CHALMERS
UNIVERSITY OF TECHNOLOGY

Trajectory Generation using Dynamic Movement Primitives under Hierarchical Constraints

Master's thesis in Systems, Control and Mechatronics

Valentin Dambly

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Trajectory Generation using Dynamic Movement Primitives under Hierarchical Constraints

VALENTIN DAMBLY

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Signals and Systems
Chalmers University of Technology
Gothenburg, Sweden 2019

Trajectory Generation using Dynamic Movement Primitives under Hierarchical Constraints
VALENTIN DAMBLY

Supervisor and Examiner: Yiannis KARAYIANNIDIS, Assistant Professor, Department of Signals and Systems, Chalmers University of Technology

Master's Thesis 2019:EENX30-50118
Department of Electrical Engineering
Division of Signals and Systems
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2019

Trajectory Generation using Dynamic Movement Primitives under Hierarchical Constraints

VALENTIN DAMBLY
Department of Electrical Engineering
Division of Signals and Systems
Chalmers University of Technology

Abstract

Human and robots are expected to work in collaboration or share a common workspace for a wide range of robotic applications. Such scenario requires the use of reactive robots and new ways of programming where the ability to be reactive and to learn from demonstrations take precedence over preprogrammed behaviours. We propose a framework that combines optimisation based control and reactive planning. In this thesis we consider a use case where the robot is operating in a environment with obstacles. We design a kinematic controller using the Hierarchical Quadratic Programming method that can accommodate an ensemble of constraints/tasks ordered in a stack. Dynamic Movement Primitives (DMP) are chosen as a real time trajectory generation method owing to their flexibility. Following the learning from demonstration approach, DMP can encode a trajectory that can be easily adapted in new conditions. Obstacle avoidance is an important aspect that is addressed in our approach. Several methods were explored such as artificial potential fields methods specialised in collision avoidance and obstacle circumvention. We also investigate how obstacle avoidance can be integrated either as a task of the stack or as a non-linear term in the trajectory generation method. We studied, implemented and tested each of quoted approaches in a UR10 robot in order to develop a control approach that can guarantee safety when humans and robots are working together.

Keywords: Kinematic control, hierarchical quadratic programming, dynamic movement primitives, artificial potential fields, obstacle avoidance, motion planning

Acknowledgements

This work wouldn't have seen the light of the day without the support and intervention of many people, sincere thanks to Yiannis Karayiannidis for making this thesis possible, for the advice provided and his trust. I would also like to thank Diogo Almeida, Albin Dahlin and Ramin Jaberzadeh Ansari for their help. A thankful thought also goes to Olivier Verlinden and Véronique Feldheim for their help in preparing the student exchange. I finally want to thank my family for the daily support and my fellow Erasmus mates for the priceless journey.

Valentin Dambly, Gothenburg, June 2019

Contents

List of Figures

List of Tables

1

Introduction

1.1 Context

Traditionally robots employed in industry operate in appropriately designed cages where they perform preprogrammed tasks. Modern collaborative robots are expected to be involved in manufacturing processes or domestic tasks and to work in the vicinity of humans. In order to interact with its environment, the robot perceives it through several sensors such as cameras, lidar, force and tactile sensors. To this end, the robot movements must be flexible and adjustable rather than preprogrammed.

One basic scenario of a collaborative robot interacting with its environment is the flexible movement generation of a six degrees of freedom robot acting under constraints. More precisely, the generation of a flexible trajectory is driven possible by Dynamic Movement Primitives and the motion is generated for the joints as velocity commands to address redundancy and additional constraints.

Robots have originally been conceived to achieve tasks considered as repetitive, tedious or boring for a human being. Nevertheless robots have been evolved and the time to change that paradigm has come. Their field of use has widened, leading to the progressive replacement of human operators by autonomous robots.

The development of collaborative robots, also called cobots, is the opportunity to keep human operators and to take advantage of the technology. Many benefits arise from this collaboration. Since those robots are made to work and interact with humans in the same work-space, the safety is a major concern during their development. Those safety requirements can also be achieved through constraints imposed by the control and can create a safer work environment. Furthermore, cobots are made to be flexible with a wide field of use. Consequently, the cobots embody the opportunity for small and medium sized companies to automate or simplify several tasks.

Two main directions can be distinguished in this research work. The first one being the design of a kinematic control aware of the environment and the limitations and the second being the trajectory generation.

1.2 Related Work

A well-established method to achieve the flexible trajectory generation is the Dynamic Movement Primitives (DMPs) [21]. This method enables the generation of a parametrized trajectory for the end-effector. The trajectory to be reproduced and/or modified is first learned. The learning methods presented in [23] are reinforcement learning and learning from demonstration. There are different ways to formulate DMP for trajectory generation. A theoretical overview of the DMP is presented by S. Schaal *et al.* in [23]. Various dynamical systems are presented such as the discrete and rythmic motion. Alternative versions are introduced by A. Ijspeert *et al.* where modifications are made in the dynamical systems in order to improve the robustness to changes. The trajectory generation is made with the integration in time of the dynamical system. Depending on the systems considered, the stability, the robustness or the fidelity in time can be enhanced [8][22]. The authors of [23] propose a way to do so for a multiple degrees of freedom system.

The improvements made to the dynamical system are not limited to computation properties and issues. Planning features are also offered such as the possibility to avoid obstacles [18]. This possibility has a special importance in this thesis since the final goal is to control a robot sharing an unstructured workspace with humans. All these possible modifications make planning flexible.

Since motion control of the robot is performed at the velocity level a closed-loop inverse differential kinematic control can be used. There is a vast literature on the kinematic control of robots. Some approaches are presented in [24] and [4] where well known methods are broached such as the Single Value Filtering, Jacobian Weighting (JW) and the Gradient Projection. The stability analysis of inverse-kinematic control is addressed in [1].
The notion of prioritised tasks is covered through predefined matrices such as weighting matrices which adapt their contents depending on the robot behaviour and the tasks. The prioritised inverse kinematic is also addressed in [9] and the way to deal with multitasking is explored with [20] and [27]. It is also interesting to mention that the neural network approach can also be used at the service of the kinematic control as in [14], the mapping of inverse kinematic is realised with several hidden layers where the weights are learned and continuously uploaded.

Optimisation based control can also be used to address inequality constraints. For multiple objective problems, the ensemble of tasks to whom the robot is subjected can be considered as a stack where each level is a task. Several ways to solve this optimisation problem configuration have been proposed. For a stack composed of tasks expressed as equalities, the quadratic programming optimisation method is applied to every level of the stack where the minimisation criterion is the deviation of the task.
However when some tasks are expressed as inequalities, the optimisation process must be modified. A prioritised task regulation framework was presented by Kanoun [10] based on a sequence of quadratic programs. This solution is an optimisation

method where the position in the stack gives the priority order. Solving the stack means fitting at best with the prioritised levels and minimising as far as possible the violation of lower priority tasks. This method is further referred as Sequential Quadratic Programming in the literature.

Furthermore, A. Escande *et al.* proposed an improved version of the SQP method called Hierarchical Quadratic Programming (HQP) [5]. The method is improved in the sense of computation speed and regarding to the way the priority is considered. The notion of "hierarchy" implies that the lower priority levels are solved at best if the higher priority ones are not damaged by the improved optimum. Another feature of this method is that the stack is examined to define which constraints are active and then to restrain the optimal value computation those constraints only.

The inverse-kinematic control and tasks building can also be approached by the artificial potentials field method (APF) [24]. As previously mentioned, the collision avoidance is an important element of the robot control. In the literature relative to the APF theory, not only the collision avoidance is addressed but also the obstacle avoidance. The control prevent the collision but also still reach the goal by going around obstacles. Several methods are developed such as the common repulsive fields detailed in [12] and also circumventing fields [15], [17] to guarantee the avoidance. The vortex fields are used to force the robot to go around the obstacle and this behaviour has been also implemented for dynamical systems used to generate trajectories [26], [16].

1.3 Approach and Structure

This thesis is focused on the study and the implementation of the Hierarchical Quadratic Programming (HQP) method for the kinematic control of robots sharing a workspace with humans. The general aim is to link HQP with a flexible trajectory generation method embodied by the Dynamic Movement Primitives.

The robot's work is made safer thanks to a hierarchy of tasks which cannot be violated. Safety tasks are then occupying the higher priority levels. Humans and other elements located in the workspace are then considered as obstacles to avoid. The obstacle avoidance will then be a key point in the thesis. It can be approached by two different possibilities. The first one is counting on the HQP method with a level of the stack of tasks dedicated to the avoidance. The second one is to have a trajectory planning that is taking the environment into account and exploit the DMP formulation that assures the obstacle avoidance.
The following structure is used to guide the reader in this research work: Chapter 2 is dedicated to the introduction of the HQP method as developed in [5] and the next one presents the use of this method for a kinematic controller. Chapter 4 is an analysis of the artificial potential fields methods designed for obstacle avoidance for robot control. Following this analysis, several tests and simulations are carried out where the junction of artificial potential field methods and HQP is also covered. Chapter 5 is a study of the different Dynamical Movement Primitives existing for-

mulations and their use for trajectory generation. Finally Chapter 6 regroups tests made on a UR10 robot, and the last one is of course the conclusion of this work accompanied by the perspectives of development.

2

Theoretical Background of HQP

This first chapter consists in a brief description of the Hierarchical Quadratic Programming (HQP) method proposed in [5]. HQP is an optimisation method that finds the solution in the least-square sense of a multiple objective problem expressed with equalities and/or inequalities ordered in a predefined hierarchy.

2.1 Hierarchy of tasks

The problem is formulated as a list of tasks ordered in a stack. Each task must be shaped as in Eq.(2.2) or Eq.(2.3). The optimisation problem is then expressed as follows:

$$\text{Find } x^* \in \underset{x}{\text{Arg min}} \parallel Ax - b \parallel \tag{2.1}$$

$$\text{with} \quad Ax = b \tag{2.2}$$

$$\text{and/or} \quad b_- \leqslant Ax \leqslant b_+ \tag{2.3}$$

where x is the vector of decision variables, A the task function and b the corresponding target for Eq.(2.2) or limits for Eq.(2.3).

2.2 HQP for equalities-only system

First, a system exclusively composed of equalities will be considered to emphasise the shape of the solution, and then the method to deal with inequalities will be introduced. The solution of a system of ordered equations is based on the approach proposed by Siciliano and Slotine [25]. The system consists of p linear equalities stacked in global matrices as follows:

$$A_i x = b_i \text{ with } i = 1, ..., p \tag{2.4}$$

$$\underline{A}_p = \begin{bmatrix} A_1 \\ \vdots \\ A_p \end{bmatrix} , \underline{b}_p = \begin{bmatrix} b_1 \\ \vdots \\ b_p \end{bmatrix} \tag{2.5}$$

One of the solutions x^* of the first constraint can be calculated using the Moore-Penrose pseudo-inverse of A_1 as follows:

$$x^* = A_1^\dagger b_1 \ , \tag{2.6}$$

The other constraints will be solved at best in the null space[1] of the first constraint (A_1). To illustrate that, the computation of the solution suited for the first and the second constraints is detailed. Using the projection matrix P_1 into the null space of A_1, another solution of the first constraint can be computed as follows:

$$x^* = A_1^\dagger b_1 + P_1 \tilde{x}_2 \ , \tag{2.7}$$

where $\tilde{x}_2$ is a vector of the decision variables. Thanks to P_1, $P_1\tilde{x}_2$ do not interfere with the optimal solution of $A_1 x = b_1$. The second constraint will be solved at best but only in the null space of the first one, which gives the following equation used for optimising the rest of the constraints:

$$\min_{\tilde{x}_2} \parallel A_2 P_1 \tilde{x}_2 - \left(b_2 - A_2 A_1^\dagger b_1\right) \parallel \tag{2.8}$$

The solution of the minimisation problem (2.8) is $\tilde{x}_2$ which corresponds to the contribution of the second constraint to the optimum value of the decision variables and is given below:

$$\tilde{x}_2 = (A_2 P_1)^\dagger \left(b_2 - A_2 A_1^\dagger b_1\right) + \tilde{P}_2 \tilde{x}_3 \ , \tag{2.9}$$

where $\tilde{P}_2$ is the projector in the null space of $(A_2 P_1)^\dagger$ and $\tilde{x}_3$ a vector of configuration parameter available for a third constraint.

And thus, an optimal solution of the two first constraints becomes:

$$x^* = A_1^\dagger b_1 + (A_2 P_1)^\dagger \left(b_2 - A_2 A_1^\dagger b_1\right) + P_2 \tilde{x}_3 \ , \tag{2.10}$$

with $P_2 = P_1 \tilde{P}_2$ being the projection matrix in the common null space of A_1 and A_2. For the further matrix decomposition in section 2.2.3, the projector P_k is expressed with a basis of the null space Z_k as follows : $P_k = Z_k Z_k^T$. This expression allows an easier computation. The expression of the optimal configuration when the p levels are considered is:

$$x_p^* = \sum_{k=1}^{p} (A_k P_{k-1})^\dagger (b_k - A_k x_{k-1}^*) + P_p \tilde{x}_{p+1} \tag{2.11}$$

and in turn becomes:

$$x_p^* = \sum_{k=1}^{p} Z_{k-1} (A_k Z_{k-1})^\dagger (b_k - A_k x_{k-1}^*) + Z_p \tilde{x}_{p+1} \tag{2.12}$$

2.2.1 Slack variable and lexicographic order

If a constraint cannot be satisfied (in the null space of the previous ones), its violation must be minimised. Therefore the vector of slack variables w is introduced. When the method will be extended to inequalities, this variable will be used to relax

[1] The null space (or kernel) of a matrix, or more generally of a linear map $\mathcal{F} : X \to Y$ between two vector spaces X and Y is the set of vectors x such as : $\{x \in X | \mathcal{F}(x) = 0\}$

unfeasible constraints. If we consider a system of equality and inequality constraints in a total of p constraints, the problem formulation becomes then:

$$\min_{x_k, w_k} \parallel w_k \parallel \text{ for } k = 1, ..., p \tag{2.13}$$

$$\text{subject to } A_i x = b_i + w_i \text{ with } i = k \text{ if equality} \tag{2.14}$$

$$\text{or } A_j x \leqslant b_j + w_j \text{ with } j = k \text{ if inequality} \tag{2.15}$$

Taking the inequalities into account we will now make the formulation of the solution relevant when the inequalities will be added.
The hierarchical quadratic programming method aims to solve a quadratic problem following a strict hierarchy. To do so, the optimisation process cannot move the optimal value closer to a lower priority objective if it will move away the optimum from the higher priority objectives. The optimisation process respect a lexicographic order. This concern of priority is explicitly shown by the equation (2.16).

$$\operatorname*{lex\,min}_{x, w_1, ..., w_p} \{\parallel w_1 \parallel, ..., \parallel w_p \parallel\} \tag{2.16}$$

with $w_k = A_k x - b_k$.

2.2.2 Optimality conditions

The optimisation problem at the k level for an equality system is expressed as follows:

$$\min_{x_k, w_k} \parallel w_k \parallel \tag{2.17}$$

$$\text{subject to } A_k x_k = b_k + w_k \tag{2.18}$$

$$\underline{A}_{k-1} x_k = \underline{b}_{k-1} + \underline{w}^*_{k-1}, \tag{2.19}$$

with $\underline{A}_{k-1}, \underline{b}_{k-1}, \underline{w}^*_{k-1}$ the stacked quantities from level 1 to $k-1$ where $\underline{w}^*_{k-1}$ is fixed due to the computation of the previous levels optima and due to the necessity of respecting the lexicographic order.

The Lagrangian of the system is given by the following equation:

$$\mathcal{L}_k = \frac{1}{2} w_k^T w_k + \underline{\lambda}_k^T \left(\underline{A}_{k-1} x_k - \underline{b}_{k-1} - \underline{w}^*_{k-1} \right) + \lambda_k^T \left(A_k x_k - b_k - w_k \right) \tag{2.20}$$

and the corresponding optimality conditions are given below:

$$w_k = A_k x_k - b_k \tag{2.21}$$

$$\underline{A}_{k-1} x_k = \underline{b}_{k-1} + \underline{w}^*_{k-1} \tag{2.22}$$

$$\lambda_k = w_k \tag{2.23}$$

$$\underline{A}_{k-1}^T \underline{\lambda}_k = -A_k^T w_k \tag{2.24}$$

with x_k being the system parameters, w_k the slack variables, λ_k the Lagrange multiplier for the k level (corresponding to constraint (2.18)) and $\underline{\lambda}_k$ the Lagrange multiplier for the stack equation (2.19).

2.2.3 Hierarchical Complete Orthogonal Decomposition

A key element in the process execution is the computation of Moore-Penrose pseudo-inverse of the A_k matrix. Such matrix operation turns out to be costly in terms of computation, a simple procedure should be used to avoid expensive computations. To do so, a complete rank revealing decomposition can be employed by using either a singular value decomposition (SVD) or a complete orthogonal decomposition (COD) method. In [5] the COD is preferred owing to its simplicity of implementation and its execution speed. As an illustration, the complete orthogonal decomposition of the first level is given below:

$$A_1 = \begin{bmatrix} V_1 & U_1 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ L_1 & 0 \end{bmatrix} \begin{bmatrix} Y_1 & Z_1 \end{bmatrix}^T = U_1 L_1 Y_1^T \tag{2.25}$$

where Z_1 is the basis of the kernel (or null space), Y_1 is a projection matrix in the space of A_1, U_1 is a basis of the range space of A_1, V_1 is an identity matrix which dimension is equal to the rank deficiency of A_1 and finally L_1 is a triangular matrix resulting from the decomposition. The Moore-Penrose pseudo-inverse becomes:

$$A_1^\dagger = \begin{bmatrix} Y_1 & Z_1 \end{bmatrix} \begin{bmatrix} 0 & L_1^{-1} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1 & U_1 \end{bmatrix}^T = Y_1 L_1^{-1} U_1^T \tag{2.26}$$

In regard of the second stage, $A_2 Y_1$ represents the coupling between the two first levels meaning that the corresponding part of the parameters (x) is already assigned and cannot be modified. The matrix $A_2 Z_1$ gives the free space and so the part of the parameters that can still be assigned. Since we need to take into account the common null space of the previous stages, the decomposition of the k^{th} level becomes:

$$A_k = \begin{bmatrix} V_k & U_k \end{bmatrix} \begin{bmatrix} N_k & 0 & 0 \\ M_k & L_k & 0 \end{bmatrix} \begin{bmatrix} \underline{Y}_{k-1} Y_k & Z_k \end{bmatrix}^T = W_k H_k \underline{Y}_k^T \tag{2.27}$$

with $M_k = U_k^T A_k \underline{Y}_{k-1}$ and $N_k = V_k^T A_k \underline{Y}_{k-1}$ the matrices that embodies the coupled part of A_k with all the previous levels expressed by $A_k \underline{Y}_{k-1}$, $\underline{Y}_{k-1} = \begin{bmatrix} Y_1 & \cdots & Y_{k-1} \end{bmatrix}$, $W_k = \begin{bmatrix} V_k & U_k \end{bmatrix}$, $H_k = \begin{bmatrix} N_k & 0 \\ M_k & L_k \end{bmatrix}$ and $\underline{Y}_k = \begin{bmatrix} \underline{Y}_{k-1} & Y_k \end{bmatrix}$. The free space is still expressed with $A_k Z_{k-1}$.

Finally the decomposition for the stacked data is obtained by a recursive process using the following equation:

$$\begin{bmatrix} \underline{A}_{k-1} \\ A_k \end{bmatrix} = \begin{bmatrix} \underline{W}_{k-1} & 0 \\ 0 & W_k \end{bmatrix} \begin{bmatrix} \underline{H}_{k-1} & 0 & 0 \\ N_k & 0 & 0 \\ M_k & L_k & 0 \end{bmatrix} \begin{bmatrix} \underline{Y}_{k-1} & Y_k & Z_k \end{bmatrix}^T = \underline{W}_k \underline{H}_k \underline{Y}_k^T \tag{2.28}$$

This decomposition is known as the hierarchical complete orthogonal decomposition of the all system (HCOD) [5]. The computation of the HCOD is detailed in the section 2.4.

2.2.4 Optimum computation

Regarding the optimality conditions previously expressed, the parameters that will be computed are the optimal values for the p levels system parameters x_p^*, the corresponding slack variables $\underline{w}_p$, the Lagrange multiplier at each level λ_k and the Lagrange multiplier for the stack equation $\underline{\lambda}_k$. The resolution of the following equations is known as the equality hierarchical quadratic programming algorithm or eHQP in [5].

The system parameters x_p^* :
The solution of a "two constraints" problem given in (2.10) can be generalised for the k level by including the hierarchical complete orthogonal decomposition as follows:

$$x_k^* = x_{k-1}^* + Y_k L_k^{-1} U_k^T \left(b_k - A_k x_{k-1}^*\right) Z_k z_{k+1} \tag{2.29}$$

Since the solution at the level k depends on this level of the HCOD and the solution at the previous levels, it can be reshaped to make the recurrence appears in the matrix computation. A new matrix $\underline{A}_p^\ddagger$ is defined such that $x_p^* = \underline{A}_p^\ddagger \underline{b}_p$. This matrix is computed by the following recurrent relation using the HCOD from (2.28):

$$\underline{A}_p^\ddagger = \underline{Y}_p \underline{H}_p^\ddagger \underline{W}_p^T \tag{2.30}$$

with

$$\underline{H}_k^\ddagger = \begin{bmatrix} \underline{H}_{k-1}^\ddagger & 0 & 0 \\ -L_k^{-1} M_k \underline{H}_{k-1}^\ddagger & 0 & L_k^{-1} \end{bmatrix} \tag{2.31}$$

An alternative way to compute the optimum is with the Y basis. This basis allows to emphasise the contribution of each additional task as shown below:

$$\underline{y}_k^* = \begin{bmatrix} \underline{y}_{k-1}^* \\ y_k^* \end{bmatrix} \tag{2.32}$$

with $y_k^* = L_k^{-1} \left(U_k^T b_k - M_k \underline{y}_{k-1}^*\right)$ being the additional contribution of the considered level. Once the computations are made in this basis, the optimal value of the system parameters are obtained using : $\underline{y}_k^* = \underline{Y}_k^T x_k^*$.

The slack variables $\underline{w}_p^*$:
The computation of the slack variables results from the optimum (2.21). The slack variables for the k level can be computed by using the decomposition of the A_k matrix in (2.21) as follows:

$$w_k^* = V_k N_k \underline{y}_{k-1}^* - V_k V_k^T b_k \tag{2.33}$$

or alternatively:

$$w_k^* = V_k V_k^T \left(A_k x_{k-1}^* - b_k\right) \tag{2.34}$$

The computation of λ_k is straight forward since $\lambda_k = w_k^*$ (2.23).

