Multiobjective Kinematic Trajectory Planning
An Application to the
Captive Trajectory Simulation (CTS) System

by

Alexis Guigue

B.Sc. equivalent, École Nationale Supérieure de l'Aéronautique et de l'Espace,

M.A.Sc., École Polytechnique de Montréal

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

Ottawa-Carleton Institute for
Mechanical and Aerospace Engineering

Department of
Mechanical and Aerospace Engineering
Carleton University
Ottawa, Ontario
December 22, 2009

The undersigned recommend to

the Faculty of Graduate Studies and Research

acceptance of the thesis

Multiobjective Kinematic Trajectory Planning

An Application to the

Captive Trajectory Simulation (CTS) System

submitted by **Alexis Guigue, B.Sc. equivalent, M.A.Sc.**

in partial fulfillment of the requirements for

the degree of Doctor of Philosophy

Dr. M. Ahmadi
Thesis Supervisor

Dr. M.J.D. Hayes
Thesis Supervisor

Dr. R.G. Langlois
Thesis Supervisor

Dr. M.I. Yaras
Chair, Department of
Mechanical and Aerospace Engineering

Carleton University

December 22, 2009

Abstract

In this thesis, a new approach to the resolution of multiobjective trajectory planning problems is proposed. Rather than providing a single solution with the weighting method, it is proposed to provide the entire Pareto optimal set, or more generally the entire minimal element set when the objective space is partially ordered by a cone. This approach is applied to the general class of multiobjective deterministic finite horizon optimal control problems to which many multiobjective trajectory planning problems, including the joint space trajectory planning problem arising from a wind-tunnel experimental setup called captive trajectory simulation (CTS) system, can be shown to belong. A new discrete dynamic programming (DDP) approximation method for the resolution of this class of problems is developed, from which an approximate minimal element set is obtained. Partial convergence results of this approximate set towards the original minimal element set is obtained using set convergence in the sense of Hausdorff. When the DDP approximation method is applied to the joint space trajectory planning problem for the CTS system, the approximate Pareto optimal set is shown to provide a better representation of the Pareto optimal set than the set obtained with the weighting method. In order to improve the computational efficiency of the DDP approximation method, a clustering approach is introduced. With the application of clustering, the DDP approximation method is shown to have polynomial complexity, which yields a dramatic reduction in the resolution time. When the DDP approximation method with clustering is applied to the joint trajectory planning problem for the CTS system, the resulting set still provides a better representation of the Pareto optimal set than the set obtained with the weighting method. The Hausdorff distance is proposed as a measure for comparison. Finally, the complete CTS trajectory planning problem is solved by sequentially solving the problem in the task and joint spaces. For the resolution of the task space problem, an algorithm based on the resolution of a *rural postman problem* is proposed. The proposed sequential approach for the resolution of the CTS trajectory planning problem allows for fast resolution times.

Contents

List of Tables

List of Figures

List of Symbols

$\mathbf{p}$	end-effector position and orientation
$\mathcal{W}$	operating envelope of the CTS manipulator
$\dot{\mathbf{p}}$	end-effector velocity.
$\dot{\mathbf{p}}_{\text{max}}$	maximum end-effector velocity.
$\mathbf{q}$	joint configuration
$\mathcal{Q}$	joint range of the CTS manipulator, $\mathcal{Q} = [\mathbf{q}_{\text{min}},\mathbf{q}_{\text{max}}]$
$\dot{\mathbf{q}}$	joint speed
$\dot{\mathbf{q}}_{\text{max}}$	maximum joint speed
$\ddot{\mathbf{q}}$	joint acceleration
v	redundancy parameter
$\mathcal{S}$	set of subtasks to be completed during a grid survey experiment
T_{tunnel}	run time of the wind tunnel
∂Z	boundary of the set Z
$\lvert Z\rvert$	cardinality of the set Z
B	closed unit ball
$B(x, r)$	closed ball with center x and radius r
$\text{cl}(Z)$	closure of the set Z
$\lVert \cdot \rVert$	Euclidian norm in $\mathbf{R}^n$
$\text{int}(Z)$	interior of the set Z
$\langle \cdot, \cdot \rangle$	scalar product in $\mathbf{R}^n$
$SO(3)$	Special Orthogonal group in three dimensions

BVP	Boundary Value Problem
CTS	Captive Trajectory Simulation
DH	Denavit-Hartenberg
DOF	Degrees Of Freedom
DDP	Discrete Dynamic Programming
HJB	Hamilton-Jacobi-Bellman
IAR	Institute for Aerospace Research
IVP	Initial Value Problem
NBI	Normal-Boundary Intersection
NRC	National Research Council

Chapter 1

Introduction

1.1 Background

The motion of a robotic manipulator can be observed in three different spaces: the task, joint, and actuation spaces. The *task space* is the space in which the task is specified. Task specification is generally done in the space $\mathbf{R}^3 \times SO(3)$, where $\mathbf{R}^3$ is the three-dimensional Euclidian space used to describe the position, and $SO(3)$ is the group of all rigid body rotations used to describe the orientation. The *joint space* and the *actuation space* are the set of all possible values that can be taken by the joint and actuator variables, respectively. In the absence of actuator redundancy, both the joint space and the actuation space are subsets of $\mathbf{R}^n$, where $\mathbf{R}^n$ is the n-dimensional Euclidian space, and n is the number of joints or degrees of freedom (DOF) [2, p. 5]. Although the task is specified in the task space, a robotic manipulator is commanded either in the joint or actuation space. A trajectory planning problem is therefore the problem of finding a joint or actuator trajectory to complete the task. A major difficulty in trajectory planning problems arises when the task can be completed with an infinite number of joint or actuator trajectories, which occurs, for example, when the robotic manipulator is either kinematically or task redundant [3], i.e., when the dimension of the joint space is greater than the dimension of the task space. This thesis aims at solving such a problem for a captive trajectory simulation (CTS) system, which will be hereinafter referred to as the *CTS trajectory planning problem*.

The CTS system [4, 5, 6, 7], currently under development at the National Research Council (NRC) Institute for Aerospace Research (IAR), is a wind-tunnel experimental setup devised to investigate the release of *stores* from aircraft. In this context, a store is any object that can be released from an aircraft. The purpose of these investigations is to check whether the release of a store from its parent aircraft is safe, i.e., occurs without any collision. The most common type of experiment conducted with the CTS system is the grid survey experiment [5, 8]. The objective of a grid survey experiment is to build a look-up table of interference coefficients between the store and the parent aircraft as a function of the store angle of attack, trajectory angle, and Mach number. The information contained in this table will later serve as an input to an equation of motion solver to simulate the store release trajectory [5]. This look-up table is built by moving the store model along segments in $\mathbf{R}^3$ at constant angles of attack within the interference flow field of the parent aircraft model, and measuring the loads acting on the store model. At each point along these segments, the interference coefficients are calculated by dividing the measured aerodynamic loads by the freestream loads (measured for the same store model at the same angle of attack and Mach number in the absence of the parent aircraft model). From the above discussion, a grid survey experiment can therefore be interpreted as a set of (segment,store angle of attack,Mach number) that needs to be covered by the store model.

As illustrated in Figure 1.1, the main component of the CTS system is a redundant robotic manipulator, which will be hereinafter referred to as the *CTS manipulator*. The CTS manipulator holds at its end-effector a balance to which the store model is attached.

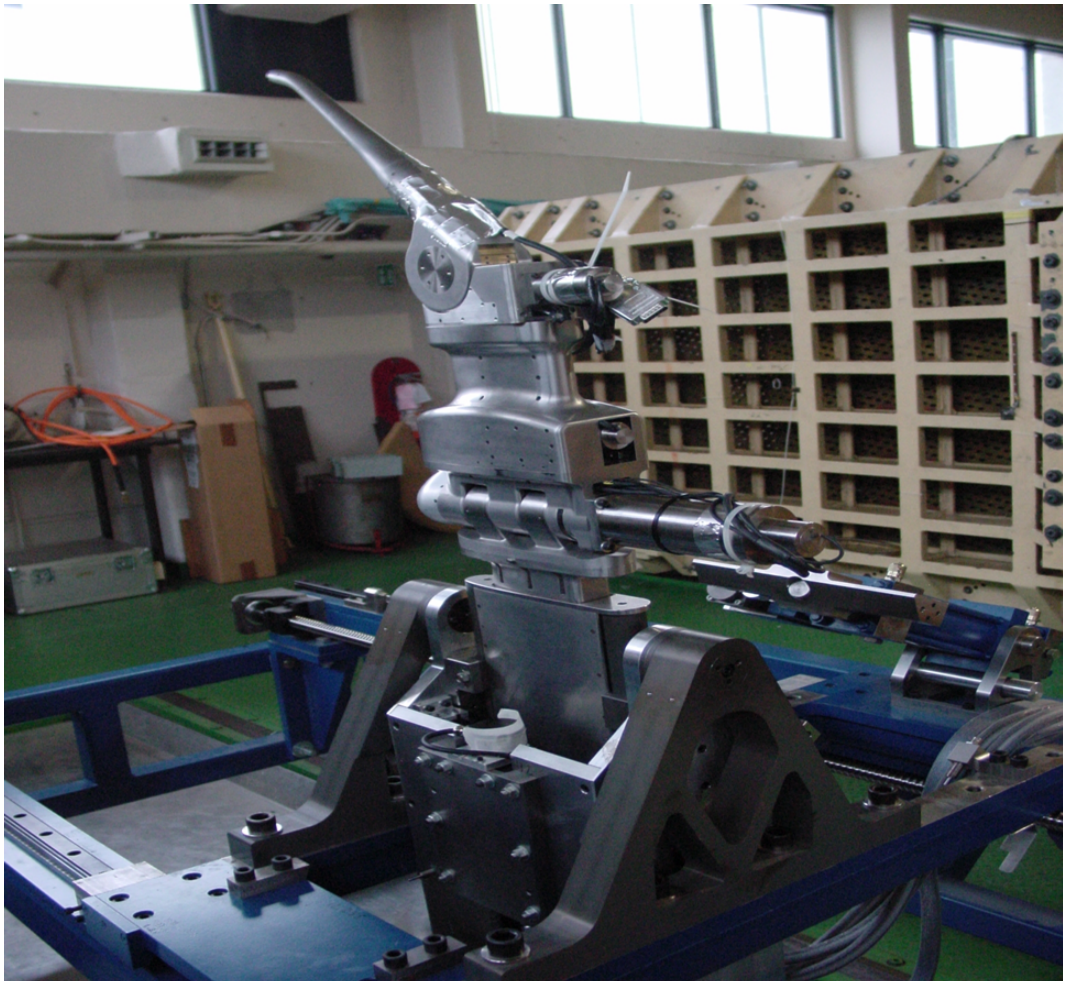


Figure 1.1: The CTS manipulator.

The store model can therefore be positioned and oriented anywhere within the interference flow field of the parent aircraft model where the loads acting on the store model can be measured. The task for the CTS manipulator is to perform a grid survey experiment. Therefore, the CTS trajectory planning problem is the problem of finding a set of actuator or joint trajectories such that the store model covers the set of (segment,store angle of attack,Mach number) prescribed by the given grid survey experiment. From the CTS trajectory planning problem, two subproblems can be identified. The first subproblem is the problem of building a set of store trajectories such that the store model covers the set of (segment,store angle of attack,Mach number) prescribed by the grid survey experiment. This subproblem will be referred to as the *task space CTS trajectory planning problem*. The second subproblem is the problem of building an actuator or a joint trajectory such that the end-effector of the CTS manipulator follows a given store trajectory. This subproblem will be referred to as the *joint space CTS trajectory planning problem*. It will be shown in Chapter 2 that the CTS trajectory planning problem can be solved by solving these two subproblems sequentially.

The joint space CTS trajectory planning problem is discussed first. In this thesis, most of the attention is given to this problem. The main reason is that, as it will be detailed in Chapter 2, the joint space CTS trajectory planning problem is a typical example of multiobjective kinematic trajectory planning problems [9]. Therefore, it is expected that the contributions made in this thesis can be applied to other problems of this type. The first step when formulating a trajectory problem is to choose between a kinematic or a dynamic formulation [9]. With a kinematic formulation, the trajectory planning problem is purely kinematic and its resolution produces a joint trajectory. It is implicitly assumed that joint trajectories can be executed by the actuation system of the robotic manipulator. With a dynamic formulation, the dynamic equations of motion are involved. Both the actuator trajectory and its corresponding joint trajectory are produced. For the CTS trajectory planning problem, an accurate model of the aerodynamic loads acting on the CTS manipulator is not available. Therefore, a kinematic formulation has been preferred. The actuation system of the CTS manipulator has been selected at the design phase of the CTS system [6] such that any joint trajectory satisfying the kinematic constraints in the joint space CTS trajectory planning problem can be executed even in the presence of the worst-case aerodynamic loads.

The next step in the formulation of a trajectory planning problem is to identify the constraints to be satisfied by the trajectories. The constraints in the joint space CTS trajectory planning problem are typical constraints that can be found in kinematic trajectory planning problems [10]. These constraints, which can only be kinematic, include joint speed and joint mechanical limits. Self-collisions between parts of the CTS manipulator and collisions [10, 11] between the CTS manipulator and its surrounding environment, namely the wind-tunnel and the parent aircraft model, must also be avoided.

For redundant robotic manipulators, even in the presence of constraints, the task can generally still be completed with an infinite number of trajectories. The most common method used to choose a final trajectory is to find a trajectory that corresponds to the optimum value of a performance criterion defined over the set of trajectories. Many performance criteria have been considered in the literature [12]. For dynamic trajectory planning problems, some common performance criteria are the final time [13] and the norm or the weighted norm of the joint torque to minimize the energy required by the actuators [11]. For kinematic trajectory planning problems, the norm of the joint speed is generally used instead of the norm of the joint torque [10]. The identification of relevant performance criteria is therefore part of the formulation of a trajectory planning problem. For example, for the joint space CTS trajectory planning problem, it is desirable to have the main body of the CTS manipulator as far as possible from the parent aircraft since its presence disturbs the flow field, and this disturbance adversely affects the measurement of the aerodynamic loads acting on the store.

By choosing a trajectory that corresponds to the optimum of a performance criterion, trajectory planning problems mathematically become extremum problems involving functions. *Optimal control theory* [14], *dynamic programming for continuous-time systems* [15], and *calculus of variations* [16] are the theories that have been developed for solving such problems. Nakamura [17], Martin [18], and Shin [19] have been, respectively, pioneers in applying optimal control theory, calculus of variations, and dynamic programming to the resolution of trajectory planning problems. Nakamura [17] stated the problem of following a given Cartesian trajectory for the end-effector of a redundant robotic manipulator as an optimal control problem. Both kinematic and dynamic formulations for this trajectory planning problem were considered. Martin [18] stated the same kinematic trajectory planning problem as in [17] as a problem in calculus of variations. Shin [19] stated the problem of moving the end-effector of a non-redundant robotic manipulator along a path in $\mathbf{R}^3 \times SO(3)$ under joint torque and joint torque derivative constraints as an optimal control problem. To solve this optimal control problem, discretization in the path parameter variables was first performed. The resulting discrete problem was then solved using dynamic programming.

The main reason why the trajectory planning problems considered in the pioneering work discussed above are rather simple is that problems in calculus of variations and optimal control theory are generally very difficult to solve numerically, particularly in the presence of constraints. As a result, the more recent work in trajectory planning has been more focused on applying existing efficient numerical algorithms [20, 21, 22, 23, 24] to trajectory planning problems with constraints. Galicki [11] stated the problem of moving the end-effector of a redundant robotic manipulator along a path in $\mathbf{R}^3 \times SO(3)$ under joint torque constraints and in the presence of obstacles as an optimal control problem. The proposed numerical algorithm consisted of the resolution of a sequence of linear programs obtained from the discretization of the negative formulation of the Pontryagin's maximum principle. Shen [10] stated the problem of moving a robotic manipulator under kinematic constraints, and with a fixed final time, as a problem in calculus of variations. The proposed numerical algorithm started with a discretization of the Lagrange function which converted the original variational problem into a finite-dimensional optimization problem. Newton's algorithm was then applied to obtain a solution to this finite-dimensional optimization problem. Cahill [13] stated the problem of moving the end-effector of a non-redundant robotic manipulator along a path in $\mathbf{R}^3 \times SO(3)$ under joint torque constraints as an optimal control problem. This optimal control problem was solved using dynamic programming. The first step was to derive the Hamilton-Jacobi-Bellman (HJB) partial differential equation satisfied by the value function of the optimal control problem. An approximate solution to the HJB equation was finally obtained using a finite difference method.

It can be seen that trajectory planning problems have been widely studied. However, it appears that there is one important aspect of these problems that has not received sufficient attention. It is a fact that, for a given trajectory planning problem, many different performance criteria can be considered for the selection of the final trajectory. For the joint space CTS trajectory planning problem, together with the performance criterion mentioned above, it is also desirable for example to operate under low joint speeds. In [10], several performance criteria including the joint speed norm and the joint acceleration norm were also considered. The approach in this paper was to combine linearly the objective functions with positive weights to obtain a new single performance criterion. In the multiobjective optimization literature, this classical approach is referred to as the *weighting method* [25, pp. 78–85]. Different optimal joint trajectories with respect to this new performance criterion could then be obtained by varying the weights. The weighting method seems to be the only approach that has been applied to solve trajectory planning problems with multiple performance criteria, which will be hereinafter referred to as *multiobjective trajectory planning problems.*

1.2 A Multiobjective Approach to Trajectory Planning Problems

In this thesis, the joint space CTS trajectory planning problem is considered as a multiobjective trajectory planning problem. Before defining what an optimal solution for such a problem is, some preliminary definitions are needed. The *objective space* [25, p. 5] is the set of all possible values that can be taken by the p objective functions, or p performance criteria, seen as a vector in $\mathbf{R}^p$. The objective space is therefore a subset of $\mathbf{R}^p$. An element of the objective space is an *objective vector* [25, p. 5]. The concept of optimality in multiobjective optimization problems derives from the choice of a preference in the objective space. The following preference is considered for the joint space CTS trajectory planning problem, which is the natural extension of the "less than or equal to" or "$\leq$" preference when $p = 1$: given two vectors $\mathbf{z}_1$ and $\mathbf{z}_2$ in $\mathbf{R}^p$, $\mathbf{z}_1$ is said to be *preferred* to $\mathbf{z}_2$ if and only if each component of $\mathbf{z}_1$ is less than or equal to its corresponding component of $\mathbf{z}_2$. Therefore, an objective vector $\mathbf{z}_1$ is defined as optimal (with respect to this preference) if there is no other objective vector $\mathbf{z}_2$, $\mathbf{z}_2 \neq \mathbf{z}_1$, that can be preferred to it. This classical concept of optimality is known as *Pareto optimality* in the multiobjective optimization literature [25, 26, 27] and the optimal objective vectors are referred to as *Pareto objective vectors*. The difficulty of multiobjective optimization problems is precisely that the optimal solutions form a set, the *Pareto optimal set*, which does not generally reduce to a singleton, as is the case for single objective optimization problems.

Based on the problem description from Section 1.1, the joint space CTS trajectory planning problem can finally be formulated as a multiobjective problem in calculus of variations. The resolution of this problem therefore consists of determining the Pareto optimal set and the corresponding joint trajectories. The first proposed resolution method in this thesis is the weighting method. The weighting method belongs to the class of scalarization methods [27, pp. 86–101]. The general principle for these methods is to transform the original multiobjective optimization problem into a parameterized single objective optimization problem and vary the parameters to obtain Pareto optimal solutions. For the weighting method, the parameters are the weights, and it can be shown that the objective vector corresponding to the optimal solution of the scalarized problem is Pareto optimal [25, pp. 78–79]. Because of its simplicity, the weighting method is still widely used. However, it suffers from important known weaknesses [28], which will be illustrated with the joint space CTS trajectory planning problem. In particular, it will be shown that the objective space for the joint space CTS trajectory planning problem is not convex, which results that a large subset of the Pareto optimal set cannot be obtained with the weighting method.

One of the main contributions of this thesis is the development of a new discrete dynamic programming (DDP) approximation method to solve the joint space CTS trajectory planning problem. Dynamic programming [29] is a very general principle that applies in particular to variational problems such as trajectory planning problems. Dynamic programming for continuous-time systems, calculus of variations, and optimal control, are theories that are intimately connected [15]. However, the primary focus of calculus of variations and optimal control theory is to find an optimal trajectory. Meanwhile, the primary focus of dynamic programming is to determine the *value function* [15, p. 9] or *return function* [30, p. 129] defined as the optimal value of the objective function as a function of the initial conditions of the trajectories. This is the main reason why dynamic programming is believed to be very suitable for multiobjective optimization problems: the return function takes its value precisely in the space where the preference is defined. The proposed DDP approximation method details as follows. The first step is a first-order discretization in the time variable. For the resulting discrete problem, a *set-valued return function* is defined as the Pareto optimal set as a function of the initial conditions of the discrete joint trajectories. The application of the dynamic programming principle then yields a *multiobjective dynamic programming equation* satisfied by this set-valued function. This equation is finally discretized in the state-space variable from which results an *approximate multiobjective dynamic programming equation* that can be easily solved by performing a finite number of comparisons. In the resolution process of this approximate equation, the optimal discrete joint trajectories corresponding to the approximate Pareto objective vectors are obtained. It will be shown that, when compared to the weighting method, the DDP approximation method provides a much better representation of the Pareto optimal set.

Interestingly, the proposed DDP approximation method can also be applied to the class of multiobjective deterministic finite horizon optimal control problems [31] to which the unconstrained joint space CTS trajectory planning problem can be shown to belong. This method also remains valid for a general class of preferences defined in terms of a pointed convex closed cone $D \subset \mathbf{R}^p$ containing the origin [26]. Pareto optimality, as defined above, is just a particular case of these preferences with $D = \mathbf{R}^p_+$. When considering these preferences, the term *minimal element* will be used instead of Pareto objective vector. Accordingly, the term *minimal element set* will be used instead of Pareto optimal set.

A natural question that arises when deriving approximations is the question of convergence. For the proposed DDP approximation method, this question amounts to determining whether the set of approximate minimal elements converges in some sense towards the minimal element set for the original problem. Note that for this convergence study, it is not assumed that the objective space is convex. Therefore, it is expected that the proposed DDP approximation will provide a good representation of the minimal element set regardless of the convexity of the objective space for the given problem. The first step to answer the question of convergence is to properly define convergence. This is done by introducing a topology on the family of compact sets of $\mathbf{R}^p$ defined from the Hausdorff distance [1]. With this topology, the minimal element map, which is the map that associates with each compact set its minimal element set, is shown to be continuous. Other results related to the existence of minimal elements and the *external stability property* [26, p. 59] for compact sets are also stated. As detailed above for the joint space CTS trajectory planning problem, the DDP approximation method starts with a first-order discretization in time. This discretization yields a discrete multiobjective optimal control problem, called the *discrete problem*. By choosing a particular sequence of time steps, and using the external stability property, it is shown that convergent sequences of minimal elements of the corresponding discrete problems can be constructed. Using the dynamic programming principle, a multiobjective dynamic programming equation with respect to the ordering cone D is then obtained. The solution to this equation is shown to be the minimal element set of the discrete problem. The final step of the DDP approximation method is a state-space discretization of the multiobjective dynamic programming equation. Using the continuity of the minimal element map, the solution to the resulting approximate multiobjective dynamic programming equation is shown to converge towards the minimal element set of the discrete problem in the sense of Hausdorff.

For dynamic programming based resolution methods, such as the proposed DDP approximation method, a well-known limiting factor is the *curse of dimensionality*, which is intrinsic to dynamic programming [29]. For the joint space CTS trajectory planning problem, the curse of dimensionality appears as follows: the algorithmic complexity of the resolution of the approximate multiobjective dynamic programming equation is exponential in the discretization size. This problem can be addressed by *clustering* [32, pp. 325–329] the Pareto optimal set at each step of the resolution of the approximate multiobjective dynamic programming equation, i.e., by substituting into the actual Pareto optimal set one of its subsets. It is proposed to choose for this subset, the closest subset to the actual Pareto optimal set in terms of the Hausdorff distance among all the subsets with a given cardinality. Note that this idea of clustering can also be generalized to the class of multiobjective deterministic finite horizon optimal control problems.

The clustering method based on the Hausdorff distance can be shown to be equivalent to the resolution of the restricted central k-clustering problem for a set of cardinality n included in $\mathbf{R}^p$, which is a well-known problem in geometric location theory [33] [34, pp. 325–327]. Unfortunately, in dimension two and above ($p \geq 2$), the restricted central k-clustering problem is a very difficult problem to solve: it is NP-hard [35, pp. 114–145] for approximation factors of δ in the central cluster size if $\delta < \sqrt{3}$ ([33], Theorem 12). Nevertheless, polynomial algorithms, such as the farthest-point clustering algorithm [34, p. 326] [36] can be used. The farthest-point clustering algorithm, which is the "best" known polynomial algorithm for solving the restricted central k-clustering problem, is a $O(nk)$ algorithm that guarantees a solution at most within two times the minimum central cluster size. The $O(nk)$ complexity was later improved to $O(n \log(k))$ in [33]. Using the farthest-point clustering algorithm, the algorithmic complexity of the proposed DDP approximation method reduces to polynomial, and it will be shown that a dramatic reduction in the resolution time results.

By adding another layer of approximation, it is natural to wonder whether the DDP approximation method with clustering will still provide a better representation of the Pareto optimal set than the weighting method. The difficulty of providing a measure to the quality of a representation of the Pareto optimal set is that the Pareto optimal set for the joint space CTS trajectory planning problem is unknown. Nevertheless, using again the Hausdorff distance, it is possible to provide a lower bound and an upper bound to the quality of a given representation of the Pareto optimal set. By calculating these bounds for the sets obtained with the weighting method and the DDP approximation method, it will be shown that the DDP approximation method with clustering still outperforms the weighting method.

