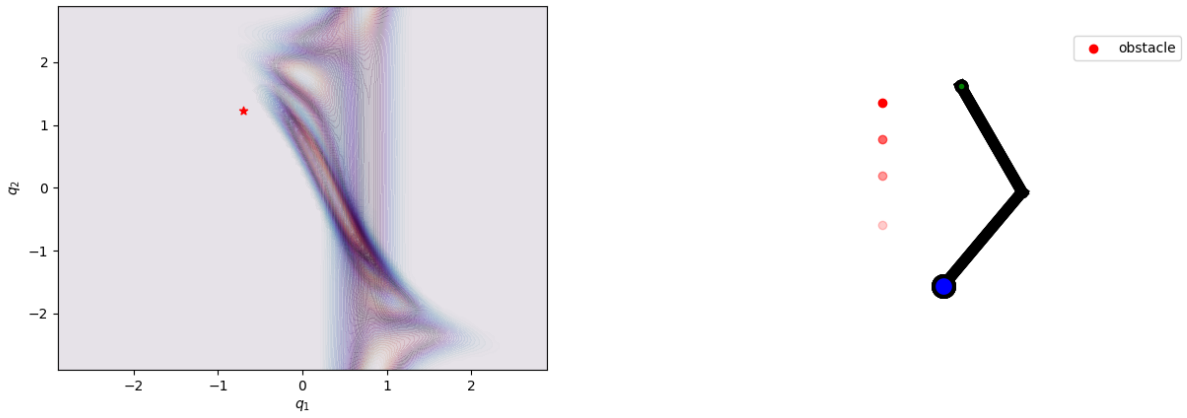
To illustrate our method, we use a 2-dimensional manipulator. We arbitrarily choose reasonable joint limits, such as $\mathbf{q}^- = \begin{bmatrix} -\pi, & -\pi \end{bmatrix}^T$ and $\mathbf{q}^+ = \begin{bmatrix} \pi, & \pi \end{bmatrix}^T$. To test this joint limitation strategy, we create a situation where a geodesic motion leads the system towards the limits, see figure 5.6a. Figure 5.6b shows that the joint limits can effectively be combined with an obstacle.

5.5 Dynamic obstacles

5.5.1 Moving obstacles

In a situation where an obstacle is moving through time, the equations governing our DS do not change. Indeed, the sole obstacle's property that have an influence on a single state is its position. As long as $\mathbf{x}$ has no dependency on $\mathbf{q}$, the expression of the DS remains the same.

The only difference with the static case is that the curvature is shifting in the joint space embedded representation. Figure 5.7 shows the evolution of the curvature when an obstacle is dropped in the manipulator's task space. At its lowest, the curvature form a barrier for q_1, preventing the first joint angle to increase past it.



(a) Evolution of the configuration space w.r.t time

(b) Moving obstacle in task space

Figure 5.7: Moving obstacle scenario

5.5.2 Multiple obstacles

With the same reasoning, adding more obstacles does not influence the expression of the DS, as the only impacted variable is the collision probability ϕ. In general, if there are n obstacles in the environment, the collision probability is re-written as:

$$\psi(\cdot, \mathbf{q}) = \sum_{i=1}^{n} \sum_{j=1}^{d} \psi_j(\mathbf{x}_i, \mathbf{q})$$

Of course, depending on the position of the obstacles, the curvatures can merge and profiles as in figure 5.9.

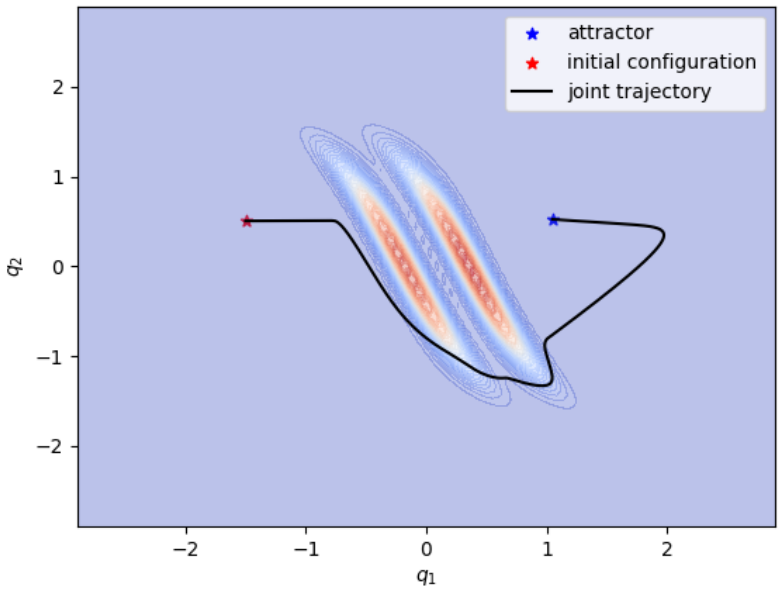
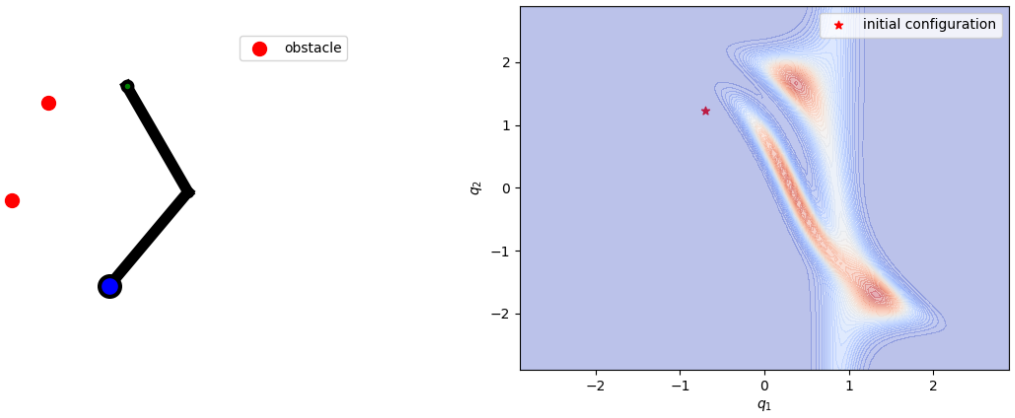


Figure 5.8: Embedding of a task space with multiple obstacles



(a) Task space (b) Configuration space

Figure 5.9: Merging of curvatures in configuration space

Chapter 6

Results

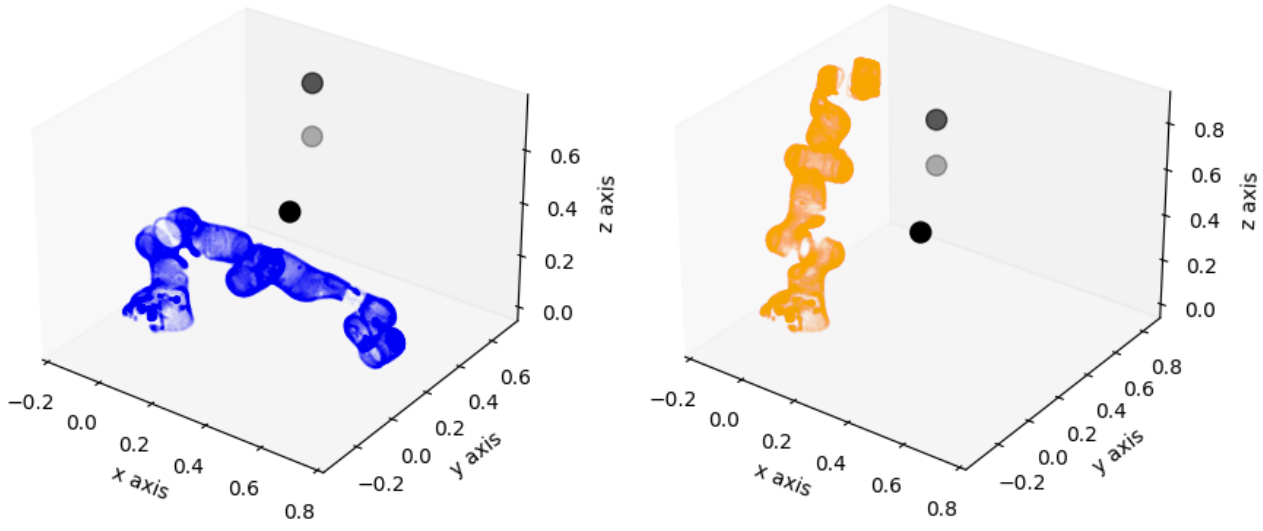
6.1 7d manipulator

To test the dynamical system, it was applied to the Franka Research 3, a 7-dofs manipulator, on several obstacle avoidance scenarios.