The (stacked) Lagrange multipliers $\underline{\lambda}_k$:
Using the hierarchical inverse in (2.30) in (2.24), the $\underline{\lambda}_k$ becomes:

$$\underline{\lambda}_k = -\underline{A}_{k-1}^{\ddagger T} A_k^T w_k^* \tag{2.35}$$

Since the HQP method aims to be fast, the computation of the Lagrange multipliers is modified to establish a recurrent relation. To this end, a new variable called cumulative variable, $\underline{\rho}^{j+1}$ is then introduced. This vector has two parts, defined by the separation of the stack in $\begin{bmatrix} \underline{A}_j \\ A_{j+1} \end{bmatrix}$. For j going from $k-1$ down to 1, the cumulative variable is computed as follows:

$$\underline{\rho}^{(j)} = \underline{Y}_j^T \left(\sum_{i=j+1}^{k-1} A_i^{Ti} \underline{\lambda}_k + A_k^T w_k \right) \tag{2.36}$$

with

$${}^j\underline{\lambda}_k = U_j L_j^{-T} Y_j^T \left(\sum_{i=j+1}^{k-1} A_i^{Ti} \underline{\lambda}_k + A_k^T w_k \right) \tag{2.37}$$

where $Y_j^T \left(\sum_{i=j+1}^{k-1} A_i^{Ti} \underline{\lambda}_k + A_k^T w_k \right)$ corresponds to the second part (the last element) of the cumulative vector. This computation process is more complicated to implement than the hierarchical inverse since it requires a loop for j going from $k-1$ down to 1 for k going from 2 to p but since the HCOD, x_p^* and $\underline{w}_p^*$ have already been computed, the calculation is still faster than the whole hierarchical inverse.

2.3 HQP for inequalities and equalities systems

The problem of optimisation for a system composed of inequalities has been addressed in [10]. The resolution is similar since it is based on the minimisation of the slack variables and the decision variables. Nevertheless, this method can be considered as slow due to the large amount of computation required to solve the whole stack. An improved method is then set up as described in [5].
As highlighted in [5], two classes of algorithms are designed to handle quadratic programming problems with inequality constrains: the interior-point methods and the active-search algorithms. The chosen method implemented in HQP solver is the active-search method.

The principle of the active-search is to iterate in order to establish the set of constraints from the stack that are active. A task is considered as active if it holds as an equality. The case of bounded output clearly illustrates this statement. Let an example of stack of tasks be composed of two tasks. The first one (with the highest priority) being bounds for the output (typically expressed as inequalities) and the second one being a command the output has to follow.
At first, the output values are not computed yet and so by default the bounds aren't violated. The computation of the optimum output values regarding the command (second task in this example) is made. If these output values do not violated the

bounds, the active set of tasks will consist of the second task. In contrast, if the bounds are violated, the output cannot be true and the violated bounds will hold as equality in the optimum computation. In this last case, the active set of tasks will be composed of the bounds that hold as equalities and the second task.

The active search method needs an initial guess of the active set of tasks. However, at the very beginning of the execution, the active set is empty since the algorithm does not even know the stack yet. The initial guess of the set proposed in [5] is composed of the equalities of the stack. Once an active set is determined, an initial point can be computed through eHQP. The given result is then modified recursively:

$$x^{(i+1)} = x^{(i)} + \tau(x^{(*i)} - x^{(i)}) \tag{2.38}$$

with i being the iteration number, $x^{(*i)}$ the optimum of eHQP, $x^{(i)}$ the previous result of equation 2.38 ($x^{(i)} = \underline{0}$ when $i = 0$) and τ a parameter called step length. The step length is computed for each row of each constraint level. The aim is to check if the inactive constraints are not violated. The mathematical definition of τ for the r^{th} row of the k^{th} constraint is :

$$\tau_{k,r} = \begin{cases} \frac{b_{k,r} - A_{k,r}x^{(i)}}{A_{k,r}\left(x^{(*i)} - x^{(i)}\right)} & \text{if the constraint is violated} \\ 1 & \text{otherwise} \end{cases} \tag{2.39}$$

After the computation of τ for each row of the level k task, the one presenting the minimum value of τ is the row that will be added in the set of active constraints for the next active search iteration. By opposition, if at the end of the τ computation for the level, its value remains equals to one, no row of the level k should be added in the active set.

2.4 Hierarchical active search - Implementation and Computation

The algorithm is composed by two loops. The first one, called inner loop, aims to activate the constraints that requires to be taken into account in the active set of tasks and also to check if the non-activated constraints are satisfied. The second one, called the outer loop, aims to remove the unnecessary constraints while scanning the stack of tasks. At each iteration of the active search algorithm, the eHQP is computed with the actual active set. Depending on the results given by the eHQP, the inner and outer loops will change, if necessary, the active set. When no more changes are required, the final optimum is returned.

As previously quoted, a key element in this method is the decomposition of the stack of (active) tasks. Furthermore when the set of active tasks is modified (upgraded by the inner loop or downgraded by the outer loop), a new decomposition is needed. Nevertheless the whole decomposition is costly in terms of computational resources. In terms of continuity, the inner or outer loops will not generate a new active-set

radically different of the previous one[2] therefore it is not necessary to recompute the whole decomposition but only the part of the stacked matrix $\underline{A}_p$ affected by the changes. Three main cases are to consider: the whole decomposition (executed on the first iteration of active search algorithm), the addition of a constraint and the deactivation of a constraint.
Since the hierarchical orthogonal decomposition is an iterative process, the k level introduced in the subsection (2.2.3) is considered. The aim of the decomposition is to express A_k as in the equation (2.40):

$$A_k = W_k \begin{bmatrix} N_k & 0 & 0 \\ M_k & L_k & 0 \end{bmatrix} \begin{bmatrix} \underline{Y}_k & Z_k \end{bmatrix}^T \tag{2.40}$$

The first matrices to be computed are Y_k from $\underline{Y}_k = \begin{bmatrix} \underline{Y}_{k-1} & Y_k \end{bmatrix}$ and Z_k. To do so, the rank-revealing LQ decomposition[3] of the matrix $A_k Z_{k-1}$ (being the k level matrix projected in the free-space of the $k-1$ level) is computed. A practical way to obtain such result is to perform a QR factorisation on the transpose of $A_k Z_{k-1}$. The QR factorisation of the transpose is illustrated on the figure 2.1. When this factorisation is applied, the matrix $A_k Z_{k-1}$ can be rewritten as follows:

$$A_k Z_{k-1} = \Pi \begin{bmatrix} L^R & 0 & \cdots & 0 \\ L^N & 0 & \cdots & 0 \end{bmatrix} Q^T \tag{2.41}$$

with Q being the product of the r_k Householder reflections matrices used in the QR factorisation process to nullify the tails of the rows and Π being the product of all the r_k permutations needed during the nullifying process (cfr figure 2.1). By comparing the two sides of Eq. (2.41), Q is a transformation of Z_{k-1} that reveals the rank of the level k.
Finally, the matrices Y_k and Z_k are found using the following equality:

$$\begin{bmatrix} Y_k & Z_k \end{bmatrix} = Z_{k-1} Q \tag{2.42}$$

Matrix Y of the whole stack is found after the decomposition of the p levels with $Y = \begin{bmatrix} \underline{Y}_p & Z_p \end{bmatrix}$ and will be further used for the update of the stack matrix decomposition.

The matrix W_k is built by applying a series of Givens rotations that aims to nullify the L^N part of the matrix in (2.41). The matrices M_k and N_k appear thanks to the r_k Givens rotations [4] applied on $A_k \underline{Y}_k$. The algebraic process is detailed in [5] and more information about the matrices manipulation quoted in the section can be found in [3].

[2] At each active search algorithm iteration, one constraint (or a part of it) is either added or removed, the active set is then modified by one element only.
[3] Matrix decomposition that aims to nullify the tails of the matrix's rows
[4] Matrix manipulation, more information in [3]

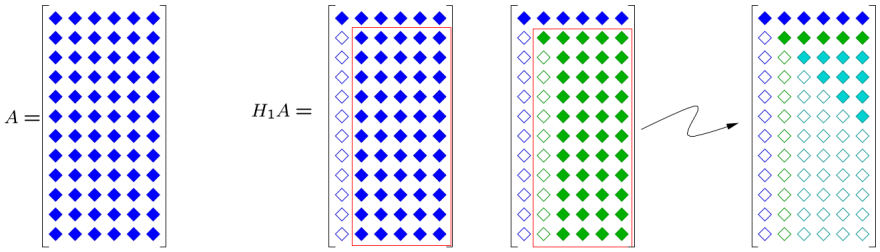


Figure 2.1: Illustration of the QR factorisation - [6]

The activation of a new constraint corresponds to the addition of a new row in the $\underline{A}_p$. This procedure is triggered depending on the result of the step length computation previously mentioned. Adding a line in the decomposition does not mean that the whole HCOD must be recomputed. Let consider the level k. The update of the W_k matrix is straightforward since the previous levels are not affected by this, $W_{new} = \begin{bmatrix} W_k & 0 \\ 0 & 1 \end{bmatrix}$. Nevertheless, depending on the amount of zeros in the tail of the new row, it becomes more complicated for the rest of the decomposition. Indeed the new row can either increase the rank of the level by one or leave it unchanged. For the following levels, their rank can either stay the same or decrease by one since they do not have any additional row but they only deal with the propagation of this change.
The global idea is to apply several Givens rotation on Y basis to nullify the tail of the new row which will give a new matrix Y_{up}. Once the decomposition of the level k is completed, the propagation of the modification thanks to application Y_{up} at each level is done.

The deactivation means to remove a line in the matrix $\underline{A}_p$ and this removal influences its HCOD. The row to be removed is considered as the last one the level k and if this is not the case, a preliminary permutation must be done to place that row at the last position. The first step is to build the matrix W_{down} so that the deactivated line has a unitary contribution in W_k as in Eq.(2.43). The way to build W_{down} matrix is through a series of Givens rotations that aim to nullify the tail of the row to be removed (the last, by previous definition) in H_k. The new matrices for the level k are W_{new} in the equation (2.43) and H_{new} is the H_k without the last row.

$$W_k W_{down} = \begin{bmatrix} & & & 0 \\ & W_{new} & & \vdots \\ & & & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix} \tag{2.43}$$

Once the level k has been updated by the deactivation, the modification must be propagated to the lower levels. To do so, the matrix Y_{down} is applied on the next level[5]. Depending on the rank of the H_i (with $i > k$), Givens rotations applied on the left will be needed to nullify the tails of the N_i matrix from the COD of the i

[5]Recall : Y is the right basis projection matrix used to project the next levels in the common space of the previous stages.

level in order to recover the HCOD shape initially shown in (2.28).

The global active-search algorithm detailed in this chapter is summarised:

Algorithm 1: Hierarchical Quadratic Programming Algorithm

Data: Every Constraints respecting the shape 2.2
Result: Optimal solution x^* and active set of tasks

```
Building the stack from constraints;
Active search:
if First call then
    initialisation of the active set of tasks by activating all the equalities
     detected in the stack;
else
    Go to while loop;
while index ≠ length stack do
    eHQP optimal computation;
    Constraints violation research among whole stack;
    Innerloop:
    if Violation of constraint detected then
        Activation of the constraint having the highest priority among
         violated one;
    else
        Go to outerloop;
    Outerloop:
    if no deactivations needed then
        index is incremented;
    else
        index is not changed and the stack is reinspected ;
Optimum solution x* = Actual solution
```

3
Application of HQP in Robotics

The third chapter is about the employment of the HQP method in the case of the kinematic control of a robot. For robotic applications, finding the best input for a robot subjected to several objectives and constraints is a common issue.

First of all, an introduction to the robotic kinematic control method is given. The robots are considered as actuated multi-body systems. They are composed by links connected through joints. An analogy with the human would give the following correspondence: the arm and the forearm are the links and the shoulder and the elbow are the joints. The robot end-effector operates in a space called workspace (or operational space). The joint space is the space of the joint and its dimension is equal to the number of the degrees of freedom[1]. Example: for a six-joints robot, the joint space is six-dimensional.

A very basic illustration of close loop kinematic control with the Jacobian inverse is shown in figure 3.1.

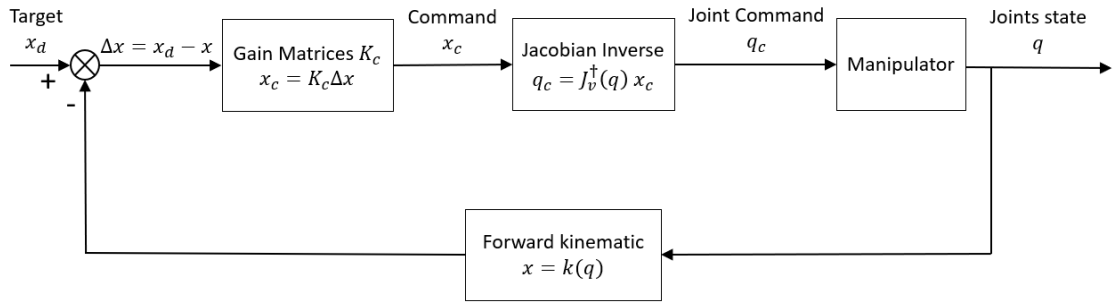


Figure 3.1: Illustration of close loop kinematic control: Jacobian pseudo-inverse. x is the robot position in the workspace, q is the joint position, x_d is the position command, J_v is the Jacobian matrix and $J_v^{\dagger}$.

Using HQP as an optimal kinematic controller implies that it will have to solve the inverse differential kinematics of the robot. Other methods used for the kinematic control of a robot are similar to HQP since they are also based on the use of the pseudo inverse differential kinematics. An example is the Prioritised Inverse Kinematic method (PIK) [9] which also aims to take the concept of hierarchy into

[1] A so called degree of freedom is an actuated joint of the robot

account. Some more common methods, purely based on the closed loop kinematic control, are described in [1].

3.1 Hierarchical Quadratic Programming in robotic framework

As illustrated in [5], HQP is used to solve the inverse kinematic problem with additional constraints to generate motion.

In order to fit with the equations formalism of the HQP method[2], the objectives to be performed by the robot are expressed with the task-function approach. It consists in expressing the task as a differentiable function e of q (q being the actuated joints of the robot). In order to represent the evolution of this task with the configuration parameter, the task Jacobian is defined as $J = \frac{\partial e}{\partial q}$. The task is finally expressed as follows:

$$\dot{e} = J\dot{q} \tag{3.1}$$

Finding the best motion for the robot according to the specified task (3.1) is equivalent to solving a quadratic programming problem (2.1) that is in this case expressed as follows:

$$\text{Find } \dot{q}^* \in \text{Arg}_{\dot{q}} \min \parallel J\dot{q} - \dot{e} \parallel \tag{3.2}$$

The HQP method is used to solve the inverse kinematic problem for the stack of tasks Jacobians. HQP behaves then as an optimisation-based controller where the decision variables x are the joint velocities $\dot{q}$. As mentioned in the previous chapter, the aim of the stack is to solve the quadratic programming problem while respecting the hierarchy imposed by the location of the tasks in the stack. In robot control, a pre-hierarchy can be built. The higher priority levels will be dedicated to safety constraints such as joints position, velocity and acceleration bounds. Then will come the collision avoidance task and finally the trajectory tracking.

3.2 Expression of the tasks

Regarding to the task-function approach, different tasks such as trajectory tracking and obstacle avoidance must be expressed at the velocity level. The tasks introduced in the following section are illustrated with their implementation in HQP method and simulation in MatLab for the simplified case of a three links planar arm.

3.2.1 Trajectory tracking tasks

The trajectory tracking task command is expressed as an attractor system. The task displayed in Eq.(3.3) represents the square of the error between the trajectory

[2] $Ax = b$ or $b_- \leqslant Ax \leqslant b_+$

point and the end-effector position.

$$e_{Trajectory} = (p_t(t) - p_e)^T (p_t(t) - p_e) \tag{3.3}$$

As mentioned in the beginning of this chapter, the equation considered by the solver is the derivative of the task. The differentiation of this equation gives the desired form Eq.(3.1). Since the task (3.3) is the tracking error, an asymptotic convergence is induced to drive the error to zero. This convergence is achieved by considering $\dot{e} = -K_t e$, with K_t being a positive gain that reflects the strength of the attraction and thus tunes the convergence speed.
Once the differentiation is completed and the asymptotic convergence is taken into account, the task is expressed as follows:

$$J_v \dot{q} = \dot{p}_t + \frac{1}{2} K_t (p_t(t) - p_e) \ , \tag{3.4}$$

with J_v being the Jacobian : $J_v = \begin{bmatrix} \cdots & \frac{\partial x}{\partial q_i} & \cdots \\ \cdots & \frac{\partial y}{\partial q_i} & \cdots \\ \cdots & \frac{\partial z}{\partial q_i} & \cdots \end{bmatrix}$ and K_t a positive constant.

3.2.2 Bounds

The safety constraints are the expressions of the bounds. The solver considers the joint velocities as its parameters, the velocity bounds are then simply expressed with Eq.(3.6). The joint positions bounds are defined with Eq.(3.5) where the joints position were replaced by $q_t = q_{t-1} + \dot{q}\Delta t$. The acceleration bounds can also be defined with the same reasoning Eq.(3.7) using an approximation made with the backwards difference $\ddot{q}_t = \frac{\dot{q}_t - \dot{q}_{t-1}}{\Delta t}$.

$$\frac{\kappa}{\Delta t} (q_{min} - q_{t-1}) \leqslant I_n \dot{q}_t \leqslant \frac{\kappa}{\Delta t} (q_{max} - q_{t-1}) \tag{3.5}$$

$$\dot{q}_{min} \leqslant I_n \dot{q} \leqslant \dot{q}_{max} \tag{3.6}$$

$$\ddot{q}_{min} \Delta t + \dot{q}_{t-1} \leqslant I_n \dot{q}_t \leqslant \ddot{q}_{max} \Delta t + \dot{q}_{t-1} \tag{3.7}$$

To illustrate the activation of the bounds, the trajectory used for this example is a circle in the workspace that has to be completed in 5 seconds. As shown in Eq.(3.3), the velocity bounds are activated one after another to follow the trajectory in the best possible way.

3.2.3 Workspace limitation

The limitation of the workspace means that the end-effector must remain in a limited area of the space. The task naturally linked to workspace limitation is the following inequality constraint:

$$p_{min} \leqslant p_e \leqslant p_{max} \tag{3.8}$$

While expressed in the velocity level, we get the following the inequality:

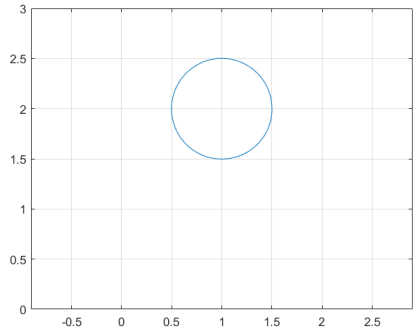


Figure 3.2: Trajectory of the end-effector in the workspace

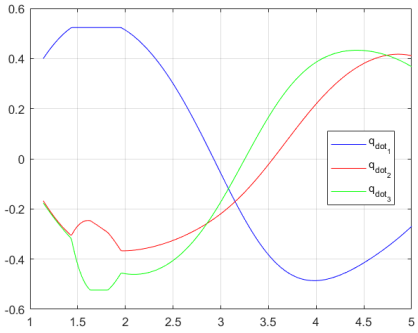
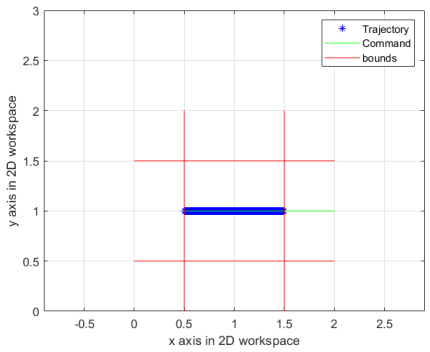


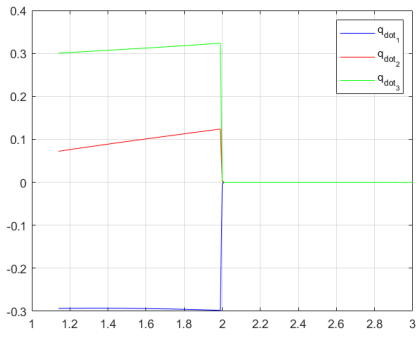
Figure 3.3: Joint velocities in time

$$\frac{\kappa}{\Delta t}\left(p_{min} - p_e\right) \leqslant J_v \dot{q} \leqslant \frac{\kappa}{\Delta t}\left(p_{max} - p_e\right) \tag{3.9}$$

Since the objective is to force the end-effector to remain in the defined area, the limitation task must have a higher priority level than the trajectory tracking. Figures 3.4 (a),(b) illustrate the satisfactory of the hierarchy since the robot stops following the trajectory when it reach the workspace border.



(a)



(b)

Figure 3.4: Illustration of the workspace limitation task. **(a)** End-effector motion in the limited workspace. **(b)** Joints velocities.

3.2.4 Obstacle avoidance tasks

There are several way to express the obstacle avoidance. In a first instance, an obstacle in the workspace is represented by a circle (2D) or a sphere (3D) that includes the 'real' obstacle. Therefore, each obstacle must be described by a radius R and the coordinates of the centre p_o. The task that prevents the collision between the end-effector (at the position p_e) and the obstacle is expressed as follows:

$$d \geqslant d_0 \quad \text{with} \quad d = \sqrt{\left(p_e - p_o\right)^T \left(p_e - p_o\right)} \quad \& \quad d_0 = R \tag{3.10}$$

Eq.(3.10) does not take the shape of the robot into account since the collision avoidance is based on the position of a control point (the end-effector in this case) with respect to the area R occupied by the obstacle.
In the velocity level, the task becomes:

$$\frac{\kappa}{\Delta t}(d_0 - d) \leqslant \frac{(p_e - p_o)^T}{d} J_v \dot{q} \tag{3.11}$$

An example of obstacle avoidance for the end-effector position as the control point is shown in the figures 3.5 and 3.6. The straight line trajectory is violated in order to avoid the obstacle until there is no risk of collisions as proves the evolution of the error in X and Y in the figures 3.8 and 3.8.