As mentioned above, the task space CTS trajectory planning problem is given less attention in this thesis. Nevertheless, this problem is still extremely important for the operation of the CTS system. Recall that the task space CTS trajectory planning problem is the problem of building a set of store trajectories such that the store model covers the set of (segment,store angle of attack,Mach number) prescribed by the grid survey experiment. Knowing that a store trajectory corresponds to a wind-tunnel run, the objective for this problem is to reduce the operation cost for the CTS system by building the minimum number of store trajectories. The difficulty for the task space CTS trajectory planning problem is that for a given Mach number, the prescribed set of (segment,store angle of attack) is not necessarily connected. Therefore, when building the store trajectories, some extra motions, such as moving the store at a constant angle of attack to connect two segments, or changing the angle of attack at the edge of a segment, might have to be added. An algorithm based on the resolution of a rural postman problem [37] is proposed to solve the task space CTS trajectory planning problem. The motions that need to be added are generated during the resolution process. It will be shown that, compared to the method previously used at IAR to generate store trajectories, the proposed algorithm yields a reduction in the number of wind-tunnel runs, and therefore a reduction in the operation cost for the CTS system.

1.3 Synopsis and Summary of Results

Chapter 2: Problem Definition

The CTS system is first described in detail. Particular attention is given to the CTS manipulator and its kinematics. The CTS trajectory planning problem is discussed next, and it is explained how this problem can be solved by solving sequentially the task space CTS trajectory planning problem and the joint space CTS trajectory planning problem. The formulation for these two problems follows starting with the task space CTS trajectory planning problem. For the joint space CTS trajectory planning problem, a detailed description of the constraints and performance criteria is first provided, and a formulation as a multiobjective problem in calculus of variations is then presented.

Chapter 3: Resolution of the Problem in the Joint Space

The weighting method is first described in general. This method is then applied to the joint space CTS trajectory planning problem. The scalarized problem is a problem in calculus of variations that is solved by solving the boundary value problem (BVP) obtained from the first-order necessary conditions for optimality or Euler-Lagrange equations. The difficulty in evaluating the performance of the weighting method for providing a good representation of the Pareto optimal set is that the Pareto optimal set is not available. Therefore, a method for deriving the boundary of the objective space is proposed. Knowing the boundary of the objective space, the Pareto optimal set is derived. The well-known drawbacks of the weighting method are then illustrated. The DDP approximation method is presented next. For clarity, it is first presented with a single objective, i.e., the scalarized problem obtained from the weighting method. The reasons for proceeding this way are to simplify the presentation of the DDP approximation method, and to validate the results obtained with this method by comparing them to the results obtained by solving the BVP. Finally, the DDP approximation method is applied to the joint space CTS trajectory planning problem and the results are compared to those obtained with the weighting method.

Chapter 4: A Multiobjective Optimal Control Problem

The unconstrained joint space CTS trajectory planning problem is first shown to belong to a class of multiobjective deterministic finite horizon optimal control problems. The DDP approximation method developed for the joint space CTS trajectory planning problem is then generalized to this class of problems. The convergence of the DDP approximation for this class of problems is studied.

Chapter 5: Practical Considerations

The complexity of the resolution of the approximate multiobjective dynamic programming equation obtained from the DDP approximation method for the joint space CTS trajectory planning problem is first determined. It is then shown that with clustering, the algorithmic complexity of the resolution of this equation can be reduced. Three clustering methods are proposed and compared: the first method is a purely random method, the second is inspired from the normal-boundary intersection (NBI) method, and the third method is based on the Hausdorff distance. The problem of giving a measure to the quality of a representation of the Pareto optimal set is then discussed. The weighting method and the DDP approximation method with clustering are then evaluated with this measure and compared. Finally, a description of how continuous trajectories can be derived from the discrete joint trajectories obtained from the resolution of the approximate multiobjective dynamic programming equation is presented.

Chapter 6: Resolution of the Problem in the Task Space

This chapter is entirely dedicated to the task space CTS trajectory planning problem. A detailed formulation for this problem is first provided. Given a grid survey experiment, it is shown how a lower bound on the number of required store trajectories to complete the experiment can be obtained. The relation between the task space CTS trajectory planning problem and a well-chosen rural postman problem is highlighted next. From there, an algorithm to solve the task space CTS trajectory planning problem is proposed. This algorithm provides an upper bound on the number of required store trajectories to complete the experiment. Numerical simulations using a grid survey experiment previously conducted at IAR are finally performed. The number of store trajectories obtained with the algorithm is compared to the number obtained with the method previously used at IAR to generate store trajectories.

Chapter 7: Conclusions and Future Work

The results obtained in this thesis are summarized and the direction proposed for future work is discussed.

Contributions

The new results and new areas of investigation presented in this thesis are summarized below.

1. A new approach for the resolution of multiobjective trajectory planning problems is proposed. Rather than providing a single solution through scalarization, it is proposed to provide the entire Pareto optimal set, or more generally the entire minimal element set when the objective space is partially ordered by a cone.

2. A new DDP approximation method for the class of multiobjective deterministic finite horizon optimal control problems is developed. Partial convergence results of the approximate minimal set obtained from the DDP approximation method towards the original minimal element set are obtained using set convergence in the sense of Hausdorff. These results do not require the assumption that the objective space is convex.

3. The DDP approximation method is applied to the joint space trajectory planning problem arising from a CTS system for which a complete formulation is provided. For this problem, the approximate Pareto optimal set is shown to provide a much better representation of the Pareto optimal set than the set obtained with the weighting method through varying the weights. This example confirms that the weighting method is not an appropriate method to solve multiobjective optimization problems.

4. For the practical applicability of the DDP approximation method, the idea of clustering is introduced. With clustering, the complexity of the method is shown to reduce to polynomial, which therefore yields a dramatic reduction in the resolution time. For the joint space CTS trajectory planning problem, the set obtained with the DDP approximation method with clustering is shown, using a measure derived from

the Hausdorff distance, to still provide a much better representation of the Pareto optimal set than the set obtained with the weighting method.

5. A sequential approach for the resolution of the complete CTS trajectory planning problem is proposed. For the resolution of the task space problem, an algorithm based on the resolution of a rural postman problem is developed. For the resolution of the joint space problem, the DDP approximation method with clustering is used.

Chapter 2

Problem Definition

This chapter begins by describing in Section 2.1 what the CTS system is and what are the two types of experiment that can be performed with such a system. In this thesis, only the grid survey experiment is considered and the related trajectory planning problem is referred to as the *CTS trajectory planning problem*. A detailed kinematic analysis of the CTS manipulator, which is the main component of the CTS system, follows in Section 2.2. Finally, a complete formulation for the CTS trajectory planning problem is provided in Section 2.3.

2.1 The CTS System

2.1.1 Overview

The store release research community defines a *store* as any object that can be released from an aircraft, including bombs, fuel tanks, and missiles. Figure 2.1 presents an example of a store, the MK-83 BSU, which is a free-fall non-guided general-purpose 1,000 pound bomb. The certification of a new store consists of ensuring that the separation of the store from the parent aircraft is safe. In other words, when released in the surrounding

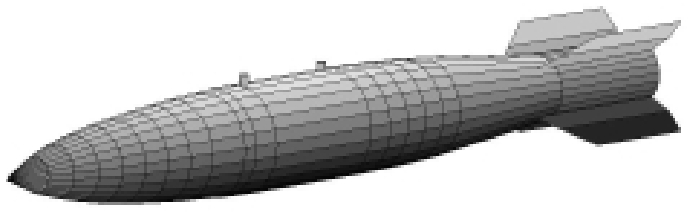


Figure 2.1: Graphical model of the MK-83BSU.

airflow, the store must not collide with any other store attached to the parent aircraft or the parent aircraft itself. The investigation of the store separation can be accomplished with real-flight tests [8]. However, these tests are expensive, take time to perform, and represent a serious risk for the pilot safety. An alternative is to simulate the store release trajectory by integrating the 6-DOF equations of motion of the store [8]. However, this approach is limited by the difficulty in accurately modelling the aerodynamic loads acting on the store within the interference flow field of the parent aircraft, particularly in the transonic flow regime [8]. Therefore, another alternative is to experimentally obtain these loads by performing wind-tunnel tests on a scaled model of the parent aircraft and the store.

The 1.5 m Trisonic Blowdown wind tunnel of IAR is used for the investigation of store separation. A detailed description and the specification of this wind tunnel can be found in [38]. As it will be explained in Section 2.3, two parameters about the wind tunnel, important to the CTS trajectory planning problem, are the run time and the time between two consecutive wind-tunnel runs. The run time of the wind tunnel T_{tunnel}, which depends

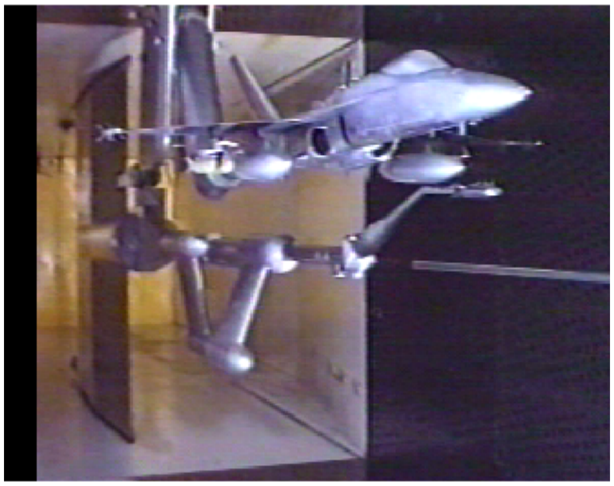


Figure 2.2: The original experimental setup.

on the airflow conditions, is typically 30 s. The time between two consecutive wind-tunnel runs, which corresponds to the time needed to pressurize the plenum chamber of the wind tunnel, is typically 30 min.

Figure 2.2 presents the original wind-tunnel experimental setup designed by IAR to investigate store separation. This system featured a scaled model of the store and the parent aircraft and an articulated sting, which could be described as a 4-DOF kinematic chain with only two DOF being actuated. The parent aircraft model was attached to a sting mounted to the roof of the wind-tunnel test section, while the store model was attached to the extremity of the articulated sting. The articulated sting was instrumented with a 5-DOF balance, which allowed measurement of the forces and moments acting on the store model with the exception of the axial force. Due to its limited positioning and orienting capability, the articulated sting is in the process of being replaced by a fully actuated 8-DOF manipulator, the CTS manipulator, shown in Figure 2.3.

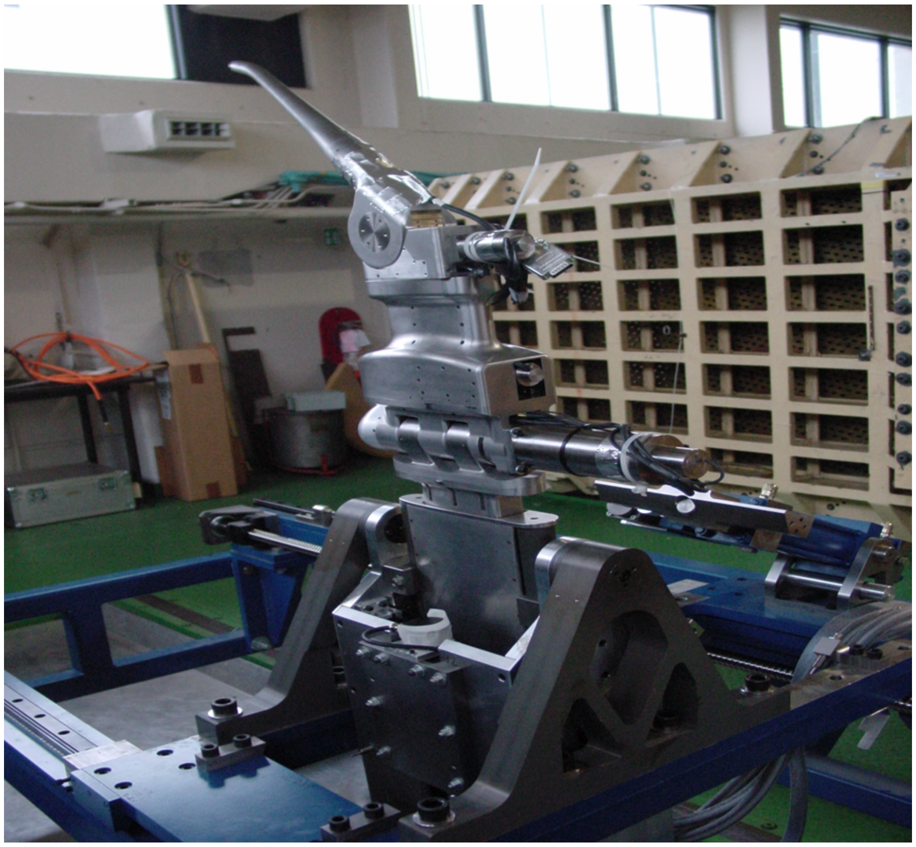


Figure 2.3: The CTS manipulator.

2.1.2 Wind-Tunnel Testing with the CTS System

Two different types of experiment can be performed with the CTS system, namely "grid survey" and "captive load" [8]. Conceptually, a grid survey experiment corresponds to an "open-loop" approach for the simulation of the store release trajectory, while the captive load experiment corresponds to a "closed-loop" approach. Both experiments are described below. However, only the grid survey experiment, being the most commonly performed experiment [8], is considered in this thesis.

A captive load experiment starts with the store model positioned close to the carriage position, which corresponds to the position on the parent aircraft where the store is initially mounted. The forces and moments measured by the balance serve as inputs to the 6-DOF equations of motion of the store. At the initial moment of the separation, the ejection forces are also included. The integration of the equations of motion provides the position and orientation that the store should reach after a small fixed time increment. The CTS manipulator then moves the store model to this new position and orientation. This process is repeated until either a potential collision is detected [4], or the store model has been moved a sufficient distance from the parent aircraft model, or the wind-tunnel run ends.

A grid survey experiment [5] consists of building a look-up table of interference coefficients between the store and the parent aircraft as a function of the store angle of attack, the trajectory angle η, and the Mach number. This look-up table is built by moving the store model along segments at constant velocities and angles of attack within the interference flow field of the parent aircraft. The interference coefficients are then calculated by dividing the measured aerodynamic loads by the freestream loads (measured for the same store model at the same store angle of attack and Mach number in the absence of the parent aircraft model). Once the look-up table is built, the interference coefficients at any store position and orientation and Mach number can be obtained using a multi-dimensional interpolation. The store trajectory can therefore be simulated by solving the same equations of motion as in the captive load experiment. In this case, instead of the measured values, the forces and moments acting on the store at the current position and orientation are calculated by multiplying the freestream aerodynamic coefficients of the store and the interpolated interference coefficients.

Figure 2.4 [5] illustrates a typical example of a grid survey experiment. The objective of this experiment is to obtain the interference coefficients along the segments (1,2), (1,3),

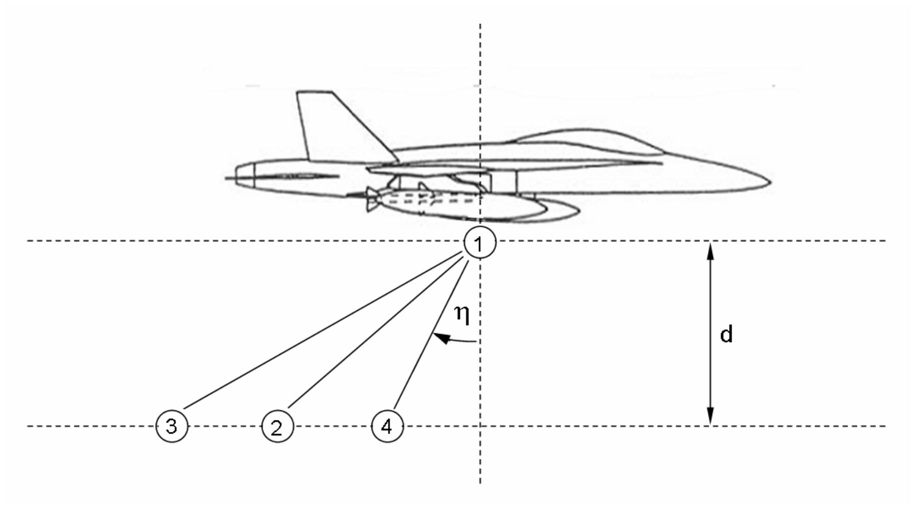


Figure 2.4: An example of a grid survey experiment.

and (1,4) for the store angles of attack $\{0°, -3°, -6°, -9°, -12°, -15°, -18°\}$ and a given Mach number. The segments (1,2), (1,3), and (1,4) correspond to the trajectory angles 25°, 50°, and 3° respectively. The distance d represents the distance after which the store separation is considered safe. With the articulated sting, this experiment was performed as illustrated in Figure 2.5. For each store angle of attack, the store model was moved along the path 1-2-3-1-4. Each motion could be completed in less time than the run time of the wind tunnel. Therefore, seven wind-tunnel runs were needed to perform this grid survey experiment.

The main reason why the grid survey experiment described above was performed as illustrated in Figure 2.5 is that, with the articulated sting, it was not possible to change the store angle of attack during a wind-tunnel run. Therefore, to build a path for the store, either the segment (2,3) or (2,4) could be added. The segment (2,3) was added instead of (2,4) because it is shorter ($|\tan 25° - \tan 3°| < |\tan 50° - \tan 25°|$), and therefore, takes less time to complete. In other words, if the segment (2,4) was added instead of (2,3), it might not have been possible to move the store model along the resulting path in less

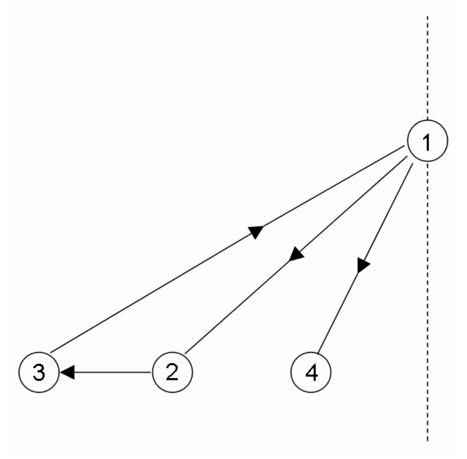


Figure 2.5: The store path 1-2-3-1-4 generated to complete the grid survey experiment.

time than the run time of the wind tunnel. With the CTS manipulator, it is possible to change the store angle of attack during a wind-tunnel run. Therefore, the number of possible paths that can be built to perform the experiment is much larger. Indeed, motions such as, moving the store model along the segment (2,3), changing the store angle of attack at position 3, and moving back along the segment (2,3), now become available. From this added flexibility might result a reduction in the number of wind-tunnel runs needed to complete a grid survey experiment. As discussed in Section 2.3, it is precisely the objective of the CTS trajectory planning problem to address this question.

2.2 The CTS Manipulator

2.2.1 Design

From the capabilities of the existing CTS systems around the world, an exhaustive list of specifications for the CTS manipulator was defined by IAR. Two of these specifications directly concern the CTS trajectory planning problem: the operating envelope $\mathcal{W}$ of the CTS manipulator and the maximum end-effector velocity within this envelope. The operating envelope, provided in Table 2.1, is defined relative to the carriage position and corresponds to the set of positions and orientations that the end-effector must be able to reach.

Table 2.1: Operating envelope and end-effector velocity specification.

Axial	Vertical	Lateral	Roll	Pitch	Yaw
± 15 in	+3, -18 in	+5, -25 in	± 180 deg	± 30 deg	± 35 deg
5 in$\cdot$s^{-1}	2 in$\cdot$s^{-1}	2 in$\cdot$s^{-1}	20 deg$\cdot$s^{-1}	2 deg$\cdot$s^{-1}	2 deg$\cdot$s^{-1}

The maximum end-effector velocity, also provided in Table 2.1, corresponds to the maximum velocity at which the store model can be moved during a grid survey experiment. It was chosen such that, starting from the carriage position, the end-effector could reach any position and orientation within $\mathcal{W}$ in less time than the run time of the wind tunnel. The resulting design for the CTS manipulator [6] is described in Section 2.2.2. The joint mechanical limits $\mathbf{q}_{\text{min}}$ and $\mathbf{q}_{\text{max}}$ were selected to satisfy the operating envelope specification. Let $\mathcal{Q} = [\mathbf{q}_{\text{min}}, \mathbf{q}_{\text{max}}]$ be the set of joint configurations within the joint mechanical limits. The actuation system was chosen to be able to achieve the maximum joint speed $\dot{\mathbf{q}}_{\text{max}}$, which guarantees that the maximum end-effector velocity specification is satisfied anywhere within $\mathcal{W}$ [39].

2.2.2 Description

The CTS manipulator, presented in Figure 2.6, is an 8-DOF manipulator. The joint arrangement of the CTS manipulator, provided in Table 2.2, consists of a prismatic joint at the base followed by a revolute joint connected to two more prismatic joints. The last four joints are all revolute, and mainly contribute to the orienting of the end-effector.

The prismatic joints 3 and 4 form a telescopic arrangement resulting from design con-

Table 2.2: Joint type for the CTS manipulator. R for revolute or P for prismatic.

Joint	1	2	3	4	5	6	7	8
Type	P	R	P	P	R	R	R	R

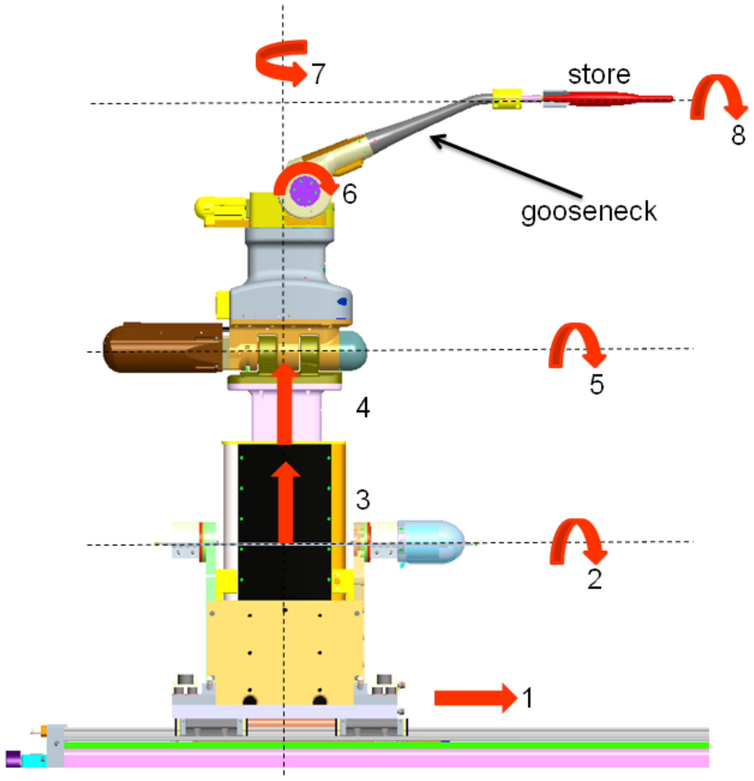


Figure 2.6: Joint motions for the CTS manipulator.

straints [6]. These two joints are therefore kinematically equivalent to a single prismatic joint and will be considered as such in this thesis. With this simplification, the number of DOF for the CTS manipulator reduces to 7. The joint configuration $\mathbf{q}$, joint speed $\dot{\mathbf{q}}$, and joint acceleration $\ddot{\mathbf{q}}$ become 7-dimensional vectors. However, for clarity, the index of the joint will be preserved.

As simplified, the CTS manipulator is a kinematically redundant manipulator with only one degree of redundancy [3]. Within $\mathcal{W}$, the CTS manipulator therefore has the capability of positioning and orienting the store model at a given position and orientation with an infinite number of joint configurations. An illustration of this fact is provided in Figure 2.7. Redundancy tends to considerably complicate the resolution of trajectory

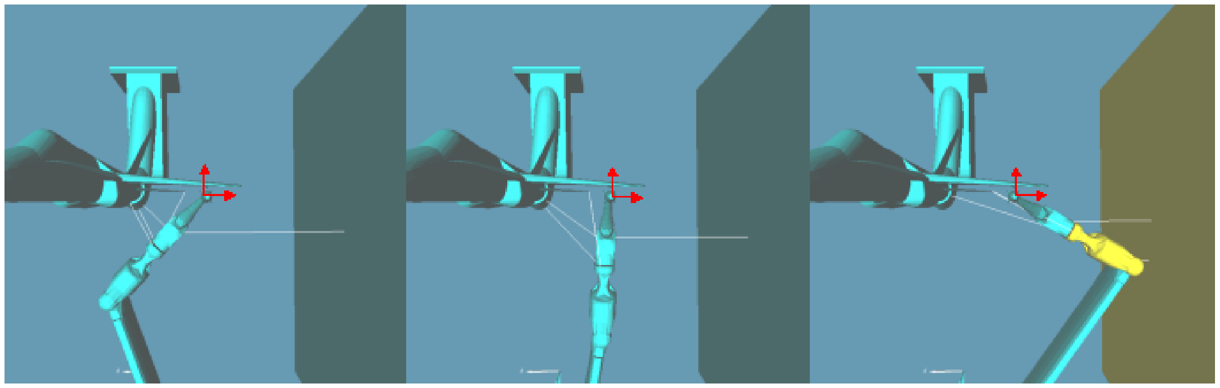


Figure 2.7: Three different joint configurations for the same store position and orientation.

planning problems by adding the problem of choosing a joint configuration. On the other hand, redundancy provides freedom in the joint motion and this is precisely the reason why it was chosen to design a redundant manipulator for the CTS system [6]. This freedom can be used in many ways. For example, in Figure 2.7, the first two joint configurations are preferable to the last one which brings the CTS manipulator too close to the walls of the wind tunnel. Between the first and the second position, the first one is worse as it brings the CTS manipulator closer to the parent aircraft model. It is known that the presence of the CTS manipulator disturbs the interference flow field of the parent aircraft model, and this disturbance affects the aerodynamic loads acting on the store model.