Given the high dimensionality of the manipulator's joint space, visualizing the latter's embedded representation is no longer an option, or at least does not reveal as many information as the 2-dimensional case (e.g figure 3.7). More information on multi-dimensional visualization in appendix-A.

To evaluate the theoritical success of our approach, we study the joint trajectory and verify that the system reaches the attractor while deviating from the collision-inducing configurations.

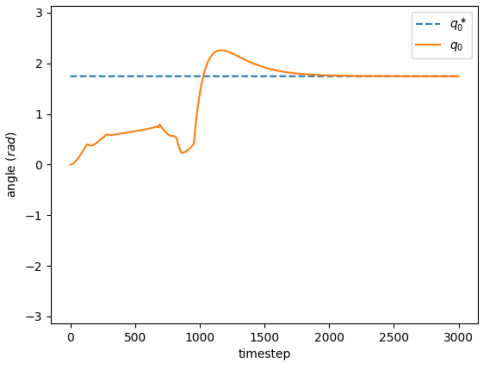
Applying equation 5.6 to the situation displayed in figure 6.1 yields the joint trajectories presented in figure 6.2.



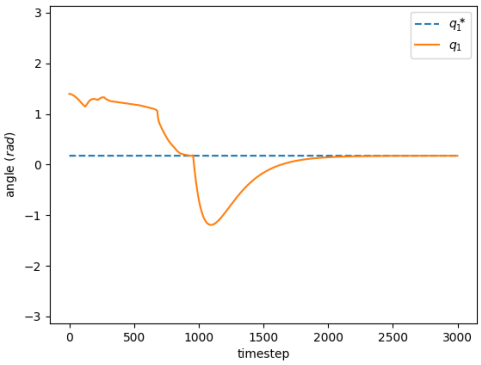
(a) Initial configuration

(b) Final configuration

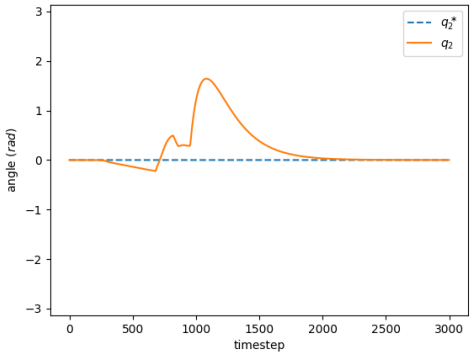
Figure 6.1: Example of an obstacle avoidance scenario with a 7-dofs manipulator



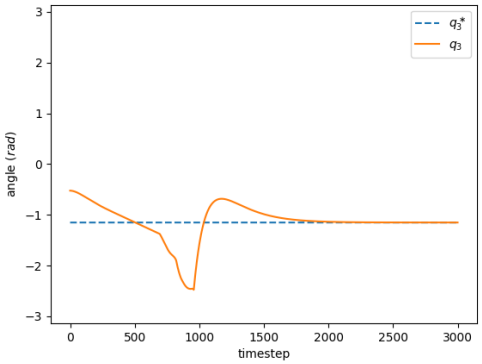
q_1



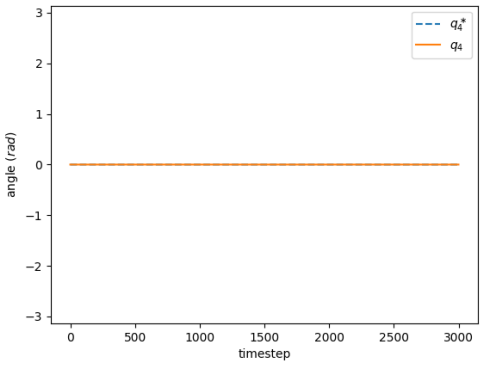
q_2



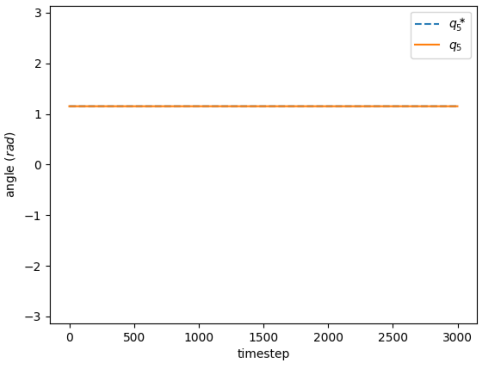
q_3



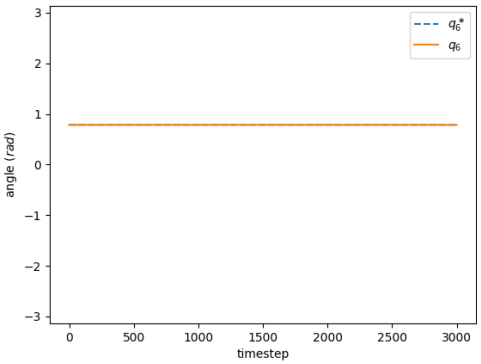
q_4



q_5

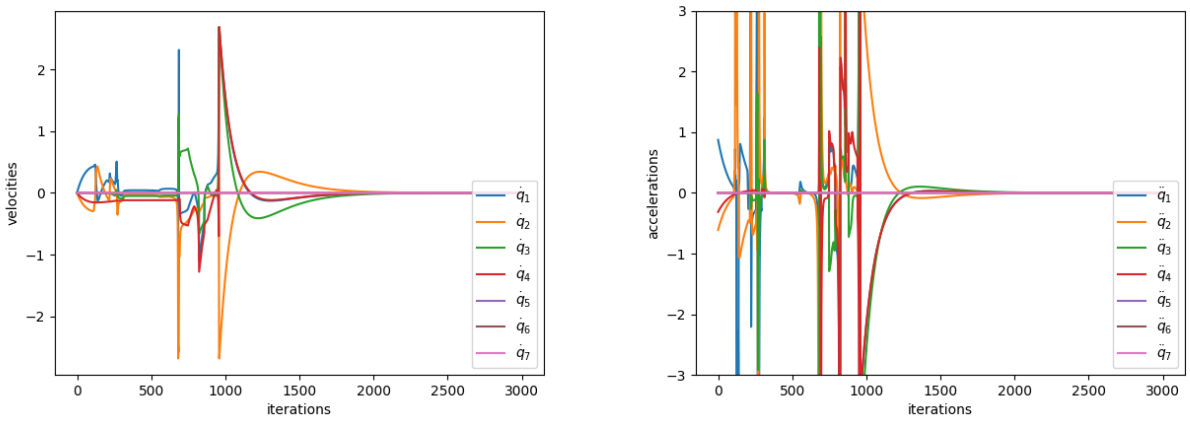


q_6



q_7

Figure 6.2: Joint angle profiles with equation 5.6



(a) Joint velocities

(b) Joint accelerations

Figure 6.3: Joint velocities and accelerations resulting from equation 5.6

While figure 6.2 indicates that the system reaches the attractor, the saccades on the joint angles and the burst of accelerations and velocities on figure 6.3 suggest that the switching between the harmonic and geodesic terms is too abrupt. Indeed, taking a closer look at the evolution of the switching weight, the profile is switching at a high frequency, introducing an unwanted behaviour in the dynamical system. Although we could try to tune the probability threshold κ to get rid of this noisy profile, we could achieve a more reliable result by improving the switching function.