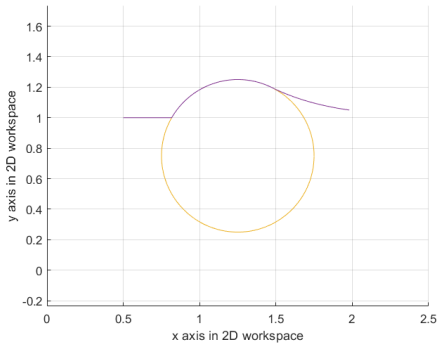


Figure 3.5: Trajectory of the End-Effector

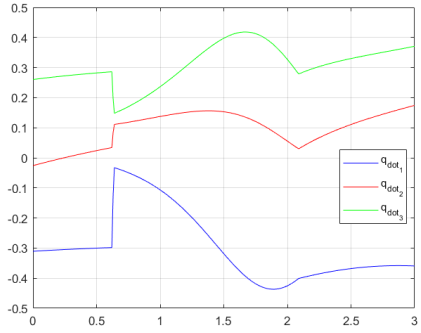


Figure 3.6: Joint velocities in time

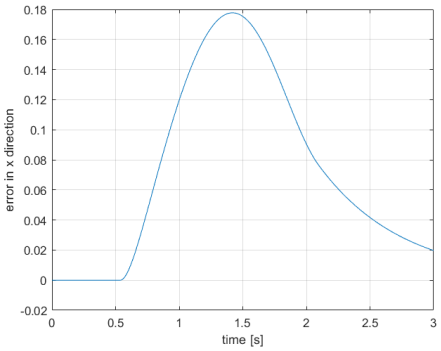


Figure 3.7: Error in time in the x direction of the workspace

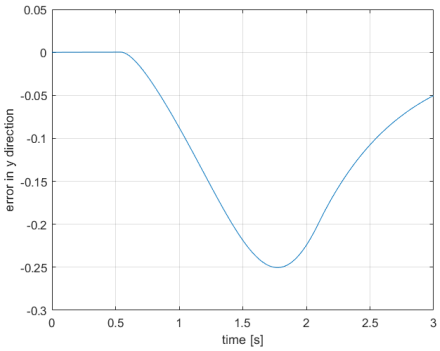


Figure 3.8: Error in time in the y direction of the workspace

The collision avoidance was made so far for the end-effector only. In order to prevent collisions with the rest of the robot, more control points must be taken into account.

3.2.5 Improved collision avoidance

The previous expression of the obstacle avoidance are expressed as a point to point constraint with an obstacle point and a control point. However, if the collision avoidance policy is made for the whole robot, the previous collision avoidance constraint

must be applied for each new control point, which is the same as adding a new level (Eq.3.11) for each control point. The advantage of this method is the simplicity of the constraint's expression. However, a major drawback is the number constraints generated for completely covering the robot. As an example, if n_{ctrl} control points are taken on the robot's skeleton with n_o obstacles detected in the workspace, the total number of constraints is $n_{ctrl}n_o$. It has been detailed in the chapter 2 that the active search algorithm inner and outer loops are going through each constraint level several times in one active search iteration. It is therefore important to limit the number of tasks to reduce the execution time.

To do so, the collision avoidance task can be expressed as a "line to point constraint". The constraint will remain the same as in the equation (3.10) but the control point becomes the closed point to the obstacle on the line linking two joints. Considering two points p_1 and p_2 representing the coordinates of the joints on both sides of the link and p_o being the coordinates of the centre of the sphere incorporating the obstacle. The equivalent task of Eq.(3.11) for a line is expressed at the velocity level as follows:

$$\frac{\kappa}{\Delta t}(d_0 - d) + \frac{1}{d}(p_{cl} - p_o)^T \left(\dot{p}_o - \frac{\overrightarrow{d_{12}}\overrightarrow{d_{12}}^T}{\|\overrightarrow{d_{12}}\|^2}\dot{p}_o \right) \leqslant \frac{1}{d}(p_{cl} - p_o)^T J_{v_{cl}}\dot{q} \tag{3.12}$$

where

$$\begin{aligned} J_{v_{cl}} =& J_{v_1} + \frac{\overrightarrow{d_{12}}\overrightarrow{d_{10}}^T}{\|\overrightarrow{d_{12}}\|^2}(J_{v_2} - J_{v_1}) - \frac{\overrightarrow{d_{12}}\overrightarrow{d_{12}}^T}{\|\overrightarrow{d_{12}}\|^2}J_{v_1} \\ & - 2\frac{\overrightarrow{d_{12}}\overrightarrow{d_{12}}^T\overrightarrow{d_{10}}\overrightarrow{d_{12}}^T}{\|\overrightarrow{d_{12}}\|^4}(J_{v_2} - J_{v_1}) + \frac{\overrightarrow{d_{12}}^T\overrightarrow{d_{10}}}{\|\overrightarrow{d_{12}}\|^2}(J_{v_2} - J_{v_1}) \end{aligned} \tag{3.13}$$

with $\overrightarrow{d_{12}}$ being the line direction vector $< p_2 - p_1 >$, J_{v_i} the Jacobian corresponding to point p_i, $\overrightarrow{d_{10}}$ the direction vector $< p_0 - p_1 >$, d_0 the radius of the obstacle with the girth of the link considered and $d = \sqrt{(p_{cl} - p_o)^T (p_{cl} - p_o)}$ the shortest distance between the link and the obstacle centre where $p_{cl} = p_1 + \overrightarrow{d_{12}}\frac{\overrightarrow{d_{12}}^T\overrightarrow{d_{10}}}{\|\overrightarrow{d_{12}}\|^2}$.

Nevertheless, the task must be further elaborated. This method relies on the point of the link closest to obstacle. This point has to be located on the link, which implies that $\frac{\overrightarrow{d_{12}}^T\overrightarrow{d_{10}}}{\|\overrightarrow{d_{12}}\|^2} \in [0, 1]$. This case is illustrated in figure 3.9. If the point is out of the link-line (figures 3.10 and 3.11), the obstacle avoidance task is Eq.(3.11) where the controlled point is the either p_1 or p_2 depending on which one is closer to the obstacle.

The different cases are summarised in the following Algorithm 2:

Algorithm 2: Collision avoidance Task

Finding closest point : $p_{cl} = p_1 + \overrightarrow{d_{12}} t_{line}$ where $t_{line} = \frac{\overrightarrow{d_{12}}^T \overrightarrow{d_{10}}}{\|\overrightarrow{d_{12}}\|^2}$

if $t_{line} \in [0, 1]$ **then**

  The collision avoidance task is 3.12 ;

else if $t_{line} < 0$ **then**

  The collision avoidance task is 3.11 with $p_e = p_1$;

else

  then $t_{line} > 1$ and the collision avoidance task is 3.11 with $p_e = p_1$;

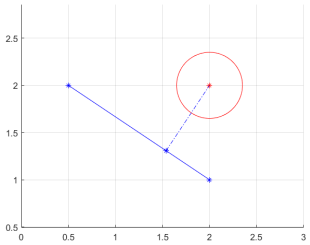


Figure 3.9: Link wise constraint - Intersection in the link

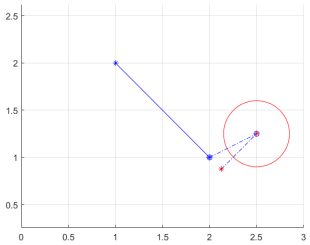


Figure 3.10: Link wise constraint - Intersection after the link

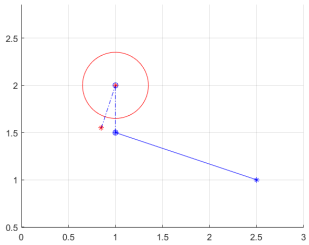


Figure 3.11: Link wise constraint - Intersection before the link

An illustration of this task is found in the figure 3.12. In this example, the three links planar arm is asked to reach the point (2;2) in a field of obstacle where the line collision avoidance is applied on the last link. The end-effector's motion prevents then the collision between the link and the obstacles. The goal will not be reached since the collision avoidance has a higher priority in the stack and that reaching the goal implies an interference between link and obstacle.

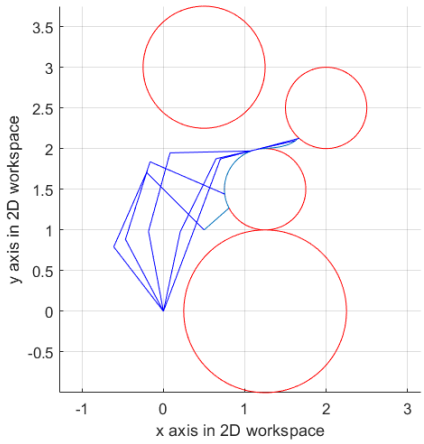


Figure 3.12: Trajectory of the three links planar arm is the workspace.

3.2.6 Concave obstacle issue

An issue encountered with the kind of collision avoidance constraints presented is that the robot get stuck in concave obstacle configurations. The HQP method uses the concept of lexicographic order to arrange the optimisation process, which makes it impossible for the solver to increase the slack variables. In this case, the bounds are top priority constraints, then comes the obstacle collision avoidance and finally the goal attraction. When the end-effector is stuck in the circles intersection, the action of going back does not create any conflict with the top priority level neither with the obstacle avoidance level but increase the slack variables for the trajectory tracking level. The result of this situation is that the robot will remain blocked in this position. An illustration is provided at the figure 3.13.

Alternative ways to express the obstacle avoidance task are studied in order to prevent this situation. Instead of using a constraint based on distance comparison, more sophisticated ways are studied as the artificial potential approach such as repelling or circumventing constraints in the Chapter 4.

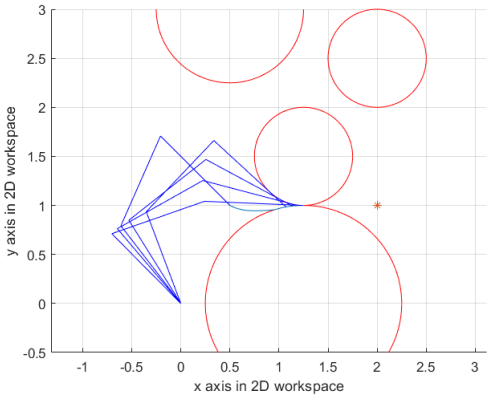


Figure 3.13: Illustration of the consequence of a lexicographic order based optimisation solver with concave obstacles.

3.3 Evaluation of HQP

In order to study the accuracy and stability of HQP inverse made with a Hierarchical Complete Orthogonal method (HCOD), we compare HQP and the sequential quadratic programming method (SQP) which is part of MatLab toolbox [10] for a simple trajectory tracking task in a straight line a two-dimensional space. The accuracy and stability are assessed with the computation error and its propagation in time. The results of the simulation are shown in figures 3.14,3.16,3.15 and 3.17.

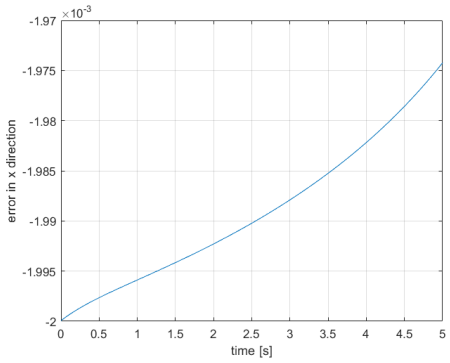


Figure 3.14: Error in x for a trajectory tracking task computed with HQP.

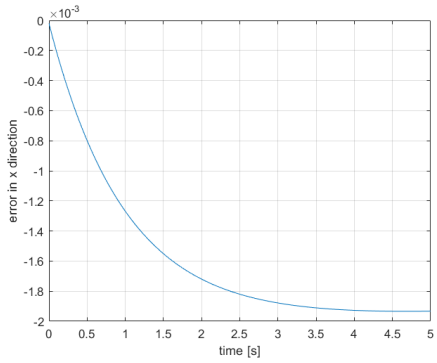


Figure 3.15: Error in x for a trajectory tracking task computed with SQP.

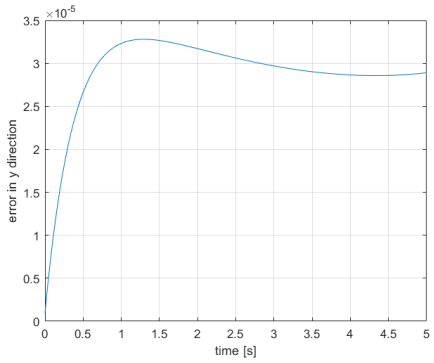


Figure 3.16: Error in y for a trajectory tracking task computed with HQP.

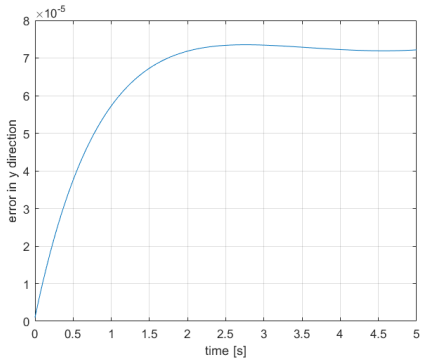


Figure 3.17: Error in y for a trajectory tracking task computed with SQP.

The error magnitude and behaviour for HQP method are the same as for SQP method. The numerical uncertainties linked to the additional operations inherent to the hierarchical way of managing the stack of tasks are not critical for the global computation accuracy.

4

Artificial Potential Fields and their use with HQP method

The artificial potential fields are similar to the task functions allowed in the HQP solver when they are expressed at the kinematic level (cfr. chap 12 [24]). The objective of this chapter is to provide an overview of the state of art of the existing artificial potential fields for obstacle avoidance and to test if and how they can be combined with the stack of tasks approach.

The artificial potential theory aims to express the tasks as potential functions generating force fields in the space. An analogy will be made further between those fields and inviscid fluid flows in fluid dynamics. Those fields can either be expressed in the operational space or in the joint space. Once the potential functions are built, the negative gradient with respect to the corresponding degree of freedom represents a force field for the end-effector. When the potentials are expressed in the workspace, the global force field is obtained by superimposing each of the fields. Theses force fields can be used at the kinematic level to command the motion of the robot as follows [24]:

$$J_v \dot{q} = f_{total} = \sum_{i=1}^{N} f_i = \sum_{i=1}^{N} -\nabla_p U_i \tag{4.1}$$

where J_v is the Jacobian matrix and U_i represents any potential defined in this chapter.

4.1 Attraction Field

An attraction field is a function that aims to drive the controlled point to another position called the goal. Two attraction fields are presented in this section. Both of their expressions contain the error between the controlled point and the goal $e(q) = p_{goal} - p_e$. Since the potential is used through its negative gradient, the resulting force field aims to minimise this error $e(q)$. Depending on the expression of the potential, the convergence to the goal will be different. The first kind is the paraboloidal shape (4.2) and the second one is the conical shape (4.3).

Potentials

$$U_{a,1}(q) = \frac{1}{2} k_a e^T(q) e(q) = \frac{1}{2} k_a \|e(q)\|^2 \tag{4.2}$$

$$U_{a,2}(q) = \frac{1}{2} k_a \|e(q)\| \tag{4.3}$$

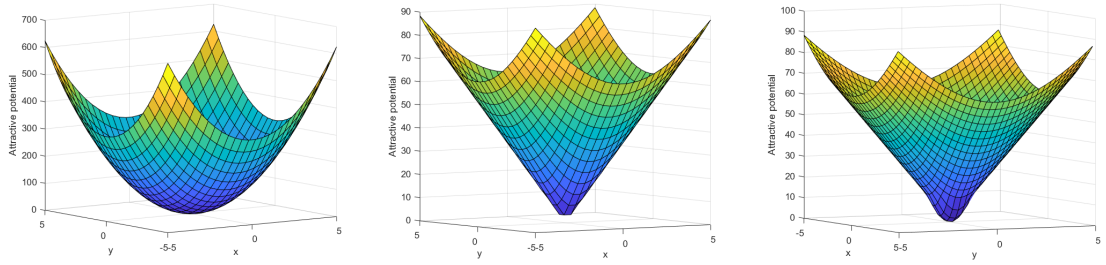
where k_a is an constant reflecting the strength of the attraction. The resulting attractive forces are given in Eq.(4.4) and Eq.(4.5) for paraboloidal and conical shape respectively.

Force fields

$$f_{a,1}(q) = -\nabla U_{a1}(q) = k_a e(q) \tag{4.4}$$

$$f_{a,2}(q) = -\nabla U_{a2}(q) = k_a \frac{e(q)}{\|e(q)\|} \tag{4.5}$$

The advantage of the paraboloidal potential is its continuous differentiability, however when the controlled point is far away of the target, the field strength can become large. Concerning the conical potential, the attraction is steady for far away points but the potential function is not differentiable on the goal point. A combination of these two attractive functions is proposed in [24] to take the best of the two methods. When $\|e(q)\| > 1$ the conical potential is used and changes to the paraboloidal one when $\|e(q)\| \leqslant 1$ in order to guaranty a smooth transition. The potentials are illustrated in Fig. 4.1.



(a) Paraboloidal attractive potential. **(b)** Conical attractive potential. **(c)** Combined attractive potential.

Figure 4.1: Illustration of the different types of attractive potentials.

The attractive force field is similar to the trajectory tracking task Eq.(3.4) if the centre of the potential (the goal) is moving. In that case, the velocity of the goal must be taken into account.

4.2 Avoidance Fields

There exist several ways to express avoidance fields. A first way is basically to consider the obstacle as a repulsive force that will repel the robot. However other methods have emerged to guarantee the obstacle avoidance such as the use of vortex fields rather than repelling fields. In the following section we test several avoidance approaches. The test performed for each of the proposed avoidance methods considers a goal attractor system with the starting point and goal on either side of the obstacle. The parameters for these tests are displayed in the table 4.1.

Parameters	Value
Starting point	$[0.5, 1]^T$
Goal	$[0.5, 1]^T$
Obstacle centre	$[1.25, 0.75]^T$
Obstacle radius	0.5

Table 4.1: Parameters for the obstacle avoidance methods test.

The obstacles considered are defined as included in an ellipsoid of equation:

$$h(p) = \left(\frac{p_x - x_h}{a}\right)^{2m} + \left(\frac{p_y - y_h}{b}\right)^{2m} + \left(\frac{p_z - z_h}{c}\right)^{2m} \tag{4.6}$$

with $p = (p_x, p_y, p_z)$ being a point on the obstacle surface and $p_o = (x_h, y_h, z_h)$ being the centre of the ellipsoid. The following avoidance approaches use the distance between the end-effector and the obstacle in order to design the avoidance potentials. This distance is then defined as the shortest path between the end-effector and the obstacle surface from Eq.(4.6).

4.2.1 Repulsive Field

The field force produced by repulsive potential repels the controlled point from the obstacle. The potential function depends on the distance between the controlled point and the obstacle and prevents them to collide since the force field tends toward an infinite value in the vicinity of the obstacle. The common repulsive field expression is developed in [24] and used in [12]. The potential is expressed in Eq.(4.7) and the corresponding force field in Eq.(4.8).

Potential

$$U_{r,i}(q) = \begin{cases} \frac{k_{r,i}}{\gamma}\left(\frac{1}{\eta_i(q)} - \frac{1}{\eta_{0,i}}\right)^{\gamma} & \text{if } \eta_i(q) \leqslant \eta_{0,i} \\ 0 & \text{if } \eta_i(q) \geqslant \eta_{0,i} \end{cases} \tag{4.7}$$

Force field

$$f_{r,i}(q) = \begin{cases} \frac{k_{r,i}}{\eta_i^2(q)}\left(\frac{1}{\eta_i(q)} - \frac{1}{\eta_{0,i}}\right)^{\gamma-1} \nabla\eta_i(q) & \text{if } \eta_i(q) \leqslant \eta_{0,i} \\ 0 & \text{if } \eta_i(q) \geqslant \eta_{0,i} \end{cases} \tag{4.8}$$

with $\eta_i(q) = min_{q' \in CO^i} \|q - q'\|$ being the distance from q to the closest point of the obstacle i expressed is the configuration space and $\eta_{0,i}$ the range of influence of the obstacle. If expressed in the workspace, the field equation is formulated with p and η_i becomes the distance between the end-effector and the obstacle surface and $\gamma = 2$. The results for a simple study case of a goal attraction with an obstacle are shown on the figures 4.2 and 4.3.

The main drawback of the repulsive force field is related to stability. Since it acts as a source that repels the end-effector a time-step not small enough can lead to an oscillating behaviour as shown on the figure 4.4.

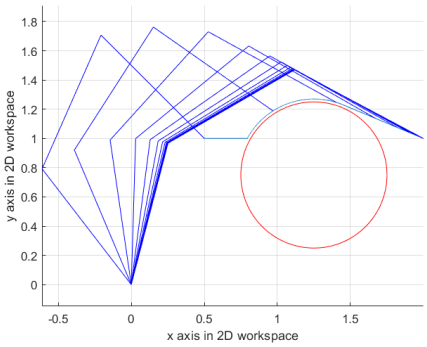


Figure 4.2: Three links planar arm configurations in time.

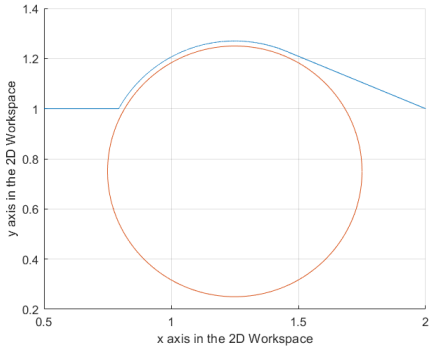


Figure 4.3: Trajectory generated using the repulsive field method.

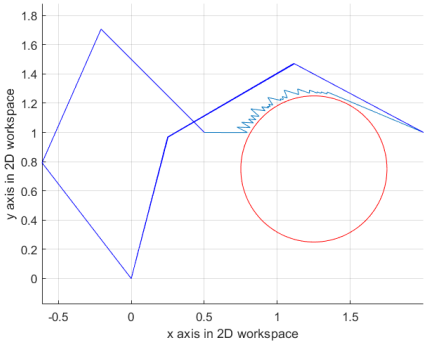


Figure 4.4: Three links planar arm motion in the workspace for a repelling field with a bigger time step.

4.2.2 Vortex field

Another inconvenience with the repulsive force field applied on the end-effector is that it could lead to a local minima problem. In order to avoid this scenario, another kind of fields can be considered namely the vortex force field.

The aim is not to repel but to force the end-effector to go along the obstacle surface as far as it's necessary. Two ways of building those vortexes will be detailed in this section.

Force field

The first avoidance field expressed as a vortex is detailed in [15] in a 2D workspace.

$$\begin{cases} f_{ox} = p_y \left(x - x_o\right)\left(y - y_o\right) - p_x \left(y - y_o\right)^2 \\ f_{oy} = p_x \left(x - x_o\right)\left(y - y_o\right) - p_y \left(x - x_o\right)^2 \end{cases} \tag{4.9}$$

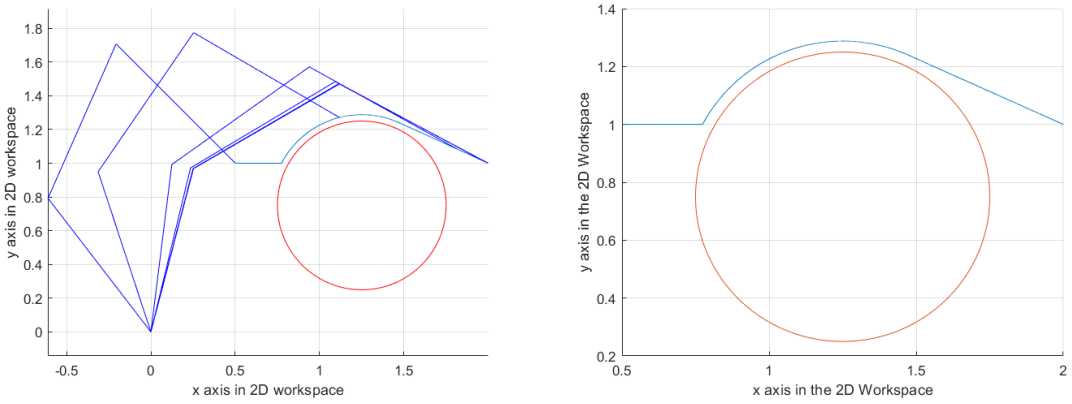
with $p_x \triangleq \cos\Phi$, $p_y \triangleq \sin\Phi$ being the components of the direction vector for the line linking the centre of the obstacle to the goal, computed with the direction angle $\Phi \triangleq \text{atan2}(y_g - y_o, x_g - x_o)$. More generally, the family of force field used in [15] is expressed as:

$$f_r = \lambda(p^T r)r - p(r^T r) \tag{4.10}$$

where λ is a constant that defines the shape of the field ($\lambda = 0$ for a uniform field, $\lambda = 1$ for a vortex field and $\lambda = 2$ for an attractive field) and r is a relative position vector which depends on the aim of the field (defined by λ). For an avoidance field, this position r is taken relatively to the obstacle position $r = p_e - p_o$.