2.2.3 Forward Kinematics

Forward Kinematics at the Position Level

To describe the position and orientation of the end-effector of the CTS manipulator, two frames, the tool frame and the base frame, are needed [2, p. 5]. The tool frame is attached to the end-effector while the base frame is attached to the non-moving part of the CTS manipulator. These two frames are represented in Figure 2.8 by the frames indexed 0 to 8 respectively. The position and orientation of the end-effector therefore correspond to the relative position and orientation of the tool frame with respect to the base frame. In this thesis, the convention used to describe the orientation is the Z-Y-X Euler angles (roll (γ), pitch (β), yaw (α)) convention [2, p. 41]. Let $\mathbf{p} = (x, y, z, \gamma, \beta, \alpha)^T$ be the 6-dimensional vector representing the position and orientation of the end-effector.

The forward kinematics equation provides the position and orientation of the end-effector $\mathbf{p}$, given the joint configuration $\mathbf{q}$:

$$\mathbf{f}(\mathbf{q}) = \mathbf{p}. \tag{2.1}$$

The most common method to derive this highly nonlinear equation is to use the Denavit-Hartenberg (DH) convention [40, p. 38]. The DH convention provides a systematic way of attaching frames to the links. The relative position and orientation between two neighbouring links is described with four parameters, the DH parameters, which include the joint configuration. The frames assigned by the DH convention to the 8-DOF CTS manipulator are illustrated in Figures 2.8, A.1, and A.2. The corresponding set of DH parameters is provided in Appendix A.

Forward Kinematics at the Velocity Level

The forward kinematics equation at the velocity level provides the end-effector velocity $\dot{\mathbf{p}} = (\dot{x}, \dot{y}, \dot{z}, \dot{\gamma}, \dot{\beta}, \dot{\alpha})^T$, given the joint speed $\dot{\mathbf{q}}$:

$$\widetilde{\mathbf{J}}(\mathbf{q})\dot{\mathbf{q}} = \dot{\mathbf{p}}, \tag{2.2}$$

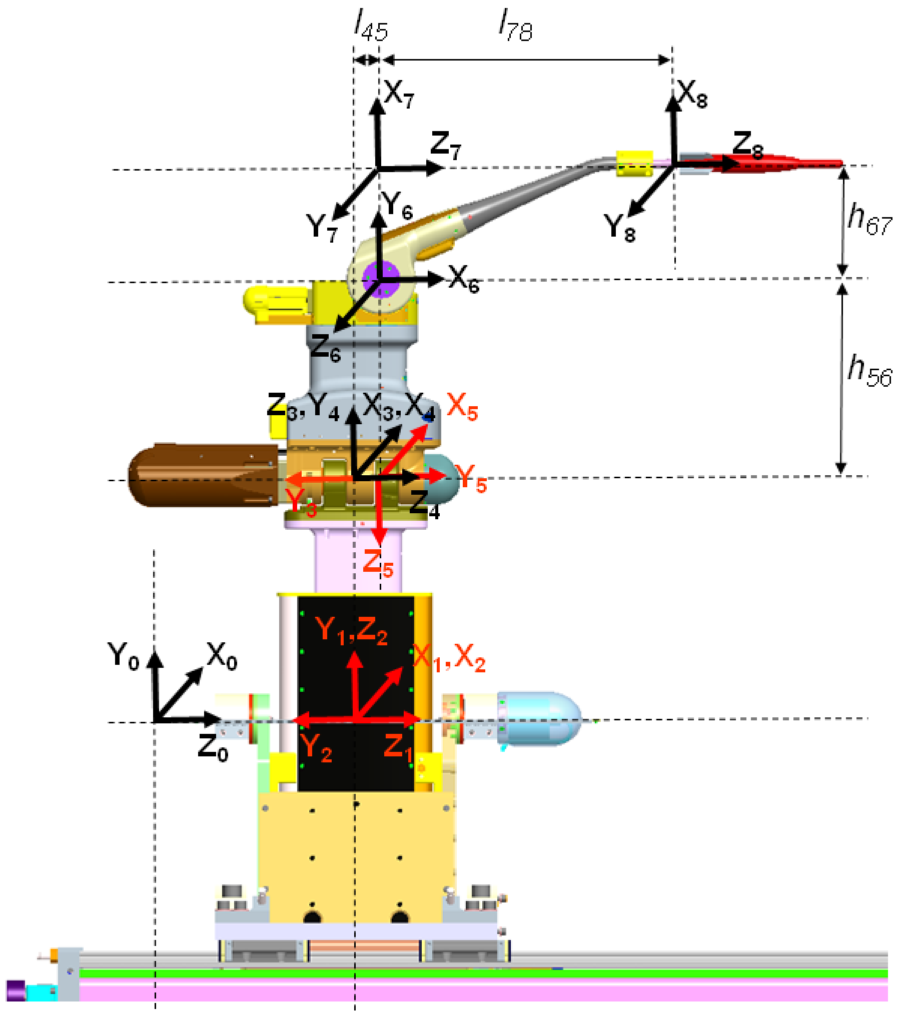


Figure 2.8: The frames obtained from the DH convention.

where $\widetilde{\mathbf{J}}(\mathbf{q})$ is a 6×7 matrix. This equation can be derived in two steps. The first step is to obtain the relation between the vector $\dot{\tilde{\mathbf{p}}} = (\dot{x}, \dot{y}, \dot{z}, w_x, w_y, w_z)^T$ of the linear $(\dot{x}, \dot{y}, \dot{z})$ and angular (w_x, w_y, w_z) end-effector velocity and the joint speed [40, p. 69]. This relation involves the 6×7 Jacobian matrix $\mathbf{J}(\mathbf{q})$ of the CTS manipulator:

$$\mathbf{J}(\mathbf{q})\dot{\mathbf{q}} = \dot{\tilde{\mathbf{p}}}. \tag{2.3}$$

The second step is to derive the relation between $\dot{\tilde{\mathbf{p}}}$ and $\dot{\mathbf{p}}$:

$$\dot{\tilde{\mathbf{p}}} = \widetilde{\mathbf{J}}_{\mathbf{A}}(\mathbf{p})\dot{\mathbf{p}}, \tag{2.4}$$

where the 6×6 matrix $\widetilde{\mathbf{J}}_{\mathbf{A}}(\mathbf{p})$ is given by

$$\widetilde{\mathbf{J}}_{\mathbf{A}}(\mathbf{p}) = \begin{bmatrix} \mathbf{I}_3 & 0 \\ 0 & \mathbf{J}_{\mathbf{A}}(\mathbf{p}) \end{bmatrix},$$

and $\mathbf{J}_{\mathbf{A}}(\mathbf{p})$ is the 3×3 matrix that transforms the Euler rates to the angular velocity [41, p. 44]:

$$\mathbf{J}_{\mathbf{A}}(\mathbf{p}) = \begin{bmatrix} 1 & 0 & -\sin(\beta) \\ 0 & \cos(\gamma) & \sin(\gamma)\cos(\beta) \\ 0 & -\sin(\gamma) & \cos(\gamma)\cos(\beta) \end{bmatrix}.$$

From the operating envelope specification provided in Table 2.1, it follows that

$$\det(\mathbf{J}_{\mathbf{A}}(\mathbf{p})) = \cos(\beta) \neq 0.$$

Hence, the matrix $\widetilde{\mathbf{J}}_{\mathbf{A}}(\mathbf{p})$ is always invertible. Therefore, combining (2.3) [1] and (2.4) yields

$$\widetilde{\mathbf{J}}(\mathbf{q}) = \widetilde{\mathbf{J}}_{\mathbf{A}}^{-1}(\mathbf{p})\mathbf{J}(\mathbf{q}). \tag{2.5}$$

2.2.4 Singularities

A singularity for a robotic manipulator [2, p. 151] is a joint configuration where the manipulator Jacobian matrix loses rank. As a consequence, at a singularity, there is at least one direction in the task space in which the end-effector cannot move. Another conse-

[1] In this thesis, equations will be referred with respect to SIAM's TEX and File Submission Guidelines, e.g., (2.3) will be used instead of Equation (2.3).

quence is that, near a singularity, very large joint speeds are required to achieve very small end-effector velocities in some direction in the task space. The presence of singularities is therefore an important problem, which has been investigated extensively in the early robotics literature [42]. The CTS manipulator has been designed such that the singularities are outside the joint mechanical limits [6]. Therefore, issues related to singularities, such as singularity avoidance [43], are not included in the CTS trajectory planning problem.

2.2.5 Inverse Kinematics

Inverse Kinematics at the Position Level

At the design stage [6], the joint arrangement of the CTS manipulator was chosen such that a closed-form solution to the inverse kinematics problem [2, p. 101] in $\mathcal{W}$ could be derived. The inverse kinematics equation, detailed in Appendix B, can be formally stated as follows:

$$\mathbf{q} = \mathbf{g}(\mathbf{p}, \upsilon), \tag{2.6}$$

where $\mathbf{q} \in \mathcal{Q}$, $\mathbf{p} \in \mathcal{W}$, and the scalar $\upsilon = \mathbf{q}_2 + \mathbf{q}_5$ is the redundancy parameter. The set of values $\mathcal{A}(\mathbf{p})$ that can be taken by υ for a given end-effector position and orientation $\mathbf{p}$ is a function of $\mathbf{p}$. From the definition of the redundancy parameter, it follows that $\mathcal{A}(\mathbf{p}) \subset [\mathbf{q}_{\min,2} + \mathbf{q}_{\min,5}, \mathbf{q}_{\max,2} + \mathbf{q}_{\max,5}]$, denoted $[\upsilon_{\min}, \upsilon_{\max}]$.

Inverse Kinematics at the Velocity Level

The function $\mathbf{g}(\cdot, \cdot)$ being smooth means (2.6) can be differentiated with respect to time, yielding

$$\dot{\mathbf{q}} = \widetilde{\mathbf{G}}(\mathbf{q})\dot{\mathbf{p}} + \mathbf{N}(\mathbf{q})\dot{\upsilon}, \tag{2.7}$$

where

$$\widetilde{\mathbf{G}}(\mathbf{q}) = \mathbf{G}(\mathbf{q})\widetilde{\mathbf{J}}_{\mathbf{A}}, \tag{2.8}$$

and $\mathbf{G}(\mathbf{q})$ is a 7×6 matrix and $\mathbf{N}(\mathbf{q})$ a 7-dimensional vector. The matrix $\mathbf{G}(\mathbf{q})$ is a generalized inverse of the manipulator Jacobian matrix $\mathbf{J}(\mathbf{q})$:

$$\mathbf{J}(\mathbf{q})\mathbf{G}(\mathbf{q}) = \mathbf{I}_7,$$

where $\mathbf{I}_7$ is the 7×7 identity matrix. The vector $\mathbf{N}(\mathbf{q})$ represents the null space of the manipulator Jacobian matrix $\mathbf{J}(\mathbf{q})$:

$$\mathbf{J}(\mathbf{q})\mathbf{N}(\mathbf{q}) = \mathbf{0}_7,$$

where $\mathbf{0}_7$ is the 7-dimensional vector whose components are all identically zero.

To accelerate the computations in the resolution of the CTS trajectory planning problem, the forward kinematics equation (2.1), the matrices $\mathbf{J}(\mathbf{q})$ and $\mathbf{G}(\mathbf{q})$, and the vector $\mathbf{N}(\mathbf{q})$ have been derived analytically with the symbolic computer algebra software Maple [44].

2.3 The CTS Trajectory Planning Problem

2.3.1 A Sequential Approach to the Resolution of the CTS Trajectory Planning Problem

Based on the example provided in Section 2.1.2, a grid survey experiment can be formally defined as follows. Assume the Mach number fixed. At this Mach number, let $\mathcal{A} = \{A_i,\ i = 1, \ldots, I\}$ be a set of positions defined relative to the parent aircraft model, and $\mathcal{B} = \{\beta_j,\ j = 1, \ldots, J\}$ be a set of store angle of attacks. A grid survey experiment consists of moving the store along a set of segments $\mathcal{K} = \{(A_i, A_{i'})\}$ (by convention, $i' > i$).

Along each of these segments, the store angle of attack is constant and is taken to be in the set $\mathcal{B}_{i,i'}$. The velocity is also assumed to be constant. Note that the set $\mathcal{B}_{i,i'}$ does not need to be the same for each segment and does not need to be equal to $\mathcal{B}$. Note also that the set $\mathcal{K}$ does not need to be the set of all possible segments between the points A_i. For example, for the grid survey experiment presented in Section 2.1.2, $\mathcal{A} = \{1, 2, 3, 4\}$, $\mathcal{B} = \{0°, -3°, -6°, -9°, -12°, -15°, -18°\}$, $\mathcal{K} = \{(1, 2), (1, 3), (1, 4)\}$ and it happens that $\mathcal{B}_{i,i'} = \mathcal{B}$ for all the segments in $\mathcal{K}$. If a subtask is defined as the task of moving the store model from A_i with an angle of attack β_j to $A_{i'}$ with an angle of attack $\beta_{j'}$, or equivalently by the quadruplet (i, j, i', j'), then a grid survey experiment consists of completing all the subtasks (i, j, i', j') that satisfy $(A_i, A_{i'}) \in \mathcal{K}$ and $j' = j$ with $\beta_j \in \mathcal{B}_{i,i'}$. The set of such subtasks is denoted $\mathcal{S}$.

For a given grid survey experiment as defined above, the CTS trajectory planning problem, denoted PCTS, can therefore be defined as follows.

Problem PCTS: Find the minimum number of joint trajectories such that all the subtasks in $\mathcal{S}$ are completed under the constraint that the duration of each joint trajectory is less than the run time of the wind tunnel.

The first important difficulty in PCTS relates to the task space. Indeed, the set of segments $\mathcal{K}$ being not necessarily connected, it might therefore be required to add some idle motions to move the store from one segment of the grid to another segment. These idle motions correspond to subtasks that do not belong to $\mathcal{S}$. For example, for the grid survey experiment presented in Section 2.1.2, seven subtasks, i.e., $\{(2, j, 3, j),\ j = 1, \ldots, J\}$ were added. Moreover, as mentioned in Section 2.1.2, it is now possible with the CTS manipulator to change the angle of attack from β_j to $\beta_{j'}$ at a position A_i. These other idle motions correspond to the subtasks (i, j, i, j') which also do not belong to $\mathcal{S}$. Adding these subtasks obviously has a cost in time, but overall might result in a reduction in the number of joint trajectories needed to complete $\mathcal{S}$. Therefore, they have to be considered. The problem of finding the minimum number of store trajectories such that all the subtasks in $\mathcal{S}$ are completed is referred to as the *task space CTS trajectory planning problem* or PCTS_1.

Problem PCTS_1: Find the minimum number of store trajectories such that all the subtasks in $\mathcal{S}$ are completed under the constraint that the duration of each store trajectory is less than the run time of the wind tunnel.

The second important difficulty in PCTS relates to the joint space. As the CTS manipulator is redundant, there is an infinite number of joint trajectories such that the end-effector follows a given trajectory. The problem of finding a joint trajectory given a store trajectory is referred to as the *joint space CTS trajectory planning problem* or PCTS_2.

Problem PCTS_2: Find a joint trajectory such that the end-effector follows a given store trajectory.

The resolution approach to PCTS consists of solving PCTS_1 and PCTS_2 sequentially. The main motivation for choosing such a simplified approach concerns the resolution time, which must be less than the time between two consecutive wind-tunnel runs. Indeed, it may very well happen that, based on the results obtained during a wind-tunnel run, it becomes necessary to modify the grid [5]. The velocity used to plan the store trajectories in PCTS_1 is the end-effector velocity specified in Table 2.1. From the design, it is therefore guaranteed that there will always be a solution to PCTS_2.

2.3.2 The Store Trajectory in PCTS$_1$

As mentioned in Section 2.3.1, the velocity for each subtask in $\mathcal{S}$ and each subtask that eventually could be added is constant and equal to the end-effector velocity $\dot{\mathbf{p}}_{\mathrm{max}}$ specified in Table 2.1. The trajectory for each subtask is therefore known. A store trajectory is then obtained by arranging sequentially the individual subtask trajectories for all the subtasks that should be completed by the store trajectory. However, because the velocity for each subtask is different and the store must be moved along the store path, it is necessary that the velocity at the beginning and the end of each subtask trajectory is zero. How to build such a trajectory is now discussed.

Formally, a subtask consists of moving the store from an initial position and orientation $\mathbf{p_0}$ at t_0 to a final position and orientation $\mathbf{p_f}$ at the constant velocity $\dot{\mathbf{p}}_{\mathrm{max}}$. This motion can be realized with the following trajectory:

$$\mathbf{p}(t) = \mathbf{p_0} + \frac{t - t_0}{t_f - t_0}(\mathbf{p_f} - \mathbf{p_0}), \tag{2.9}$$

where

$$t_f = t_0 + \max\left\{\left|\frac{\mathbf{p_f}_i - \mathbf{p_0}_i}{\dot{\mathbf{p}}_{\mathrm{max}_i}}\right|,\ i = 1, \ldots, 6\right\}. \tag{2.10}$$

However, the initial and final velocities are not zero for the trajectory (2.9). This problem can be addressed by adding two parabolic blends at t_0 and t_f [2, p. 210]. As a consequence, t_f increases. With these modified subtask trajectories, the velocity along the resulting store trajectory is therefore continuous and the initial and final velocities are zero. However, considering the high aerodynamic loads acting on the CTS manipulator during a wind-tunnel run, it is preferred to have an even more regular store trajectory, i.e., a trajectory with a continuous acceleration and zero initial and final acceleration. It is detailed in Appendix C how the subtask trajectories can be built to obtain such a store trajectory.

Therefore, for the resolution of PCTS$_1$, which will be performed in Chapter 6, it will be assumed that, for each subtask in $\mathcal{S}$ and each subtask that eventually could be added, the trajectory and the duration of this trajectory are known.

2.3.3 PCTS$_2$

Given a store trajectory obtained from the resolution of PCTS$_1$, the problem PCTS$_2$ is to find a corresponding joint trajectory $\mathbf{q}(\cdot)$. Even in the presence of constraints, the number of joint trajectories for which the end-effector follows the prescribed store trajectory is infinite. The final joint trajectory can then be selected as the optimal joint trajectory with respect to an objective function, in which case PCTS$_2$ becomes a problem in calculus of variations. However, several objective functions can be considered for the choice of the final joint trajectory.

The Constraints

The joint trajectory is subject to four different constraints.

1. The forward kinematics equation (2.1) imposes that

$$\mathbf{f}(\mathbf{q}(t)) = \mathbf{p}(t). \tag{2.11}$$

2. The joint configuration is limited by the joint mechanical limits:

$$\mathbf{q}_{\text{min}} \leq \mathbf{q}(t) \leq \mathbf{q}_{\text{max}}. \tag{2.12}$$

3. The joint speed is limited by the maximum joint speeds:

$$-\dot{\mathbf{q}}_{\max} \leq \dot{\mathbf{q}}(t) \leq \dot{\mathbf{q}}_{\max}. \tag{2.13}$$

4. Self-collision between parts of the CTS manipulator must be avoided. The danger of self-collision between the CTS manipulator links was addressed during the design stage of the CTS manipulator [6] through a proper selection of the joint mechanical limits. For the collision between the CTS manipulator and its surrounding environment, an efficient distance determination algorithm for the CTS system was proposed in [4]. This algorithm starts by including the CAD-generated geometrical models of the CTS manipulator links, the parent aircraft model, and the wind tunnel into a set of convex sub-hulls. From there, the minimum distance between any CTS manipulator link geometry and any parent aircraft model or wind-tunnel geometry can be calculated using any convex distance determination algorithm. Therefore, the collision constraint is a one-dimensional inequality constraint which can be written as

$$\mathrm{d}_{\text{safe}} - c(\mathbf{q}(t)) \leq 0, \tag{2.14}$$

where $c(\cdot)$ is a function that returns the minimum distance between the CTS manipulator and its surrounding environment for any given joint configuration and d_{safe} is a safety distance defined by the operator of the CTS system.

From the design of the CTS manipulator [6], it is known that there exists at least one joint trajectory satisfying the constraints (2.11)-(2.14).

The Objective Functions

As the CTS manipulator is redundant, the number of joint trajectories satisfying the constraints (2.11)-(2.14) is infinite. The final joint trajectory can then be selected as the optimal joint trajectory with respect to an objective function. A detailed review of the objective functions commonly used in the robotic manipulation literature as well as a new categorization of these objective functions can be found in [12]. In general, the choice of the objective functions depends on the application. For a grid survey experiment, at least two objective functions have been identified.

- To obtain a good tracking performance from the CTS manipulator control system, particularly at large Mach numbers, it is highly desirable for the CTS manipulator to operate under low joint speeds. Therefore, the first objective function $f_1(\cdot)$ to be considered for PCTS$_2$ is the joint speed norm, given by

  $$f_1(\mathbf{q}(t)) = \frac{1}{2}\langle \dot{\mathbf{q}}(t), \dot{\mathbf{q}}(t) \rangle, \tag{2.15}$$

  where $\langle \cdot, \cdot \rangle$ denotes the scalar product in $\mathbf{R}^7$. The weighted joint speed norm can also be considered instead of the joint speed norm, given by

  $$\frac{1}{2}\langle \dot{\mathbf{q}}(t), \mathbf{W}\dot{\mathbf{q}}(t) \rangle, \tag{2.16}$$

  where $\mathbf{W}$ is 7×7 positive definite symmetric matrix. The matrix $\mathbf{W}$ is used for normalization purposes, and allows taking into account the fact that the joints can be different in nature (revolute or prismatic) and have different speed ranges.

- The presence of the CTS manipulator close to the parent aircraft is highly undesirable as it disturbs the interference flow field, which affects the reliability of the measurement of the aerodynamic loads acting on the store model. This presence also imposes additional loads on the CTS manipulator links. This effect can be reduced by maintaining the last three links of the manipulator as close as possible to the vertical. As illustrated in Figure 2.9, this amounts to minimizing the angle between the plane P passing through the origins O_6, O_7, and O_8 of the frames 6, 7, and 8 defined in Figure 2.8 and the vertical plane, i.e., the plane with normal $\mathbf{X_0}$. The

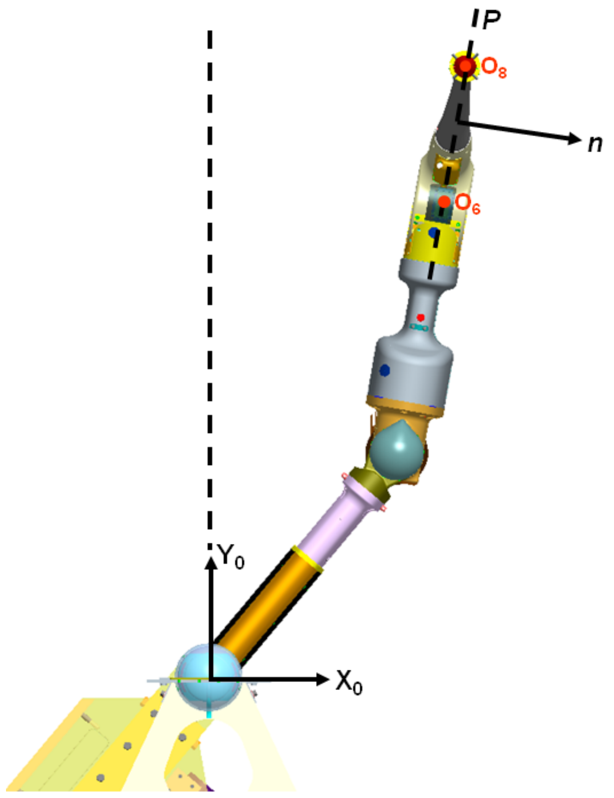


Figure 2.9: Illustration of the plane P.

normal $\mathbf{n}$ to the plane P is

$$\mathbf{n} = \frac{\overrightarrow{O_7O_8} \times \overrightarrow{O_7O_6}}{\|\overrightarrow{O_7O_8} \times \overrightarrow{O_7O_6}\|} = \begin{pmatrix} \cos(\mathbf{q}_6)\cos(\mathbf{q}_2+\mathbf{q}_5) \\ \cos(\mathbf{q}_6)\sin(\mathbf{q}_2+\mathbf{q}_5) \\ \sin(\mathbf{q}_6) \end{pmatrix},$$

where the coordinates of the origins O_6, O_7, and O_8 can be obtained in the frame 0 with the forward kinematics. Therefore, the angle between the plane P and the vertical plane is $\arccos(\langle \mathbf{X_0}, \mathbf{n} \rangle) = \arccos(\cos(\mathbf{q}_6)\cos(\mathbf{q}_2 + \mathbf{q}_5))$. Note that in Figure 2.9, $\mathbf{q}_6$ is equal to zero, in which case the angle between the plane P and the vertical plane is directly given by $\mathbf{q}_2 + \mathbf{q}_5 \bmod \pi$. The second objective function $f_2(\cdot)$ to be considered for PCTS$_2$ is the aerodynamic interference function defined by

$$f_2(\mathbf{q}) = \arccos(\cos(\mathbf{q}_6)\cos(\mathbf{q}_2 + \mathbf{q}_5)).$$

Figure 2.10 illustrates that the aerodynamic interference function has only one minimum in $\mathcal{Q}$. This minimum is zero and is reached when both $\mathbf{q}_2 + \mathbf{q}_5$ and $\mathbf{q}_6$ are zero. When both $\mathbf{q}_2 + \mathbf{q}_5$ and $\mathbf{q}_6$ are zero, the last three links of the CTS manipulator are vertical as desired.

Having described the constraints and the objective functions appearing in PCTS$_2$, the concept of solution for optimization problems involving multiple objective functions must be investigated before being able to finally formulate PCTS$_2$.

The Definition of Optimality in Multiobjective Optimization

Let $\mathbf{F}(\cdot) = (\mathbf{F}_1(\cdot), \ldots, \mathbf{F}_p(\cdot))$ be a real vector-valued function defined on a subset K of a normed linear space X. The *objective space* Z is defined as the set $\mathbf{F}(K) \subset \mathbf{R}^p$. An *objective vector* is an element of the objective space. The definition of an *optimal objective vector* derives from the choice of a preference in the objective space. In a general setting, preferences can be represented by a binary relation denoted $\succ$ [26, p. 25]. Let $\mathbf{z_1},\ \mathbf{z_2} \in Z$, $\mathbf{z_1} \succ \mathbf{z_2}$ implies that the objective vector $\mathbf{z_1}$ is *preferred* to $\mathbf{z_2}$.