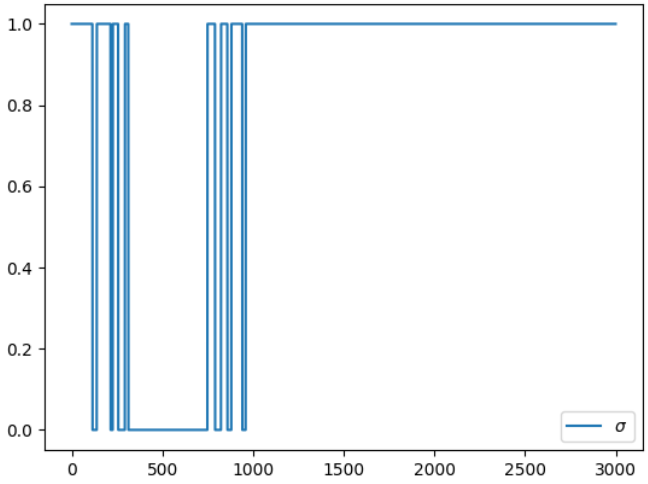


Figure 6.4: Dynamic weights for a binary switch (equation 5.5) for $\kappa = 0.5$

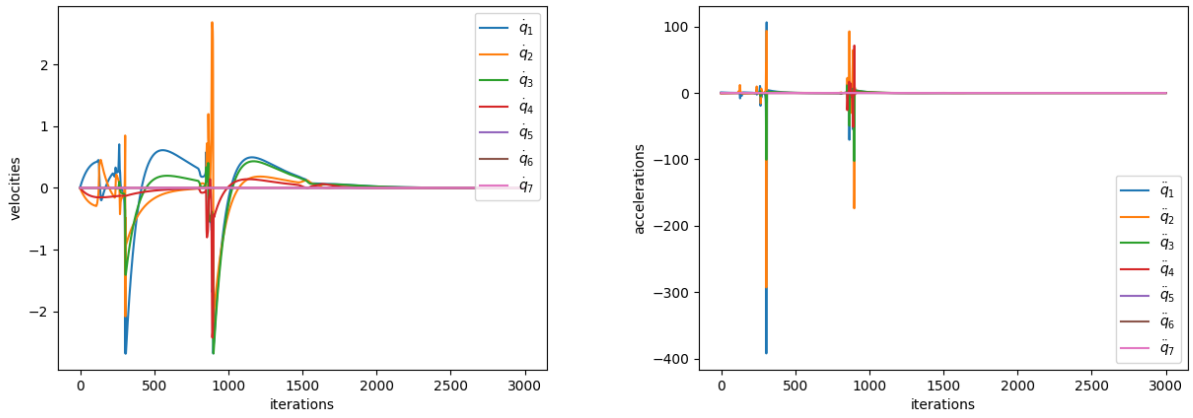
6.1.1 Sigmoid-based switching

A regular smoothing function is a sigmoid. Although it adds another tuning parameter, it already provides an intuition of how the switching method should be improved.

We replace equation 5.5 by

$$\sigma = \frac{-1}{1 + e^{-\alpha(\psi - \kappa)}} + 1,\ \alpha, \kappa \in \mathbb{R}$$

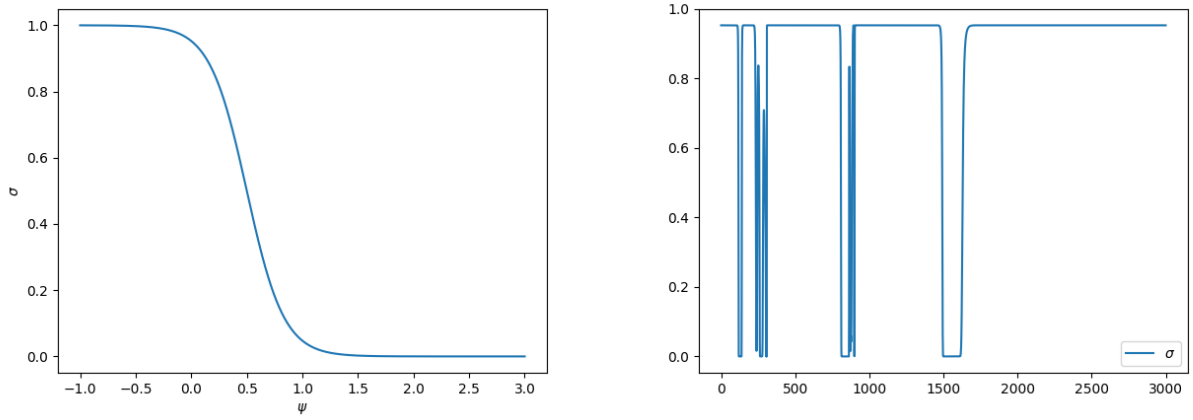
and choose the slope α to smooth the transition around the probability threshold κ.



(a) Joint velocities

(b) Joint accelerations

Figure 6.5: Joint velocities and accelerations resulting from equation 5.6 with a sigmoid-like switch



(a) Joint velocities

(b) Joint accelerations

Figure 6.6: Dynamic weights for a sigmoid-like switch

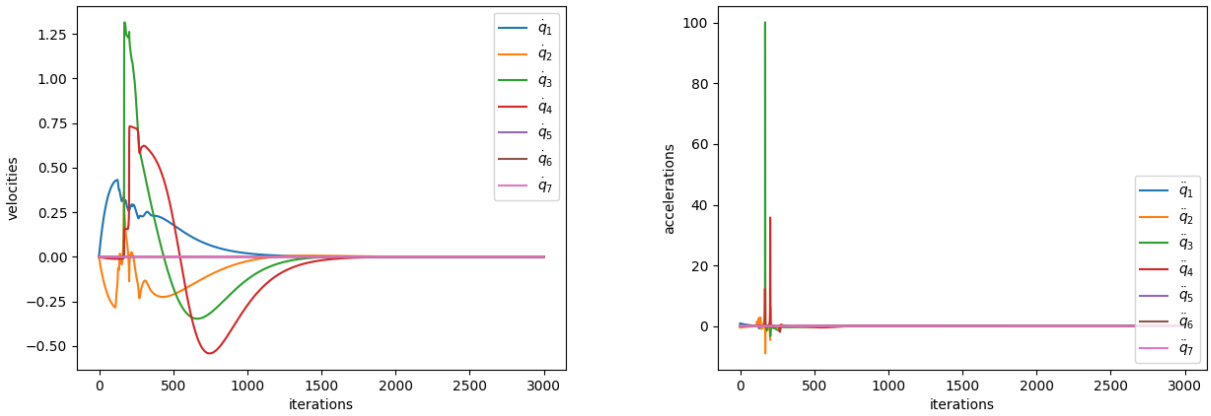
From figure 6.6, we notice that the slopes are indeed smoother, yet less localized. Even if the final joint angles curves are less hectic, the velocity spikes and the high-frequency switching is still present. For comparison with figure 6.2, the plots of the joint angles tracking is available in the appendix, see figure B.1.

6.1.2 A distance-based switching

Overall, the fact that the transition weight σ is the same for every joint is not really adapted. Indeed, there is not necessarily a need for a joint that barely affects the collision probability to enter in geodesic mode, for example. Hence, the natural solution is to compute a different transition weight for each link, depending on their impact on the collision probability. With this, the joints are decoupled and act according to their 'local' situation, without considering the situation on the manipulator scale.

To improve further the switching between harmonic and geodesic, we can also address the fact that the threshold value depends a lot on the scenario. Also, the value of the collision probability, which we recall is unnormalized, is hard to interpret, and we would like to avoid fine-tuning it each time a new scenario is entered.

Forturnately, we can easily provide a physical meaning to our threshold, by using the between the means of the Gaussian components to the obstacles. As explained during the choice



(a) Joint velocities

(b) Joint accelerations

Figure 6.7: Joint velocities and accelerations resulting from equation 6.2

of the embedding function, this solution is not perfect because it does not account for the links' orientations, it still addresses the problem. For each link, we can compute the minimum Euclidean distance between the link's GMM's means and the obstacles, and use a distance threshold to decide whether to switch between harmonic and geodesic mode. Thanks to this, the decision threshold would be in meters, and thus much more interpretable.

Formally, we are constructing a switching weight $\boldsymbol{\sigma} \in \mathbb{R}^7$ respecting the following:

$$\boldsymbol{\sigma} = \frac{1}{1 + e^{-\alpha(\mathbf{d}-\boldsymbol{\kappa})}} \tag{6.1}$$

where $\alpha \in \mathbb{R}$ is the growth rate of the sigmoid, $\boldsymbol{\kappa} \in \mathbb{R}^7$ the vector containing the distance theshold (e.g 0.2 meters) and $\mathbf{d} \in \mathbb{R}^7$ the distance vector whose coefficients are:

$$d_i = \min \|\boldsymbol{\mu}_k - \mathbf{x}\| \, \forall k \mid 1 \leq k \leq K \in \mathbb{N}$$

with K being the number of Gaussian components representing the i-th link.