In this case, the resulting force field is weighted between the attractive field and the avoidance field depending on the position of the end-effector with a factor σ_i (σ_i=1 when the obstacle is not a threat and σ_i=0 when it is). A generalised expression for n obstacles can be expressed as follows [15]:

$$f_{total} = \prod_{i=1}^{N} \sigma_i f_a + \sum_{i=1}^{N} (1 - \sigma_i) f_{o,i} \tag{4.11}$$



(a) **(b)**

Figure 4.5: **(a)** Three links planar arm configurations in time. **(b)** Trajectory generated using the vortex field method in Eq.(4.9).

For this avoidance approach, a the time-step which is not small enough does not lead to an undesirable oscillating behaviour as shown on the figure 4.6.

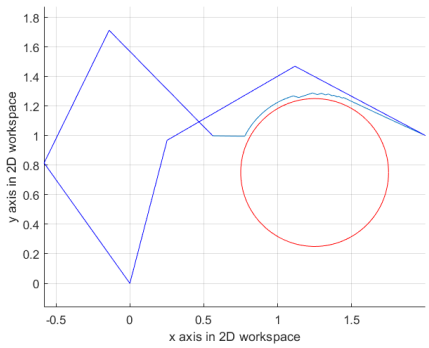


Figure 4.6: Three links planar arm motion in the workspace for a vortex field with a bigger time step.

The obstacle avoidance for the whole robot is a superposition of several potentials expressed for each additional control point as in Eq.(4.12) [24].

$$f_o = \sum_{i=1}^{n_{ctrl}} f_{r,i}(p_i) \tag{4.12}$$

where f_o is the global avoidance field for one obstacle, n_{ctrl} are the number of control points taken and $f_{r,i}(p_i)$ is the avoidance field computed for the point p_i.

4.2.3 Avoidance field in dynamical systems and transferring into artificial force field

There are several methods designed to achieve an obstacle avoidance task with dynamical systems. Among those methods, two kinds can be identified. The first one is detailed in [11], it consists in the computation of a modulation matrix M_O. This modulation matrix is then used to change the force field in which the controlled point is evolving. The final field is then a product of this modulation matrix and the force field without the obstacle: $F_{final} = M_O F_{no-obstacles}$. This method will not be further examined since the perturbation brought by the obstacles is applied through a product on the global field instead of being an external perturbation force.

On the contrary, the second kind takes the obstacle into account through an additional perturbation term in the dynamical system. Two methods will be analysed. The first avoidance field expressed in a dynamical system is detailed in [17] and [16]. It is called dynamic potential field and consists in a combination of repulsive and vortex fields [17]. The field takes the velocity and the position of the controlled point (the end-effector in the common case) into account to activate or deactivate the force field. The potential has the following shape:

$$U_o = \begin{cases} \lambda(-\cos\theta)^{\beta} \frac{\|v\|}{p(x)} & : \frac{\pi}{2} < \theta \leqslant \pi \\ 0 & : 0 \leqslant \theta \leqslant \frac{\pi}{2} \end{cases} \tag{4.13}$$

with λ being a constant representing the strength of the field, β a constant set at 2 and θ the angle between the position vector and the velocity vector. The cosine of that angle is computed as: $\cos\theta = \frac{v^T x}{\|v\| p(x)}$. The gradient of this potential is given in Eq.(4.14) and the corresponding field force is given in Eq.(4.15).

$$\psi(x, v) = \lambda(-\cos\theta)^{\beta-1} \frac{\|v\|}{p(x)} \left(\beta \nabla_x \cos\theta - \frac{\cos\theta}{p(x)} \nabla_x p(x) \right) \tag{4.14}$$

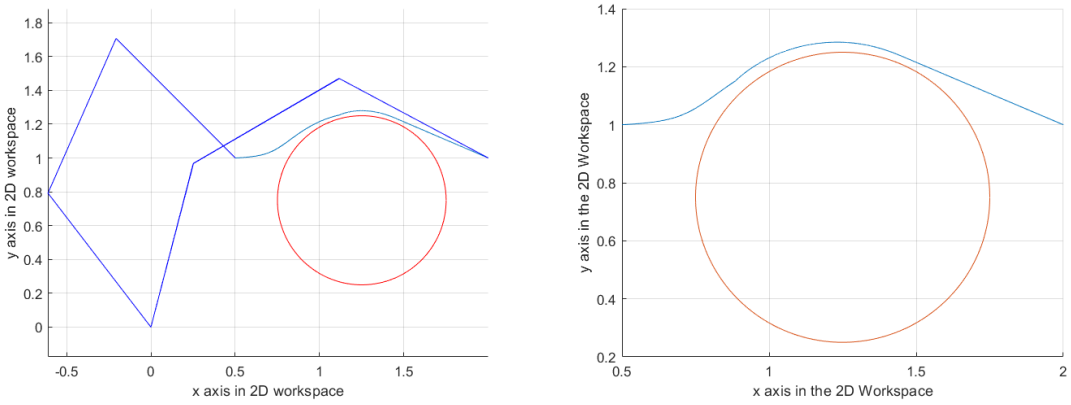
$$f_o = \begin{cases} \lambda(-\cos\theta)^{\beta-1} \frac{\|v\|}{p(x)} \left(\beta \nabla_x \cos\theta - \frac{\cos\theta}{p(x)} \nabla_x p(x) \right) & \text{if } \pi/2 < \theta \leqslant \pi \\ 0 & \text{else} \end{cases} \tag{4.15}$$

The avoidance force (4.15) is used as a perturbation term in the following dynamical system which corresponds to the transformation system of a modified Dynamical

Movement Primitives[1]:

$$\begin{cases} \tau\dot{v} = K(g - x) - Dv - K(g - x_0)s + Kf(s) + \psi(x, v) \\ \tau\dot{x} = v \end{cases} \tag{4.16}$$

The results of the simulation for the case summed up in table 4.1 are displayed in the figure 4.7.



(a) **(b)**

Figure 4.7: **(a)** Three links planar arm configurations in time. **(b)** Trajectory generated using the vortex field method in [17]

The second method proposing obstacle avoidance as a perturbation of a dynamical system is detailed in [26]. This method is only based on the distances between the elements composing the workspace: the end-effector (the controlled point), the starting point, the obstacle and finally the goal. In this case, the dynamical system proposed is the following one:

$$\begin{cases} \dot{p} = v \\ \dot{v} = -\alpha v - \beta(p - p_t) + u_h(t) \end{cases} \tag{4.17}$$

where p is the position, v the velocity, p_t the goal and $u_h(t)$ the obstacle avoidance contribution similarly to f_o in Eq.(4.15). The obstacle avoidance contribution is defined in a way that guarantee the avoidance thanks to a logarithmic transformation.

The avoidance contribution u_h is computed using the deviation between the distance from the end-effector to the centre of the obstacle being $d(p) = \|p - p_h\|$ and a new attractor scalar parameter $d_m(p)$. To do so, several distances are defined: the distance (end-effector - obstacle centre) $d(p)$, the distance (initial position - obstacle) d_o and the distance (goal - obstacle d_T). The attractor scalar parameter $d_m(p)$ is defined as the distance between the obstacle surface and the goal. And in turn, the deviation from this attractor is defined as $\sigma(t) = d(p) - d_m(p)$. Then this new distance is bounded between d_{min} and d_{max} which represent the lower and

[1] cfr. Chapter 5

upper bound distances between the obstacle and the target position. To ensure the avoidance of the obstacle, a natural logarithmic transformation is applied on $\sigma(t)$[2]:

$$T(\sigma(t)) = \ln\left(\frac{1+\frac{\sigma}{d_{min}}}{1-\frac{\sigma}{d_{max}}}\right)$$

Finally the avoidance term $u_h(t)$ is defined with:

$$u_h = -k_h \frac{\partial T}{\partial \sigma} T(\sigma(t))\xi \tag{4.18}$$

with ξ being:

$$\xi = (1 - f(p))\frac{p - p_h}{\|p - p_h\|} + \frac{f(p)}{\cos\phi}\frac{n_h(p)}{\|n_h(p)\|}$$

where $n_h(p)$ is the normal vector to the obstacle surface Eq.(4.6), $f(p) = h(p)^{\frac{1}{2m}}$ and $\cos\phi = \frac{(p-p_h)^T n_h(p)}{\|p-p_h\|\|n_h(p)\|}$. The test was performed with the parameters displayed in the table 4.2 for the dynamical system and the potential.

Parameters	Value
α	$5\sqrt{\beta}$
β	50
μ	0.5
k_h	0.1

Table 4.2: Parameters for the dynamical system (4.17) with the potential u_h from (4.18).

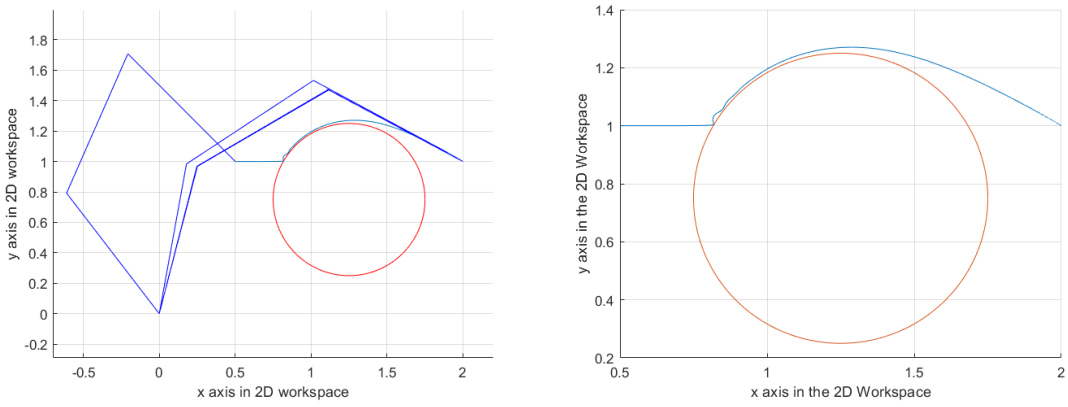


Figure 4.8: Obstacle avoidance realised with the potential developed in Eq.(4.18).

Those two avoidance methods stemmed from dynamical system implementation can be used as collision avoidance potential for goal attraction task only. If the goal is mobile and follows a pre-defined trajectory and that the obstacle interferes with this goal trajectory, then the conditions for activating these potentials do not make sense

anymore and the avoidance will not be guaranteed. This behaviour is explained by the nature of these avoidance terms. They are initially acting in the differential equations system which generates the end- effector trajectory going from a point to the goal. This implies that they were designed to impact this trajectory generation to make it deviates from the obstacle.

4.3 Avoidance Field Proposition

As previously expressed, the HQP method by respecting the lexicographic order can lead the robot in a meta-stable position like in the case of a concave obstacle (Figure 3.13). However, it is possible to avoid those singular configurations by combining several potentials acting at different priority levels. In order to clarify the upcoming reasoning, a quick introduction to the fluid flows theory is proposed.

4.3.1 The theory of potentials in fluid mechanics

An analogy between the artificial potentials and fluid flows is straightforward. The theory of incompressible irrotational[2] flows for an inviscid fluid[3] in two dimensions shows that the fluid motion (and so the velocity field) can be expressed using potential functions. Every function ϕ that respects the following conditions can be considered as a velocity potential:

- The flow is irrotational $\nabla \times \overrightarrow{V} = 0$,
- The conservation of mass leads to $\nabla^2 \phi = 0$.

In addition with potential functions, stream functions ψ can also be introduced. Stream functions are defined such as the stream lines are constantly parallel to the velocity vectors. Those functions are used to represent the stream lines and then visualise the fluid motion and intensity. The velocity (in two dimensional Cartesian space) is then obtained as follows:

$$\overrightarrow{V} = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \frac{\partial \phi}{\partial x} \\ \frac{\partial \phi}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{\partial \psi}{\partial y} \\ -\frac{\partial \psi}{\partial x} \end{bmatrix} \tag{4.19}$$

Regarding the properties of the potential functions, the superposition of two valid potentials is a valid potential (superposition principle).

As an example, combining a dipole potential[4] with an uniform field potential is equivalent as having a circular obstacle in a uniform flow. This superposition is illustrated in the figures 4.9, 4.10 and 4.11.
An interesting stream function for the concave issue is the irrotational vortex stream function:

$$\psi = -K_\psi \ln r \tag{4.20}$$

[2]Fluid flow where the particles do not rotate, leading to a cross gradient of the velocity being null: $\nabla \times \overrightarrow{V} = 0$

[3]Fluid where the frictions are neglected.

[4]A dipole is the name given to the association of a source potential and a sink potential close to each other.

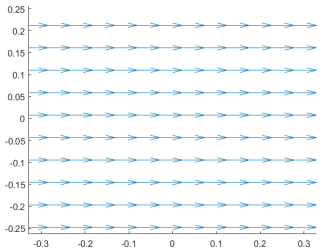


Figure 4.9: Uniform field $\phi_{uni} = Ux$

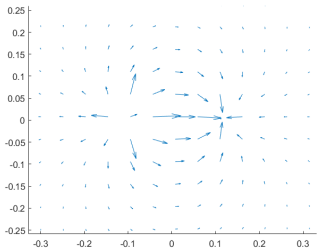


Figure 4.10: Dipole field $\phi_{dipole} = \frac{1}{2} m \frac{\ln\left((x+a)^2+y^2\right)}{\ln\left((x-a)^2+y^2\right)}$

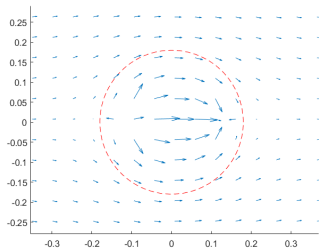


Figure 4.11: Combined field $\phi_{uni} = \phi_{uni} + \phi_{dipole}$

where K_ψ is constant[5] and r is the distance between a point in the flow p and the centre of the vortex p_o: $r = \sqrt{(p-p_o)^T(p-p_o)}$. An illustration of this potential is presented in the figure 4.12.

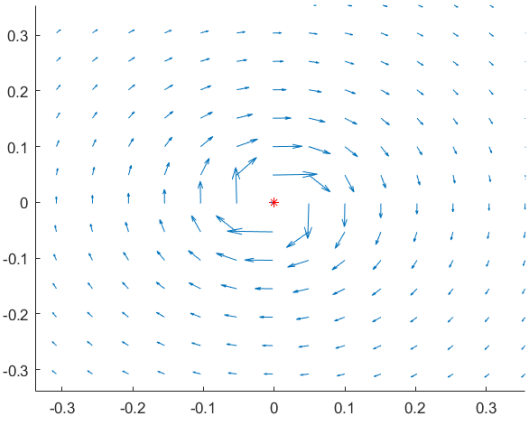


Figure 4.12: Illustration of a vortex field: $\psi = -K_\psi \ln r$ centred in (0,0).

In fluid mechanics, the theory of potential flows is detailed in two dimensions to respect the assumptions made [13], however it is necessary to find a way to adapt this theory and in particular the free-vortex stream function in a three dimensions space.

4.3.2 Transferring the two-dimensional stream function in the artificial potential task function

The free-vortex stream function can be used on condition that it is built in the plan containing the controlled point, the centre of the obstacle and the goal. Instead of adapting this vortex function in three dimensions, the proposed method is to keep the stream in a plan but make that plan varying in space with the controlled point. This stream is then applied on this adaptive plan. The following geometrical method was applied to compute the stream function and transfer it back in the

[5] $K_\psi = \frac{\Gamma}{2\pi}$ with Γ being the fluid flow strength.

global reference frame.

First, the three positions in the global reference frame $\mathcal{R}_0$ must be known: the controlled point (in this case, the end-effector coordinates) $p_{e|_0}$, the obstacle $p_{ob|_0}$, the goal $p_{g|_0}$. Then, a reference frame specific to the stream, r_{str}, is built regarding to the relative positions. Considering $\overrightarrow{p}_{eo} =< p_{ob|_0} - p_{e|_0} >$ and $\overrightarrow{p}_{og} =< p_{g|_0} - p_{ob|_0} >$ being the relative position vectors defining the stream plan.

The specific reference frame $\mathcal{R}_{str}$ is defined as follows:

$$\mathcal{R}_{str} = \begin{bmatrix} \overrightarrow{x}_{str} \\ \overrightarrow{y}_{str} \\ \overrightarrow{z}_{str} \end{bmatrix} = \begin{bmatrix} \frac{\overrightarrow{p}_{eo}}{\|\overrightarrow{p}_{eo}\|} \\ \overrightarrow{z}_{str} \times \overrightarrow{x}_{str} \\ \frac{\overrightarrow{x}_{str} \times \overrightarrow{p}_{og}}{\|\overrightarrow{x}_{str} \times \overrightarrow{p}_{og}\|} \end{bmatrix} \tag{4.21}$$

An illustration of this new reference frame computation is proposed on the figure 4.13.

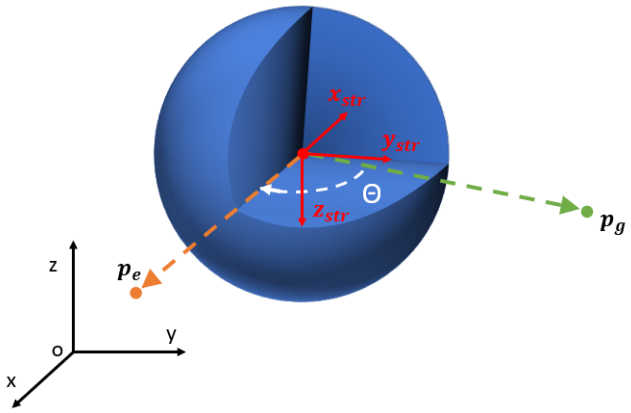


Figure 4.13: Illustration of the stream frame definition

Once the coordinates of the specific frame are known, the homogeneous transformation matrix from the stream frame $\mathcal{R}_{str}$ to the global frame $\mathcal{R}_0$ and the corresponding rotation matrix are:

$$T_0^{str} = \begin{bmatrix} R_0^{str} & \overrightarrow{p}_{ob|_0} \\ 0 & 1 \end{bmatrix}, \quad \text{with } R_0^{str} = \begin{bmatrix} \overrightarrow{x}_{str} & \overrightarrow{y}_{str} & \overrightarrow{z}_{str} \end{bmatrix} \tag{4.22}$$

The controlled point can then be expressed in the stream frame using the inverse of the transformation matrix (4.22). From this point, the stream function is built with the controlled point in $\mathcal{R}_{str}$: $p_{e|_{str}}$ and the stream force is:

$$\begin{cases} f_{x_{str}} = \frac{K_s}{\|p_{e|_{str}}\|^2} p_{e|_{str}}(x) \\ f_{y_{str}} = \frac{K_s}{\|p_{e|_{str}}\|^2} p_{e|_{str}}(y) \\ f_{z_{str}} = 0 \end{cases} \tag{4.23}$$

Now that the stream force is computed in the stream frame, it can be expressed back in the global frame to derive the obstacle avoidance contribution $f_{o_{vortex}}$:

$$f_{o_{vortex}} = R_0^{str} \begin{bmatrix} f_{y_{str}} \\ -f_{x_{str}} \\ 0 \end{bmatrix} \tag{4.24}$$

The figures 4.14 and 4.15 show the change of rotation for the vortex flow which depends on the orientation of the z_{str} vector.

Now that the potential is built, it is necessary to design its activation condition. The figures 4.16, 4.17, 4.18 and 4.19 illustrate the different cases depending on the distance between obstacle and controlled point and on the angle $\widehat{p_e p_{ob} p_g}$.

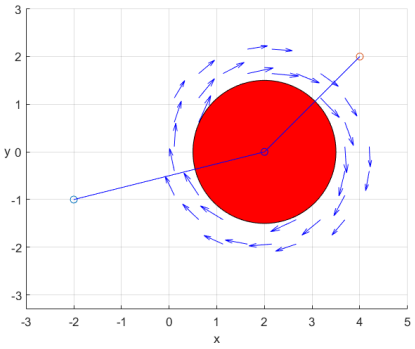


Figure 4.14: 2D application of the adaptive stream: when z_0 and z_{str} are aligned.

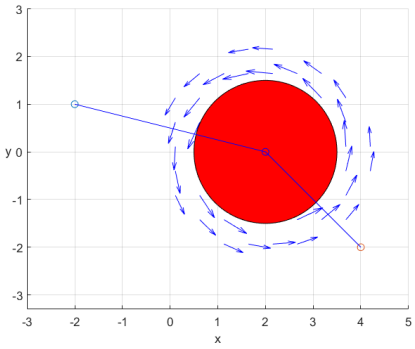


Figure 4.15: 2D application of the adaptive stream: when z_0 and z_{str} are opposed.

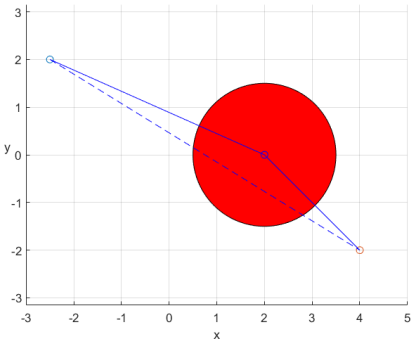


Figure 4.16: The obstacle does not represent a danger for the controlled point, the stream function is not activated.

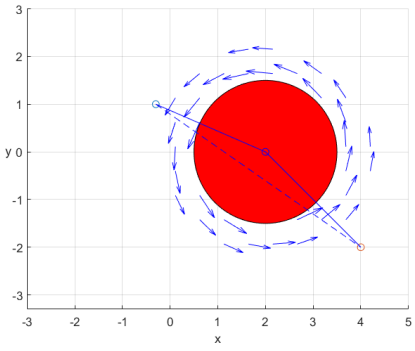


Figure 4.17: The controlled point entered the danger area, the stream function is then applied.