Example 2.1. Assume $p = 1$. The function $\mathbf{F}(\cdot)$ is then a real-valued function, denoted $F(\cdot)$. Two examples of binary relation are:

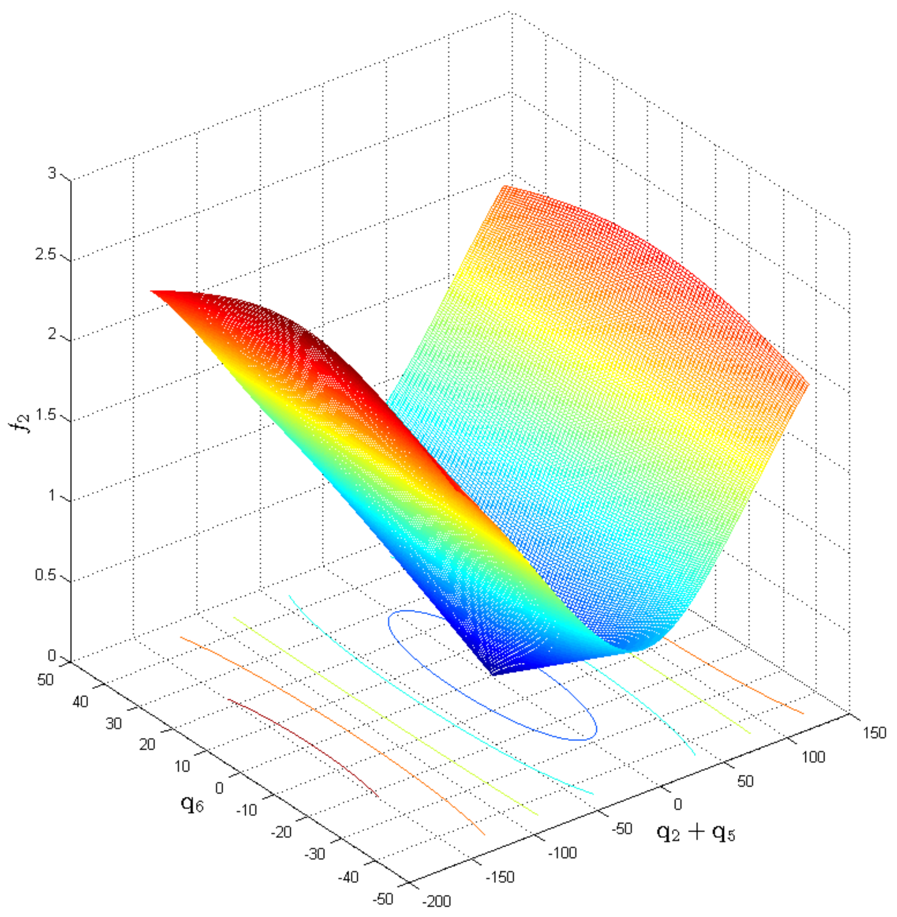


Figure 2.10: The aerodynamic interference function as a function of $\mathbf{q}_6$ and $\mathbf{q}_2 + \mathbf{q}_5$.

- Let $z_1,\ z_2 \in F(K)$, $z_1 \succ z_2$ if $z_1 \leq z_2$. Therefore, a lower value for the objective function is preferrable.

- Let $z_1,\ z_2 \in F(K)$, $z_1 \succ z_2$ if $z_1 \geq z_2$. Therefore, a higher value for the objective function is preferrable.

Example 2.2 (Pareto optimality [27, p. 39], [25, p. 10], [26, p. 30])**.** Let $\mathbf{z_1},\ \mathbf{z_2} \in Z$, $\mathbf{z_1} \succ \mathbf{z_2}$ if $\forall i = 1, \ldots, p,\ \mathbf{z_1}_i \leq \mathbf{z_2}_i$. Therefore, a lower value for all the objective functions is preferrable. Pareto optimality is by far the most used binary relation in multiobjective optimization.

Example 2.3 (Lexicographic order [26, p. 31])**.** Let $\mathbf{z_1},\ \mathbf{z_2} \in Z$, $\mathbf{z_1} \succ \mathbf{z_2}$ if $\exists k \in \{1,\ldots,p\}$ such that $\forall i < k,\ \mathbf{z_1}_i = \mathbf{z_2}_i$, and $\mathbf{z_1}_k < \mathbf{z_2}_k$.

Example 2.4 (Partial order generated by a cone $D \subset \mathbf{R}^p$ [45])**.** By definition, a set $D \subset \mathbf{R}^p$ is a cone if $\lambda D = D$, for every $\lambda \in \mathbf{R}$, $\lambda > 0$ [26, p. 7]. A cone D is said to be pointed if $D \cap -D \subset \{0\}$ [26, p. 7]. It is possible to define a binary relation in terms of a cone D. Let $\mathbf{z_1},\ \mathbf{z_2} \in Z$, $\mathbf{z_1} \succ \mathbf{z_2}$ if $\mathbf{z_2} \in \mathbf{z_1} + D$. Conditions on D can be imposed such that this binary relation defines a partial order. By definition, a binary relation is a partial order if it is reflexive, antisymmetric, and transitive. These three properties impose D to contain the origin, to be pointed, and convex respectively. A cone satisfying these properties is referred to as an *ordering cone.*

The binary relation defined in Example 2.4 is the most general binary relation that will be considered in this thesis. It contains as particular cases Pareto optimality and the lexicographic order where $D = \mathbf{R}^p_+$, and $D = \{d \in \mathbf{R}^p : \exists k \in \{1,\ldots,p\}$ such that $\forall i < k, d_i = 0$, and $d_k > 0\} \cup \{0\}$ [26, p. 31] respectively. It also contains as particular cases the two binary relations defined in Example 2.1 where $D = \mathbf{R}_+$ and $D = \mathbf{R}_-$ respectively. However, note that these two binary relations are in fact total orders.

An objective vector $\mathbf{z_1}$ can now be defined as optimal with respect to the partial order defined in terms of a cone D if there does not exist an objective vector $\mathbf{z_2}$, $\mathbf{z_2} \neq \mathbf{z_1}$, that can be preferred to it [26, p. 28]. In other words, the objective vector $\mathbf{z_1}$ is optimal if there does not exist an objective vector $\mathbf{z_2}$, $\mathbf{z_2} \neq \mathbf{z_1}$, such that $\mathbf{z_1} \in \mathbf{z_2} + D$. For the two binary relations defined in Example 2.1, this definition of an optimal objective vector corresponds to the definition of the minimum and the maximum value of the function $F(\cdot)$ over the set K respectively. The set of optimal objective vectors is denoted $\mathcal{E}(Z, D)$ and is referred to as the *minimal element set.* An element of the minimal element set is referred to as a *minimal element.* However, for Pareto optimality, the classical terminology of *Pareto* *optimal set* and *Pareto objective vector* is used instead. Finally, the objective in multiobjective optimization is to find the minimal element set, which generally does not reduce to a singleton, and the elements in K corresponding to the minimal elements.

Before going further, the question of existence of minimal elements needs to be addressed. For Example 2.1, it is known that if the function $F(\cdot)$ is continuous, and the set K is compact, then the set $F(K)$ is also compact ([46, p. 15], Theorem 1). Therefore, the minimum z_1^* and the maximum z_2^* exist, i.e., there is at least one $x_1^* \in X$ and $x_2^* \in X$ such that $z_1^* = F(x_1^*)$, and $z_2^* = F(x_2^*)$ ([46, p. 16], Theorem 2). In the general case of Example 2.4, the existence of minimal elements can be stated under similar conditions.

Proposition 2.1 ([47, 48])**.** *If the set Z is compact, then there exists a minimal element.*

Proof. Two different approaches for the proof of Proposition 2.1 can be found in the literature. The first proof [47], valid in infinite-dimensional spaces, uses Zorn's lemma but requires $\mathrm{cl}(D)$ to be pointed, which, for example, is not satisfied by the ordering cone generating the lexicographic order. The second proof [48] consists of an induction argument on the dimension p and assumes a weaker property than compactness for Z. □

Formulation of PCTS$_2$

From the definition of optimality for multiobjective optimization problems provided above, PCTS$_2$ can now be formulated. To obtain a good tracking performance from the CTS manipulator control system, the joint trajectories must at least be C^1. Therefore, let $\mathcal{T}$ be the subset of $C^1([t_0, t_f], \mathbf{R}^7)$ such that the constraints (2.11)-(2.14) are satisfied. The real vector-valued objective function $\mathbf{F}(\cdot)$ is then defined as:

$$\mathbf{F}(\cdot): \; \mathbf{q}(\cdot) \in C^1([t_0, t_f], \mathbf{R}^7) \rightarrow \left(\int_{t_0}^{t_f} f_1(\mathbf{q}(\cdot))\mathrm{d}t, \; \int_{t_0}^{t_f} f_2(\mathbf{q}(\cdot))\mathrm{d}t \right). \tag{2.17}$$

The objective space Z is the set $\mathbf{F}(\mathcal{T})$. PCTS$_2$ can therefore be reformulated as follows.

Problem PCTS$_2$: Find the minimal element set $\mathcal{E}(\mathrm{cl}(Z), D)$ and the optimal joint trajectories $\mathbf{q}^*(\cdot) \in \mathcal{T}$ corresponding to the minimal elements, if they exist.

Once PCTS$_2$ is solved, the final joint trajectory can be chosen from the joint trajectories corresponding to the minimal elements in the minimal element set. This choice will depend on the operating conditions of the CTS system. For example, at larger Mach numbers, it is more desirable to have low joint speeds, therefore, a minimal element with a lower value for $\mathbf{F}_1$ will be chosen. At lower Mach numbers, having low joint speeds is less critical, therefore, a minimal element with a lower value for $\mathbf{F}_2$ will be chosen. Such a choice will increase the reliability of the measurement of the aerodynamic loads acting on the store model.

The resolution of PCTS$_2$ is the subject of Chapter 3. A new DDP approximation method for the resolution of PCTS$_2$ is proposed. The results obtained with this method are compared to the results obtained with the commonly used weighting method.

Chapter 3

Resolution of the Problem in the Joint Space

For multiobjective optimization problems, methods for finding the minimal element set are referred to as *a posteriori* methods [25, p. 77]. Among the a posteriori methods, the weighting method has been chosen to solve PCTS_2 in Section 3.1. There are several reasons for this choice. First, the weighting method has been the only method used to solve multiobjective trajectory planning problems [10, 13, 17, 18, 49]. Second, the resolution of PCTS_2 provides an illustration of the well-known weaknesses [28] of the weighting method. As an alternative to the weighting method for solving PCTS_2, a DDP approximation method is proposed. To check the validity of this approximation method, it is first applied in Section 3.2 to the scalarized problem resulting from the application of the weighting method to PCTS_2. The DDP approximation method is then applied to PCTS_2 in Section 3.3, and the results are compared with the Pareto objective vectors obtained with the weighting method and the Pareto optimal set [50].

3.1 The Weighting Method

The formal description of the weighting method is first provided in Section 3.1.1. The application of the weighting method to PCTS$_2$ follows in Section 3.1.2. Solving PCTS$_2$ with the weighting method amounts to solving a set of problems in calculus of variations with a single objective parameterized by a weighting coefficient. The numerical resolution of these problems is performed in Section 3.1.3 by solving the BVP obtained from the first-order necessary conditions for optimality. To evaluate whether the weighting method is able to provide a good representation of the Pareto optimal set, the Pareto optimal set must be determined. This is done in Section 3.1.4 by finding the boundary of the objective space.

3.1.1 Description

Basically, the weighting method consists of combining the objective functions linearly, thereby transforming the original multiobjective optimization problem into a single objective optimization problem. Formally, the weighting method can be described as follows. Let $D \subset \mathbf{R}^p$ be a cone, the strict positive polar D^{so} is defined by [26, p. 8]

$$D^{so} = \{\mathbf{d}^* \in \mathbf{R}^p : \ \langle \mathbf{d}^*, \mathbf{d} \rangle > 0, \ \forall \mathbf{d} \in D, \ \mathbf{d} \neq 0\}.$$

For example, the strict positive polar of the cone $\mathbf{R}^p_+$ is $\mathrm{int}(\mathbf{R}^p_+)$. Consider now the general multiobjective optimization problem defined in Section 2.3.3, and let $S(\mathbf{d}^*, Z), \ \mathbf{d}^* \in D^{so}$ be the set of solutions to the single objective optimization problem

$$\inf_{\mathbf{z} \in Z} \langle \mathbf{d}^*, \mathbf{z} \rangle,$$

i.e.,

$$S(\mathbf{d}^*, Z) = \{\mathbf{z}^* \in Z,\ \langle \mathbf{d}^*, \mathbf{z}^* \rangle = \inf_{\mathbf{z} \in Z} \langle \mathbf{d}^*, \mathbf{z} \rangle\}.$$

It can be shown that every element of $S(\mathbf{d}^*, Z)$ is a minimal element ([26, p. 72], Theorem 3.4.3). Hence, by varying $\mathbf{d}^*$ in D^{so}, a subset $\mathcal{D}(Z, D)$ of the minimal element set $\mathcal{E}(Z, D)$ of Z can be obtained:

$$\mathcal{D}(Z, D) = \bigcup_{\mathbf{d}^* \in D^{so}} S(\mathbf{d}^*, Z) \subset \mathcal{E}(Z, D). \tag{3.1}$$

In other words, the set of minimal elements that can be obtained with the weighting method is included in the minimal element set. However, the main weakness of the weighting method is that not all the minimal elements can be obtained in general unless Z is convex. Indeed, if Z is convex, then ([26, p. 72], Theorem 3.4.4)

$$\mathcal{E}(Z, D) \subset \bigcup_{\mathbf{d}^* \in D^o \backslash \{0\}} S(\mathbf{d}^*, Z), \tag{3.2}$$

where D^o is the positive polar cone defined by [26, p. 8]

$$D^o = \{\mathbf{d}^* \in \mathbf{R}^p :\ \langle \mathbf{d}^*, \mathbf{d} \rangle \geq 0,\ \forall \mathbf{d} \in D\}.$$

The most favorable case for the weighting method occurs when D is open, in which case ([26, p. 74], Corollary 3.4.1):

$$\mathcal{E}(Z, D) = \mathcal{D}(Z, D) = \bigcup_{\mathbf{d}^* \in D^o \backslash \{0\}} S(\mathbf{d}^*, Z),$$

which means that any minimal element can be obtained with the weighting method. However, on the other end, it is possible that no minimal element can be obtained with the weighting method, as shown by the following example. Let $p = 2$, $X = \mathbf{R}$, $\mathbf{F}(\cdot) :\ t \rightarrow$

$(\cos(t), \sin(t))$, $K =]0, \pi/2[$, and $D = \mathbf{R}^2_+$. Then, $Z = \mathbf{F}(K) = \{x^2+y^2 = 1,\ x > 0,\ y > 0\}$. It is easy to check that $\mathcal{E}(Z, \mathbf{R}^2_+) = Z$, and $S(\mathbf{d}^*, Z) = \emptyset,\ \forall \mathbf{d}^* \in \mathrm{int}(\mathbf{R}^2_+)$.

3.1.2 Application of the Weighting Method to PCTS_2

Applying the weighting method to PCTS_2, as formulated in Section 2.3.3, yields the single objective optimization problem, for some $\mathbf{d}^* \in D^{so}$,

$$\inf_{\mathbf{z}\in \mathrm{cl}(Z)} \langle \mathbf{d}^*, \mathbf{z}\rangle = \inf_{\mathbf{z}\in Z} \langle \mathbf{d}^*, \mathbf{z}\rangle = \inf_{\mathbf{q}(\cdot)\in \mathcal{T}} \int_{t_0}^{t_f} (\mathbf{d}^*_1 f_1(\mathbf{q}(t)) + \mathbf{d}^*_2 f_2(\mathbf{q}(t)))\mathrm{d}t. \tag{3.3}$$

For the rest of this chapter, the binary relation used is the Pareto optimality, i.e., $D = \mathbf{R}^p_+$. As in general, $S(\mathbf{d}^*, Z) = S(\alpha\mathbf{d}^*, Z),\ \forall \alpha > 0$, the objective function in (3.3) can be normalized, yielding the equivalent scalarized problem PCTS_{2,s_1}:

Problem PCTS_{2,s_1}:

$$\inf_{\mathbf{q}(\cdot)\in \mathcal{T}} \int_{t_0}^{t_f} (f_1(\mathbf{q}(t)) + w f_2(\mathbf{q}(t)))\mathrm{d}t, \tag{3.4}$$

where $w > 0$ is referred to as the *weighting coefficient*. Moreover, a variant of PCTS_2 is introduced where the terminal cost, rather than the integral cost, is taken for the aerodynamic interference function f_2. The resulting scalarized problem PCTS_{2,s_2} is:

Problem PCTS_{2,s_2}:

$$\inf_{\mathbf{q}(\cdot)\in \mathcal{T}} \int_{t_0}^{t_f} f_1(\mathbf{q}(t))\mathrm{d}t + w f_2(\mathbf{q}(t_f)). \tag{3.5}$$

The main motivation for introducing this new problem is to have an example of a trajectory planning problem with terminal cost. Introducing such a problem is not strictly necessary since a terminal cost can always be reformulated as an integral cost and conversely. Regardless, it is interesting to detail how a terminal cost can be specifically treated.

One of the main objectives in this section is to illustrate the weaknesses of the weighting method. Therefore, for simplicity and for this section only, all the constraints described in Section 2.3.3 except the store trajectory constraint (2.11) are removed from PCTS_{2,s_1} and PCTS_{2,s_2}. The initial joint configuration $\mathbf{q_0}$, which satisfies $\mathbf{f}(\mathbf{q_0}) = \mathbf{p}(t_0)$, is also supposed to be known.

3.1.3 Resolution of PCTS_{2,s_1} and PCTS_{2,s_2}

The minimizers to PCTS_{2,s_1} and PCTS_{2,s_2} are obtained by solving the BVP formed by the Euler-Lagrange equations [16, p. 45] and the necessary boundary conditions for optimality.

The Euler-Lagrange Equations

For PCTS_{2,s_1} and PCTS_{2,s_2}, a system of first-order differential equations equivalent to the Euler-Lagrange equations was obtained in [18], which is recalled for clarity. Let $\gamma = \mathbf{N}^T(\mathbf{q})\dot{\mathbf{q}}$, where $\mathbf{N}(\mathbf{q})$ is the null space of the CTS manipulator Jacobian matrix $\mathbf{J}(\mathbf{q})$ obtained in Equation (2.7). The necessary conditions for optimality obtained in [18] are as follows:

- For PCTS_{2,s_1},

$$\left\{\begin{array}{lcl} \dot{\mathbf{q}} & = & \mathbf{J}^{+}\widetilde{\mathbf{J}}_{\mathbf{A}}\dot{\mathbf{p}} + \mathbf{N}(\mathbf{N}^T\mathbf{N})^{-1}\gamma \\ \dot{\gamma} & = & \dot{\mathbf{N}}^T\dot{\mathbf{q}} + w\mathbf{N}^T\nabla f_2 \end{array}\right. , \tag{3.6}$$

  where $\mathbf{J}^{+}$ denotes the Moore-Penrose inverse of $\mathbf{J}$, and ∇f_2 is the gradient of the function f_2. Moreover [18],

$$\dot{\mathbf{N}}^T = \dot{\mathbf{q}}^T\frac{\partial \mathbf{N}}{\partial \mathbf{q}},$$

  where the 7×7 (symmetric) matrix $\partial\mathbf{N}/\partial\mathbf{q}$ can be derived analytically with any symbolic computer algebra software, such as Maple [44].

- For PCTS$_{2,s_2}$,

$$\left\{\begin{array}{rcl}\dot{\mathbf{q}} & = & \mathbf{J}^{+}\widetilde{\mathbf{J}}_{\mathbf{A}}\dot{\mathbf{p}} + \mathbf{N}(\mathbf{N}^{T}\mathbf{N})^{-1}\gamma \\ \dot{\gamma} & = & \dot{\mathbf{N}}^{T}\dot{\mathbf{q}}\end{array}\right. . \tag{3.7}$$

The Boundary Conditions

For PCTS$_{2,s_1}$ and PCTS$_{2,s_1}$, the initial joint configuration is known. This condition together with the natural boundary condition at t_f [18] yields:

- For PCTS$_{2,s_1}$,

$$\left\{\begin{array}{rcl}\mathbf{q}(t_0) & = & \mathbf{q_0} \\ \gamma(t_f) & = & 0\end{array}\right. . \tag{3.8}$$

- For PCTS$_{2,s_2}$,

$$\left\{\begin{array}{rcl}\mathbf{q}(t_0) & = & \mathbf{q_0} \\ \gamma(t_f) & = & -w\mathbf{N}^{T}(\mathbf{q}(t_f))\nabla f_2(\mathbf{q}(t_f))\end{array}\right. . \tag{3.9}$$

The systems of first-order differential equations (3.6) and (3.7) together with their respective boundary conditions (3.8) and (3.9) form two BVPs. It is therefore not possible to impose additional desirable conditions for PCTS$_{2,s_1}$ and PCTS$_{2,s_2}$, such as zero initial and final joint speeds. One possibility to do so would be to formulate PCTS$_2$ at the acceleration level. Another possibility would be to keep the current formulation at the velocity level, and define the initial and final joint speeds as objective functions to be minimized. Neither of these approaches are pursued. It will be shown in Chapter 5 how the issues of zero initial and final joint speeds can be practically addressed.

Numerical Experiments

For the numerical experiments presented here, only one task in $\mathcal{S}$ is considered. The store trajectory $\mathbf{p}(\cdot)$ is planned as described in Section 2.3.2. For simplicity, only PCTS$_{2,s_1}$ is discussed. A minimizer $\mathbf{q}^*(\cdot)$ to PCTS_{2,s_1}, if it exists, necessarily satisfies the BVP (3.6) and (3.8). However, without some convexity assumptions [16, p. 46], the converse is not necessarily true. A solution to the BVP (3.6) and (3.8) could, for example, correspond to a maximizer. Therefore, to find the minimizer(s) to PCTS_{2,s_1}, all the solutions to the BVP (3.6) and (3.8) need to be found. The value of the objective function can then be computed for each of these solutions, from which the solution(s) corresponding to the minimizer(s) can be determined. To obtain all the solutions to the BVP (3.6) and (3.8), it is taken advantage of the fact that γ is a scalar. The key point is to observe that finding the solutions to the BVP (3.6) and (3.8) amounts to finding the set of values γ_0 such that the solution to the initial value problem (IVP) formed by (3.6) and the boundary conditions $(\mathbf{q_0}, \gamma(t_0) = \gamma_0)$ verifies $\gamma(t_f) = 0$. By performing a one-dimensional search, approximate values $\widetilde{\gamma}_0$ to each γ_0 can be found. The solution to the IVP (3.6) and boundary conditions $(\mathbf{q_0}, \widetilde{\gamma}_0)$ then serves as an initial guess for the collocation method used to solve the BVP (3.6) and (3.8) [51].