Finally, we can re-write equation 5.6:

$$\ddot{\mathbf{q}} = -\boldsymbol{\sigma}(\mathbf{G}(\mathbf{q})^{-1}\mathbf{K}(\mathbf{q} + \mathbf{q}^*) - \mathbf{D}\dot{\mathbf{q}}) - (1 - \boldsymbol{\sigma})\boldsymbol{\Xi}\dot{\mathbf{q}} \tag{6.2}$$

The resulting velocities, observed in figure 6.7 are smoother and the peaks that were observed with the previous switching methods are no longer present.

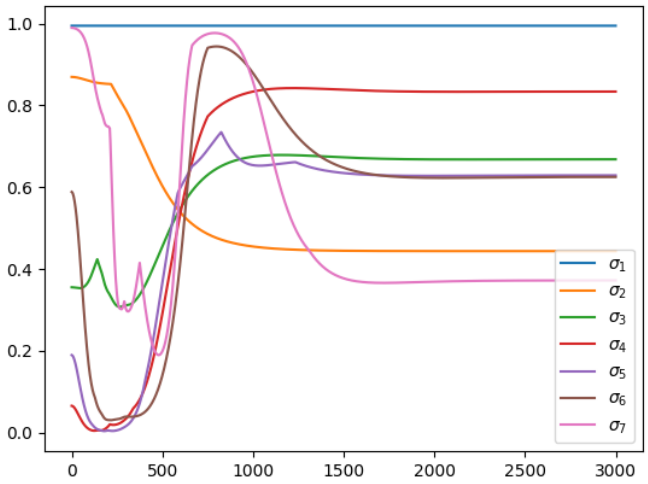


Figure 6.8: Dynamic weights for an Eulidean distance based switching

6.2 Simulation

Before testing on a real manipulator, the algorithms were implemented on Beautiful Bullet[1], a C++ wrapper of the Bullet engine. The setup to simulate a real scenario is composed of a Python program, that computes the joint accelerations based on the joint positions and velocities, that the physics engine gives as feedback.

To evaluate our method, we create different obstacle avoidance scenarios and verify that the following conditions are fullfilled:

- the manipulator reaches the desired configuration
- the joint angles deviate from the 'obvious' path that would result in a collision
- the manipulator does not collide with any obstacle. This can be verified by checking if the obstacles moved from their original position
- the joint accelerations generated by the QP solver match the accelerations computed by the dynamical system

6.2.1 Quadratic Programming

In the previous chapters, we showed several trajectories made of a succession of joint configurations $\mathbf{q}$. These positions are obtained after integrating twice the accelerations yielded by the dynamical system, following the Euler method, i.e:

$$\mathbf{q}_{t+1} = \mathbf{q}_t + \dot{\mathbf{q}}_t dt + \frac{1}{2}\ddot{\mathbf{q}} dt^2$$

In practice, simply setting the state of the robot to the result of the double integration of the acceleration is naive, since it does not account for the model of the robot. The model of the robot provides intrinsic information about the manipulator such as its inertia, the friction of the joints and the Coriolis forces associated to their movements. These are necessary to determine if

[1] `https://github.com/nash169/beautiful-bullet`

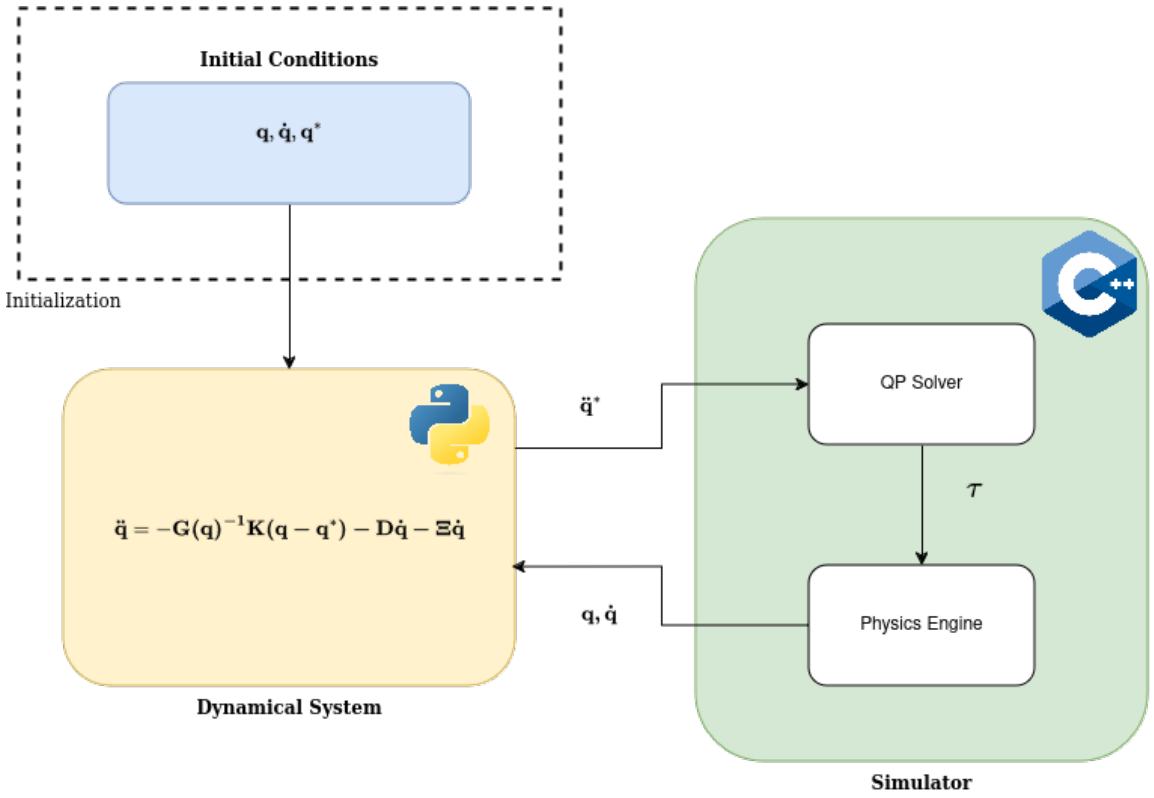


Figure 6.9: Simulation architecture

a force applied on it is enough to overcome the actuator's friction. In fact, the motion required to reach a specific configuration may not be achievable by the manipulator, because the motion of an articulated system follows some equations, and is subjects to some constraints, induced by its articulations. These constraints are notably on the level of the joint angles, velocities and accelarations, as well as the torques that the manipulator can support. For example, a joint cannot accelerate more than a certain threshold due to hardware limitations.

Ideally, the commands that one sends to a manipulator to control it are torques. Indeed, we want compliant behavior for safe human-robot interaction Thus, the problem that we are trying to solve can be resumed as: 'How much torque should be sent to the joints to make the manipulator reach the desired acceleration, knowing it should respect the physical constraints of the manipulator?'. This corresponds typically to an optimization problem, which can be formulated as:

$$\begin{aligned} \min_{\ddot{\mathbf{q}},\boldsymbol{\tau},\boldsymbol{\zeta}} \quad & \frac{1}{2}(\ddot{\mathbf{q}} - \ddot{\mathbf{q}}^*)^T\mathbf{Q}(\ddot{\mathbf{q}} - \ddot{\mathbf{q}}^*) + \boldsymbol{\tau}^T\mathbf{R}\boldsymbol{\tau} + \boldsymbol{\zeta}^T\mathbf{W}\boldsymbol{\zeta} \\ \text{s.t.} \quad & \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} = \dot{\mathbf{q}}^T\mathbf{C}(\mathbf{q})\dot{\mathbf{q}} + \boldsymbol{\tau} + \boldsymbol{\zeta} \\ & (\mathbf{q}^- \leq \mathbf{q} \leq \mathbf{q}^+) \\ & \dot{\mathbf{q}}^- \leq \dot{\mathbf{q}} \leq \dot{\mathbf{q}}^+ \\ & \ddot{\mathbf{q}}^-, \boldsymbol{\tau}^- \leq \ddot{\mathbf{q}}, \boldsymbol{\tau} \leq \ddot{\mathbf{q}}^+, \boldsymbol{\tau}^+ \end{aligned} \tag{6.3}$$

If, at each iteration of the dynamical system, the current manipulator's joint acceleration $\ddot{\mathbf{q}}$ can match with the desired joint acceleration $\ddot{\mathbf{q}}^*$, the system will eventually converge towards the targeted joint configuration $\mathbf{q}^*$.