This method allows to avoid the singular configuration due to a concave obstacle. In two dimensions, the concave obstacle is created by the fusion of two circles. If such fusion occurs the streams are synchronised to prevent the streams to act against

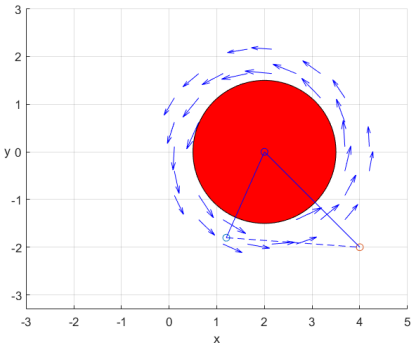


Figure 4.18: The stream function remains active as long as the angle $\widehat{p_e p_{ob} p_g} > \pi/4$.

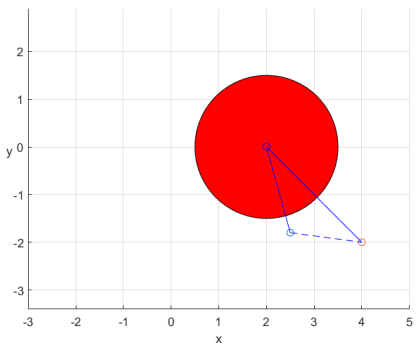


Figure 4.19: The angle $\widehat{p_e p_{ob} p_g}$ is small enough, the stream is not needed anymore.

each other as it is shown on figure 4.20.

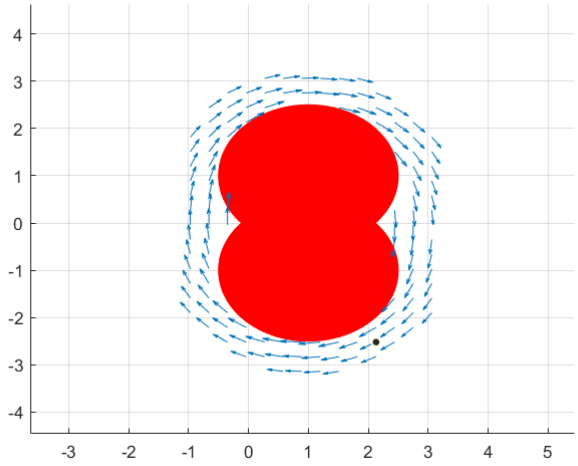


Figure 4.20: Global vortex field for a concave obstacle made of two circles.

Nevertheless, there is also a singular configuration for the presented method. If the three points used to create the adaptive plan are aligned, there is an infinity of plans available. It is then necessary to anticipate this situation.

When the points (p_e, p_{ob}, p_g) are aligned, the controlled point and the goal are on either side of the obstacle. It implies that the orientation of the adaptive plan can be set in any direction as long as the vector $< p_{ob} - p_e >$ is contained in the plan.

4.3.3 Generalisation of the adaptive plan stream for unknown p_{ob}

In the case of an obstacle detection sensor such as a lidar, the centre of the obstacle is unknown, only distance information is available. The method is still working if instead of the obstacle centre position p_{ob}, the position of a point on the obstacle surface is considered as the stream centre. In this case, the stream plan will be fixed to avoid the perpetual change of its direction since the singular configuration (points alignment) is more likely to occur.

4.4 HQP and Artificial Potential Field Tasks

This section considers artificial potential avoidance methods as a level of the stack of tasks in HQP.
The stack structure for the following tests is shown in table 4.3. The aim is to check if the obstacle avoidance potentials expressed in the literature are compatible with the stack of tasks architecture and if it can guarantee better results than the collision avoidance in Eq.(3.11). The test conditions are the same as for the potentials demonstration and are summarised in table 4.1.

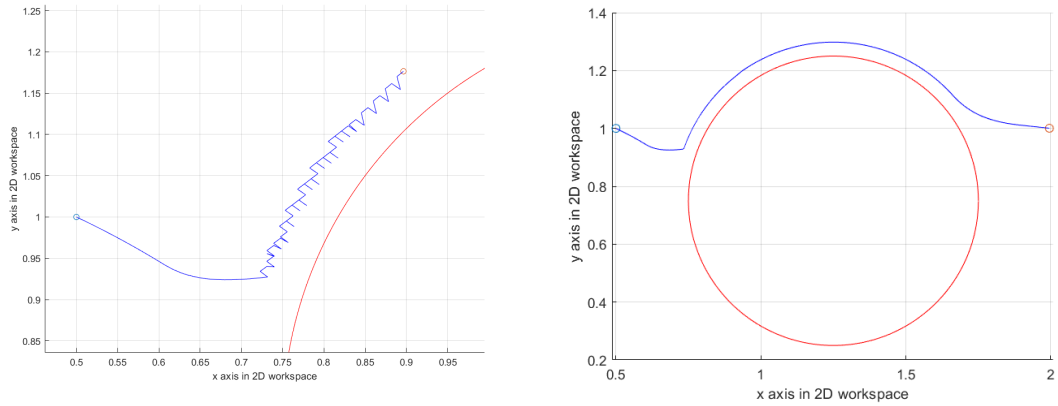
Priority level	**Name**	**Comments**	**Dimensions**	**Type**
1	Safety bounds	Impose the physical bounds of the robot.	n_{dof}	Inequalities
2	Obstacle avoidance field	Implementation of the avoidance potentials.	$n_{dimensions}$	Equalities
⋮	Other	⋮	⋮	⋮
last	Trajectory tracking	/	$n_{dimensions}$	Equalities

Table 4.3: Stack Structure for the obstacle avoidance method made with artificial potential methods.

The first avoidance method presented was the repulsive field. However this cannot be considered as an individual stack level because of the notion of hierarchy in HQP stack of tasks. Indeed, if the task is included to the stack as in table 4.3, the repulsive force will be solved in priority, and thus the end effector will be repelled until it is not in the repulsive field activation area. This situation will create an oscillating motion of the end effector around the activation area border, which is not a desired behaviour. Fig.4.21 illustrates that antagonistic behaviour.

The results of the test for the vortex field defined in Eq.(4.10) are displayed in the figure 4.22. The results of the test made with the vortex field used with dynamical system, defined in Eq.(4.15), are shown in Fig.4.24. Finally Fig.4.23 shows the results of the avoidance term Eq.(4.18) used in dynamical system.

A common observation with the stack structure proposed in table 4.3 is the oscillating velocity curve shape caused by the activation of the potential based tasks level, when the end-effector enters the trigger area. An illustration of this behaviour is displayed in the figure 4.25. By adding a acceleration bounds task (Eq.(3.7)) in top priority level, the oscillations in the generated velocity can be prevented. The results of this new bounds level are displayed in the figure 4.26.



(a) **(b)**

Figure 4.21: Obstacle avoidance realised with HQP and the potential developed in Eq.(4.8). **(a)** The repelling force is put in a higher priority level. **(b)** The action is combined in the same level (it correspond the fusion of level 2 and *last* of table 4.3.

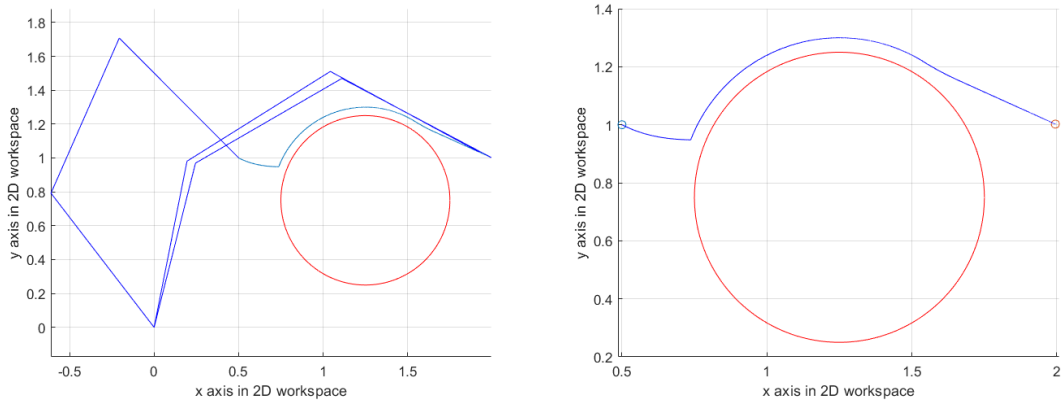


Figure 4.22: Obstacle avoidance realised with HQP and the potential developed in Eq.(4.9).

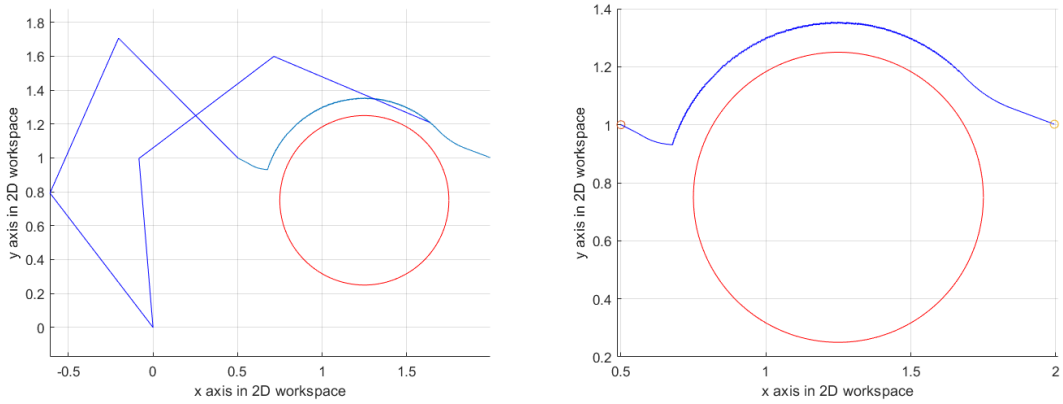


Figure 4.23: Obstacle avoidance realised with HQP and the potential developed in Eq.(4.18).

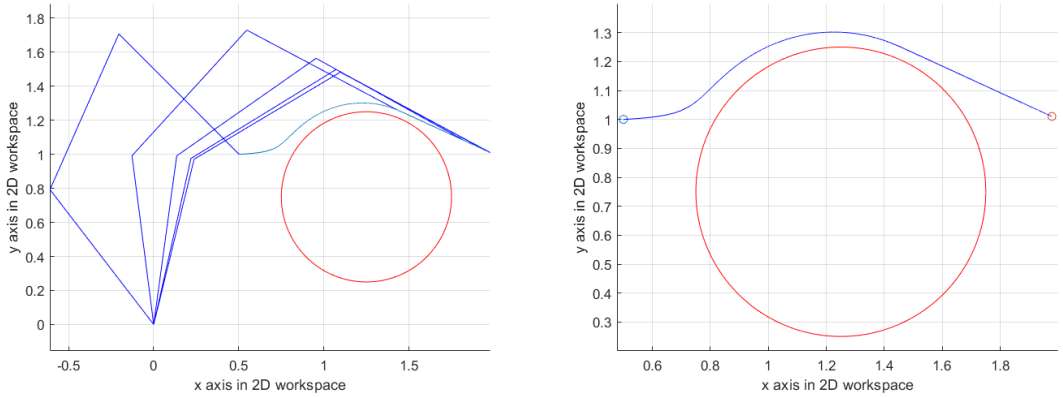


Figure 4.24: Obstacle avoidance realised with HQP and the potential developed in Eq.(4.15).

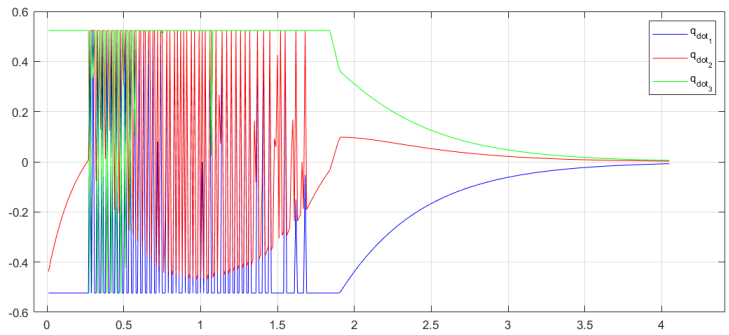


Figure 4.25: Illustration of the velocity oscillation phenomenon caused by the fluctuation activation of the potentials.

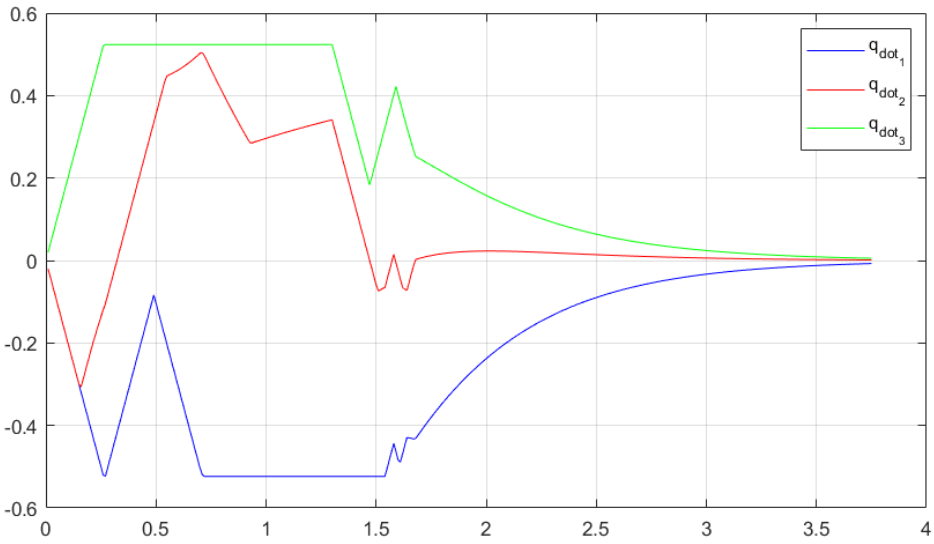


Figure 4.26: Illustration of the reduction of velocity oscillation phenomenon by adding a acceleration bounds in top priority level.

4.5 HQP and Proposed Avoidance Field

Combining the concept of hierarchy from HQP and the stream function proposed in Eq.(4.24), the concave issue addressed in the figure 3.13 can be avoided. The stack structure is proposed in the table 4.4.

A simulation of a concave obstacle avoidance with HQP and the proposed stack structure is shown in Fig.4.27 for a two dimensions implementation.

A goal attraction task in three dimension for a UR10 robot is shown in Fig.4.28. The tracking of a linear trajectory is performed in Fig.4.29. We can see that the avoidance is correctly carried out however, there are jumps in the joints velocity command generated by the HQP kinematic controller. As in the previous tests conducted for the other potentials, the velocity oscillations/jumps remain an issue. To address this problem, the acceleration bounds Eq.(3.7) has been added in the stack as the highest priority task. The results for the augmented stack are shown in the figure 4.30.

Priority level	**Name**	**Comments**	**Dimensions**	**Type**
1	Safety bounds	Impose the physical bounds of the robot.	n_{dof}	Inequalities
2	Collision avoidance	Prevent the parts of the robot to collide with obstacles.	$n_{links}n_{obstacles}$	Inequalities
3	Obstacle avoidance	Implementation of the stream function.	$n_{dimensions}$	Equalities
⋮	Other	⋮	⋮	⋮
last	Trajectory tracking	/	$n_{dimensions}$	Equalities

Table 4.4: Stack Structure for the obstacle avoidance method including the adaptive stream method.

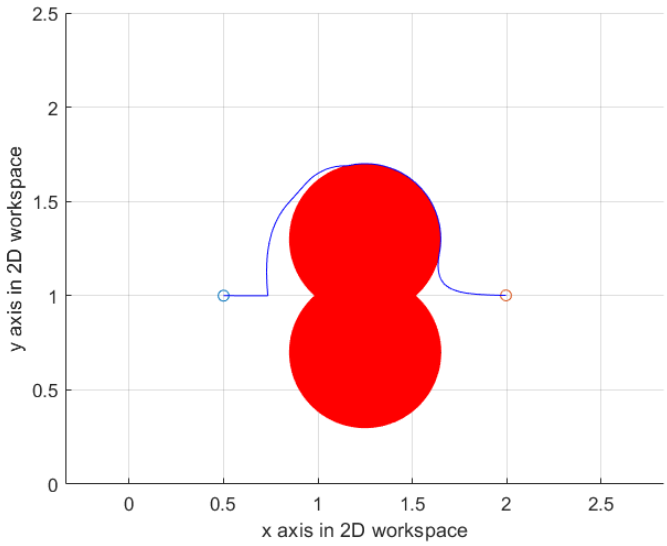


Figure 4.27: Concave obstacle avoidance with stacked potentials

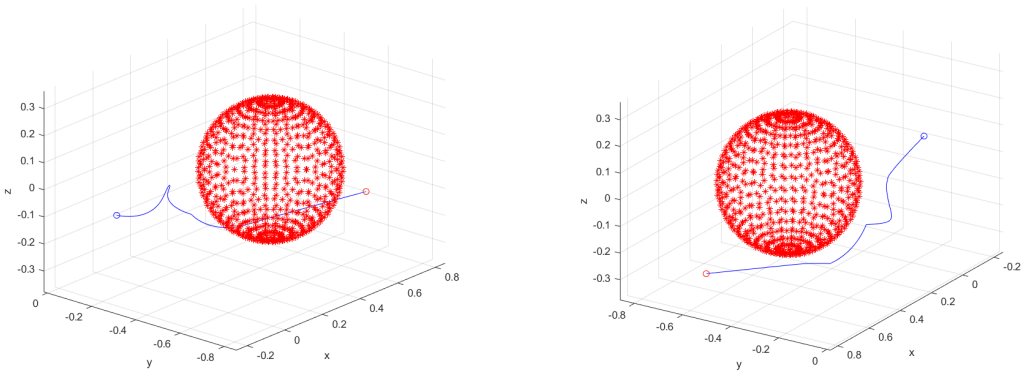


Figure 4.28: Obstacle avoidance realised with HQP and the stack in 4.4 for a goal attractor task.

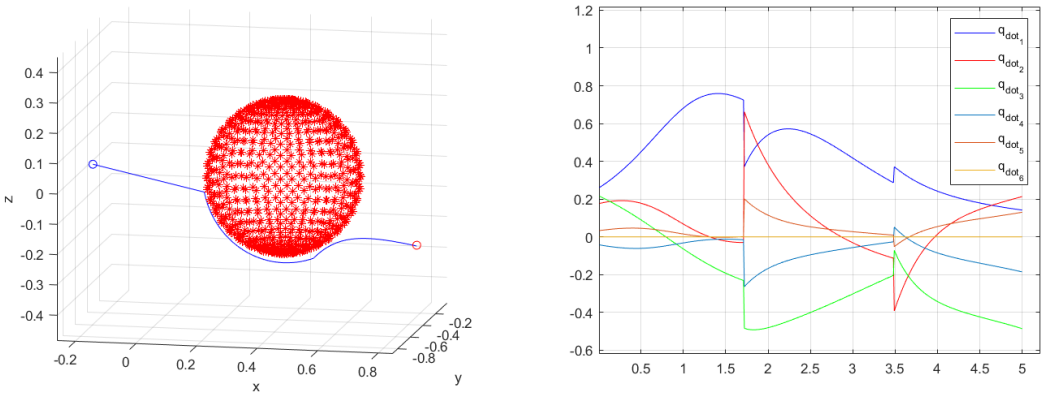
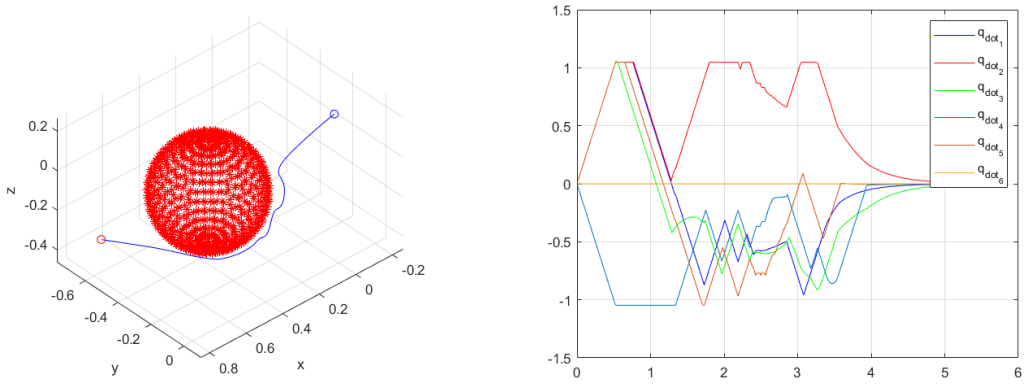


Figure 4.29: Obstacle avoidance realised with HQP and the stack in 4.4 for a linear trajectory.



(a) Trajectory in space. **(b)** Joint velocities.

Figure 4.30: Obstacle avoidance realised with HQP and the stack in 4.4 for a goal attractor task with an acceleration bounds level.

5

Trajectory Generation

The desired features for a trajectory generation method are flexibility and reactivity. Thus, the generation is performed in real time using Dynamical Movement Primitives (DMP). The DMP are used in order to encode a trajectory into a non-linear dynamical system and adapt it in new conditions. This dynamical system is then built through the learning of a known behaviour called demonstration [21]. In this thesis we consider point-to-point trajectories which can be generated from a point attractor system. A simple example of point attractor dynamical system is the string-damper.

5.1 Dynamical Movement Primitives Formulations

Two dynamical systems composed a Dynamical Movement Primitive: the transformation system and the canonical system. The transformation system is a second order differential equation split in two first order differential equations that represent the motion of the end-effector:

$$\begin{cases} \tau \dot{z} = \alpha_z \left(\beta_z (g - y) - z \right) + f \\ \tau \dot{y} = z \end{cases} \tag{5.1}$$

where g is the goal, α_z and β_z are positive constants of the systems. Constant β_z is set equal to $\alpha_z/4$ in order to have a critically damped system which allows a exponential convergence to the goal without oscillations [8], y and z are the system parameters where y represents the position and f are the forcing terms. The role of the forcing terms is to allow the transformation system to have a desired behaviour in time and space. As detailed in [23], for vanishing forcing term ($f = 0$), the system converges asymptotically to the goal. By varying the forcing terms in time and space, the transformation system can be adjusted and so have a more complex behaviour before reaching the goal. To make this possible, another dynamical system is defined, the canonical system. The canonical system is a simple asymptotically convergent system:

$$\tau \dot{x} = -\alpha_x x \tag{5.2}$$

where τ is the timescale parameter and x is the phase variable and allows to modify the temporal evolution of the dynamical system. The expression of the forcing term depends on the formulation of the DMP. For the classical attractor system in Eq.(5.1) it is given by:

$$f(x) = \frac{\sum_i^N w_i \Phi_i(x)}{\sum_i^N \Phi_i(x)} x(g - y_0), \tag{5.3}$$

with g the goal and y_0 the starting point of the trajectory and the Φ_i are N Gaussian basis functions depending on x expressed as:

$$\Phi_i = \exp\left(-\frac{1}{\sigma_i^2}(x - c_i)\right) \tag{5.4}$$

with c_i the mean and σ_i the standard deviation. The use of the canonical system allows the transformation system to be time independent since the forcing terms equation is expressed in x.