Let $w = 0, 1, \dots, 30$. The vector $(1, 0)$ corresponding to $w = 0$ belongs to the positive polar of $\mathbf{R}^p_+$, but not to the strict positive polar. However, it can be shown that, if the solution in $S(\mathbf{d}^*, Z)$ for some $\mathbf{d}^* \in D^o \backslash (D^{so} \cup \{0\})$ is unique, then the corresponding objective vector is a minimal element. Note that when $w = 0$, the objective function for the scalarized problem PCTS_{2,s_1} is simply

$$\int_{t_0}^{t_f} f_1(\mathbf{q}(\cdot))\mathrm{d}t,$$

and therefore does not include the aerodynamic interference function. The Pareto objective vectors obtained for PCTS_{2,s_1} with this set of values for w are displayed in Figure 3.1(a). It can be observed that the Pareto objective vector for $w = 0$ is isolated from the other

Pareto objective vectors obtained for $w > 0$. A closer look at the Pareto objective vectors obtained for $w > 0$ is provided in Figure 3.1(b). Recall from (2.17) that

$$\mathbf{F}_i(\mathbf{q}(\cdot)) = \int_{t_0}^{t_f} f_i(\mathbf{q}(\cdot))\mathrm{d}t,\ i = 1, 2.$$

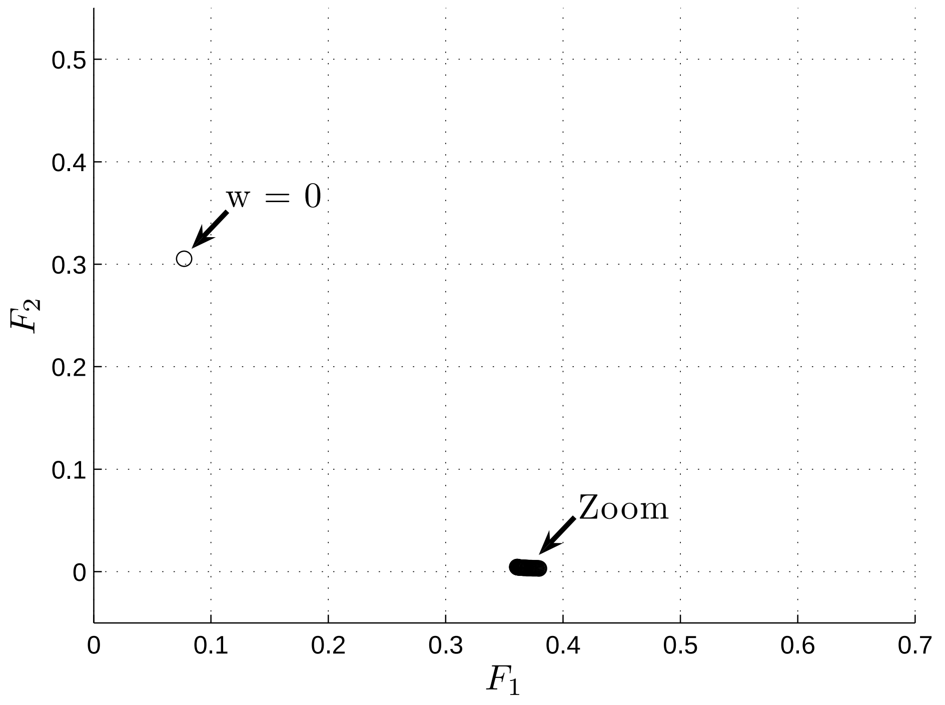
The same experiment is performed for PCTS_{2,s_2}. Note again that when $w = 0$, the objective function for the scalarized problem PCTS_{2,s_2} is simply

$$\int_{t_0}^{t_f} f_1(\mathbf{q}(\cdot))\mathrm{d}t,$$

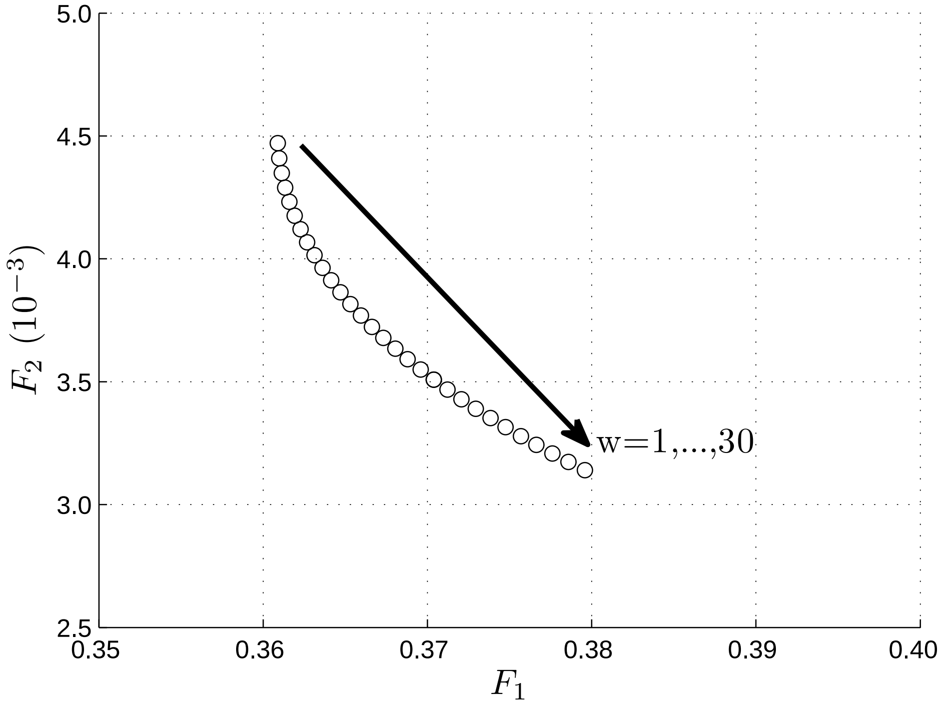
and therefore does not include the aerodynamic interference function. The Pareto objective vectors obtained for this problem are displayed in Figure 3.2(a). It can be observed that the Pareto objective vectors for $w = 0,\ 1,\ 2$ are isolated from the other Pareto objective vectors obtained for $w > 2$. A closer look at the Pareto objective vectors obtained for $w > 2$ is provided in Figure 3.2(b). In this case,

$$\mathbf{F}_2(\mathbf{q}(\cdot)) = f_2(\mathbf{q}(t_f)).$$

Figures 3.1(a)-3.2(b) provide a good illustration of the first important weakness of the weighting method. A small variation in the weighting coefficient does not necessarily result in a small variation in the Pareto objective vector. Indeed, a sudden jump in the objective vector can be observed both in Figure 3.1(a) (between $w = 0$ and $w = 1$) and in Figure 3.2(a) (between $w = 2$ and $w = 3$). A large variation in the weighting coefficient does not also necessarily results in a large variation in the Pareto objective vector, as illustrated in Figure 3.1(b) (for $w > 0$) and Figure 3.2(b) (for $w > 2$).

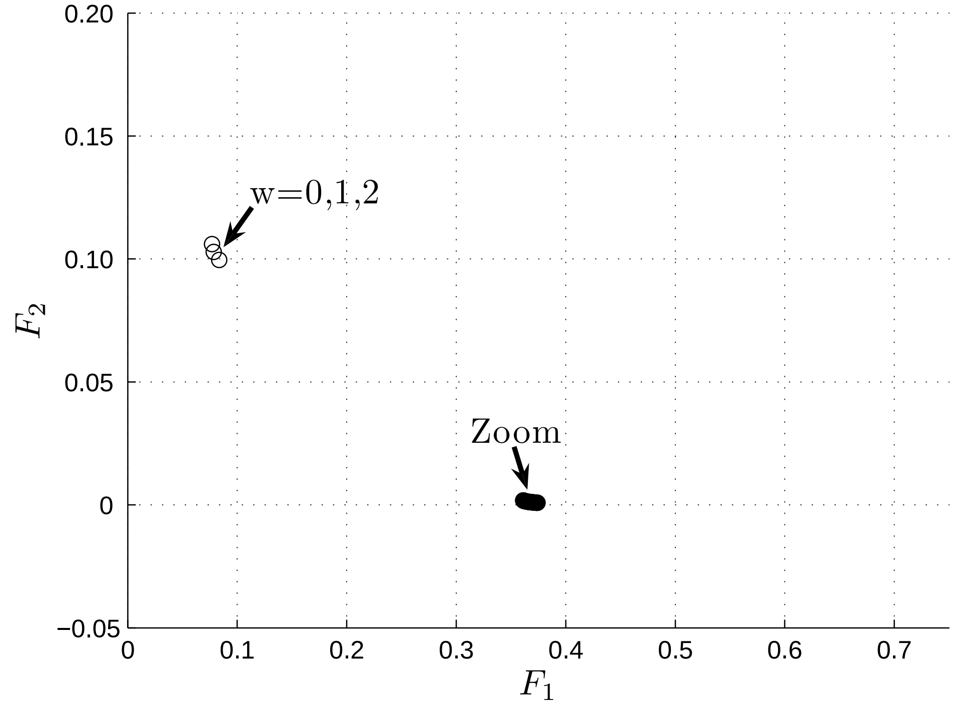


(a) $w = 0, 1, \dots, 30$.

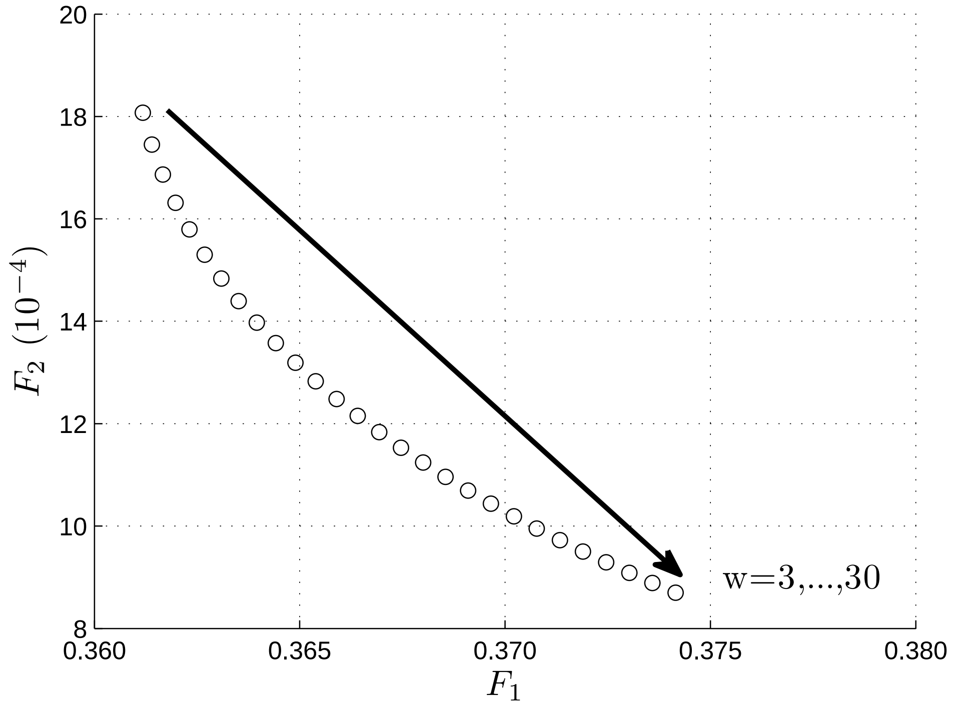


(b) Zoom for $w > 0$.

Figure 3.1: Pareto objective vectors (circles) obtained for PCTS_{2,s_1}.



(a) $w = 0, 1, \ldots, 30$.



(b) Zoom for $w > 2$.

Figure 3.2: Pareto objective vectors (circles) obtained for PCTS_{2,s_2}.

3.1.4 Determining the Objective Space for PCTS_2

Description

It is proposed to determine the objective space for PCTS_2 by finding its boundary (or finding a large set of boundary points). Pareto objective vectors being boundary points, the Pareto objective vectors obtained with the weighting method provide a first set of boundary points for PCTS_2. A second set of boundary points for PCTS_2 is given by the objective vectors corresponding to the solutions (with the exception of the minimizers to which correspond Pareto objective vectors already belonging to the first set) to the BVP obtained from the first-order optimality conditions for PCTS_{2,s_1}. Recall that, in Section 3.1.3, these solutions have all been found to determine the minimizers. However, precautions need to be taken with this second set as Proposition 3.2 only shows that these objective vectors are candidates to be boundary points. In other words, it might be possible to have a solution satisfying the first-order optimality conditions, or the BVP, whose corresponding objective vector is not a boundary point.

First, the geometric notion of *perpendicular* [52, p. 11] needs to be introduced. Let Z be a nonempty closed subset of $\mathbf{R}^p$, and let $\mathbf{z} \in \partial Z$. A vector $\mathbf{v} \in \mathbf{R}^p$ is said to be *perpendicular* to Z at $\mathbf{z}$ if $\mathbf{v} = \mathbf{z}' - \mathbf{z}$, where the point $\mathbf{z}'$ has unique closest point $\mathbf{z}$ in Z. Or, equivalently, $\mathbf{v} = \mathbf{z}' - \mathbf{z}$, where there is a closed ball centered at $\mathbf{z}'$ which meets Z only at $\mathbf{z}$. An illustration of this definition is provided in Figure 3.3. The following property of perpendiculars will prove useful:

Proposition 3.1 ([52, p. 66]).

$$\forall \mathbf{z}'' \in Z,\ \langle \mathbf{v}, \mathbf{z}'' - \mathbf{z} \rangle \leq \frac{1}{2}\|\mathbf{z} - \mathbf{z}''\|_2^2.$$

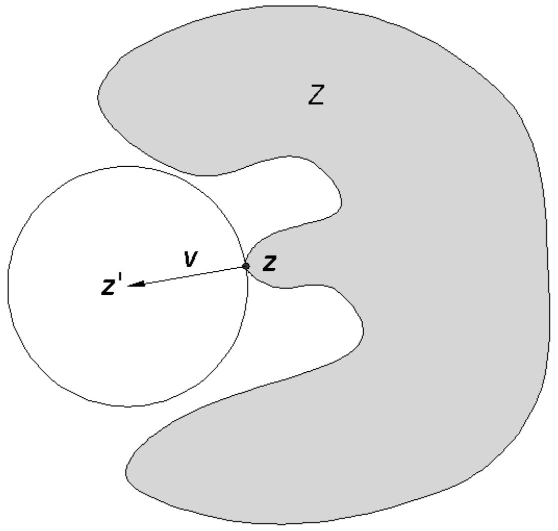


Figure 3.3: The vector $\mathbf{z}' - \mathbf{z}$ is perpendicular to Z at $\mathbf{z}$.

Consider now the general multiobjective optimization problem defined in Section 2.3.3. For simplicity, K is chosen to be equal to X. Assume that the function $\mathbf{F}(\cdot)$ is $C^1(X, \mathbf{R}^p)$. In this setting, Proposition 3.2 provides a characterization of the boundary points of Z. It is shown that, under certain conditions, the elements in X corresponding to the boundary points of Z satisfies the first-order optimality conditions for the single objective optimization problem

$$\inf_{\mathbf{x} \in X} \langle \mathbf{v}, F(\mathbf{x}) \rangle.$$

Proposition 3.2. *Let $\mathbf{z} \in \partial Z \cap Z$, and $\bar{\mathbf{x}} \in X$ such that $\mathbf{F}(\bar{\mathbf{x}}) = \mathbf{z}$. Assume that there exists a vector $\mathbf{v} \in \mathbf{R}^p$ perpendicular to Z at $\mathbf{z}$, then*

$$\forall \mathbf{h} \in X, \ \sum_{i=1}^{p} \mathbf{v}_i F_i'(\bar{\mathbf{x}})\mathbf{h} = 0, \tag{3.10}$$

where $F_i'(\bar{\mathbf{x}})$ is the Fréchet derivative at $\bar{\mathbf{x}}$.

Proof. First, recall the definition of differentiability in the setting of normed linear spaces [46, p. 115]. A function $G(\cdot) : X \to \mathbf{R}$ is differentiable at $\mathbf{x}$ if there exists $\delta > 0$ such that

$\forall \mathbf{h} \in X,\ \|\mathbf{h}\|_X < \delta,$

$$G(\mathbf{x}+\mathbf{h}) = G(\mathbf{x}) + G'(\mathbf{x})\mathbf{h} + \|\mathbf{h}\|_X \varepsilon(\mathbf{h}),$$

with

$$\lim_{\|\mathbf{h}\|_X \to 0} \varepsilon(\mathbf{h}) = 0,$$

and where the Fréchet derivative $G'(\mathbf{x})$ at $\mathbf{x}$ is an element of $L(X, \mathbf{R})$, the space of bounded linear maps from X to $\mathbf{R}$. Applying Proposition 3.1 with $\mathbf{z}'' = \mathbf{F}(\bar{\mathbf{x}}+\mathbf{h})$ yields

$$\sum_{i=1}^{p} \mathbf{v}_i (F_i(\bar{\mathbf{x}}+\mathbf{h}) - F_i(\bar{\mathbf{x}})) \leq \frac{1}{2} \sum_{i=1}^{p} (F_i(\bar{\mathbf{x}}+\mathbf{h}) - F_i(\bar{\mathbf{x}}))^2. \tag{3.11}$$

Let $\mathbf{h}$ be such that $\|\mathbf{h}\|_X < \delta$. Using the differentiability of the function $F_i(\cdot)$ at $\bar{\mathbf{x}}$, Equation (3.11) becomes

$$\sum_{i=1}^{p} \mathbf{v}_i (F_i'(\bar{\mathbf{x}})\mathbf{h} + \|\mathbf{h}\|_X \varepsilon_i(\mathbf{h})) \leq \frac{1}{2} \sum_{i=1}^{p} (F_i'(\bar{\mathbf{x}})\mathbf{h} + \|\mathbf{h}\|_X \varepsilon_i(\mathbf{h}))^2. \tag{3.12}$$

Let $0 < t < 1$, $t\mathbf{h}$ satisfies $\|t\mathbf{h}\|_X < \delta$. Therefore, substituting $\mathbf{h}$ by $t\mathbf{h}$ in Equation (3.12) yields, using the linearity of the maps $F_i'(\bar{\mathbf{x}})$,

$$t \sum_{i=1}^{p} \mathbf{v}_i (F_i'(\bar{\mathbf{x}})\mathbf{h} + \|\mathbf{h}\|_X \varepsilon_i(\mathbf{th})) \leq \frac{1}{2} t^2 \sum_{i=1}^{p} (F_i'(\bar{\mathbf{x}})\mathbf{h} + \|\mathbf{h}\|_X \varepsilon_i(t\mathbf{h}))^2. \tag{3.13}$$

Dividing both sides by t, and letting $t \to 0$ finally gives

$$\sum_{i=1}^{p} \mathbf{v}_i F_i'(\bar{\mathbf{x}})\mathbf{h} \leq 0. \tag{3.14}$$

Equation (3.14) remains valid for $-\mathbf{h}$, which proves the proposition. $\square$

Corollary 3.1. *The element* $\bar{\mathbf{x}}$ *defined in Proposition* 3.2 *satisfies the first-order optimality conditions for the single objective optimization problem*

$$\inf_{\mathbf{x}\in X} \langle \mathbf{v}, F(\mathbf{x})\rangle.$$

Proof. The first-order optimality conditions for the above problem are ([46, p. 145], Theorem 1):

$$\sum_{i=1}^{p} \mathbf{v}_i F_i'(\bar{\mathbf{x}})\mathbf{h} = 0.$$

The conclusion therefore follows from Proposition 3.2. □

Numerical experiments where Proposition 3.2 is applied to obtain boundary points of the objective space for PCTS$_2$ are presented below. For now, consider a simple illustrative example. Let $p = 2$, $X = \mathbf{R}^2$, and $\mathbf{F}(\cdot):\ (x_1, x_2) \to (x_1, x_1^3(1 + x_2^2))$. In this example, the objective space Z is easily determined, and is represented in Figure 3.4. Let $\mathbf{v} = (w, 1),\ w \in \mathbf{R}$. The function $\mathbf{F}(\cdot)$ being $C^1(\mathbf{R}^2, \mathbf{R}^2)$, the first-order optimality conditions (3.10) are:

$$\forall \mathbf{h} \in \mathbf{R}^2,\ \left\{ \begin{array}{lcl} w\partial F_1/\partial x_1 \mathbf{h}_1 + \partial F_2/\partial x_1 \mathbf{h}_2 & = & 0 \\ w\partial F_1/\partial x_2 \mathbf{h}_1 + \partial F_2/\partial x_2 \mathbf{h}_2 & = & 0 \end{array} \right. ,$$

which implies

$$\left\{ \begin{array}{lcl} w\partial F_1/\partial x_1 + \partial F_2/\partial x_1 & = & 0 \\ w\partial F_1/\partial x_2 + \partial F_2/\partial x_2 & = & 0 \end{array} \right. ,$$

and finally yields the system of nonlinear equations

$$\left\{ \begin{array}{lcl} w + 3x_1^2(1 + x_2^2) & = & 0 \\ x_1^3 x_2 & = & 0 \end{array} \right. .$$

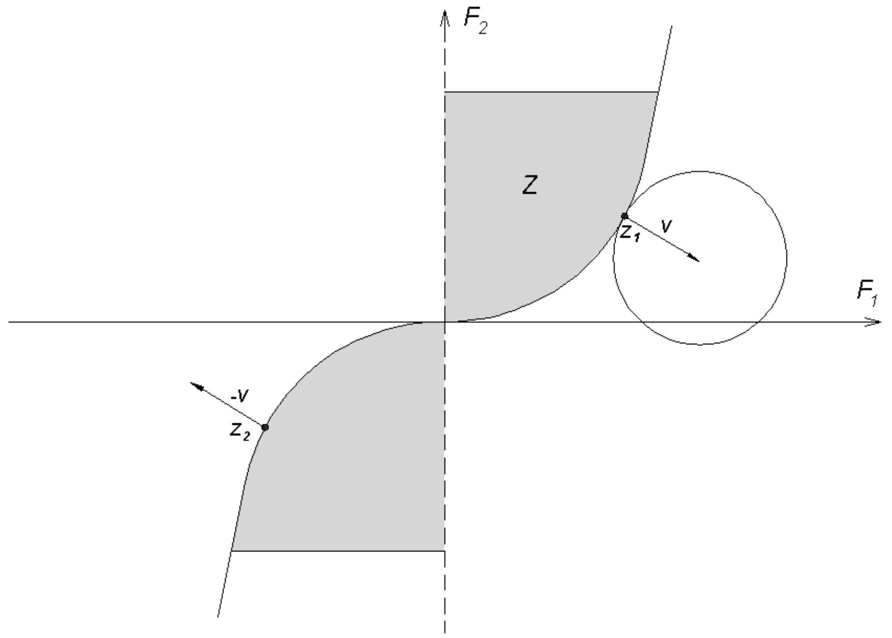


Figure 3.4: The objective space for the function $\mathbf{F}(\cdot) : (x_1, x_2) \rightarrow (x_1, x_1^3(1 + x_2^2))$.

Solving this system requires consideration of several cases:

1. $x_1 = 0$. This implies $w = 0$ and x_2 can take any value. Now, $(x_1 = 0, x_2)$ corresponds the objective vector $(0, 0)$.

2. $x_1 \neq 0$. This implies $x_2 = 0$, and therefore $x_1^2 = -w/3$. If $w < 0$, then there are two solutions for x_1 to which correspond the two objective vectors $\mathbf{z_1}$ and $\mathbf{z_2}$ as illustrated in Figure 3.4. If $w > 0$, then there is no solution to the system.

Hence, by varying w, all the points on the curve C defined by the equation $F_2 = F_1^3$ can be obtained, and it can easily be checked that this curve, together with the line $F_1 = 0$, form the boundary of Z. Some interesting facts about this example are also worth mentioning:

- The line $F_1 = 0$ does not belong to Z. Therefore, Proposition 3.2 does not apply to any point on this line.

- The origin $(0, 0)$ was obtained in Case 1 above. However, from Figure 3.4, it is clear that there does not exist any perpendicular at $(0, 0)$. This shows that a boundary point for which there does not exist any perpendicular can still satisfy the first-order optimality conditions. It also shows that a perpendicular at a boundary point does not always exist.

- In Case 2 above, the two solutions were obtained under the condition $w < 0$, which is consistent with the fact that the direction of the perpendicular at any point on C except the origin satisfies $w < 0$.

- For a given value of w, there is only one point $\mathbf{z_1}$ on C having the perpendicular $(1, w)$. However, two solutions were obtained in Case 2. This can be explained by the fact that, in general, the first-order optimality conditions (3.10) remain valid for $-\mathbf{v}$, and that, for this example, there exists a point $\mathbf{z_2}$ on C having the perpendicular $(-1, -w)$ as illustrated in Figure 3.4.

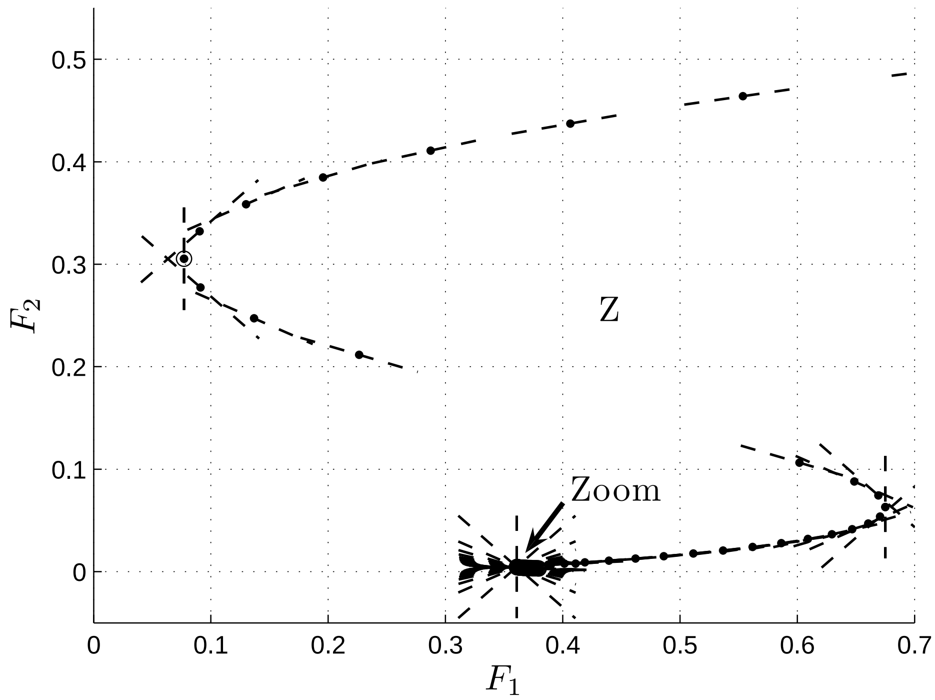
Numerical Experiments

For the numerical experiments presented now, the same instance of PCTS_2 as in Section 3.1.3 is used. First, the results with f_2 as integral cost are presented. For Figures 3.5(a)-3.7(b), to get more boundary points, PCTS_{2,s_1} is also solved for $w = -20, -19, \ldots, 0$.

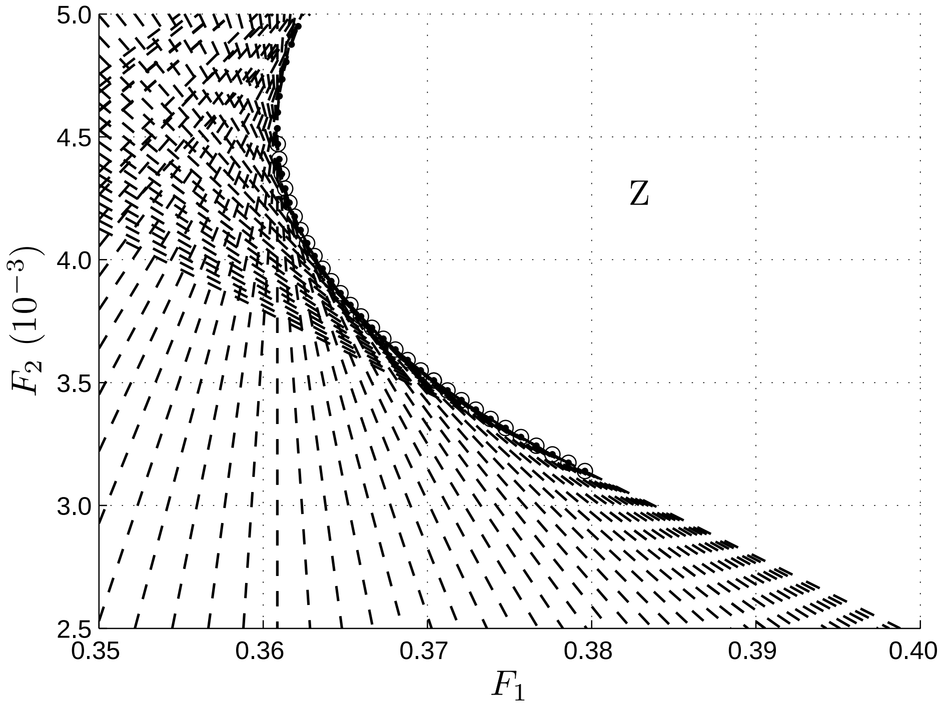
In Figures 3.5(a) and 3.5(b), the boundary points obtained as discussed at the beginning of this section are plotted. The Pareto objective vectors are identified as in Figures 3.1(a) and 3.1(b). At each boundary point, the tangent direction $(-w, 1)$ is drawn. The objective space Z is as indicated on the figures. It can be concluded that the objective space is unbounded. This is expected as any desired self-motion can be added to a joint trajectory satisfying the store trajectory constraint (2.11). It can also be concluded that the objective space is not convex. Therefore, as mentioned in Section 3.1.1, it is expected that not all the Pareto objective vectors can be obtained with the weighting method.