The first constraint corresponds to the equation of motion of an articulated robot system where the first link is assumed to be fixed. $\mathbf{M}$ is the inertia matrix and $\mathbf{C}$ the Coriolis tensor.

The remaining constraints are the physical limits of the joints, regarding the maximal values supported in terms of angle, velocity, acceleration and torques.

And so, at each iteration of the dynamical system, a joint accelaration $\ddot{\mathbf{q}}$ is computed and fed to the solver described in equation 6.3. This solver finds the smaller torques $\boldsymbol{\tau}$ that minimize the difference between $\ddot{\mathbf{q}}^*$ and $\ddot{\mathbf{q}}$. These torques are then send to the manipulator's actuators.

$\mathbf{Q}$, $\mathbf{R}$ and $\mathbf{W}$ are tunable matrices. They are the gains that affect the rate at which the references of the accelerations, torque and slack variable are tracked.

6.2.2 No obstacle in the workspace of the manipulator

The first scenario has no reachable obstacle. It serves as a benchmark for the future scenarios, and we can confirm the behaviour of the manipulator when its configuration space is not subjected to any curvature.

Here are the initial conditions:

- $\mathbf{q}^* = \begin{bmatrix} 90, & 45, & 0, & -66, & 0, & 66, & 45 \end{bmatrix}^T$
- $\mathbf{q} = \begin{bmatrix} 0, & 45, & 0, & -66, & 0, & 66, & 45 \end{bmatrix}^T$
- $\dot{\mathbf{q}} = \begin{bmatrix} 0, & 0, & 0, & 0, & 0, & 0, & 0 \end{bmatrix}^T$

Those configurations were chosen arbitrarily, but in accordance with the joint limits of the Franka Research 3[1].

Regarding the tuning of the QP solver hyperparameters, we used:

- $\mathbf{Q} = 30 \cdot \mathbf{I}_{7\times7}$
- $\mathbf{R} = 1 \cdot \mathbf{I}_{7\times7}$
- $\mathbf{W} = 0.001 \cdot \mathbf{I}_{6\times6}$

On the dynamical system side:

- $\mathbf{K} = 0.5 \cdot \mathbf{I}_{7\times7}$
- $\mathbf{D} = 1.5 \cdot \mathbf{I}_{7\times7}$
- $\boldsymbol{\kappa} = \begin{bmatrix} 0.1, & 0.1, & 0.2, & 0.2, & 0.2, & 0.2, & 0.1 \end{bmatrix}^T$

with $\boldsymbol{\kappa}$ the distance threshold as defined in equation 6.1. Only the first joint angle is changing from the initial configuration to the attractor, so the other joints will not move, with no obstacles in the environment. The tracking profiles shown in figure 6.11, will serve as a reference to define how much the joint trajectory varies when performing obstsacle avoidance.

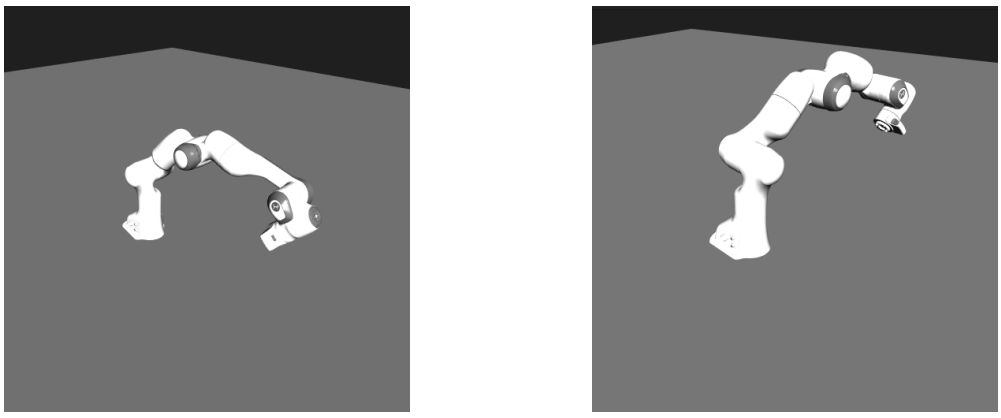
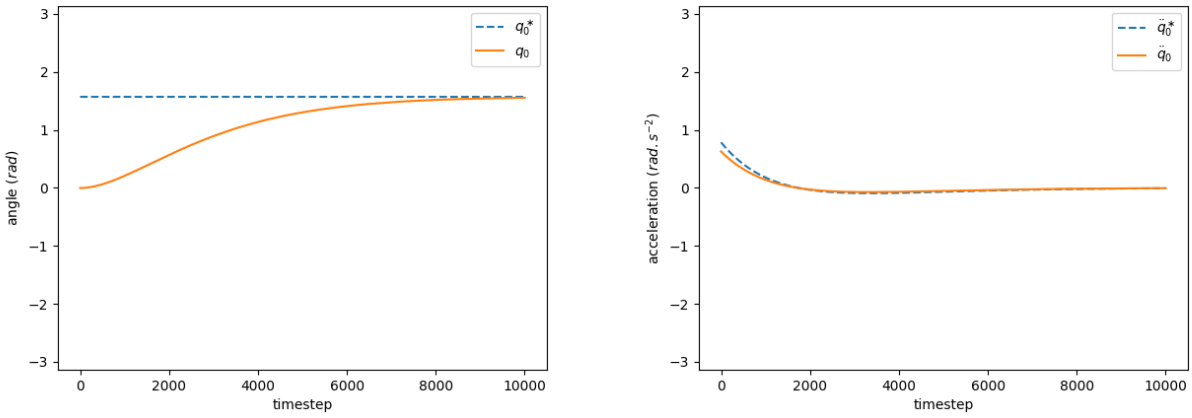


Figure 6.10: Initial and final configuration of the manipulator

[1]Franka Interface specifications.



q_1 $\ddot{q_1}$

Figure 6.11: Tracking of q_1 and $\ddot{q}_1^*$

6.2.3 Adding a cluster of obstacles to the workspace

Using the same initial conditions as before, we complexify the scenario by adding obstacles in locations that would hinder the motion of the manipulator. Also, we keep the same parameters for both the dynamical system and the QP solver.

The obstacles are placed in $\mathbf{x_1} = \begin{bmatrix}0.4, & 0.53, & 0.52\end{bmatrix}$, $\mathbf{x_2} = \begin{bmatrix}0.4, & 0.51, & 0.52\end{bmatrix}$, $\mathbf{x_3} = \begin{bmatrix}0.3, & 0.45, & 0.52\end{bmatrix}$.