There are different formulations of the dynamic movement primitives where the "basic" system formulation is augmented in order to highlight or even to allow a certain behaviour.

In [18], another way to represent the DMP system is introduced. This formulation is similar to Eq.(5.1) except that the forcing terms are separated from the goal as follows:

$$\begin{cases} \tau\dot{v} = K(g - y) - Dv + (g - y_0)\tilde{f} \\ \tau\dot{y} = v \end{cases} \tag{5.5}$$

where v is the scaled velocity of the system, y is the position. Note that the phase variable is still x while $\tilde{f}$ is an expression of the forcing terms independent of the goal:

$$\tilde{f}(x) = \frac{\sum_i^N w_i \Phi_i(x)}{\sum_i^N \Phi_i(x)} x \tag{5.6}$$

This expression of the transformation system (5.5) has several drawbacks as shown in [18] and [7]. First, if the starting point (y_0) and the goal (g) are the same, no movement will be generated, then the computation of the forcing terms is unstable for close y_0 and g and finally the change of goal from g_{init} and g_{new} can introduce strong modifications in the generated trajectory since it multiplies $f(x)$. To handle those issues, a modified transformation system is introduced in [18]:

$$\begin{cases} \tau\dot{v} = K(g - y) - Dv - K(g - y_0)x + K\tilde{f}(x) \\ \tau\dot{y} = v \end{cases} \tag{5.7}$$

Pastor *et al.* proposed an alternative version of the canonical system in [22]. The augmented version is expressed as follows:

$$\tau\dot{x} = \alpha_x x \frac{1}{1 + \alpha_c(y_{actual} - y)^2} \tag{5.8}$$

where y_{actual} is the actual value of the variable and y the expected result. The effect of this change is that a large error will slow down the phase system evolution until this error is reduced. It is used when y is subject to a multitude of external perturbations.

Another formulation is presented in [19]. This approach differs from the previous ones by weighting the terms of the transformation system with the time-dependent parameter w_g.

$$\begin{cases} \tau\dot{v} = (1 - w_g)(f_w + y_0 - y) + w_g K(g - y) - Dv \\ \tau\dot{y} = v \end{cases} \tag{5.9}$$

with

$$w_g(t) = \frac{1}{2}\left(1 + \frac{2}{\pi}\int_0^t \exp\left(-\left(\frac{t-\mu}{\sigma\sqrt{2}}\right)^2\right)\right)$$

and

$$f_w(x) = \frac{\sum_i^N w_i \Phi_i(x) w_i}{\sum_i^N \Phi_i(x)}$$

This method allows a transition between the shape and the goal attractor behaviour of the system. This switch is determined with the parameters μ and σ. It is used for situations where the goal is varying in time and space.

The previous DMP systems are expressed for one degree of freedom. For systems with many degrees of freedom, a transformation system is required for each of them. [23], [28]. The phase variable x is then common for all the degrees of freedom and thus the same canonical system is employed. Since the desired use of DMP is to generate a trajectory for the end-effector in a three dimensional space, the three degrees of freedom are therefore linked over time by sharing the phase variable. In other words, the time line must be same for each of the transformation systems.

5.2 Planning with DMPs

In order to plan with DMPs and to generate trajectories, three steps are needed. The first one is to encode a desired behaviour with a time dependent trajectory. Example: In the case of human motion imitation, minimum jerk trajectories are used. The second step is to learn that behaviour. Two different learning methods are expressed in [23]: reinforcement learning and imitation learning. It is more relevant to use the imitation learning in the context of this thesis since the global objective is to be able to reproduce a demonstrated motion and to adapt it if necessary. In this case, learning means to find the weights used in the computation of the forcing terms (5.3,5.6) by defining the values of the Gaussian basis functions parameters. And finally, the last step is to generate the trajectory through the integration in time of the differential equations system (5.1 and 5.2).

The learning process is made offline with a demonstration trajectory $(t_{demo,k}, y_{demo,k})$ with $k = 1 \rightarrow P$ and P being the number of elements in the demonstrated trajectory. The demonstration velocity and acceleration are derived from the given times and positions. The objective of this first part is to find the weights necessary for the computation of the forcing terms used for new trajectory generation. The phase variable

vector for the demonstrated trajectory comes from the solving of the ordinary differential equation (5.2) which gives: $\bar{x} = \exp\left(-\frac{\alpha_x}{\tau}\bar{t}_{demo}\right)$ where $\bar{t}_{demo} = t_{demo,k=1\rightarrow P}$.

The next step is to find the forcing terms corresponding to demonstration trajectory for each degree of freedom:

$$f_{target} = \tau^2 \ddot{y}_{demo} - \alpha_z \left(\beta_z(g - y_{demo}) - \tau \dot{y}_{demo}\right) \tag{5.10}$$

where $g = y_{demo,P}$.

Once the forcing terms are calculated, the weights can be computed by minimising the criterion $J = \sum_s (f_{target}(s) - f(s))^2$ which is a linear regression problem as expressed in [22] and [28]. If the forcing terms are expressed:

$$f = \begin{bmatrix} \frac{\Phi_1(x_1)}{\sum_{i=1}^N \Phi_i} x_1 & \cdots & \frac{\Phi_1(x_1)}{\sum_{i=1}^N \Phi_i(x_1)} x_1 \\ \vdots & & \vdots \\ \frac{\Phi_1(x_P)}{\sum_{i=1}^N \Phi_i(x_P)} x_P & \cdots & \frac{\Phi_1(x_P)}{\sum_{i=1}^N \Phi_i(x_P)} x_P \end{bmatrix} (g - y_0)w = Xw \tag{5.11}$$

where $\bar{x} = [x_1, \cdots, x_P]^T$ and $w = [w_1, \cdots, w_N]^T$ (N being the number of basis functions). The weights are finally obtained with:

$$w = (X^T X)^{-1} X^T f_{target}. \tag{5.12}$$

The offline learning also include the determination of the Gaussian basis functions used in (5.11). The parameters are computed once with the final demonstration time τ.

$$\begin{aligned} \bar{\tau} &= [0 : \frac{\tau}{N} : \tau] \\ c_i &= \exp\left(-\alpha_x \frac{\bar{\tau}_i}{\tau}\right) \\ \sigma_i &= |c_i - c_{i-1}| \end{aligned} \tag{5.13}$$

Once the weights and the basis functions are known, the generation of the trajectory is made through the integration of the transformation system for each degree of freedom (5.1).

About the modified expression of the transformation system (5.7), even though the learning process to determine the weights is very similar to the first one (5.1), there are some differences. Indeed the optimal formulation of the weights w is the same as (5.12) however the X and f_{target} are not the same since the contribution of $(g - y_0)$ is external to the forcing terms. In this case, X is computed as follows:

$$X = \begin{bmatrix} \frac{\Phi_1(x_1)}{\sum_{i=1}^N \Phi_i} x_1 & \cdots & \frac{\Phi_1(x_1)}{\sum_{i=1}^N \Phi_i(x_1)} x_1 \\ \vdots & & \vdots \\ \frac{\Phi_1(x_P)}{\sum_{i=1}^N \Phi_i(x_P)} x_P & \cdots & \frac{\Phi_1(x_P)}{\sum_{i=1}^N \Phi_i(x_P)} x_P \end{bmatrix} \tag{5.14}$$

and the f_{target} are:

$$f_{target} = \frac{\tau \dot{v} + Dv}{K} - (g - x) + (g - x_0)s \tag{5.15}$$

5.3 Learning and Planning for one degree on freedom

Before using the DMP for the trajectory generation in a three dimensional space, some simulations are made with a minimum jerk demonstration trajectory in order to evaluate the relevance of the chosen parameters. This trajectory $y(t)$ comes from [28]:

$$y(t) = y_0 + (g - y_0)\left(10\left(\frac{t}{\tau}\right)^3 - 15\left(\frac{t}{\tau}\right)^4 + 6\left(\frac{t}{\tau}\right)^5\right) \tag{5.16}$$

where y_0 is the starting point, g the goal and τ the time base for the DMP. The parameters of the demonstration trajectory are $y_0 = 0$, $g = 2$ m, $\tau = 10$ s. The aim of those tests is to observe the impact of the modifications brought on the time and space parameters.

5.3.1 Reproduction of the demonstration trajectory for several timescales

First of all, the one-dimensional trajectory is generated for different timescales τ. The parameters chosen for the transformation system (5.1) are given in the table 5.1. The ratio between α_z and β_z is defined to assure the dynamical system to be critically damped.

We can observe than both methods provide trajectories faithful to demonstration for the same timescale parameter. When another timescale is used, the shape of the trajectories is similar to the demonstration shape. The results for the transformation system Eq.(5.1) are displayed in the figure 5.1 and for the transformation system Eq.(5.7), in the figure 5.2.

Parameters	**Method (5.1)**	**Parameters**	**Method (5.7)**
Nbr_{basis}	100	Nbr_{basis}	100
α_x	-ln(0.001)	α_x	-ln(0.001)
α_z	25	K	200
β_z	$\alpha_z/4$	D	$2\sqrt{K}$

Table 5.1: Parameters of the DMP implementation in 5.1 and 5.7.

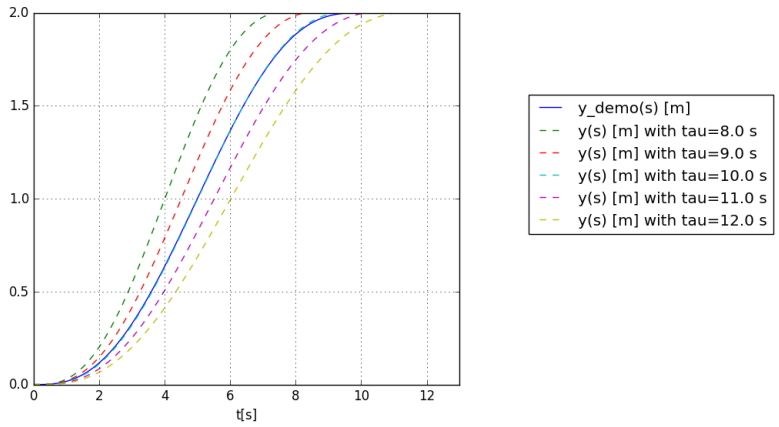


Figure 5.1: Trajectories generated with the transformation system (5.1) for several timescale factors $\tau = [8, 9, 10, 11, 12][\text{s}]$.

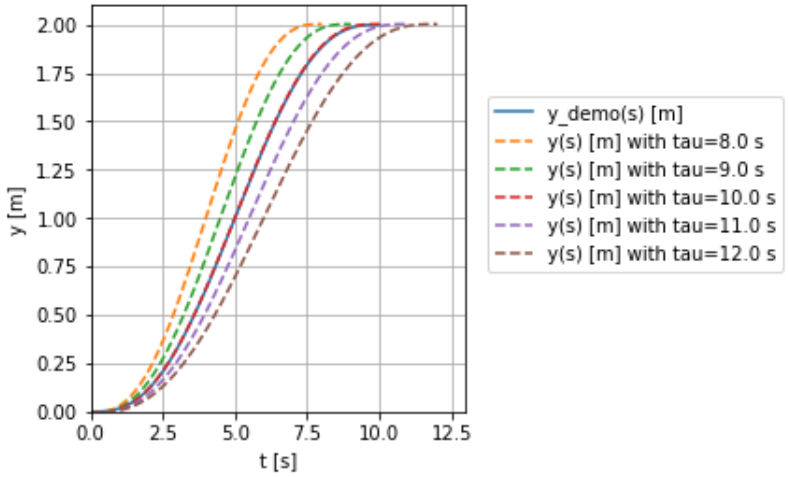


Figure 5.2: Trajectories generated with the transformation system (5.7) for several timescale factors $\tau = [8, 9, 10, 11, 12][\text{s}]$.

5.3.2 Variation of goal and timescale

In order to check the robustness of the DMP systems, two tests are made. The first series of test is the variation of the timescale factor for a different goal than the demonstration one. The results are presented in the figures 5.4 and 5.3 for each method.

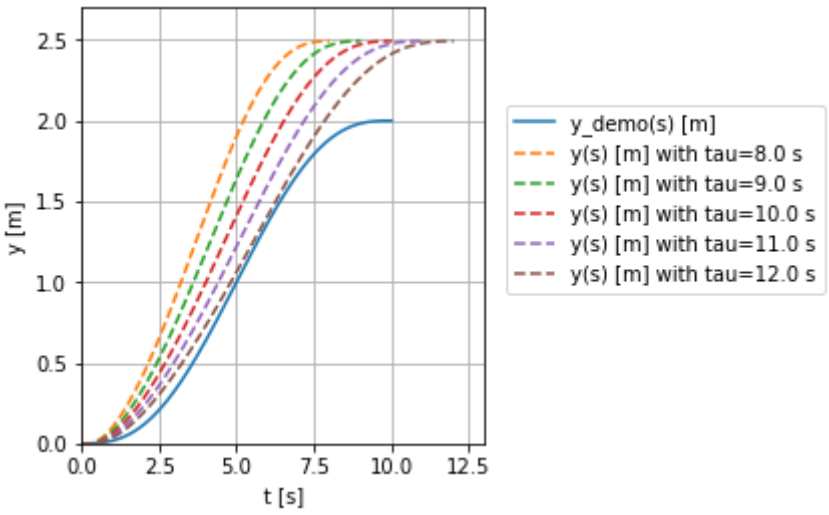


Figure 5.3: Trajectories generated with the transformation system (5.7) for a new goal with several timescale factors τ.

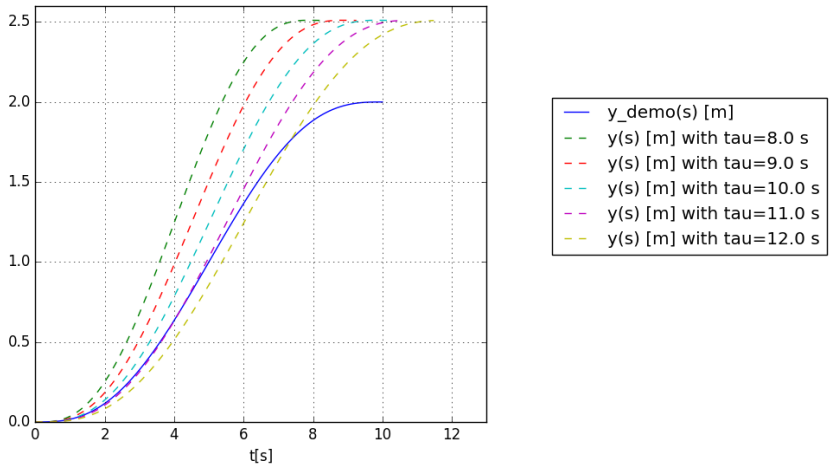
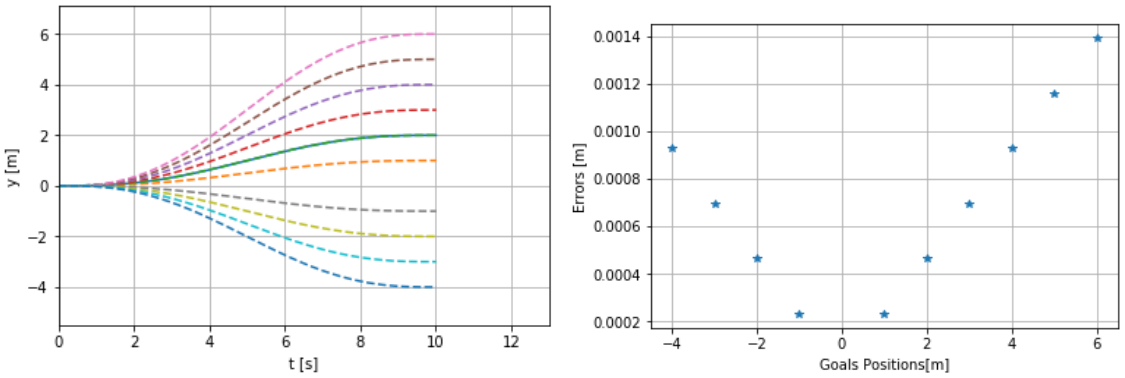


Figure 5.4: Trajectories generated with the transformation system (5.1) for a new goal with several timescale factors τ.

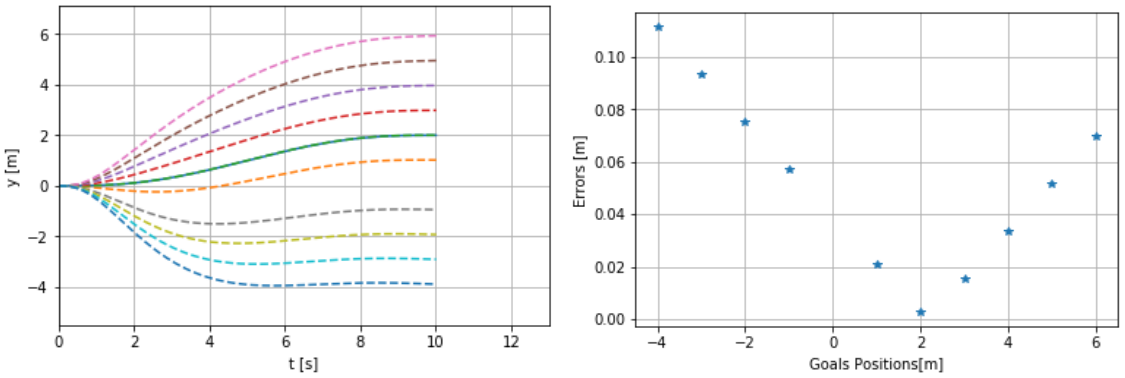
For the next series of generated trajectories, the goal was changed but the timescale factor τ remained the same (= 10 s). The trajectories generated with the first transformation system Eq.(5.1) are displayed in Fig.5.5. In comparison to those results, the trajectories generated with the other transformation system Eq.(5.7) are displayed in Fig.5.6. Even though the trajectories made with Eq.(5.7) reach

their respective goals on time, their shapes divert from the demonstration. This change is caused by the modifications made between Eq.(5.5) and Eq.(5.7).



(a) Generated Trajectories **(b)** Corresponding error

Figure 5.5: Generation of trajectories made with the transformation system 5.1 for several goals $[-4, -3, -2, -1, 1, 2, 3, 4, 5, 6]$[m]. The full line trajectory is the demonstration trajectory, the discontinued trajectories are the simulations. **(b)** Errors associated to the goals of the generated trajectories.



(a) Generated Trajectories **(b)** Corresponding times

Figure 5.6: **(a)** Generation of trajectories made with the transformation system 5.7 for several goals $[-4, -3, -2, -1, 1, 2, 3, 4, 5, 6]$[m]. The full line trajectory is the demonstration trajectory, the discontinued trajectories are the simulations. **(b)** Errors associated to the goals of the generated trajectories.

5.4 Learning and Planning in space

The demonstration trajectory used for the following comparison is an ellipsoid arc:

$$\begin{cases} x(t) = u_x \cos\left(p\frac{2\pi}{t_f}t\right) - ux + x_0 \\ y(t) = u_y \sin\left(p\frac{2\pi}{t_f}t\right) + y_0 \\ z(t) = c_z p\frac{2\pi}{t_f}t + z_0 \end{cases} \tag{5.17}$$

with the parameters:

Parameters	Value	Description
u_x	1 [m]	x radius of the ellipsoid
u_y	0.5 [m]	y radius of the ellipsoid
c_z	0.4 [m/rad]	step of the ellipsoid
$\begin{bmatrix} x_0 & y_0 & z_0 \end{bmatrix}^T$	$\begin{bmatrix} 0.5 & 0.5 & -0.5 \end{bmatrix}^T$ [m]	Starting point
p	0.5	Arc proportion
t_f	τ	Final time = Timescale

Table 5.2: Parameters of the demonstration trajectory

The results of the trajectory learning and generation for the DMP formulation Eq.(5.1) are displayed in the figure 5.7.

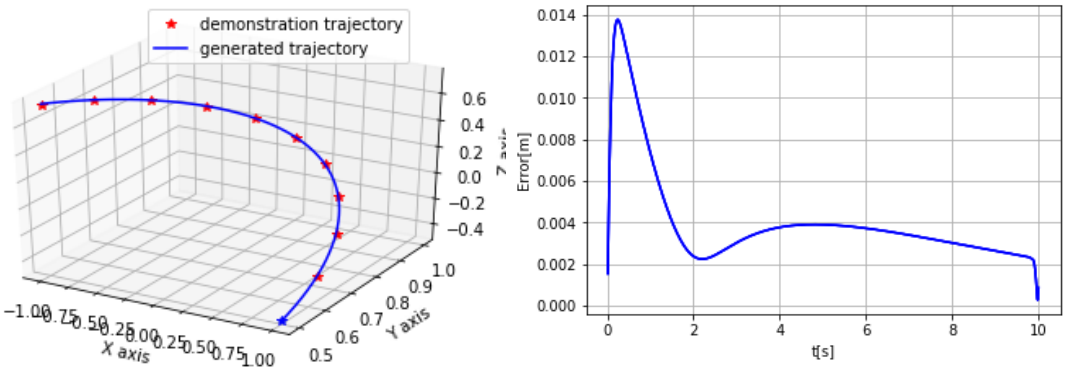


Figure 5.7: Trajectory reproduction after learning

The results of the trajectory learning and generation for the DMP formulation Eq.(5.7) are displayed in the figure 5.8.

The trajectory generation in three dimensions in faithful to the learned trajectory. However, the modified DMP (5.7) is more robust to goal modifications. This can be explained by the independence of the forcing terms with respect to the goal.

As expressed in the literature ([18] and [7]), this method is designed to be more robust to changes and to perform trajectory tracking for systems subjected to perturbations such as obstacles and goal changes.