Recall that the points obtained from Proposition 3.2 are in fact only candidates to be boundary points. To quickly verify whether the objective space obtained in Figures 3.5(a) and 3.5(b) corresponds to the "real" objective space, random joint trajectories satisfying the store trajectory constraint (2.11) could be generated and the corresponding objective vectors plotted. Another option, which is the one pursued, is to use the joint trajectories generated in Section 3.1.3 when performing the one-dimensional search to find the approximate values $\widetilde{\gamma}_0$. The corresponding objective vectors are plotted in Figures 3.6(a) and 3.6(b) which also contain the boundary points and tangent lines from Figures 3.5(a) and 3.5(b). It can be concluded that the points obtained from Proposition 3.2 are indeed boundary points.

Knowing the boundary of the objective space, the Pareto optimal set can be easily obtained. The boundary points, the Pareto optimal set obtained from these boundary points, and the Pareto objective vectors obtained with the weighting method are plotted in Figures 3.7(a) and 3.7(b). Let $\mathbf{z_1}$ and $\mathbf{z_3}$ be the Pareto objective vectors obtained with $w = 0$ and $w = 1$ respectively. Define $\mathbf{z_2}$ as the point in the plane with the same F_2 component as $\mathbf{z_1}$ and the same F_1 component as $\mathbf{z_3}$. It can be observed that the set $\mathbf{z_2} - \mathrm{int}(\mathbf{R}^2_+)$ contains Pareto objective vectors. However, none of them could be obtained with the weighting method. Note that this set corresponds to the nonconvex part of the objective space.

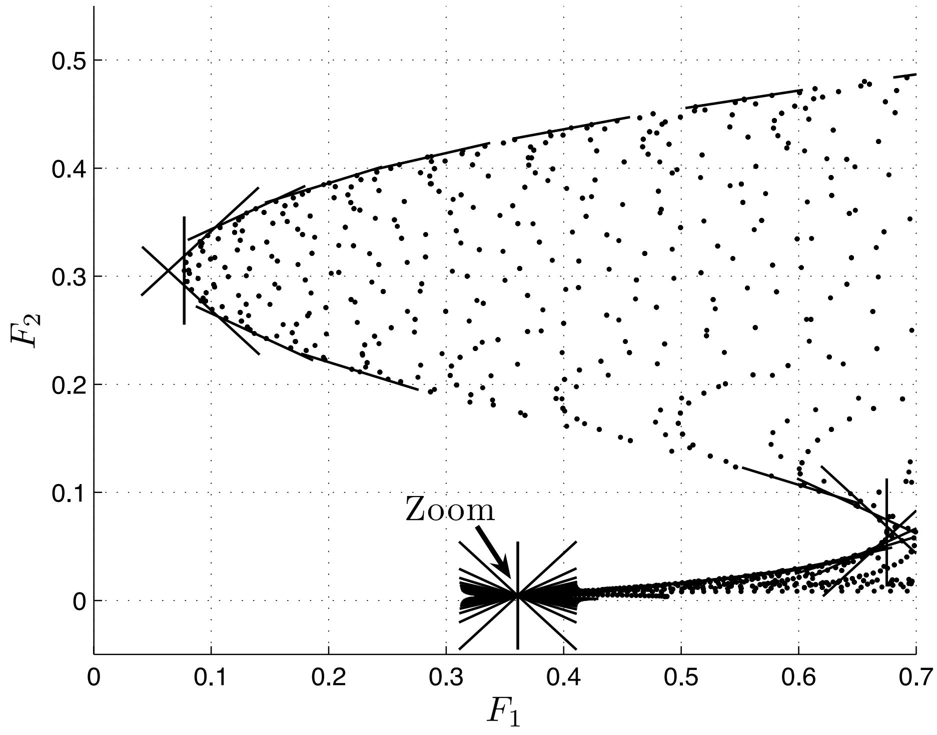


(a) Boundary points (dots), tangents at each boundary point (dotted lines), and Pareto objective vectors obtained with the weighting method (circles).

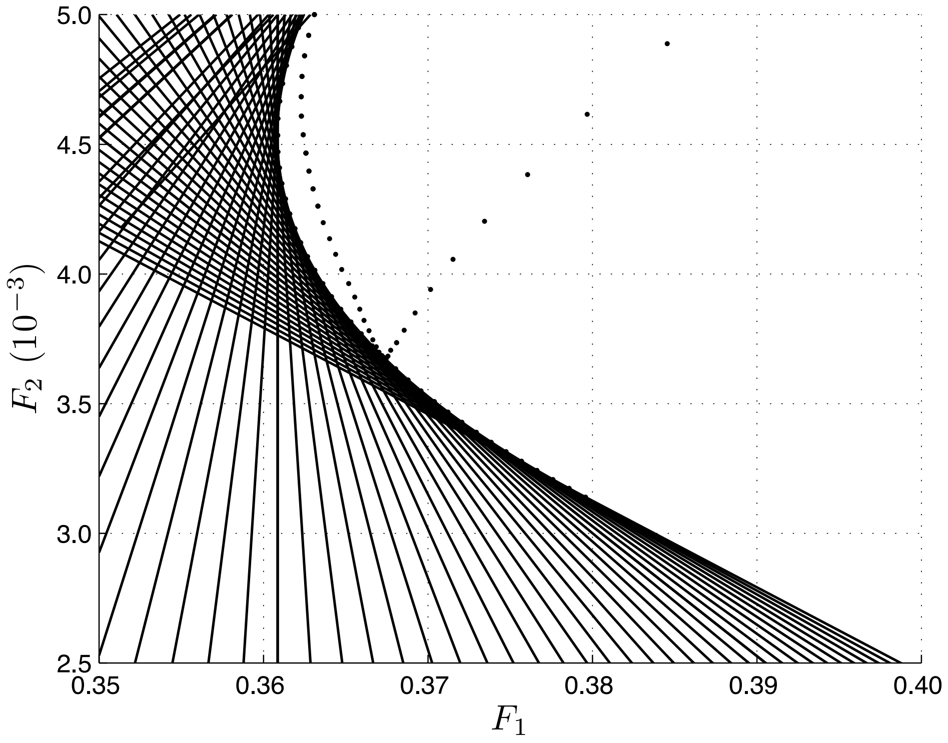


(b) Zoom.

Figure 3.5: Determining the objective space when f_2 is taken as an integral cost.

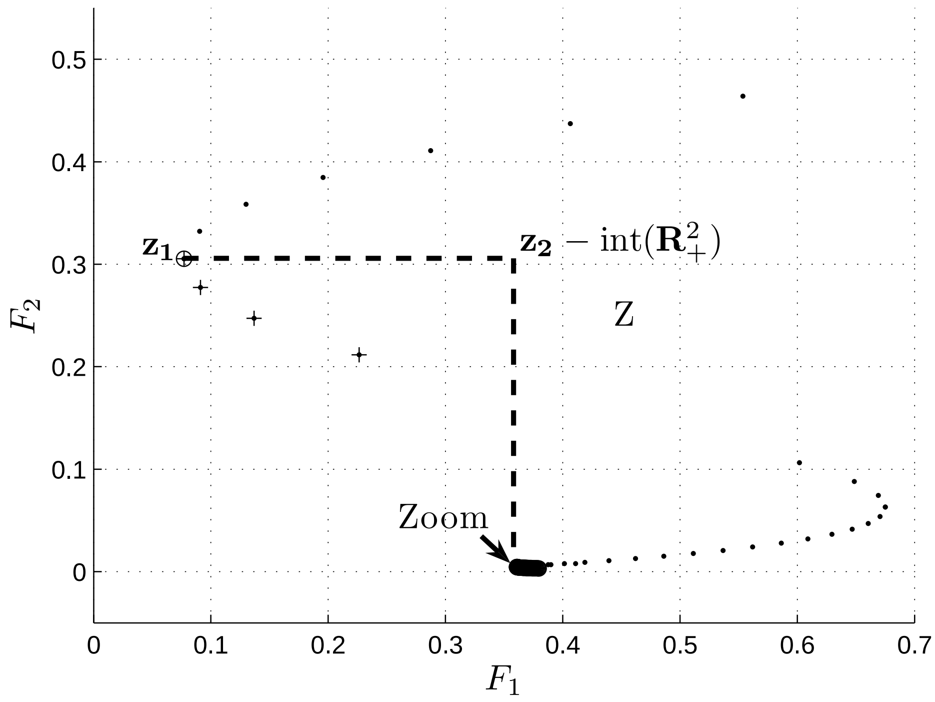


(a) Boundary points (dots), tangent lines at each boundary point (plain lines), and set of objective vectors (dots).

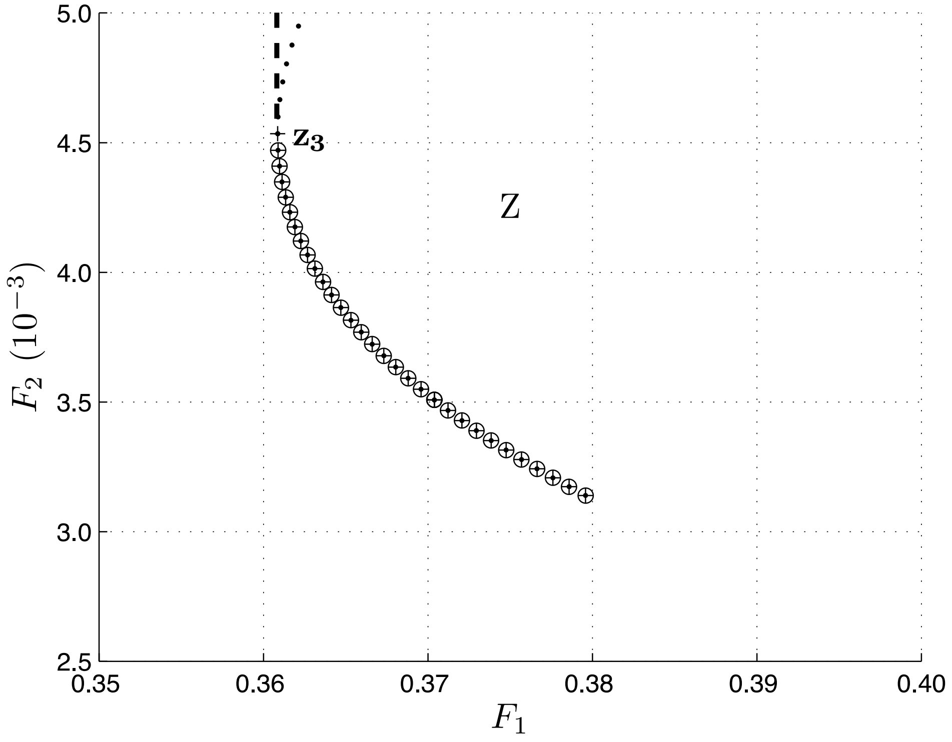


(b) Zoom.

Figure 3.6: Verifying the exactness of the objective space when f_2 is taken as an integral cost.



(a) Boundary points (dots), Pareto objective vectors obtained from the boundary points (plus signs), and Pareto objective vectors obtained with the weighting method (circles).



(b) Zoom.

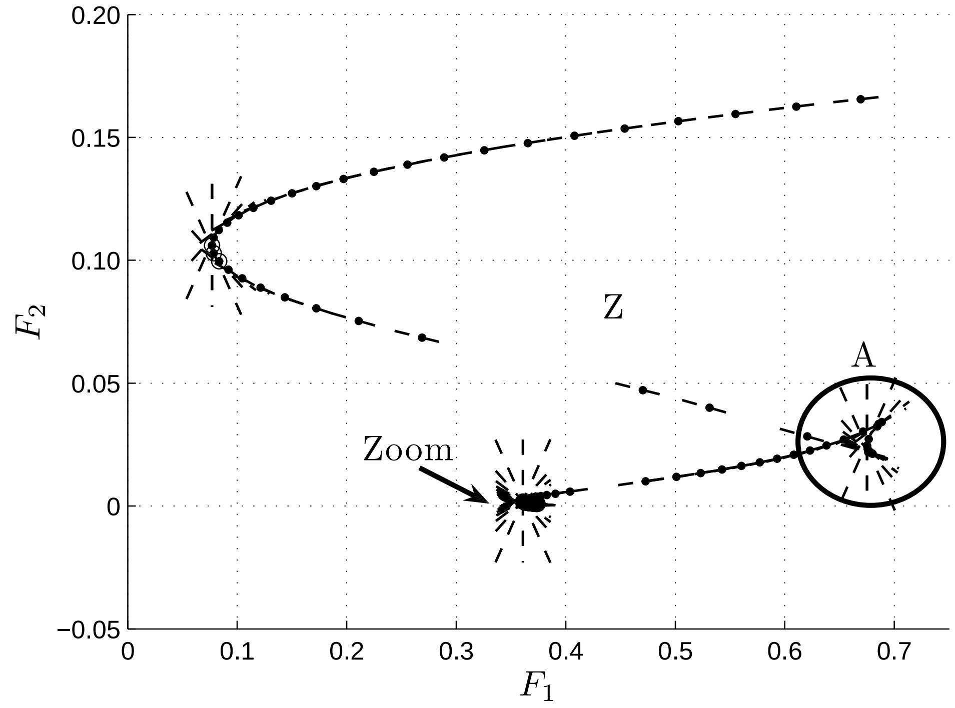
Figure 3.7: The weighting method cannot generate any Pareto objective vector in the set $\mathbf{z_2} - \mathrm{int}(\mathbf{R}^2_+)$ when f_2 is taken as an integral cost.

The results with f_2 as terminal cost follow. For Figures 3.8(a)-3.10(b), to get more boundary points, PCTS_{2,s_2} is also solved for $w = -20, -19, \ldots, 0$. Figures 3.8(a)-3.10(b) exactly correspond to Figures 3.5(a)-3.7(b). Essentially, the same observations as with f_2 as terminal cost can be made. However, several facts are worth mentioning.

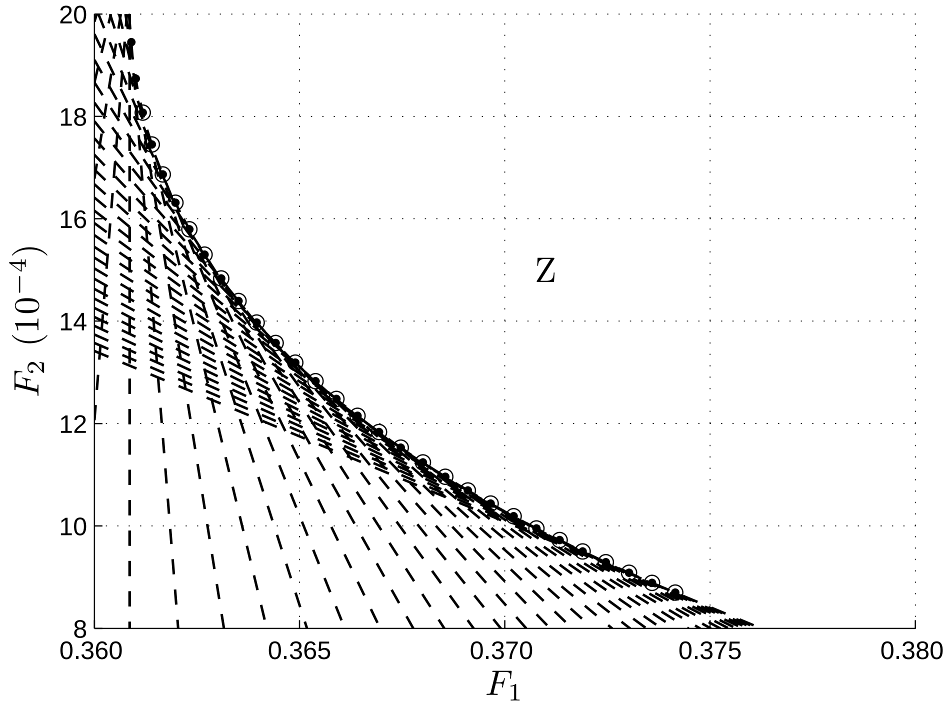
- As illustrated in Figure 3.8(a), the shape of the boundary space seems to be much more complicated, particularly in the region A delimited by a circle. It might be possible that some objective vectors in this region are in fact not boundary points, therefore providing examples of points obtained from Proposition 3.2 that are not boundary points. Further investigations about the objective space and its boundary in A were not conducted, as it is clear that A does not contain any Pareto objective vector.

- The number of objective vectors in Figure 3.9(a) is very small compared to Figure 3.6(a). This can be explained by the fact that when f_2 is taken as a terminal cost, the system of first-order differential equations (3.7) for the IVP solved to obtain these objective vectors does not depend on w. Therefore, for each value of w, the same set of objective vectors is obtained.

- In this case, $\mathbf{z_1}$ and $\mathbf{z_3}$ are the Pareto objective vectors obtained with $w = 2$ and $w = 3$ respectively.

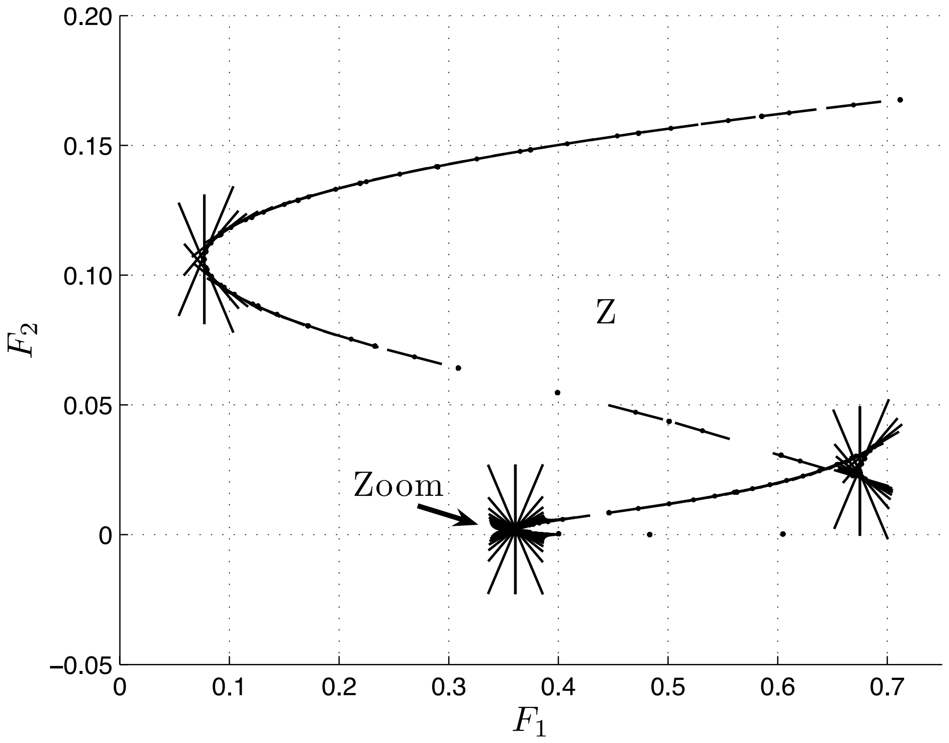


(a) Boundary points (dots), tangents at each boundary point (dotted lines), and Pareto objective vectors obtained with the weighting method (circles).

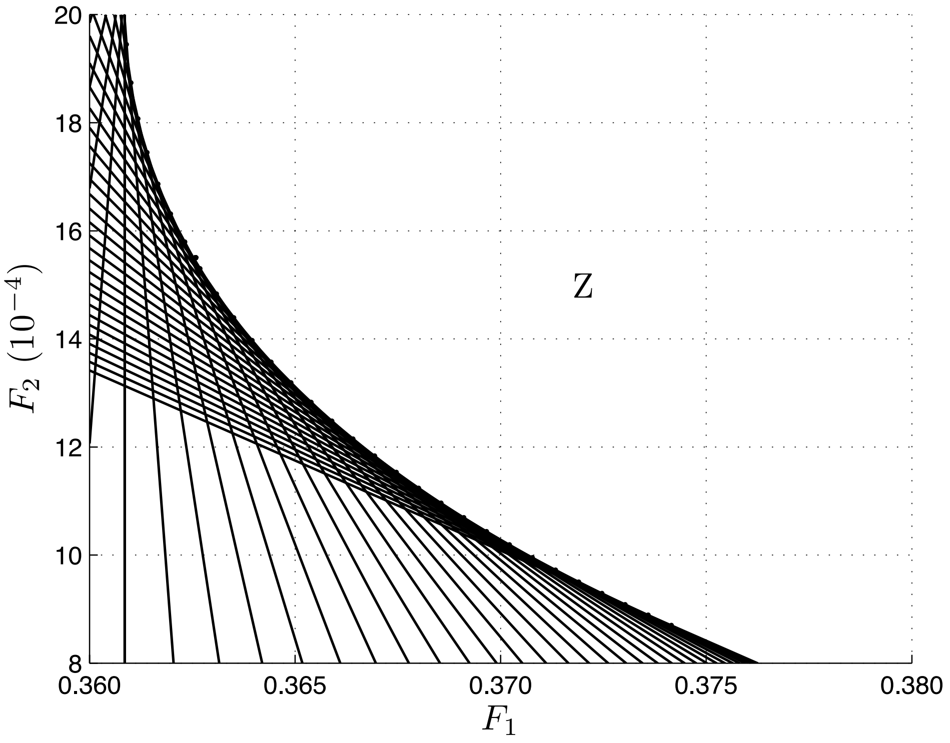


(b) Zoom.

Figure 3.8: Determining the objective space when f_2 is taken as a terminal cost.

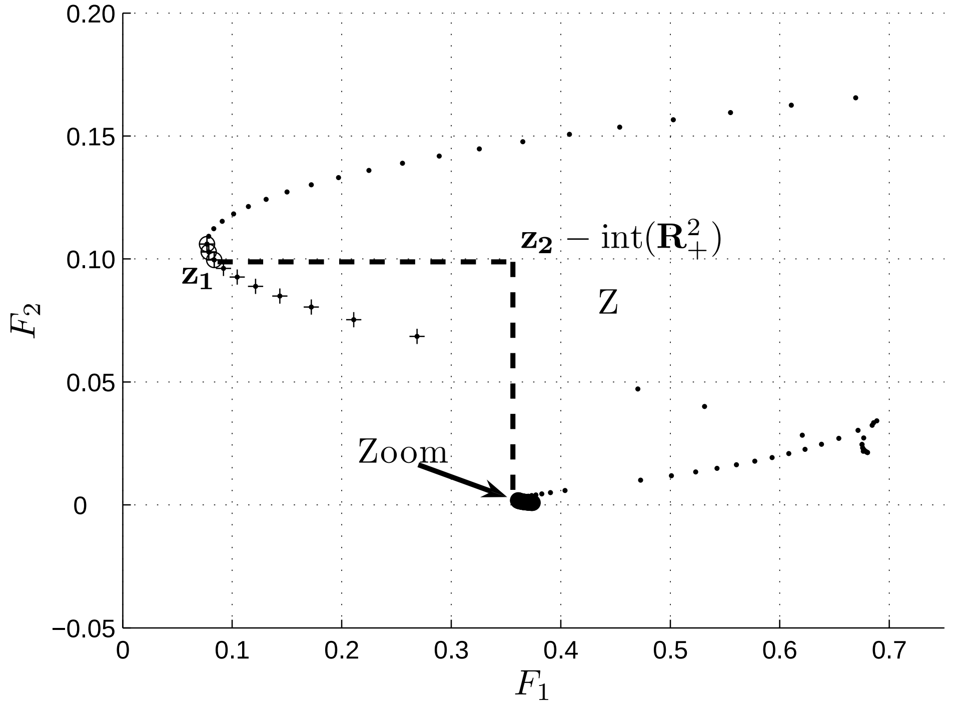


(a) Boundary points (dots), tangent lines at each boundary point (plain lines), and set of objective vectors (dots).

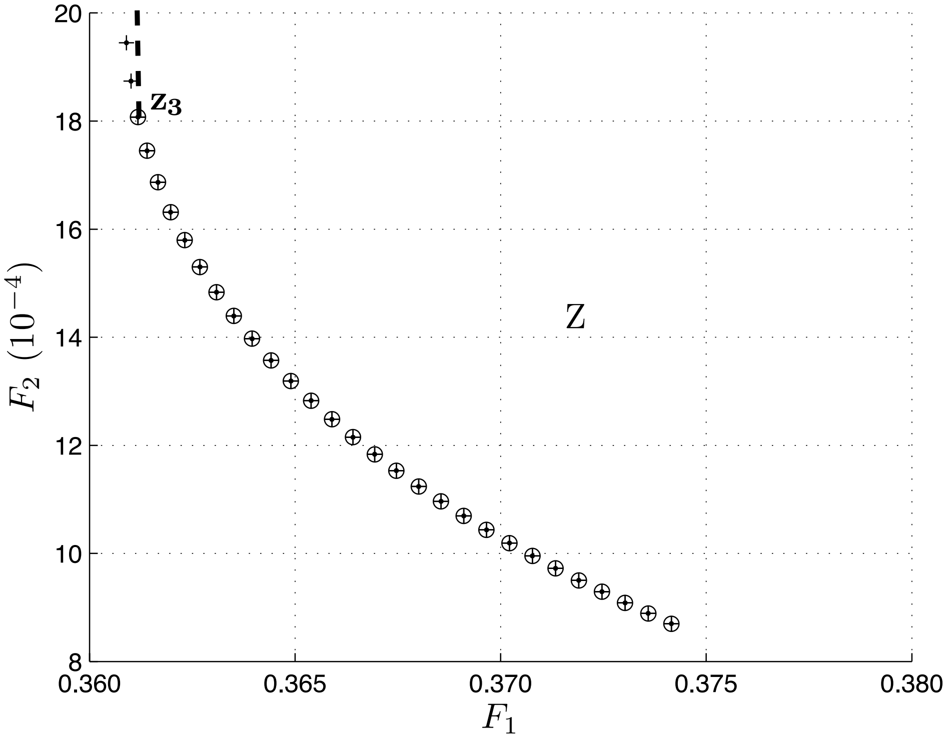


(b) Zoom.