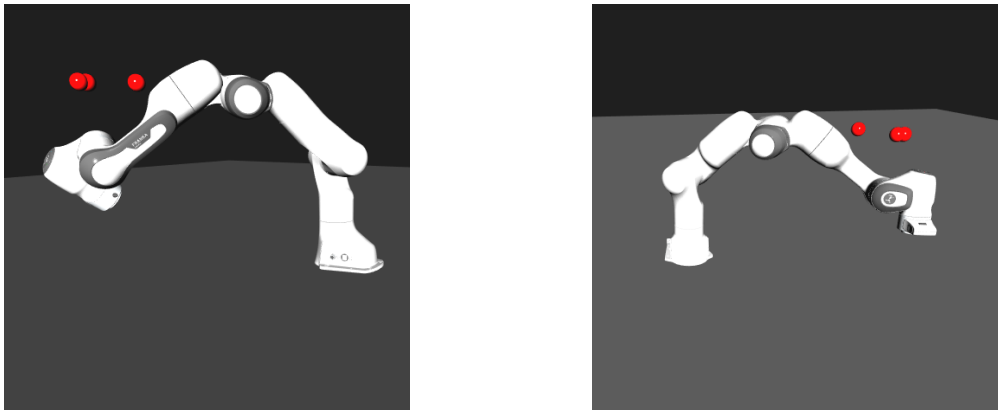
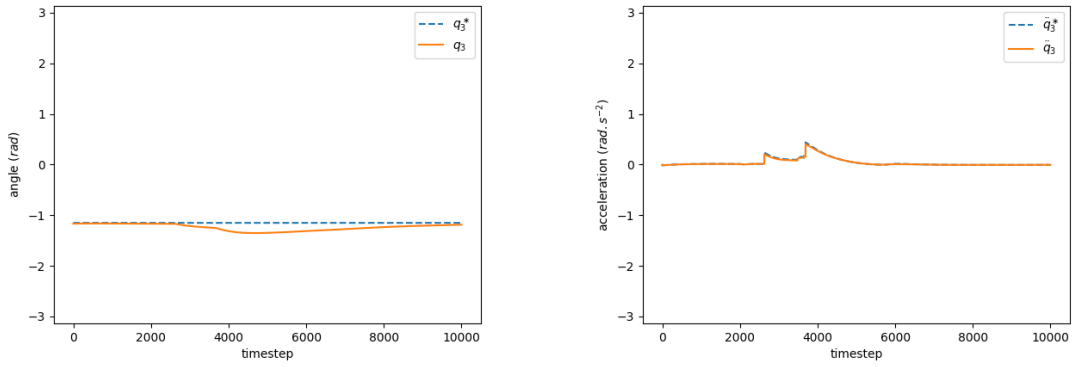


Figure 6.12: Avoidance highlights

Figure 6.12 confirm that the avoidance is conducted, and we can verify on figure 6.13 that the 4^{th} joint tilts the tip of the manipulator to pass under the cluster of obstacle, deviating the joint trajectory from the normally 'straight' path.



q_4 $\ddot{q}_4$

Figure 6.13: 4^{th} Joint angle position and acceleration

6.2.4 Diagonal trajectory

Here is another scenario, where the obstacles are in the way of an 'upper diagonal' motion. The initial conditions are now:

- $\mathbf{q}^* = \begin{bmatrix}100, & 10, & 0, & -66, & 0, & 66, & 45\end{bmatrix}^T$
- $\mathbf{q} = \begin{bmatrix}0, & 80, & 0, & -30, & 0, & 66, & 45\end{bmatrix}^T$

and the obstacles are: $\mathbf{x_1} = \begin{bmatrix}0.3, & 0.23, & 0.42\end{bmatrix}$, $\mathbf{x_2} = \begin{bmatrix}0.2, & 0.51, & 0.75\end{bmatrix}$, $\mathbf{x_3} = \begin{bmatrix}0.2, & 0.51, & 0.55\end{bmatrix}$.
Similarily to the previous scenarios, figure 6.14 confirms the avoidance of the obstacles, despite the more complex situation.

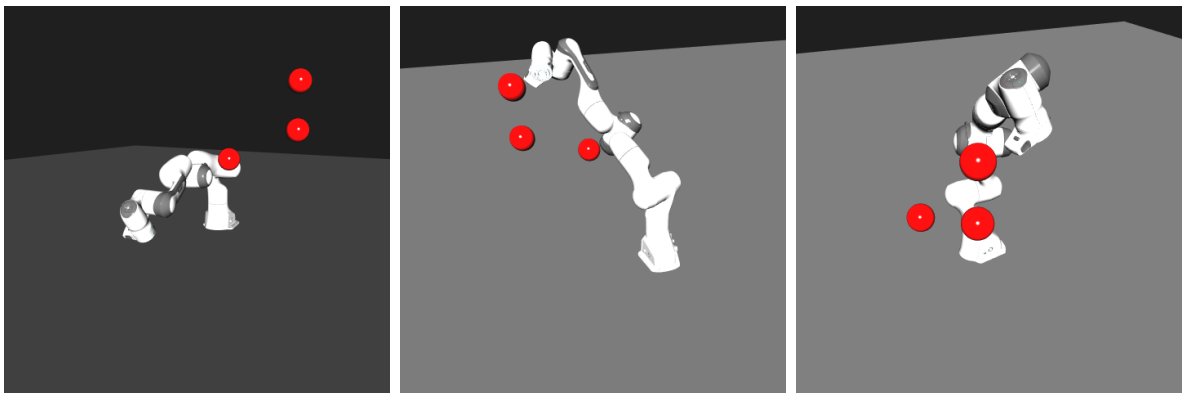


Figure 6.14: Avoidance highlights

6.3 Performance: Pytorch's Autograd vs Analytical derivatives

The algorithms presented in this work were implemented on Python. Here, we consider two different manners to compute the terms expressed in chapter–4:

- with the *autograd* features of the *pytorch* library. The implementation is rather simple and more high-level, but provide less control on the optimization and performance of the computation,
- with the analytical expressions developped in chapter–4. Although the implementation is more complex due to the many terms to be computed, it should be faster than the *pytorch* implementation, for less demanding operations are required.

To compare the performance of these two approaches, we compare the computation frequency of the dynamical system formulation:

$$\ddot{\mathbf{q}} = -\mathbf{G}(\mathbf{q})^{-1}\mathbf{K}(\mathbf{q} - \mathbf{q}^*) - \mathbf{D}\dot{\mathbf{q}} - \boldsymbol{\Xi}\dot{\mathbf{q}}$$

For each scenario, we limit the number of gaussian componenents representing each link to 1. The average frequency is computed for 1000 timesteps, and cross-validated 10 times.

number of joints	**torch.autograd**	**analytical derivative**
2 joints	133 Hz	3076 Hz
3 joints	57 Hz	1725 Hz
7 joints	4.5 Hz	214 Hz

Table 6.1: Computation frequency with respect to the number of joints

Overall, the implementation using analytical expressions to compute the derivative is noticeably faster than its *autograd* counterpart. The downside is that this method requires one to manually compute the derivative of the embedding function before implemeting it, but the substantial speed gain this approach brings motivates implementing analytically the derivative of the embedding.

The CPU used to obtain these frequencies is an Intel i7–1360P with 16 cores.

Chapter 7

Conclusion

Our study successfully demonstrated the application of differential geometry and dynamical systems in achieving full-body obstacle avoidance for robotic manipulators. This approach involves the introduction of non-linearities in place of the obstacles in the joint space of the manipulator, and making use of geodesic trajectories to avoid them. On the other hand, the system maintains stability by using the model of a damped harmonic oscillator. Because our method relies purely on a dynamical system, it benefits from a small computational cost and thus a small time cycle, enhancing the reactiveness of the manipulator. Additionally, this method offers a degree of modularity, where the control of the joint space's curvature can be used to place constraints on the trajectory of the manipulator, typically with respect to the joint limits or the minimal distance to be maintained between the manipulator and the obstacles.

Limitations and Future Work

The success of this approach relies heavily on the representation of the manipulator. We chose to use a Gaussian-based representation, implying a dilemna between an accurate representation reducing the conservativity of the motion, and computational frequency. Even though the time cycle remains decently short when using a heavy representation on a 7d robot, it has a severe impact on the computational cost, limiting the benefits of this approach compared to a hybrid DS-MPC method for example. In addition, the functions one can choose from to approximate the surface of the manipulator are limited to C^2 functions at least, by necessity of the Christoffel symbols. Still, more suitable representations could exist and offer a faster derivation with at least the same accuracy.

Also, our approach is limited in terms of velocity control. Indeed, the geodesic equation does not provide control over the velocity of the trajectory. Hence, the trajectory is left to itself during avoidance manoeuvre, and this could be a severe drawback in situations where the manipulator is transporting senstivie parts. Deepening the understanding of geodesic behaviour could adress this issue, and may also improve the transition between the harmonic and geodesic motions on the borders between curved and flat regions.