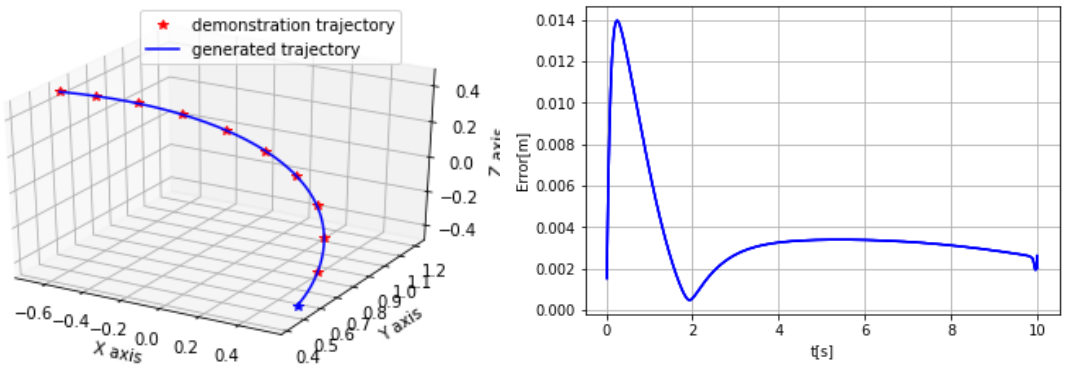


Figure 5.8: Error in time between the demonstration trajectory and the one generated by the DMPs

5.5 Reactive Planning with DMP

There is the possibility to plan the avoidance of obstacles by adding a contribution in the DMP transformation system in order to perturb it when the obstacle is a threat to the robot. The way to proceed has already been mentioned in Chapter4 with the dynamical systems for obstacle avoidance. The transformation system becomes then:

$$\begin{cases} \tau \dot{v} = K(g-y) - Dv - K(g-y_0) + Kf(x) + p(y,\dot{y}) \\ \tau \dot{y} = v \end{cases} \tag{5.18}$$

where $p(y,\dot{y})$ denotes the obstacle avoidance term that acts as an external force on the system and can be expressed as follows [18]:

$$p(y,\dot{y}) = \gamma R_u(\pi/2)\dot{y}\Phi \exp(-\beta\Phi) \tag{5.19}$$

Note $R_u(\theta)$ is a rotation matrix of angle θ about axis $u = (y-o)\times\dot{y}/\|(y-o)\times\dot{y}\|$ being the normal vector to the plan containing the relative position of the controlled point and the obstacle and is:

$$R_u(\theta) = \begin{bmatrix} \cos\theta + u_x^2(1-\cos\theta) & u_xu_y(1-\cos\theta) - u_z\sin\theta & u_xu_z(1-\cos\theta) + u_y\sin\theta \\ u_xu_y(1-\cos\theta) + u_z\sin\theta & \cos\theta + u_y^2(1-\cos\theta) & u_yu_z(1-\cos\theta) - u_x\sin\theta \\ u_zu_x(1-\cos\theta) - u_y\sin\theta & u_zu_y(1-\cos\theta) + u_x\sin\theta & \cos\theta + u_z^2(1-\cos\theta) \end{bmatrix},$$

and:

$$\Phi = \arctan\left(\frac{\|(y-o)\times\dot{y}\|}{(y-o)^T\dot{y}}\right),$$

β and γ being parameters to be tuned.

Alternative expressions for obstacle avoidance can also be found in [26] and [16]. The avoidance term from [16] has already been detailed in Eq.(4.14) where the avoidance term is recalled here:

$$p(y,\dot{y}) = \psi(y,\dot{y}) = \lambda(-\cos\theta)^{\beta-1}\frac{\|\dot{y}\|}{p(y)}\left(\beta\nabla_y\cos\theta - \frac{\cos\theta}{p(y)}\nabla_y p(y)\right)$$

An illustration of a trajectory with obstacle avoidance generated with this method is presented in the figure 5.9.

An analogy can be made between the two transformation systems in order to transfer the obstacle function to the system (5.1). The $\dot{z}$ and the $\dot{v}$ have the same dimension regarding the obstacle avoidance additional term.

The enriched version of (5.1) is simply then:

$$\begin{cases} \tau \dot{z} = \alpha_z(\beta_z(g-y) - z) + f + p(y,\dot{y}) \\ \tau \dot{y} = z \end{cases} \tag{5.20}$$

The obstacle avoidance is properly performed[1] but it is also noticeable that this method is more aggressive than the modified DMP Eq.(5.9) as shown by the rebounds in the figures 5.11 and 5.10.

[1] The obstacle was circumvented with no-collision

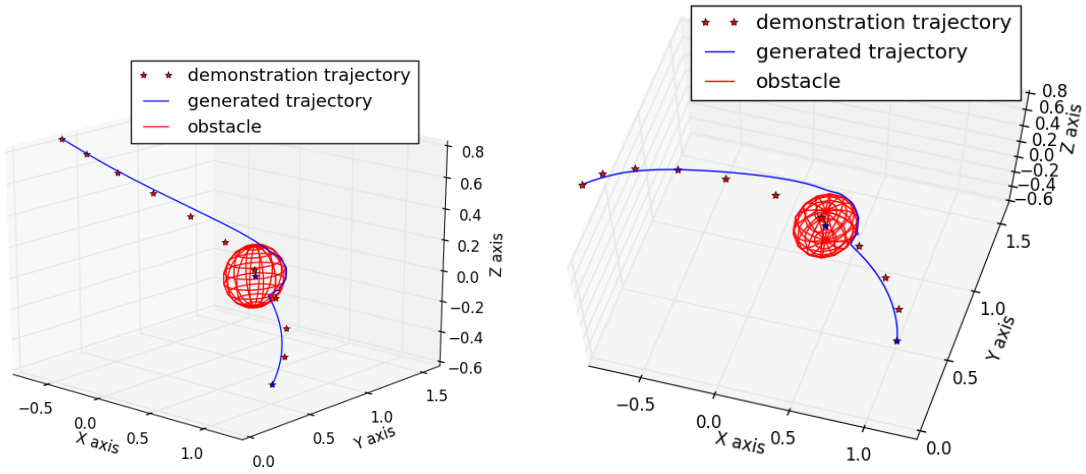


Figure 5.9: Obstacle avoidance with avoidance term from Eq.(4.14) and transformation system from Eq.(5.9)

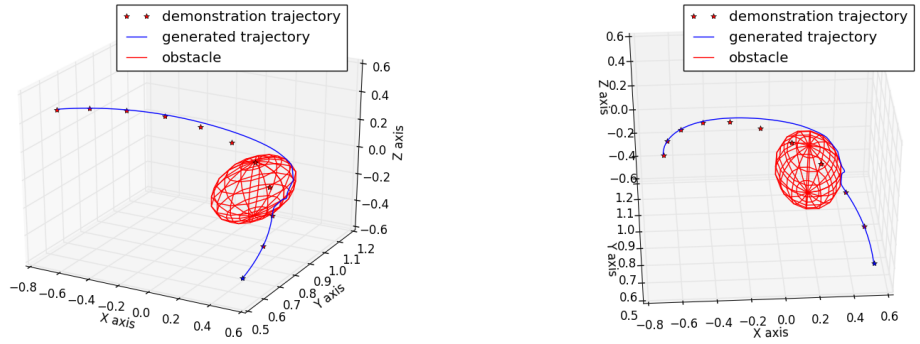


Figure 5.10: Obstacle avoidance with avoidance term from Eq.(4.14) and transformation system from Eq.5.20

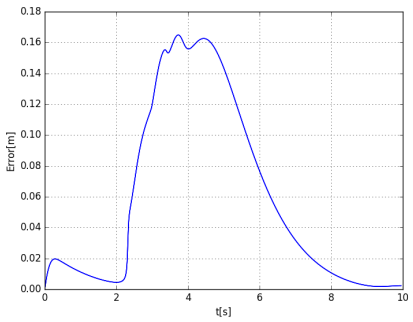


Figure 5.11: Error between the demonstrated trajectory and the trajectory generated by (5.20) with an obstacle located in $p_o = [0.2, 0.95, -0.2]^T$.

6

Experimental Results

In this chapter we present results from experiments carried out on a UR10 robot through the ROS framework. The aim is to implement the stack of task and to test its performance for robot control. The chosen frequency for the control and the position feedback is 100 Hz. The ROS implementation of the HQP method is made as a stack of tasks package represented at the figure 6.1 where the control node occupies a central position and communicates with a trajectory node (DMP trajectory generator in figure 6.1) and the robot.

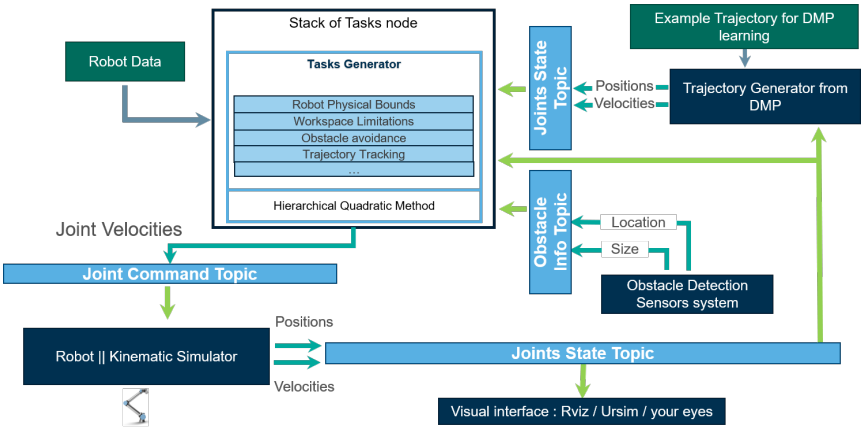


Figure 6.1: ROS communication schema of the stack of tasks package

6.1 Experiments on UR10

Experiments have been carried out and the results are presented in this section. First, in order to address an infinite loop issue we modify the HQP method proposed in [5]. There is a possibility that the active-search remains blocked in an infinite loop caused by a perpetual activation - deactivation of a constraint. No solutions were proposed even though that behaviour might seriously affect the performance of the robot control system. A condition is then added in the active search algorithm 3 with a safe exit for the x^*. Several safe exits are proposed in order to deal with this problem in algorithm. The first one is to leave the loop after a defined number of iteration with a zero value. As it is observed in the figure 6.2, this safe exit creates some hits in the trajectory since the joint velocities computed with HQP are set to zero, the robot will receive a zero command.

Algorithm 3: Hierarchical Quadratic Programming Algorithm with safe exit

Data: Every Constraints respecting the shape Eq.(2.2)
Result: Optimal solution x^* and active set of tasks
Building the stack from constraints;
Active search:
if *First call* **then**
    initialisation of the active set of tasks by activating all the equalities detected in the stack;
else
    Go to while loop;
while *index* $\neq$ *length stack* **do**
    eHQP optimal computation;
    Constraints violation search among whole stack;
    Innerloop:
    if *Violation of constraint detected* **then**
        Activation of the constraint having the highest priority among violated one;
    else
        Go to outerloop;
    Outerloop:
    if *no deactivations needed* **then**
        index is incremented;
    else
        index is not changed and the stack is reinspected ;
    Safe Exit:
    if *Iteration counter higher than a fixed maximum* **then**
        x^* = Safe exit ;
        break;
    else
        /
Optimum solution x^* = Actual solution

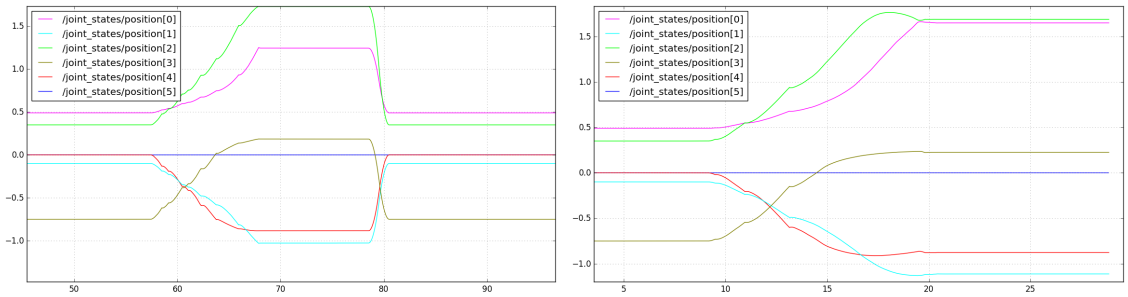


Figure 6.2: Illustration of the safe exit consequences.

An alternative safe exit consists in a prediction of the next results based on a linear interpolation made with the previous values: $x_{out} = \dot{q} + \frac{\dot{q}_{t-1} - \dot{q}_{t-2}}{\Delta t} t$ since the computation is discrete and the frequency fixed, it can be simply written as: $x_{out} = \dot{q} + (\dot{q}_{t-1} - \dot{q}_{t-2})$. With this kind of safe exit, the previous problem of hits in the trajectory is solved as shown in Fig. 6.3 with the velocity profile displayed on the figure 6.5. This safe exit is subjected to comply with the maximum tolerated acceleration defined for the robot joints which implies that $(\dot{q}_{t-1} - \dot{q}_{t-2})$ is bounded by Eq.(3.7).

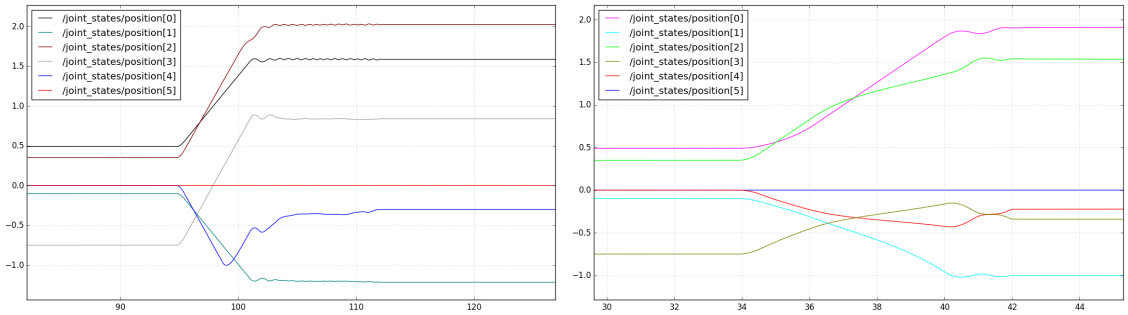


Figure 6.3: Illustration of the predicted safe exit consequences.

The first experiment is a goal-attractor task with a sphere shape obstacle located on the goal $[-0.5, 0.5, 0.5]^T$. The starting point is located in $[0.5, 0.5, -0.5]^T$.
The aim of this test is to verify that the collision is prevented and also that the robot keeps on trying to find the best position to get closer to the goal. The joints velocity profile is shown in Fig.6.4. Two worth noting behaviours are observable. The first one, from $10s$ to $17s$ highlights that the velocity bounds are respected. The second behaviour, starting from $17s$ until the end of the test, is the intervention of the obstacle. The end-effector does not interfere with the obstacle and HQP solver keeps on looking for the best configuration to reach the goal.

The next test carried out with robot is the tracking of a trajectory generated with a system of dynamical movement primitives. The DMP formulation used is based on Eq.(5.7). The trajectory parameters are gathered in table 6.1.

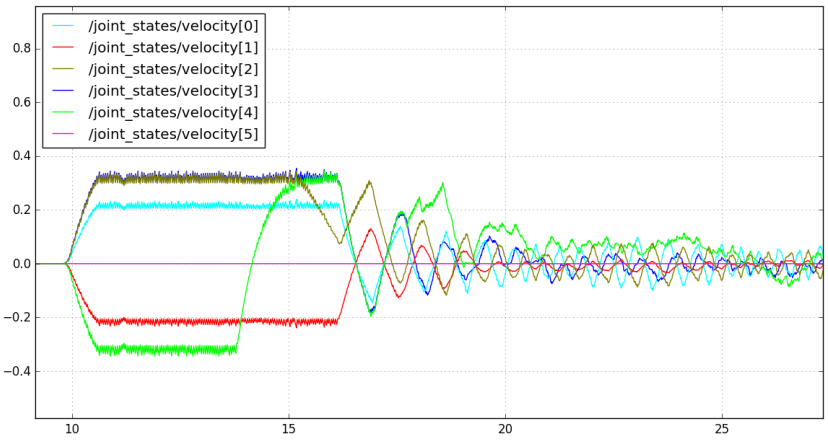


Figure 6.4: Joints velocities for a goal attractor task with the obstacle on the target.

Parameters	Value	Description
u_x	0.6 [m]	x radius of the ellipsoid
u_y	0.5 [m]	y radius of the ellipsoid
c_z	0.3 [m/rad]	step of the ellipsoid
$\begin{bmatrix} x_0 & y_0 & z_0 \end{bmatrix}^T$	$\begin{bmatrix} 0.53 & 0.57 & -0.47 \end{bmatrix}^T [m]$	Starting point
p	0.5	Arc proportion
t_f	τ	Final time = Timescale

Table 6.1: Parameters of the demonstration trajectory for a simulation with the UR10 in ROS.

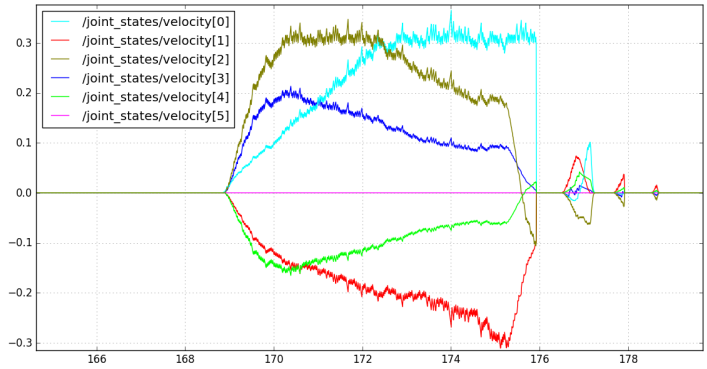


Figure 6.5: Joints velocities for a trajectory generated with the demonstration trajectory Eq.(5.17) with the parameters in table 6.1 and the DMP system Eq.(5.7).

6.2 Safety Bounds

This experiment is performed in order to highlight the effect of the bounds imposed by the safety level of the stack. The test case is a goal attractor task with an obstacle located between the starting point and the goal. The velocity limits of the joints are significantly reduced for this test compared to the real robot limits in order to visualise them easily. The acceleration bounds are observable with the slopes of the joints velocities as in time 930. The illustration of a velocity bound is noticeable in time 930.2 for the joint numbered [1].

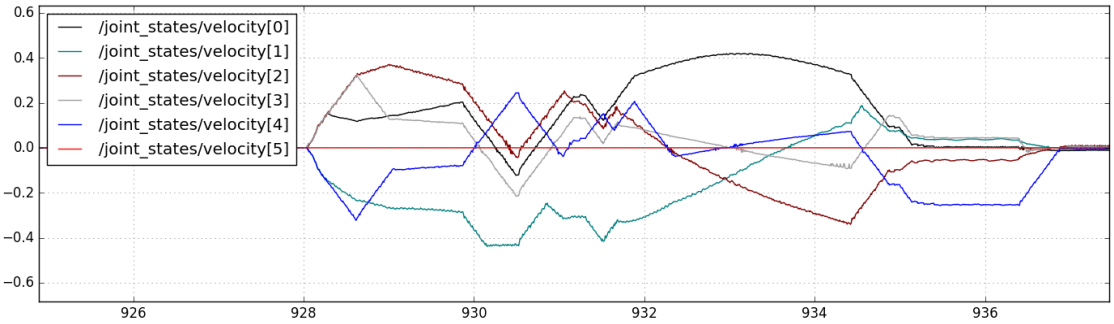


Figure 6.6: Joint velocities for a goal attractor task with the starting point and the goal on either side of an obstacle.

6.3 Obstacle avoidance

In this section, three different obstacle avoidance approaches are tested: collision avoidance in the stack, trajectory modification with modified DMP Eq.(5.18) and obstacle avoidance with vortex formulation in the stack in table 4.4. The experiment case is the same as in Section 6.1: the trajectory is summarised in the table 6.1 and the obstacle is a sphere located in $\begin{bmatrix}0.0 & 1.0 & -0.1\end{bmatrix}$ with the radius $R = 0.3$.

6.3.1 Collision avoidance approach

The collision avoidance approach is the simplest implementation. The stack is composed of three main levels. The first one is as usual the safety bounds (acceleration, velocity, position), the second one consists in the collision avoidance task Eq.(3.11) and the last level is the trajectory tracking task. The trajectory is generated through the dynamical movement primitive system Eq.(5.7).
The avoidance is correctly executed in the sense that the obstacle is not hit, nevertheless it is carried out abruptly. The joints velocities profiles in Fig.6.7 reflect this. Indeed, the collision avoidance task becomes active at time 314 and will remain so until time 318. Collision avoidance has priority over the path and therefore when it is activated, the effector deviates from this path. However, since the trajectory is dynamically generated by the DMP system, it adapts to the current position of the end-effector. This new trajectory once again encounters the obstacle in its path

and this causes the effect of rebounds which is observable between the times 314 and 318. The figure 6.8 shows the end-effector motion when the obstacle avoidance is performed with Eq.(3.11) for a trajectory generate with DMPs.

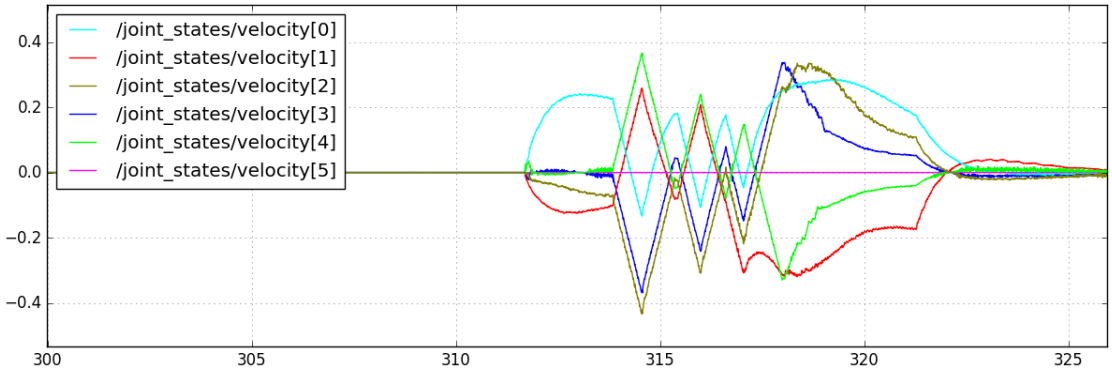
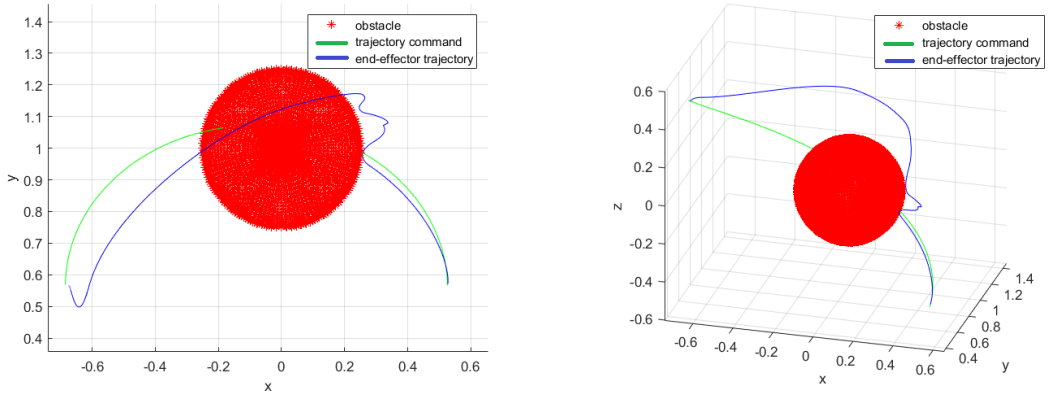


Figure 6.7: Joints velocities for an obstacle avoidance realised with Eq.(3.11).