Figure 3.9: Verifying the exactness of the objective space when f_2 is taken as a terminal cost.



(a) Boundary points (dots), Pareto objective vectors obtained from the boundary points (plus signs), and Pareto objective vectors obtained with the weighting method (circles).



(b) Zoom.

Figure 3.10: The weighting method can generate any Pareto objective vector in the set $\mathbf{z_2} - \mathrm{int}(\mathbf{R}^2_+)$ when f_2 is taken as a terminal cost.

The two examples above clearly illustrate the two main weaknesses of the weighting method in being able to provide a good representation of the Pareto optimal set [28].

- It is not possible in general to obtain every Pareto objective vector with the weighting method. The set of Pareto objective vectors that cannot be obtained with the weighting method can be large, limiting its suitability for real applications.
- The Pareto objective vector obtained with the weighting method is not, in general, a Lipschitzian function of the weighting coefficients [25, p. 83]. In other words, a uniform distribution of the weighting coefficients does not necessarily yields a uniform distribution of the Pareto objective vectors, which makes the choice of a "good" set of weighting coefficients extremely difficult.

3.2 A DDP Approximation Method for a Single Objective Function Problem

As an alternative to the weighting method for solving PCTS$_2$, a DDP approximation method is now proposed. This approximation method is first validated in this section with the scalarized problem PCTS$_{2,s_1}$ resulting from applying the weighting method to PCTS$_2$ by comparing to the results that were obtained for this problem in Section 3.1.3. For simplicity, in Section 3.1.2, the constraints (2.12)-(2.14) were not included in the definition of PCTS$_{2,s_1}$. However, in this section, these constraints are taken into account. For clarity, PCTS$_{2,s1}$ is recalled below.

$$\inf_{\mathbf{q}(\cdot)\in C_1([t_0,t_f])} \int_{t_0}^{t_f} (f_1(\mathbf{q}(t)) + w f_2(\mathbf{q}(t)))\mathrm{d}t, \tag{3.15}$$

subject to:

$$\mathbf{f}(\mathbf{q}(t)) = \mathbf{p}(t), \tag{3.16}$$

$$\mathbf{q}_{\min} \leq \mathbf{q}(t) \leq \mathbf{q}_{\max}, \tag{3.17}$$

$$-\dot{\mathbf{q}}_{\max} \leq \dot{\mathbf{q}}(t) \leq \dot{\mathbf{q}}_{\max}, \tag{3.18}$$

$$\mathrm{d}_{\mathrm{safe}} - c(\mathbf{q}(t)) \leq 0, \tag{3.19}$$

with initial joint configuration $\mathbf{q_0}$. Therefore, for comparison, the instances of PCTS_{2,s_1} will be carefully selected such that the constraints (3.17)-(3.19) are strictly satisfied by the optimal joint trajectory.

The proposed DDP approximation method consists of three steps. A preliminary step, detailed in Section 3.2.1, consists of reformulating PCTS_{2,s_1} with the redundancy parameter. This step is crucial for the applicability of the approximation method as it reduces the dimension of the state in PCTS_{2,s_1} from seven to one. The DDP approximation method applies to this reformulation of PCTS_{2,s_1}.

The first step of the proposed approximation method is to proceed to a first-order approximation in time, from which results a nonlinear programming problem. This problem can be solved with standard nonlinear programming resolution methods [53]. However, another approach using the dynamic programming principle [29] is followed. The main reason to choose this approach is that it can easily be extended to multiple objectives, as illustrated in Section 3.3. For this nonlinear programming problem, the dynamic programming principle consists of: first, defining the *return function* [30] as the optimal value of the objective function as a function of the initial conditions of the joint trajectory; second, determining the functional equation or *dynamic programming equation* satisfied by the return function. The final step of the approximation method, presented in Section 3.2.4, consists of discretizing the dynamic programming equation in the redundancy parameter, yielding an approximate dynamic programming equation that can easily be solved by performing a finite number of comparisons. The question of convergence of the approximation is briefly discussed in Section 3.2.5. For validation purposes, the results obtained with the approximation method are compared in Section 3.2.6 to those obtained in Section 3.1.3. Finally, the extension of the approximation method to PCTS_{2,s_2} is discussed in Sections 3.2.7 and 3.2.8.

3.2.1 Reformulation of PCTS_{2,s_1} with the Redundancy Parameter

Using the inverse kinematics equation at the position level (2.6) and at the velocity level (2.7), PCTS_{2,s_1} can be reformulated as follows:

$$\inf_{v(\cdot)\in C_1([t_0,t_f])} \int_{t_0}^{t_f} (\widetilde{f}_1(t, v(t), \dot{v}(t)) + w\widetilde{f}_2(t, v(t))\mathrm{d}t, \tag{3.20}$$

subject to:

$$v(t) \in \mathcal{A}(t), \tag{3.21}$$

$$\dot{v}(t) \in \mathcal{B}(t, v(t)), \tag{3.22}$$

$$v(t) \in \mathcal{A}'(t), \tag{3.23}$$

with initial redundancy parameter v_0, and where

$$\widetilde{f}_1(t, v(t), \dot{v}(t)) = \frac{1}{2}\|\widetilde{\mathbf{G}}(\mathbf{g}(\mathbf{p}(t), v(t)))\dot{\mathbf{p}}(t) + \mathbf{N}(\mathbf{g}(\mathbf{p}(t), v(t)))\dot{v}(t)\|_2^2,$$

and

$$\widetilde{f}_2(t, v(t), \dot{v}(t)) = f_2(\mathbf{g}(\mathbf{p}(t), v(t))).$$

Note the explicit dependance on the time for the functions $\widetilde{f}_1$ and $\widetilde{f}_2$ which comes from the term $\mathbf{p}(t)$. In the reformulation (3.20)-(3.23), the constraint (3.16) obviously disappears. The set $\mathcal{A}(t)$, already introduced in Section 2.2.5, represents the set of redundancy parameter values at a given time t along $\mathbf{p}(\cdot)$ such that the joint mechanical limits (3.17) are respected. The set $\mathcal{B}(t, v)$ represents the set of values at a given time t and redundancy parameter v that can be taken by the derivative of the redundancy parameter such that the maximum joint speeds (3.18) are respected. From the definition of the redundancy parameter, it follows that $\mathcal{B}(t, v(t)) \subset U$, where $U = [\dot{\mathbf{q}}_{\text{min},2} + \dot{\mathbf{q}}_{\text{min},5}, \dot{\mathbf{q}}_{\text{max},2} + \dot{\mathbf{q}}_{\text{max},5}]$. Finally, the set $\mathcal{A}'(t)$ represents the set of redundancy parameter values at a given time t along $\mathbf{p}(\cdot)$ such that the collision constraint (3.19) is respected. Both the collision constraint and joint mechanical limits have the same effect of reducing the set of values that can be taken by the redundancy parameter. Therefore, they can be grouped together. For simplicity, the set $\mathcal{A}(t) \cap \mathcal{A}'(t)$ will be denoted $\mathcal{A}(t)$ hereinafter.

3.2.2 A First-Order Discretization in Time

The first step in the proposed approximation method is to proceed to a first-order discretization in time of PCTS_{2,s_1} as reformulated in Section 3.2.1. The time step h is $(t_f - t_0)/N_T$, where N_T is the number of discretization steps. Let $v_i = v(t_i),\ i = 0, \ldots, N_T$ and $\dot{v}_i = \dot{v}(t_i),\ i = 0, \ldots, N_T - 1$. The derivative of the redundancy parameter $\dot{v}_i$ at time t_i is approximated using the forward Euler scheme:

$$\dot{v}_i = \frac{v_{i+1} - v_i}{h}.$$

The integral in the objective function (3.20) is approximated with the rectangle formula. Defining $\mathcal{A}_i^h = \mathcal{A}(t_i)$ and $\mathcal{B}_i^h(v_i) = \mathcal{B}(t_i, v_i)$, the following nonlinear programming problem results from these two approximations:

$$\inf_{\{v_i,\ i=0,\dots,N_T\}} h \sum_{i=0}^{N_T-1} \widetilde{f}_1(i, v_i, \dot{v}_i) + w\widetilde{f}_2(i, v_i), \tag{3.24}$$

subject to:

$$v_i \in \mathcal{A}_i^h. \tag{3.25}$$

$$\dot{v}_i \in \mathcal{B}_i^h(v_i), \tag{3.26}$$

with initial redundancy parameter v_0. As mentioned above, the problem (3.24)-(3.26) is a nonlinear programming problem which could be efficiently solved with standard nonlinear programming resolution methods [53]. However, considering that the objective of Section 3.2 is only to validate the DDP approximation method used in Section 3.3 to solve PCTS$_2$, the next step of this approximation method is pursued.

3.2.3 A DDP Equation

Let $\mathcal{T}_k^h(v_k)$ be the set of discrete trajectories $\{v_i,\ i = k+1,\dots,N_T\}$ with initial condition v_k such that the constraints (3.25) and (3.26) are satisfied. The return function $J_k^h(\cdot)$ is defined as the function that associates, for each $v_k \in A_k^h$, the infimum cost over all the discrete trajectories in $\mathcal{T}_k^h(v_k)$:

$$J_k^h(\cdot):\ v_k \in A_k^h \to J_k^h(v_k) = \inf_{\mathcal{T}_k^h(v_k)} h \sum_{i=k}^{N_T-1} \widetilde{f}_1(i, v_i, \dot{v}_i) + w\widetilde{f}_2(i, v_i). \tag{3.27}$$

Setting $k = 0$ in (3.27) exactly yields the nonlinear programming problem (3.24)-(3.26). Using the fact that $v_k + h\dot{v}_k = v_k + h(v_{k+1} - v_k)/h = v_{k+1}$, the return function $J_k^h(\cdot)$ can be shown to satisfy the dynamic programming equation

$$J_k^h(v_k) = \inf_{\dot{v}_k \in \mathcal{B}_k^h(v_k)} h(\widetilde{f}_1(k, v_k, \dot{v}_k) + w\widetilde{f}_2(k, v_k)) + J_{k+1}^h(v_k + h\dot{v}_k), \tag{3.28}$$

with terminal data condition

$$J_{N_T}^h(v_{N_T}) = 0. \tag{3.29}$$

The return function $J_0^h(\cdot)$, and therefore the solution to the nonlinear programming problem (3.24)-(3.26), can then be obtained by solving recursively the dynamic programming equation (3.28) starting from the terminal data condition (3.29).

3.2.4 A Discretization in the Redundancy Parameter

The return function $J_0^h(\cdot)$ is now approximated by performing a discretization in the redundancy parameter. Let $d = (v_{\max} - v_{\min})/N_X$ be the discretization step, where N_X is the number of discretization steps. Let $\mathcal{A}_k^{h,d}$ be the set resulting from the discretization of $\mathcal{A}_k^h$ in the redundancy parameter. The approximate return function $J_k^{h,d}(\cdot)$ is defined as the solution to the dynamic programming equation (3.28) with the terminal data condition (3.29) where the redundancy parameter is restricted to take on values only in $\mathcal{A}_k^{h,d}$, i.e.,

$$\forall v_k \in \mathcal{A}_k^{h,d},\ J_k^{h,d}(v_k) = \inf_{\dot{v}_k \in \mathcal{B}_k^{h,d}(v_k)} h(\widetilde{f}_1(k, v_k, \dot{v}_k) + w\widetilde{f}_2(k, v_k)) + J_{k+1}^{h,d}(v_k + h\dot{v}_k), \tag{3.30}$$

with terminal data condition

$$\forall v_{N_T} \in \mathcal{A}_{N_T}^{h,d},\ J_{N_T}^{h,d}(v_{N_T}) = 0. \tag{3.31}$$

The set $\mathcal{B}_k^{h,d}(v_k) \subset \mathcal{B}_k^h(v_k)$ is finite. The values that can be taken by $\dot{v}_k \in \mathcal{B}_k^{h,d}(v_k)$ are such that $\dot{v}_k = (v_{k+1} - v_k)/h$, where $v_k \in \mathcal{A}_k^{h,d}$ and $v_{k+1} \in \mathcal{A}_{k+1}^{h,d}$. The approximate dynamic programming equation (3.30) with terminal data condition (3.31) is straightforward to solve. Indeed, let $v_k \in \mathcal{A}_k^{h,d}$ and assume that the approximate return function $J_{k+1}^{h,d}(\cdot)$ is known. The term $h(\widetilde{f}_1(k, v_k, \dot{v}_k) + w\widetilde{f}_2(k, v_k)) + J_{k+1}^{h,d}(v_k + h\dot{v}_k)$ can be calculated for each $\dot{v}_k \in \mathcal{B}_k^{h,d}(v_k)$, which is a finite set. Therefore, to determine $J_k^{h,d}(v_k)$, $|\mathcal{B}_k^{h,d}(v_k)|$ comparisons are needed. Repeating this procedure for every $v_k \in \mathcal{A}_k^{h,d}$ yields the approximate return function $J_k^{h,d}(\cdot)$. Therefore, starting from the terminal data condition (3.31), the approximate return function $J_0^{h,d}(\cdot)$ can be recursively obtained.

3.2.5 Convergence

Let $J_0(\mathbf{q_0})$ be the optimal value for PCTS_{2,s_1}, i.e.,

$$J_0(\mathbf{q_0}) = \inf_{\mathbf{q}(\cdot)\in\mathcal{T}} \int_{t_0}^{t_f} (f_1(\mathbf{q}(t)) + wf_2(\mathbf{q}(t)))\mathrm{d}t,$$

where $\mathbf{q_0}$ is the initial joint configuration. The question is to determine whether $J_0^{h,d}(v_0)$, resulting from the DDP approximation method proposed in Sections 3.2.2, 3.2.3, and 3.2.4, converges towards $J_0(\mathbf{q_0})$ as both h and d tend towards zero. This is investigated in Section 3.2.6 by performing numerical experiments. In this section, theoretical investigations are pursued. The problem of determining the convergence of $J_0^{h,d}(v_0)$ towards $J_0(\mathbf{q_0})$ can be divided into two subproblems, i.e., the convergence of $J_0^h(v_0)$ towards $J_0(\mathbf{q_0})$ as h tends towards zero, and the convergence of $J_0^{h,d}(v_0)$ towards $J_0^h(v_0)$ as d tends towards zero. As it concerns convergence in functional spaces, the convergence of $J_0^h(v_0)$ towards $J_0(\mathbf{q_0})$ is much more difficult to establish than the convergence of $J_0^{h,d}(v_0)$ towards $J_0^h(v_0)$ which concerns convergence in the finite dimensional space $\mathbf{R}^{N_T}$. The convergence of $J_0^h(v_0)$ towards $J_0(\mathbf{q_0})$ can be studied with the theory of viscosity solutions for first-order partial differential equations of Hamilton-Jacobi type [22]. However, such a study is beyond the scope of this thesis. On the other hand, the convergence of $J_0^{h,d}(v_0)$ towards $J_0^h(v_0)$ is stated in Proposition 3.4 for the unconstrained case, i.e., without the constraints (3.17)-(3.19).

In such a case, denoting $U = [u_{\min}, u_{\max}]$, we have:

$$\left\{\begin{array}{lcl} \mathcal{A}_k^h & = & [v_0 + khu_{\min}, v_0 + khu_{\max}] \\ \mathcal{B}_k^h(v_k) & = & U \end{array}\right. . \tag{3.32}$$

The proof of Proposition 3.4 first requires showing that the return functions $J_k^h(\cdot)$ are Lipschitz, which is established in Proposition 3.3.

Proposition 3.3. *Assume that the functions $\widetilde{f}_1(t,\cdot,\xi)$ and $\widetilde{f}_2(t,\cdot)$ are Lipschitz, i.e.,*

$$\forall t \in [t_0, t_f],\ \forall \xi \in U,\ \forall (x,y) \in \mathbf{R}\times\mathbf{R},\ |\widetilde{f}_1(t,x,\xi) - \widetilde{f}_1(t,y,\xi)| \le K_1|x-y|,$$

and

$$\forall t \in [t_0, t_f],\ \forall (x,y) \in \mathbf{R}\times\mathbf{R},\ |\widetilde{f}_2(t,x) - \widetilde{f}_2(t,y)| \le K_2|x-y|,$$

then

$$\forall (x,y) \in \mathbf{R}\times\mathbf{R},\ |J_k^h(x) - J_k^h(y)| \le L_k|x-y|h, \tag{3.33}$$

where $L_k = (K_1 + wK_2)(N_T - k)$.

Proof. The proof is a proof by induction using the dynamic programming equation (3.28) with the terminal data condition (3.29).

- Let $k = N_T - 1$, we have:

$$J_k^h(x) = \inf_{\dot{v}_k \in U} h(\widetilde{f}_1(k,x,\dot{v}_k) + w\widetilde{f}_2(k,x)).$$

Therefore,

$$|J_k^h(y) - J_k^h(x)| \le h \sup_{\dot{v}_k \in U} |\widetilde{f}_1(k,y,\dot{v}_k) - \widetilde{f}_1(k,x,\dot{v}_k) + w(\widetilde{f}_2(k,y) - \widetilde{f}_2(k,x))|.$$

Using the Lipschitz assumption for the functions $\widetilde{f}_1(t,\cdot,\xi)$ and $\widetilde{f}_2(t,\cdot)$ yields

$$|J_k^h(y) - J_k^h(x)| \leq (K_1 + wK_2)|y-x|h.$$

Therefore, $L_{N_T-1} = (K_1 + wK_2)$.

- Assume now that the function $J_{k+1}^h(\cdot)$ is Lipschitz. We have:

$$J_k^h(x) = \inf_{\dot{v}_k \in U} h(\widetilde{f}_1(k,x,\dot{v}_k) + w\widetilde{f}_2(k,x)) + J_{k+1}^h(x + h\dot{v}_k).$$

Therefore, using the Lipschitz assumption for the functions $\widetilde{f}_1(t,\cdot,\xi)$ and $\widetilde{f}_2(t,\cdot)$ and the induction assumption,

$$|J_k^h(y) - J_k^h(x)| \leq (K_1 + wK_2)|y-x|h + L_{k+1}|y + h\dot{v}_k - x - h\dot{v}_k|h,$$

or,

$$|J_k^h(y) - J_k^h(x)| \leq (K_1 + wK_2 + L_{k+1})|y-x|h.$$

Therefore, $L_k = K_1 + wK_2 + L_{k+1}$.

It is a simple induction to show that $L_k = (K_1 + wK_2)(N_T - k)$. □

Proposition 3.4 provides an error estimate between the return function $J_k^h(\cdot)$ and the approximate return function $J_k^{h,d}(\cdot)$. As explained below, this error estimate allows to conclude the convergence of $J_0^{h,d}(v_0)$ towards $J_0^h(v_0)$.

Proposition 3.4. *Under the same assumptions as in Proposition* 3.4*, and the assumption that the function* $\widetilde{f}_1(t,x,\cdot)$ *is Lipschitz, i.e.,*

$$\forall t \in [t_0, t_f],\ \forall v \in \mathbf{R},\ \forall (x,y) \in U \times U,\ |\widetilde{f}_1(t,v,x) - \widetilde{f}_1(t,v,y)| \leq K_3|x-y|,$$

then, for all $x \in \mathcal{A}_k^{h,d}$,

$$0 \leq J_k^{h,d}(x) - J_k^h(x) \leq L'_k d,$$

where $L'_k = (K_3 + L_0 h)(N_T - k)$.

Proof. The proof is a proof by induction using the dynamic programming equations (3.28) and (3.30) with their respective terminal data condition (3.29) and (3.31). In the unconstrained case, the sets $\mathcal{B}_k^{h,d}(v_k)$ are identical, and denoted $U_d \subset U$.

- Let $k = N_T - 1$, and $x \in \mathcal{A}_k^{h,d}$. We have:

  $$J_k^h(x) = \inf_{\dot{v}_k \in U} h(\widetilde{f}_1(k, x, \dot{v}_k) + w\widetilde{f}_2(k, x)),$$

  and

  $$J_k^{h,d}(x) = \inf_{\dot{v}_k \in U_d} h(\widetilde{f}_1(k, x, \dot{v}_k) + w\widetilde{f}_2(k, x)).$$

  As $U_d \subset U$, $0 \leq J_k^{h,d}(x) - J_k^h(x)$. U being a compact set and the function $\widetilde{f}_1(t, x, \cdot)$ being continuous, there exists $\dot{v}_k^* \in U$ such that $J_k^h(x) = h(\widetilde{f}_1(k, x, \dot{v}_k^*) + w\widetilde{f}_2(k, x))$. Therefore, for all $\dot{v}_k \in U_d$,

  $$J_k^{h,d}(x) - J_k^h(x) \leq h(\widetilde{f}_1(k, x, \dot{v}_k) + w\widetilde{f}_2(k, x)) - h(\widetilde{f}_1(k, x, \dot{v}_k^*) + w\widetilde{f}_2(k, x)),$$

  or using the Lipschitz assumption for the function $\widetilde{f}_1(t, x, \cdot)$

  $$J_k^{h,d}(x) - J_k^h(x) \leq K_3|\dot{v}_k^* - \dot{v}_k|h.$$

  It is always possible to take $\dot{v}_k \in U_d$ such that $|\dot{v}_k^* - \dot{v}_k| \leq d/h$. Therefore, we get:

  $$J_k^{h,d}(x) - J_k^h(x) \leq K_3 d \leq (K_3 + L_0 h)d.$$

Finally, $L'_{N_T-1} = K_3 + L_0 h$.

- Assume now that for all $x \in \mathcal{A}^{h,d}_{k+1}$,

$$0 \leq J^{h,d}_{k+1}(x) - J^h_{k+1}(x) \leq L'_{k+1} d.$$

Let $x \in \mathcal{A}^{h,d}_k$. We have:

$$J^h_k(x) = \inf_{\dot{v}_k \in U} h(\widetilde{f}_1(k, x, \dot{v}_k) + w\widetilde{f}_2(k, x)) + J^h_{k+1}(x + h\dot{v}_k),$$

and

$$J^{h,d}_k(x) = \inf_{\dot{v}_k \in U_d} h(\widetilde{f}_1(k, x, \dot{v}_k) + w\widetilde{f}_2(k, x)) + J^{h,d}_{k+1}(x + h\dot{v}_k).$$

From the induction assumption, we have, for all $\dot{v}_k \in U_d$:

$$J^h_{k+1}(x + h\dot{v}_k) \leq J^{h,d}_{k+1}(x + h\dot{v}_k).$$

Therefore,

$$\inf_{\dot{v}_k \in U_d} h(\widetilde{f}_1(k, x, \dot{v}_k) + w\widetilde{f}_2(k, x)) + J^h_{k+1}(x + h\dot{v}_k) \leq J^{h,d}_k(x).$$

As $U_d \subset U$,

$$J^h_k(x) \leq \inf_{\dot{v}_k \in U_d} h(\widetilde{f}_1(k, x, \dot{v}_k) + w\widetilde{f}_2(k, x)) + J^h_{k+1}(x + h\dot{v}_k).$$

Combining the two inequalities yields

$$0 \leq J^{h,d}_k(x) - J^h_k(x).$$

To prove the right-hand side inequality, introducing again $\dot{v}_k^*$, we get, for all $\dot{v}_k \in U_d$:

$$J_k^{h,d}(x) - J_k^h(x) \leq h(\widetilde{f}_1(k, x, \dot{v}_k) - \widetilde{f}_1(k, x, \dot{v}_k^*)) + J_{k+1}^{h,d}(x + h\dot{v}_k) - J_{k+1}^h(x + h\dot{v}_k^*).$$

Using the fact that the return function $J_{k+1}^h(\cdot)$ is Lipschitz, as shown in Proposition 3.3,

$$J_{k+1}^{h,d}(x + h\dot{v}_k) - J_{k+1}^h(x + h\dot{v}_k^*) \leq J_{k+1}^{h,d}(x + h\dot{v}_k) - J_{k+1}^h(x + h\dot{v}_k) + L_{k+1}|\dot{v}_k - \dot{v}_k^*|h^2.$$

Therefore, using the Lipschitz assumption for the function $\widetilde{f}_1(t, x, \cdot)$, and knowing that $L_{k+1} \leq L_0$,

$$J_k^{h,d}(x) - J_k^h(x) \leq K_3|\dot{v}_k - \dot{v}_k^*|h + L'_{k+1}d + L_0|\dot{v}_k - \dot{v}_k^*|h^2.$$

Again, it is always possible to take $\dot{v}_k \in U_d$ such that $|\dot{v}_k^* - \dot{v}_k| \leq d/h$. Therefore, we get:

$$J_k^{h,d}(x) - J_k^h(x) \leq (K_3 + L_0 h + L'_{k+1})d.$$

Finally,

$$L'_k = K_3 + L_0 h + L'_{k+1}.$$

It is a simple induction to show that $L'_k = (K_3 + L_0 h)(N_T - k)$.