Appendix A

Visualization of 7-dimensional problems

Due to the high dimensionality of this problem, it is complex to understand the behaviour of the system in most cases. Inherently, humans can visualize 3 spatial dimensions at the same time, and eventually a fourth one when color is added. In the case of a 7-dimensional problem however, we have to compromise. For one, the computation of a high-dimensional representation has a heavy cost that varies with $\mathcal{O}(n^7)$. The accuracy of the representation is therefore already limited by the computational cost. To find a representation that would show relevant information about the system's behaviour, we decomposed the joint configuration in pairs of joint angles and displayed the relative movements of one joint with respect to another. This way, the interpretation at least was closer to a 2-dimensional problem, as in figure 3.7.

The plots in figure A.1 corresponds to the embedding seen from the point of view of every possible pairs of the 7 joints, i.e $\frac{7*(7-1)}{2} = 21$ in total. In the general case of a d-dimensional manipulator, the embedding according to the pair (q_i, q_j) is

$$\mathbf{\Psi}_{ij} = \begin{bmatrix} q_i \\ q_j \\ \psi_{ij}(\mathbf{q}) \end{bmatrix}$$

$$\psi_{ij}(\mathbf{q}) = \psi_i(\mathbf{x}, \mathbf{q}) + \psi_j(\mathbf{x}, \mathbf{q}) + \sum_{\substack{k=0 \\ k \neq i,j}}^{d} \sum_{\mathbf{q} \in \mathcal{C}} \psi_k(\mathbf{x}, \mathbf{q})$$

Put differently, each point of each plot is the the collision probability of the i-th and j-th link, plus the collision probability of the remaining joints for every joint configuration maintaining a fixed q_i and q_j.

In the case of the Franka manipulator, it is a mapping from $\mathbb{R}^7$ to $\mathbb{R}^2$. Therefore, the information about 5 dimensions are lost. The amount of information we can get from this is thereby limited, but still provides some useful insights, such as the relative motion of one joint with respect to another, or the amount of travel a specific joint requires.

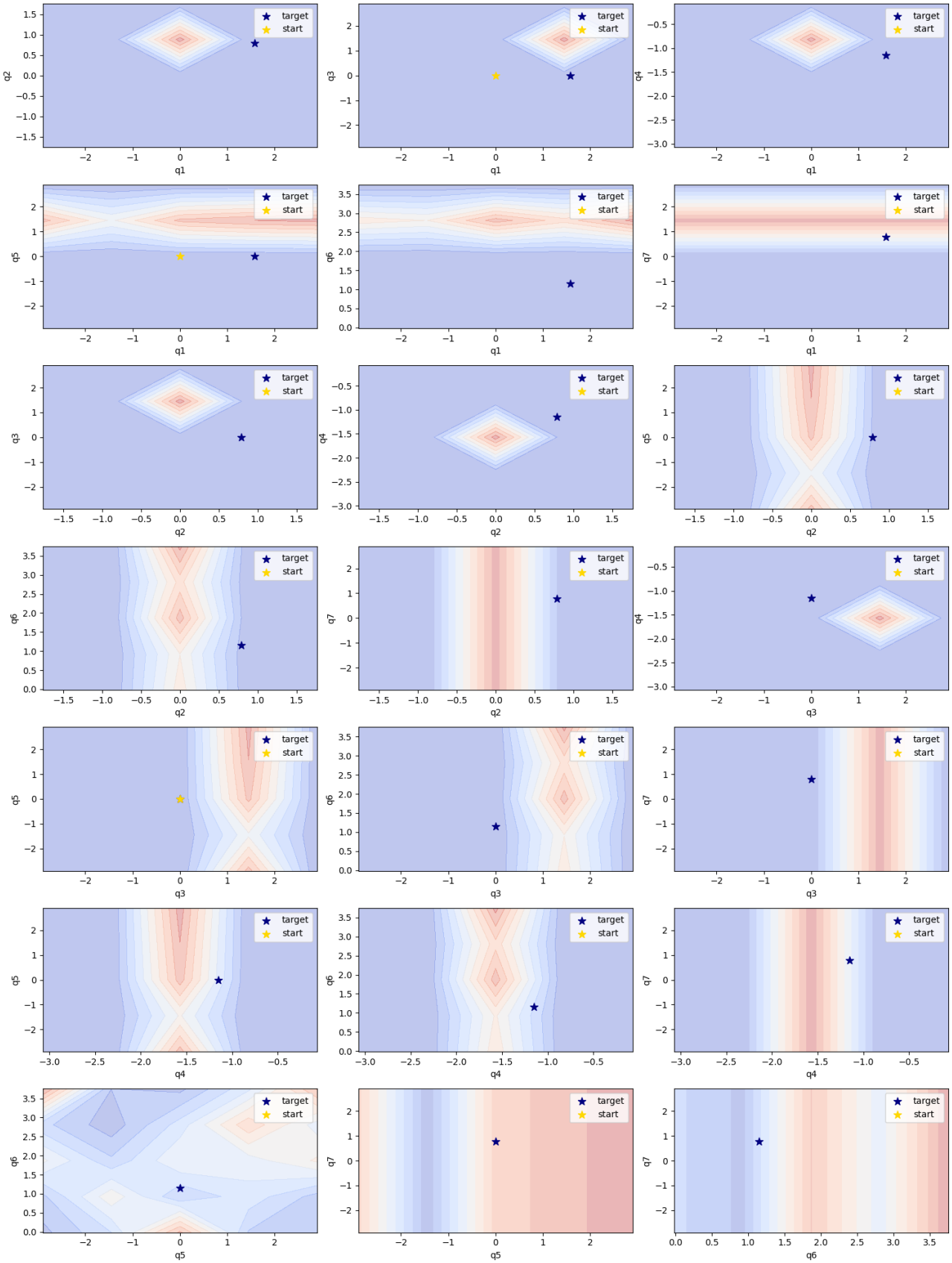
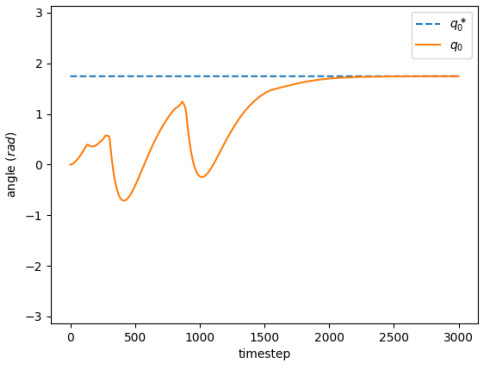