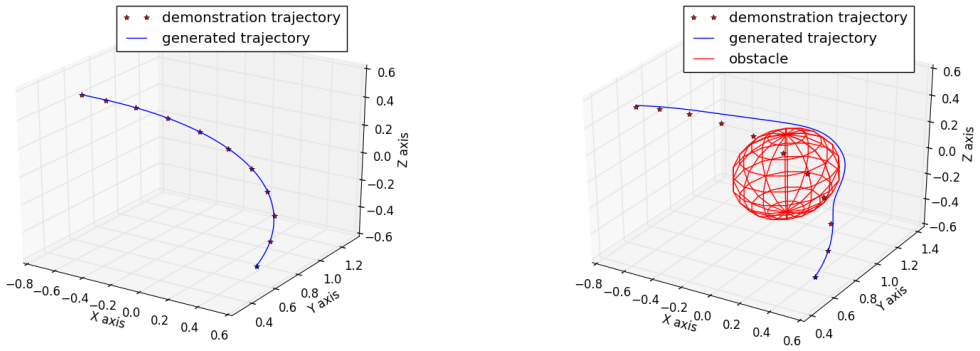


(a) **(b)**

Figure 6.8: Trajectory generated if there were no-obstacles to avoid and end-effector motion.

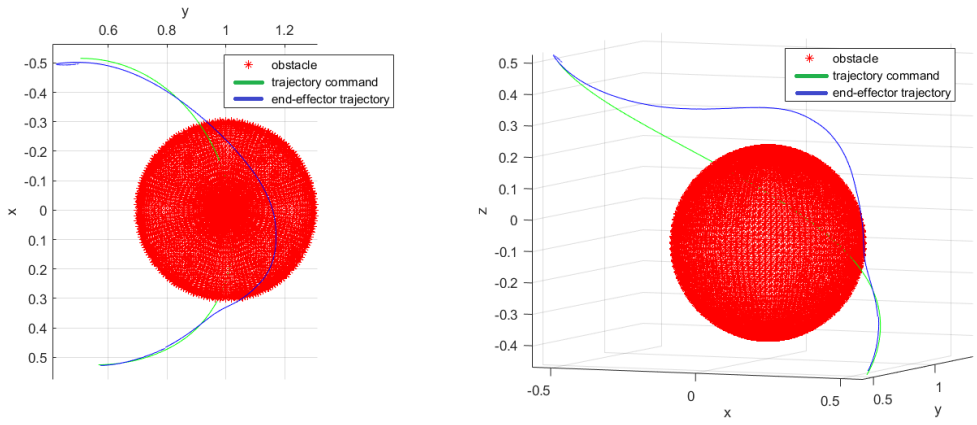
6.3.2 Modified DMP approach

In this following experiment, obstacle avoidance is taken into account in the generation of the trajectory generated by the modified DMP detailed in Eq.(5.18). The avoidance term used is Eq.(4.14) from [17]. As it can be seen in figure 6.11, the effect of the obstacle avoidance contribution in the DMP differential equations integration occurs in time 661 and is aggressive (note the slope of the velocity curve). After time 670, the goal is reached and a few adjustments are made. Compared to the previous approach, this one allows a more smooth avoidance. The influence of $p(x, v)$ of Eq.(5.18) starts to have an impact from 661 and modifies the original system until 667 to ensure the avoidance. The robot motion is shown on the figure 6.12. The motion of the end-effector in space is displayed at the figure 6.10.



(a) **(b)**

Figure 6.9: Trajectory without the obstacle and Trajectory modified by the obstacle avoidance.



(a) **(b)**

Figure 6.10: Trajectory without taking the obstacle avoidance term into account and end-effector motion.

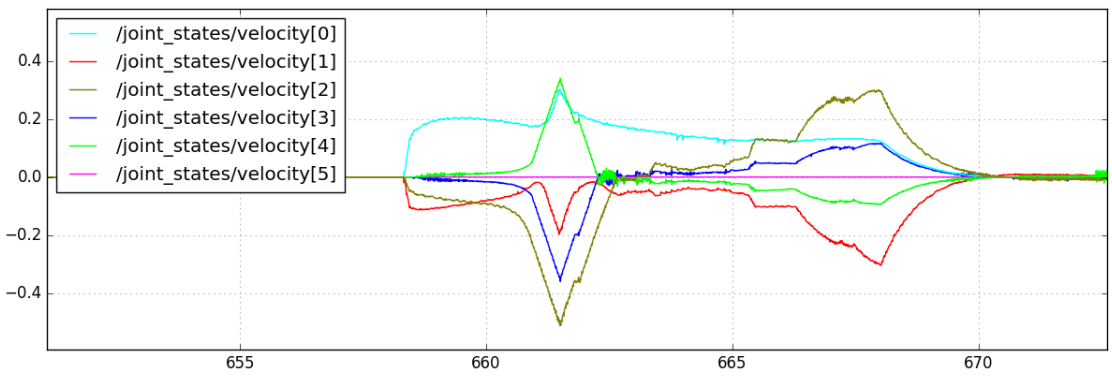


Figure 6.11: Joints velocities for an obstacle avoidance realised with the modified DMP Eq.(5.18) and the avoidance term Eq.(4.14).

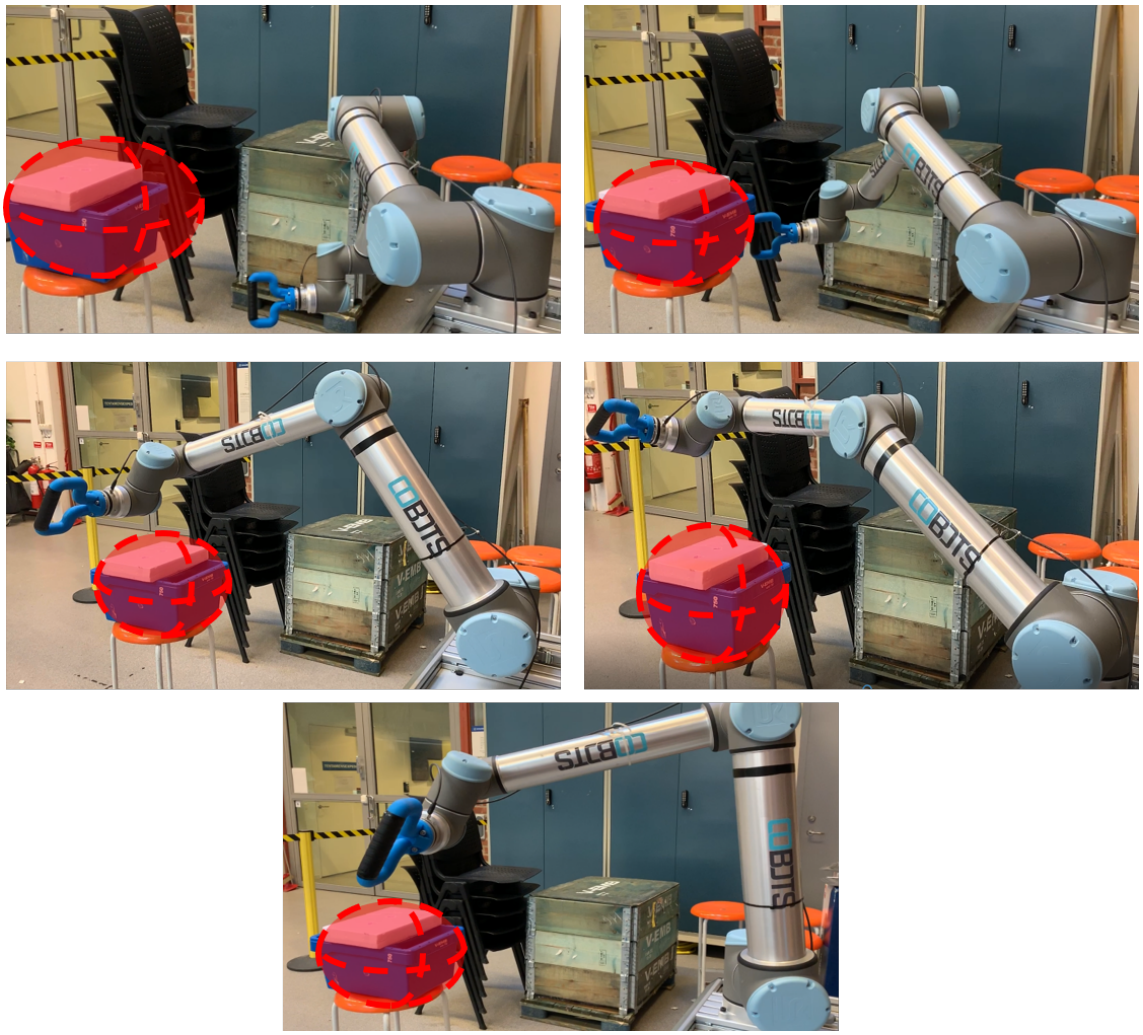


Figure 6.12: Pictures of the UR10 robot in the execution of the trajectory shown in Fig.6.9.

6.3.3 Obstacle avoidance with vortex approach

This avoidance approach is summarised in table 4.4. With this approach we aim at allowing a smooth obstacle avoidance without being supported by the trajectory generation. The trajectory is the ellipsoid presented in Fig.6.9a. Joints velocities are shown in Figure 6.13. In comparison to Fig.6.7, the avoidance is more smooth since there are no rebounds.

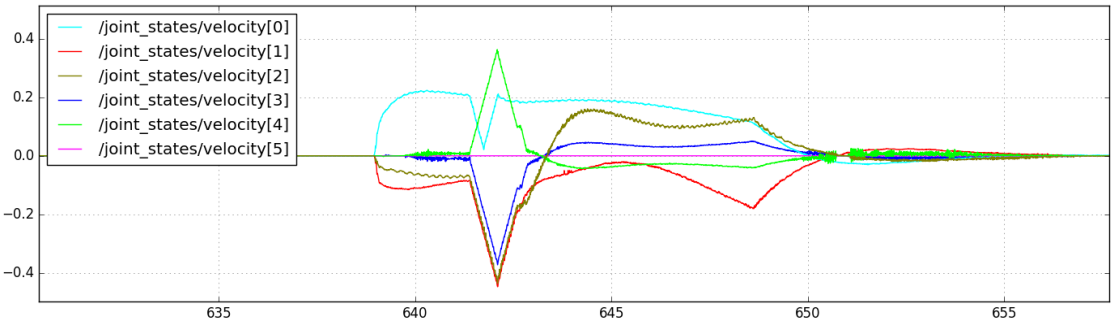


Figure 6.13: Joints velocities for an obstacle avoidance realised with the stack structure presented in table 4.4.

Another experiment has been performed to illustrate the avoidance with vortex field in the stack proposed in table 4.4. The trajectory, the obstacle and the end-effector motion are shown in the figure 6.14. The green curve represents the trajectory generated by the DMP if there was no obstacle in the workspace. The difference of shape between the end-effector motion and the green trajectory is explained by the perpetual adaptation of the trajectory generated by the DMPs since they use the actual position of the end-effector to compute the next target position.

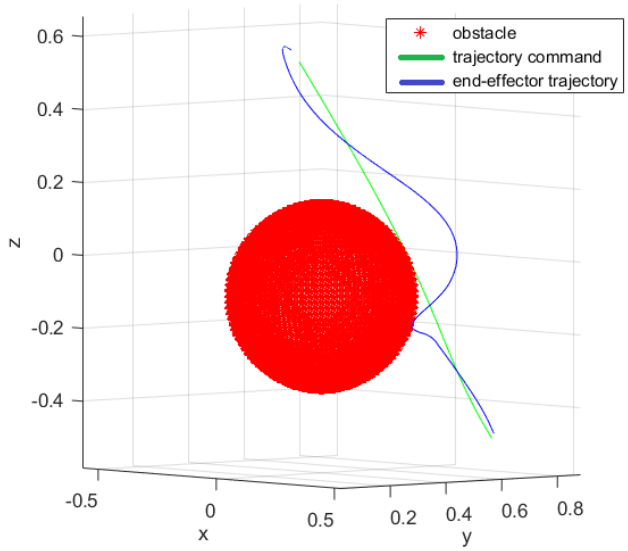


Figure 6.14: Trajectory generated if there were no-obstacles to avoid and end-effector motion in space.

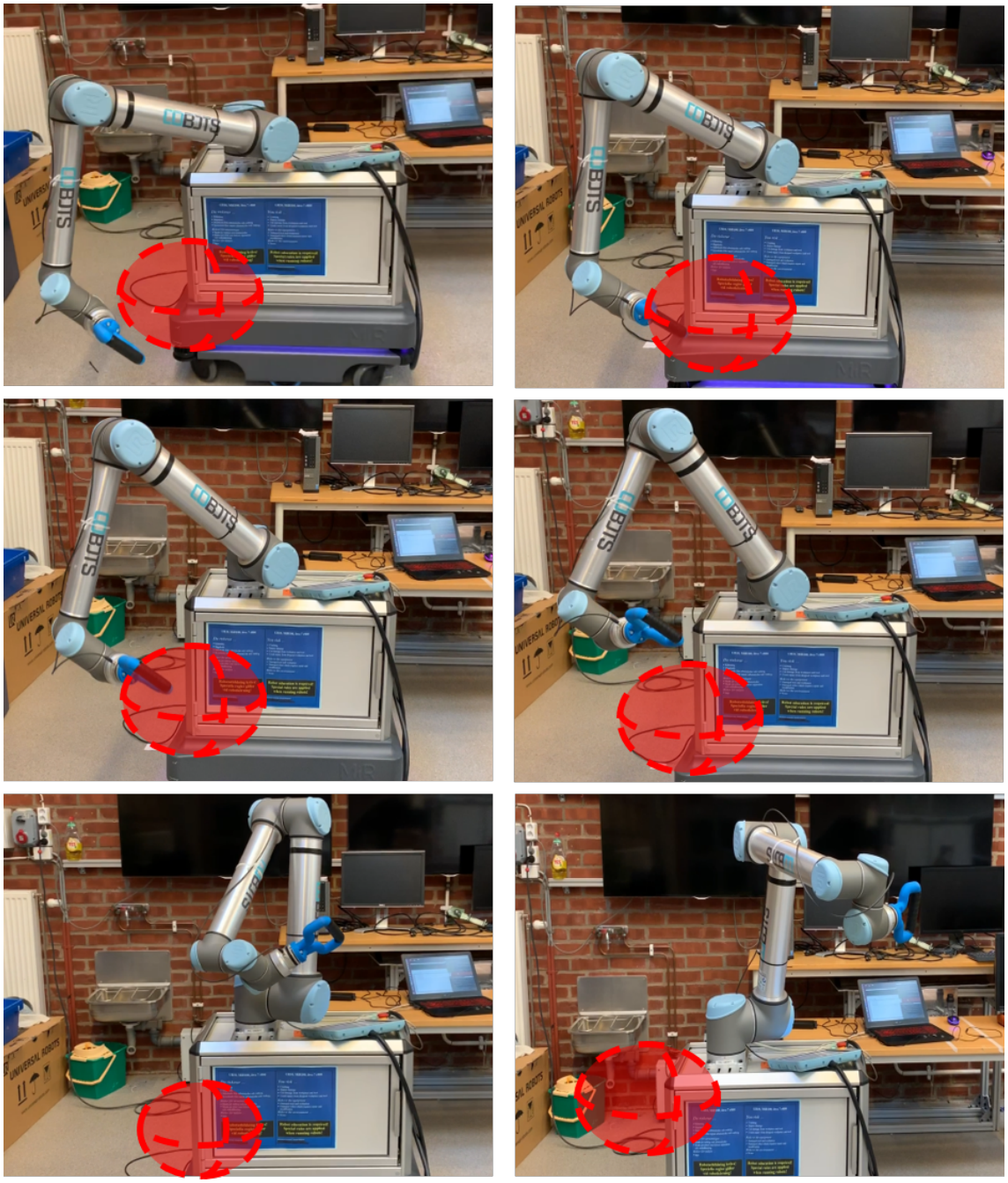


Figure 6.15: Pictures of the UR10 robot in the execution of the trajectory shown in Fig.6.14 with the stack structure presented in table 4.4.

7

Conclusions and Future Work

The objective of this thesis was to propose a framework that combines optimisation based control and reactive planning for a robot that is meant to work in an unstructured workspace with humans. To achieved this, we implemented the Hierarchical Quadratic Programming method as a kinematic controller that can accommodate equality and inequality constraints. The kinematic controller using HQP has been built and implemented with a stack structure designed for safety purpose.

In order to implement a fluid obstacle avoidance constraint, several methods have been tested and improved. The artificial potential field methods (APF) studied shared common behaviours as shown in simulations. The risk of the singular configuration being the meta-stable situation caused by the cancellation of two antagonist commands, is present in most of the APF methods. The interest shown in the vortex fields was motivated by the possibility to force the circumvention of obstacles preventing this meta-stable state. This kind of tasks are also used to escape from concave obstacles. We also proposed an artificial potential field to be used as a level of the HQP stack of tasks. A general conclusion drawn of the HQP controller is that its performance depends on how the stack is designed. The proposed potential task and stack structure shows some promising results.

The desired trajectory generation needed to be flexible and reactive which justify the use of Dynamical Movement Primitives (DMPs). Before the generation of the trajectory, a demonstrated trajectory is learned. Concerning the generation, several formulations of the transformation system were studied following the criteria of adaptability and robustness. The modified formulation of DMP showed that it can resist to perturbations. The possibility to ensure obstacle avoidance with the DMP trajectory generator was also explored and shown an effective as well as smooth avoidance.

In conclusion, it has been proven that the avoidance with the modified DMPs and with specialised potential functions were both compatible with the stack of tasks approach implicit to HQP.

There are some possibilities of improvement in the obstacle avoidance tasks. Indeed, the avoidance cases tested were simple. The theories can then be extended to moving obstacles with random shapes (other than ellipsoids). A real-time obstacle detection with exteroceptive sensors can also be used in order to make the method usable for a wider range of cases.

The whole thesis work was developed and tested for a robot having a open kinematic chain. It is then interesting to extend the theories for robots having more complex structure such as dual arm robots that can also form closed kinematic chains.

Bibliography

[1] G. Antonelli. Stability analysis for prioritized closed-loop inverse kinematic algorithms for redundant robotic systems. *IEEE Transactions on Robotics*, 25:985–994, 2009.

[2] A. Atawnih, D Papageorgiou, and Z. Doulgeri. Kinematic control of redundant robots with guaranteed joint limit avoidance. *Robotics and Autonomous Systems*, 79:122–131, 2016.

[3] D. Cherney, T. Denton, R. Thomas, and A. Waldron. *Linear Algebra*. University of California Davis, 2013.

[4] A. Colomé and C Torras. Closed-loop inverse kinematics for redundant robots: Comparative assessment and two enhancement. *IEEE/ASME Transactions on Mechatronics*, 20(2):944–955, 4 2015.

[5] A. Escande, N. Mansard, and P-B. Wieber. *Hierarchical Quadratic Programming: Fast online humanoid-robot motion generation. International Journal of Robotics Research*. SAGE Publications, 2014.

[6] F. Guyomarc'h, D. Mezher, and B. Philippe. Rank revealing qr factorisation. *CSDA*, 2005.

[7] H. Hoffmann, P. Pastor, D-H. Park, and S. Schaal. Biologically-inspired dynamical systems for movement generation: automatic real-time goal adaptation and obstacle avoidance. *IEEE International Conference on Robotics and Automation*, pages 2587–2589, 2009.

[8] AJ Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, 25(2):328–373, 2013.

[9] G. Jarquín, G. Arechavaleta, and V. Parra-Vega. Time parametrization of prioritized inverse kinematics based on terminal attractors. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011.

[10] O. Kanoun, F. Lamiraux, and P.-B. Wieber. Kinematic control of redundant manipulators: generalizing the task priority framework to inequality tasks. *IEEE Transactions on Robotics*, 27:785–792, 2011.

[11] S M. Khansari-Zadeh and A. Billard. A dynamical system approach to realtime obstacle avoidance. *Auton Robot*, 32:433–454, 2012.

[12] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5, 1986.

[13] B.R. Munson, D.F. Young, T.H. Okiishi, and W.W. Huebsch. *Fundamentals of Fluid Mechanics*. Wiley & Sons Inc., 6 edition, 2010.

[14] F. Nagataa, S. Kishimotoa, S. Kuritaa, A. Otsukaa, and K. Watanabeb. Neural network-based inverse kinematics for an industrial robot and its learning method. *4th IIAE International Conference on Industrial Application Engineering*, 2016.

[15] D. Panagou. Motion planning and collision avoidance using navigation vector. post 2012.

[16] D-H. Park, H. Hoffmann, P Pastor, and S. Schaal. Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields. *IEEE-RAS International Conference on Humanoid Robots*, 8, 2008.

[17] D-H. Park, H. Hoffmann, and S. Schaal. Combining dynamic movement primitives and potential fields for online obstacle avoidance.

[18] P. Pastor, H. Hoffmann, Tamim Asfour, and S; Schaal. Learning and generalization of motor skills by learning from demonstration. *IEEE International Conference on Robotics and Automation*, 2009.

[19] M. Prada, A. Remazeilles, A. Koene, and S. Endo. Dynamic movement primitives for human-robot interaction: comparison with human behavioral observation. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1168–1175, 2013.

[20] A. Sang-ik and L. Dongheui. Prioritized inverse kinematics with multiple task definitions. *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[21] S. Schaal. Dynamic movement primitives - a framework for motor control in humans and humanoid robotics. In *Adaptive Motion of Animals and Machines*, chapter 6, pages 262–280. Springer, Tokyo, 2006.

[22] S. Schaal, P. Mohajerian, and A. Ijspeert. Dynamics systems vs. optimal control - a unifying view. *Progress in Brain Research*, 165:425–445, 2007.

[23] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert. Control, planning, learning, and imitation with dynamic movement primitives. In *IROS 2003*, pages 1–21. Max-Planck-Gesellschaft, October 2003.

[24] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics: Modelling, Planning and Control*. Springer, 2009.

[25] B. Siciliano and J.-J. Slotine. A general framework for managing multiple tasks in highly redundant robotic systems. *IEEE Int. Conf. on Advanced Robotics (ICAR'91)*, 1991. Pisa, Italy.

[26] S. Stavridis, D Papageorgiou, and Z. Doulgeri. Dynamical system based robotic motion generation with obstacle avoidance. *IEEE Robotics and Automation letters*, 2:712–718, 2017.

[27] W. Suleiman, K. Ayusawa, F. Kanehiro, and E. Yoshida. On prioritized inverse kinematics tasks: Time-space decoupling. *15th International Workshop on Advanced Motion Control (AMC)*, 2018.

[28] D. Widmann. Adaptive control based on dynamical movement primitives for human-robot handover. Master's thesis, Chalmers University of Technology, Gothenburg , Sweden, 2016.