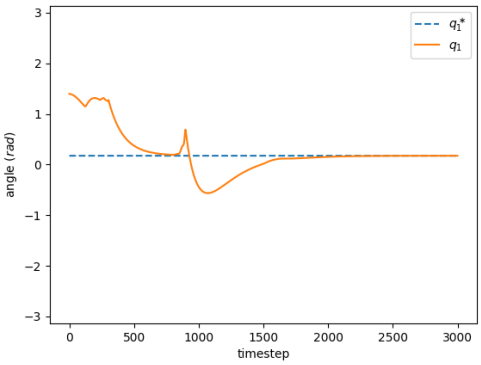
Figure A.1: 7d embedding projections

Appendix B

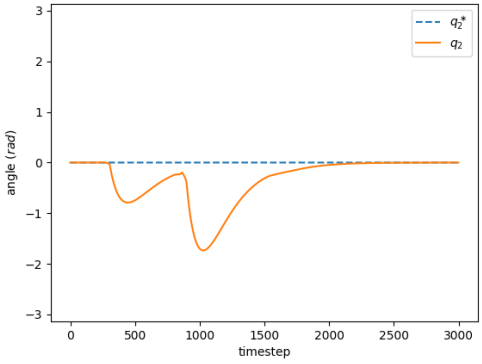
Joint tracking



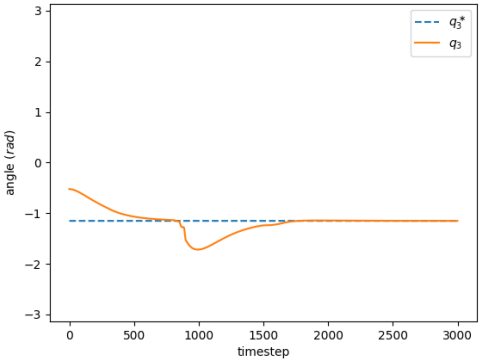
q_1



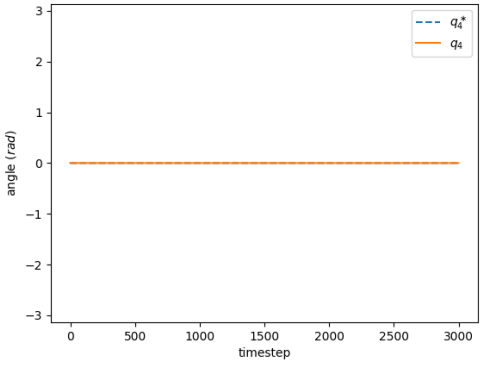
q_2



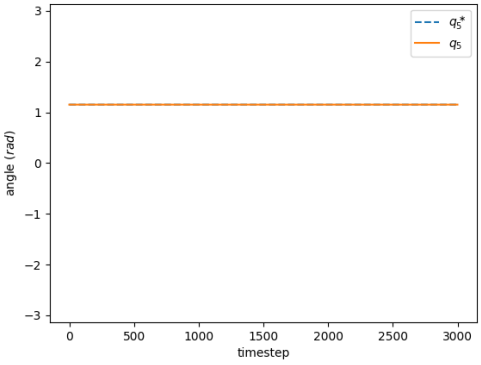
q_3



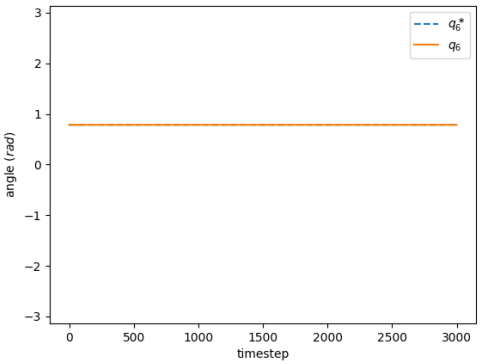
q_4



q_5

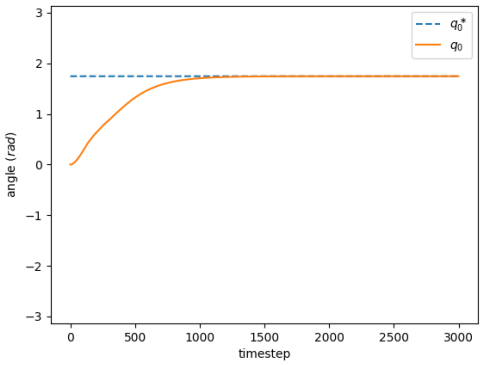


q_6

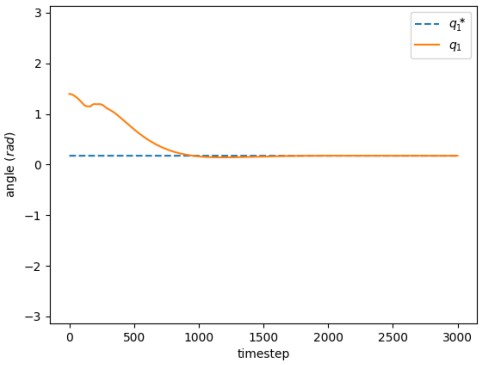


q_7

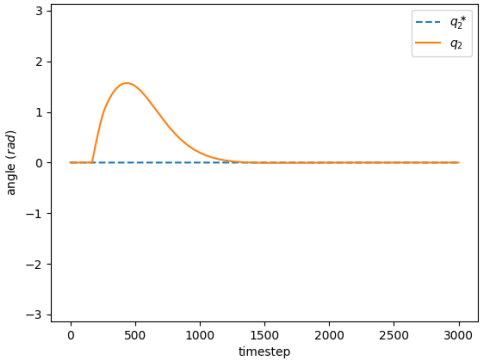
Figure B.1: Joint angle profiles with equation 5.6



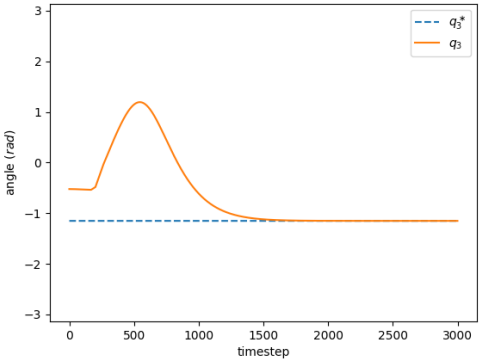
q_1



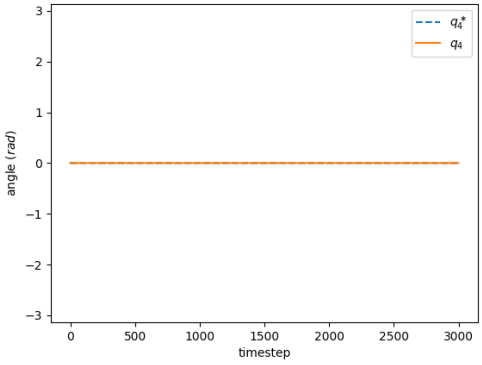
q_2



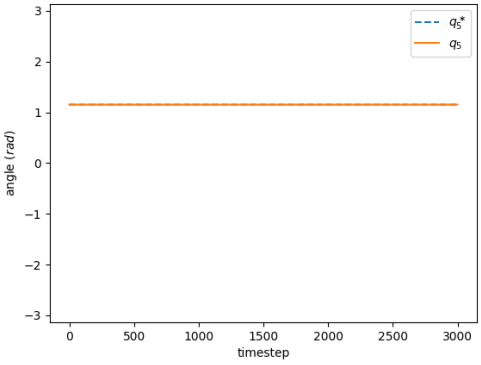
q_3



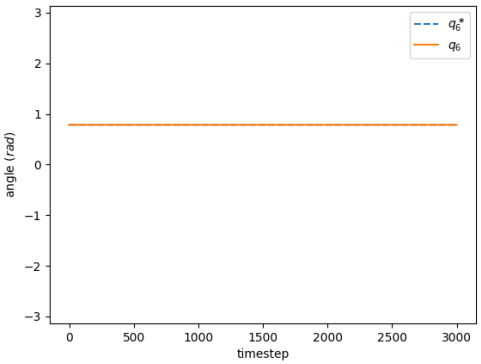
q_4



q_5



q_6



q_7

Figure B.2: Joint angle profiles with equation 6.2

Appendix C

Source codes

Source codes available at https://github.com/Bbosc/GDSOA.

Bibliography

[1] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2, pp. 500–505, 1985.

[2] K. Kronander, M. Khansari, and A. Billard, "Incremental motion learning with locally modulated dynamical systems," *Robotics and Autonomous Systems*, vol. 70, pp. 52–62, 2015.

[3] M. Koptev, N. Figueroa, and A. Billard, "Reactive collision-free motion generation in joint space via dynamical systems and sampling-based mpc," *The International Journal of Robotics Research*, p. 02783649241246557, 2024.

[4] C. Wong, E. Yang, X.-T. Yan, and D. Gu, "An overview of robotics and autonomous systems for harsh environments," in *2017 23rd International Conference on Automation and Computing (ICAC)*, pp. 1–6, IEEE, 2017.

[5] M. Koptev, N. Figueroa, and A. Billard, "Neural joint space implicit signed distance functions for reactive robot manipulator control," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 480–487, 2023.

[6] O. Stasse, A. Escande, N. Mansard, S. Miossec, P. Evrard, and A. Kheddar, "Real-time (self)-collision avoidance task on a hrp-2 humanoid robot," in *2008 IEEE International Conference on Robotics and Automation*, pp. 3200–3205, 2008.

[7] S. Koo, D. Lee, and D.-S. Kwon, "Gmm-based 3d object representation and robust tracking in unconstructed dynamic environments," in *2013 IEEE International Conference on Robotics and Automation*, pp. 1114–1121, 2013.

[8] B. Fichera and A. Billard, "Learning dynamical systems encoding non-linearity within space curvature," 2024.

[9] S. Lucey and T. Chen, "A gmm parts based face representation for improved verification through relevance adaptation," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 2, pp. II–II, 2004.

[10] K. B. Petersen and M. S. Pedersen, "The matrix cookbook," Oct. 2008. Version 20081110.

[11] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard, "The pinocchio c++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *IEEE International Symposium on System Integrations (SII)*, 2019.