□

Specialized to the case $k = 0$, Proposition 3.4 yields

$$0 \leq J_0^{h,d}(x) - J_0^h(x) \leq L'_0 d,$$

where,

$$L_0' = (K_3 + L_0 h)N_T = (K_3 + (K_1 + wK_2)N_T h)N_T.$$

Knowing that $N_T h = t_f - t_0$, we finally obtain:

$$0 \leq J_0^{h,d}(x) - J_0^h(x) \leq (K_3 + (K_1 + wK_2)(t_f - t_0))(t_f - t_0)d/h = Cd/h. \tag{3.34}$$

The interpretation of the inequality (3.34) is that $J_0^{h,d}(v_0)$ converges towards $J_0^h(v_0)$ when d converges towards zero, which is expected. However, more interestingly, it shows the importance of the ratio d/h for the convergence of $J_0^{h,d}(v_0)$ towards $J_0(\mathbf{q_0})$. This fact will be illustrated in Section 3.2.6 with numerical experiments. Intuitively, the ratio d/h corresponds to the discretization in the redundancy parameter derivative. Therefore, it makes perfect sense that this ratio should also converge towards zero, when both h and d converge towards zero. Interestingly, an error estimate in d/h was also obtained with a similar DDP approximation method applied to a general infinite horizon optimal control problem ([22, p. 476], Theorem 1.4).

3.2.6 Numerical Experiments for PCTS_{2,s_1}

The exact value for $J_0(\mathbf{q_0})$ is supposed to be the value that was obtained in Section 3.1.2. As already mentioned, the instances of PCTS_{2,s_1} have been carefully selected such that the constraints (3.17)-(3.19) are strictly satisfied by the optimal joint trajectory.

First, it is described how $J_0^{h,d}(v_0)$ can be practically obtained. Let the store trajectory $\mathbf{p}(\cdot)$ planned as described in Section 2.3.2, and h and d be given. The discretization in the time and the redundancy parameter yields a grid $\{(k, v_{k,i}),\ k = 0, \ldots, N_T,\ i = 0, \ldots, N_X\}$. At each point $(k, v_{k,i})$ of this grid, the joint configuration $\mathbf{q}$ is calculated using the inverse kinematics equation at the position level (2.6) with $v_{k,i}$ and $\mathbf{p}(t_k)$. If $\mathbf{q}$ does not satisfy the joint mechanical limits (3.17) or the collision constraint (3.19), then the point is removed from the grid. Repeating this procedure yields the sets $\mathcal{A}_k^{h,d}$. For each remaining node $(k, v_{k,i})$ of the grid, the sets $\mathcal{B}_k^{h,d}(v_k)$ are determined using the inverse kinematics equation at the velocity level (2.7) and the values of the function $\widetilde{f}_1$ and $\widetilde{f}_2$ are calculated. Finally, the approximate dynamic programming equation (3.30) with terminal data condition (3.31) is solved as discussed in Section 3.2.4, yielding the approximate return functions $J_k^{h,d}(\cdot)$, and in particular $J_0^{h,d}(v_0)$.

The immediate question is the *a priori* choice of h and d. To answer this question, practical considerations about the CTS system that were mentioned in Section 2.1 need to be recalled. In practice, it might be possible that, based on the results obtained during a wind-tunnel run, it becomes necessary to modify the grid. As a consequence, a new CTS trajectory planning problem would have to be solved and the new joint trajectories for the CTS manipulator would have to be available before the next wind-tunnel run. Therefore, it is critical that PCTS$_2$ must be able to be solved very efficiently, which explains why computation efficiency rather than accuracy is the main concern of the study below. Given that the run time of the wind-tunnel, or the duration of the store trajectory $t_f - t_0$, does not exceed 30 s, a reasonable choice for h and d, yielding a reasonable grid size, was found to be $h = 0.5$ s and $d = 1$ deg. Note that it is not possible to estimate what accuracy can be expected from these values of h and d, as: first, no global error estimate for the proposed DDP approximation is available, and second, each initial condition $\mathbf{q_0}$ and each store trajectory $\mathbf{p}(\cdot)$ yield a different instance of PCTS$_{2,s_1}$. It might be argued that the value of h, in particular, is much too high to expect the forward Euler scheme to provide a good approximation of the redundancy parameter derivative. However, recall that one of the objectives in PCTS$_2$ is to minimize the integral of the joint speed norm. Hence, the joint speeds along the optimal joint trajectory are expected to stay small.

Three numerical experiments are now presented. The objective of the first experiment is to compare $J_0(\mathbf{q_0})$ and $J_0^{h,d}(v_0)$, and the corresponding optimal joint trajectories for an instance of PCTS$_{2,s_1}$. The objective of the second experiment is to show for this same instance how the accuracy is improved by letting the ratio d/h converge towards zero as predicted from Proposition 3.4. The third experiment presents a case where the joint speeds for the optimal joint trajectory are much larger, and the chosen values for h and d prove to be inappropriate. Without changing the values of h and d, a heuristic approach is proposed that allows obtaining a "reasonable" optimal discrete trajectory.

For the first experiment, the same instance of PCTS$_{2,s_1}$ as in Section 3.1.3 is used. w is set to 0.5. The chosen values for h and d yield $(N_T, N_X) = (13, 290)$. The values of $J_0^{h,d}(v_0)$ and $J_0(\mathbf{q_0})$ are provided in Table 3.1. Table 3.1 also contains the individual optimal value for the two objective functions, i.e., $F_1(\mathbf{q}^*(\cdot))$, $F_2(\mathbf{q}^*(\cdot))$, $F_1^{h,d}$, and $F_2^{h,d}$, where, $\{v_i^*,\ i = 0, \ldots, N_T\}$ being the optimal discrete trajectory:

$$F_1^{h,d} = h \sum_{i=0}^{N_T-1} \widetilde{f}_1(i, v_i^*, \dot{v}_i^*),$$

and

$$F_2^{h,d} = h \sum_{i=0}^{N_T-1} \widetilde{f}_2(i, v_i^*).$$

Note that with these notations, $J_0^{h,d}(v_0) = F_1^{h,d} + wF_2^{h,d}$. Recall also that $J_0(\mathbf{q_0}) = F_1(\mathbf{q}^*(\cdot)) + wF_1(\mathbf{q}^*(\cdot))$. In Figures 3.11(a) and 3.11(b), are plotted the three trajecto-

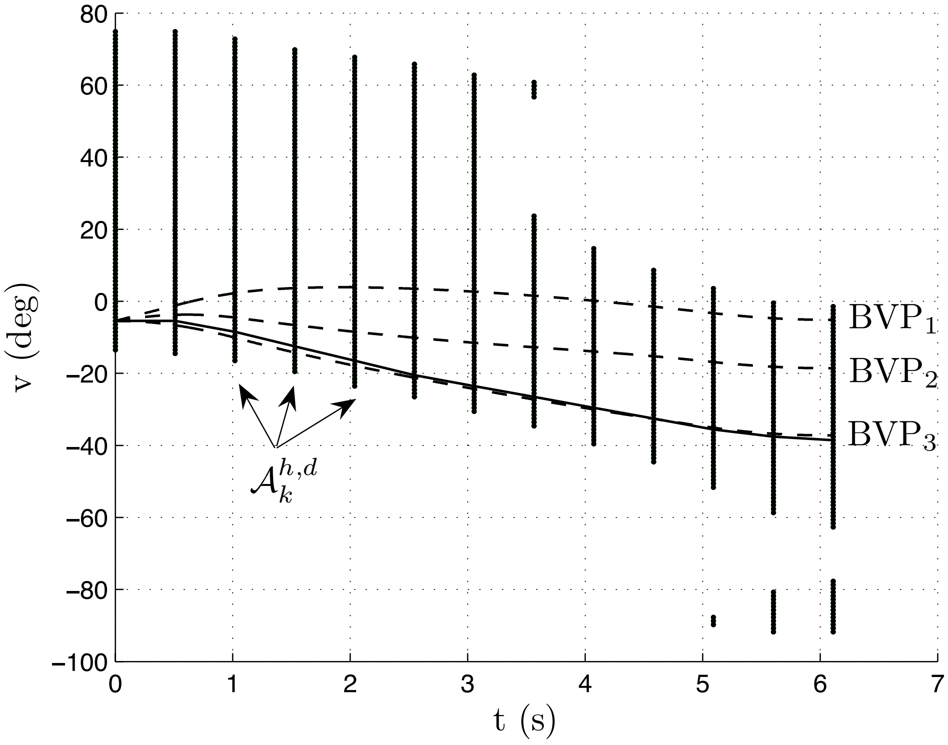
Table 3.1: Values of the objective functions for $(N_T, N_X) = (13, 290)$.

$F_1^{h,d}$	$F_1(\mathbf{q}^*(\cdot))$	$F_2^{h,d}$	$F_2(\mathbf{q}^*(\cdot))$	$J_0^{h,d}(v_0)$	$J_0(\mathbf{q_0})$
0.133054	0.080497	0.259263	0.291530	0.262685	0.226262

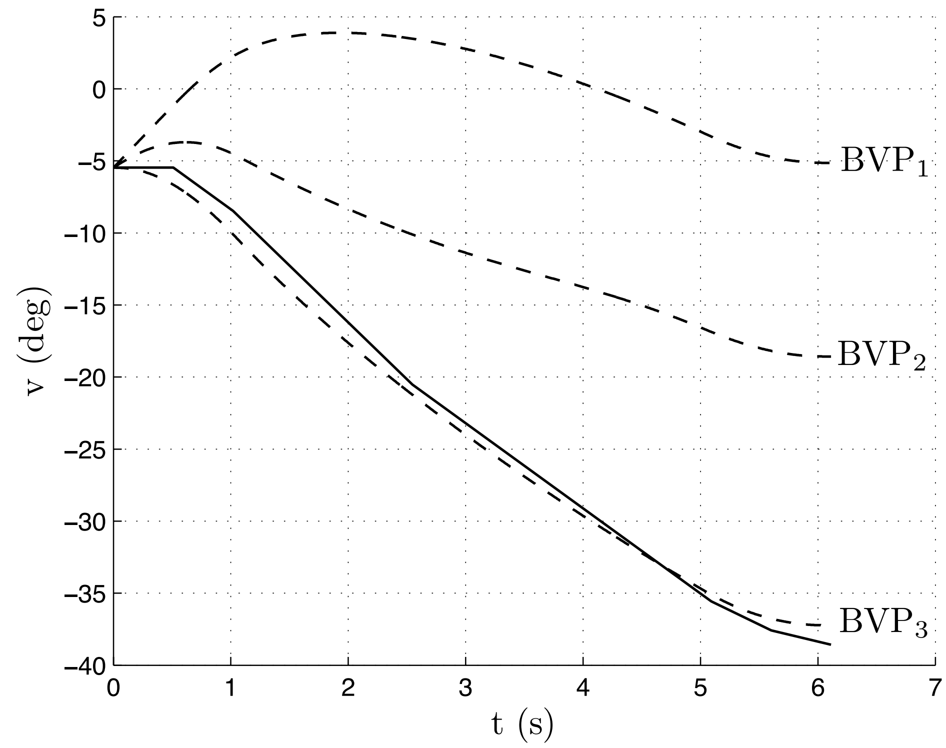
ries $v(\cdot)$ obtained from the three joint trajectories solution to the BVP (3.6) and (3.8), denoted BVP$_1$, BVP$_2$, and BVP$_3$, and the optimal discrete trajectory. Hereinafter, for plotting, the piecewise constant extension to $[t_0, t_f]$ for the optimal discrete trajectories is considered, i.e.,

$$\forall t \in [t_0, t_f],\ v^*_{h,d}(t) = v^*_i,\ i = [\frac{t_f - t_0}{h}].$$

In Figure 3.11(a), the grid is also represented, and examples of $\mathcal{A}_k^{h,d}$ are provided.



(a) With the grid resulting from the discretization.



(b) Without the grid resulting from the discretization.

Figure 3.11: The three solutions, BVP$_1$, BVP$_2$, and BVP$_3$, to the BVP (3.6) and (3.8) (dotted lines), and $v^*_{h,d}(\cdot)$ (plain line) when f_2 is taken as integral cost.

For the second experiment, the same instance of PCTS_{2,s_1} as for the first experiment is used. First, both h and d are allowed to converge towards zero with a constant ratio d/h, or a constant N_T/N_X ratio. Second, both h and d are allowed to converge towards zero with a ratio d/h also converging towards zero, or a ratio N_T/N_X converging towards zero. The resulting values for $J_0^{h,d}(v_0)$ are provided in Tables 3.2 and 3.3. In both these two tables, the column (∞,∞) indicates $J_0(\mathbf{q_0})$, while the row denoted nodes indicates the size of the grid. Note that, to have comparable grid size between Tables 3.2 and 3.3, whenever h and d are decreased by a factor δ in Table 3.2, h is decreased by $\delta^{2/3}$ and d by $\delta^{4/3}$ in Table 3.3. Therefore, in Table 3.2, the decrease in d is the square of the decrease in h. The results from Tables 3.2 and 3.3 are illustrated in Figures 3.12(a), 3.12(b), and 3.12(c). Note that because of rounding, the ratio N_T/N_X in Tables 3.2 does not exactly remain constant.

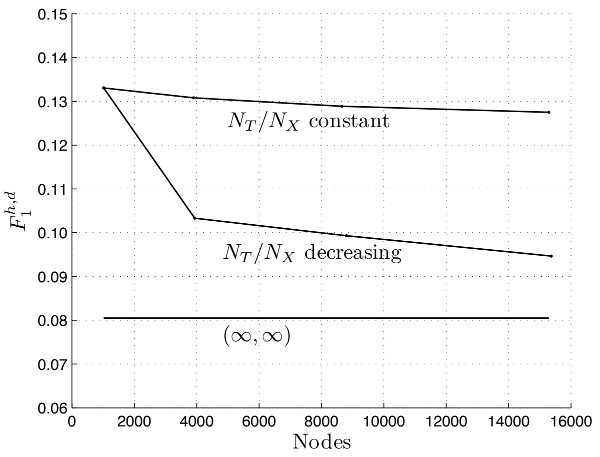
Table 3.2: Values of the objective functions when increasing both N_T and N_X with a constant N_T/N_X ratio.

(N_T, N_X)	(13,290)	(25,580)	(37,870)	(49,1160)	(∞,∞)
# nodes	1018	3898	8652	15290	-
$F_1^{h,d}$	0.133054	0.130806	0.128881	0.127499	0.080497
$F_2^{h,d}$	0.259263	0.275611	0.274619	0.277891	0.291530
$J_0^{h,d}(v_0)$	0.262685	0.268611	0.266190	0.266444	0.226262

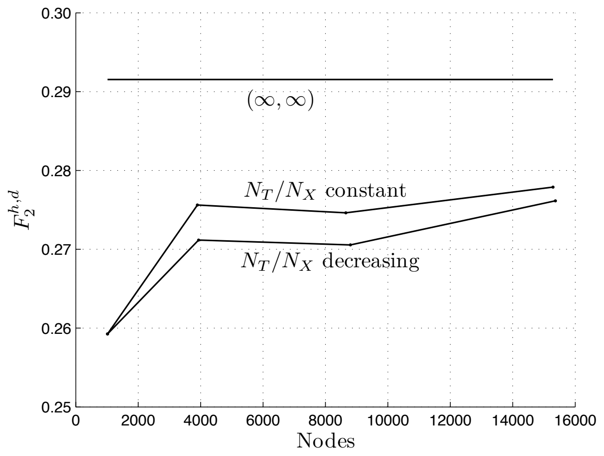
Table 3.3: Values of the objective functions when increasing both N_T and N_X with a N_T/N_X ratio converging towards zero.

(N_T, N_X)	(13,290)	(20,731)	(26,1255)	(31,1842)	(∞,∞)
# nodes	1018	3935	8798	15366	-
$F_1^{h,d}$	0.133054	0.103298	0.099311	0.094645	0.080497
$F_2^{h,d}$	0.259263	0.271164	0.270547	0.276147	0.291530
$J_0^{h,d}(v_0)$	0.262685	0.238880	0.234585	0.232718	0.226262

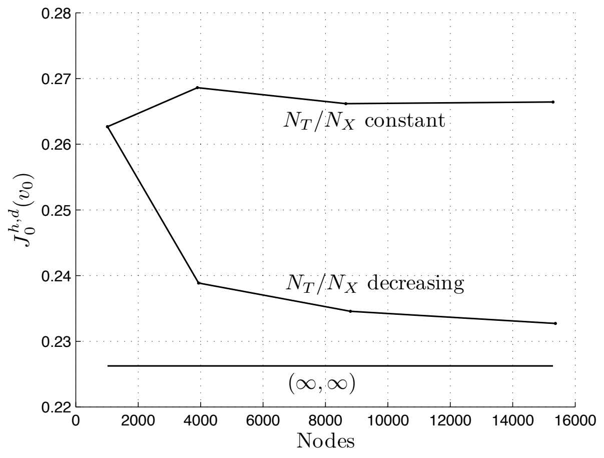
Figure 3.12(c) confirms that better convergence is obtained by letting the ratio d/h converge towards zero. Interestingly, it can be observed from Figures 3.12(a) and 3.12(b) that letting the ratio d/h converge towards zero mainly impacts $F_1^{h,d}$. This can be explained by the fact that $F_1^{h,d}$ does depend on the redundancy parameter derivative, while $F_2^{h,d}$ does not.



(a) The first objective function, $F_1^{h,d}$.



(b) The second objective function, $F_2^{h,d}$.



(c) The objective function $J_0^{h,d}(v_0) = F_1^{h,d} + wF_2^{h,d}$.

Figure 3.12: Visualization of the convergence results from Tables 3.2 and 3.3.

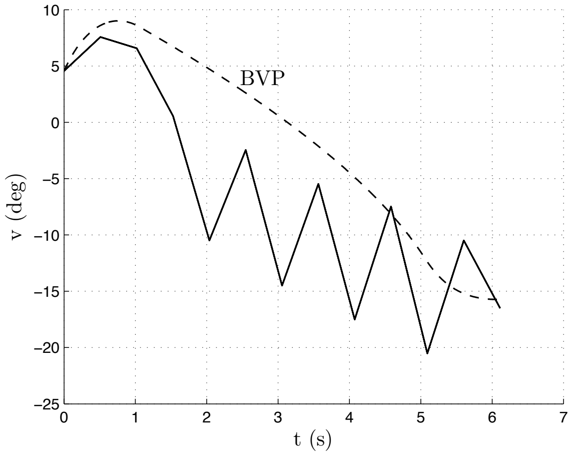
For the third experiment, a different instance of PCTS_{2,s_1} than the one used for the first two experiments is used. Again, w is set to 0.5. The values of $J_0^{h,d}(v_0)$ and $J_0(\mathbf{q_0})$ are provided in Table 3.4. Note that, for this instance, $F_1(\mathbf{q}^*(\cdot)) = 7.457361$ is much larger than the value for $F_1(\mathbf{q}^*(\cdot)) = 0.080497$ obtained in the first experiment, which indicates that the joint speeds are much larger. It can be observed in Table 3.4 that the value of $J_0^{h,d}(v_0)$ is very different from $J_0(\mathbf{q_0})$, which is confirmed in Figure 3.13(a), where the corresponding optimal trajectories are shown. The oscillations of the optimal discrete trajectory in Figure 3.13(a) are not acceptable. The first possibility to eliminate these oscillations would be to increase both N_T and N_X. As seen in Table 3.4 for $(N_T, N_X) = (78, 731)$, $J_0^{h,d}(v_0)$ approaches $J_0(\mathbf{q_0})$. The corresponding discrete optimal trajectory does not show the oscillations as illustrated in Figure 3.13(b). However, as mentioned at the beginning of this section, for computational efficiency, it is not desirable to increase the size of the grid, or change the values of d and h. Therefore, it is proposed to use the following heuristic approach to eliminate the oscillations:

$\Rightarrow$ Limit the number of times $\dot{v}_k \in \mathcal{B}_k^{h,d}(v_k)$ can change sign to m, referred to as the maximum number of modes.

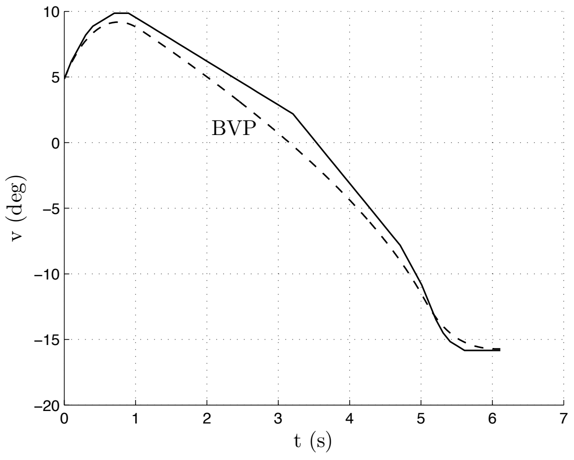
Note that this heuristic approach can easily be implemented with the dynamic programming equation (3.28). The resulting optimal discrete trajectory obtained for $m = 1$ is plotted in Figure 3.13(c). It can be observed that for this trajectory, the sign of $\dot{v}_k$ changes only one time. The sign of $\dot{v}_k$ is negative at the beginning of the trajectory, and then becomes positive. The value of $J_0^{h,d}(v_0)$ is provided for information in Table 3.4.

Table 3.4: Values of the objective functions for a second instance of PCTS$_2$.

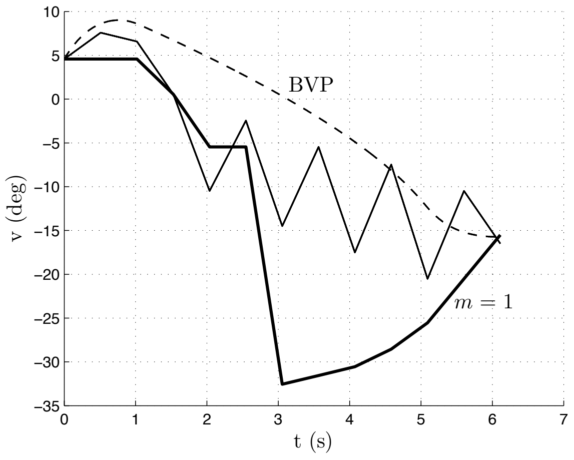
(N_T, N_X)	(13,290), $m = 1$	(13,290)	(78,731)	(∞, ∞)
# nodes	1475	1475	22254	-
$F_1^{h,d}$	4.399287	3.218412	7.314223	7.457361
$F_2^{h,d}$	0.189770	0.052837	0.033882	0.032749
$J_0^{h,d}(v_0)$	4.494172	3.244830	7.331164	7.473736



(a) $v^*_{h,d}(\cdot)$ with $(N_T, N_X) = (13, 290)$ (plain line).



(b) $v^*_{h,d}(\cdot)$ with $(N_T, N_X) = (78, 731)$ (plain line).



(c) $v^*_{h,d}(\cdot)$ with $(N_T, N_X) = (13, 290)$ (plain line), and $v^*_{h,d}(\cdot)$ with $(N_T, N_X) = (13, 290)$ and $m = 1$ (bold line).

Figure 3.13: Eliminating the oscillations by limiting the number of modes when f_2 is taken as an integral cost. For the three figures, the unique, and therefore optimal, solution to the BVP (3.6) and (3.8) is represented with a dotted line.

3.2.7 The DDP Approximation Method for PCTS_{2,s_2}

The same developments as in Sections 3.2.1, 3.2.2, 3.2.3, and 3.2.4 can be made for PCTS_{2,s_2}. The dynamic programming equation satisfied by the return function $J_k^h(\cdot)$ slightly differs, the main difference being the terminal data condition (3.36):

$$J_k^h(v_k) = \inf_{\dot{v}_k \in \mathcal{C}_k^h(v_k)} h\widetilde{f}_1(k, v_k, \dot{v}_k) + J_{k+1}^h(v_k + h\dot{v}_k), \tag{3.35}$$

with terminal data condition

$$J_{N_T}^h(v_{N_T}) = w\widetilde{f}_2(N_T, v_{N_T}). \tag{3.36}$$

Accordingly, the approximate return function $J_k^{h,d}(\cdot)$ solves:

$$\forall v_k \in \mathcal{A}_k^{h,d},\ J_k^{h,d}(v_k) = \inf_{\dot{v}_k \in \mathcal{B}_k^{h,d}(v_k)} h\widetilde{f}_1(k, v_k, \dot{v}_k) + J_{k+1}^{h,d}(v_k + h\dot{v}_k), \tag{3.37}$$

with terminal data condition

$$\forall v_{N_T} \in \mathcal{A}_{N_T}^{h,d},\ J_{N_T}^{h,d}(v_{N_T}) = w\widetilde{f}_2(N_T, v_{N_T}). \tag{3.38}$$

Propositions 3.3 and 3.4 can be easily adapted. Proposition 3.6 provides an error estimate between the return function $J_k^h(\cdot)$ and the approximate return function $J_k^{h,d}(\cdot)$. This error estimate allows to conclude to the convergence of $J_0^{h,d}(v_0)$ towards $J_0^h(v_0)$. The proof of Proposition 3.6 first requires showing that the return functions $J_k^h(\cdot)$ are Lipschitz, which is established in Proposition 3.5.

Proposition 3.5. *Assume that the functions $\widetilde{f}_1(t, \cdot, \xi)$ and $\widetilde{f}_2(t, \cdot)$ are Lipschitz, i.e.,*

$$\forall t \in [t_0, t_f],\ \forall \xi \in U,\ \forall (x, y) \in \mathbf{R} \times \mathbf{R},\ |\widetilde{f}_1(t, x, \xi) - \widetilde{f}_1(t, y, \xi)| \leq K_1|x - y|,$$

and

$$\forall t \in [t_0, t_f],\ \forall (x,y) \in \mathbf{R} \times \mathbf{R},\ |\widetilde{f}_2(t,x) - \widetilde{f}_2(t,y)| \leq K_2|x-y|,$$

then

$$\forall (x,y) \in \mathbf{R} \times \mathbf{R},\ |J_k^h(x) - J_k^h(y)| \leq L_k|x-y|h, \tag{3.39}$$

where $L_k = K_1(N_T - k) + wK_2/h$.

Proposition 3.6. *Under the same assumptions as in Proposition* 3.5*, and the assumption that the function* $\widetilde{f}_1(t,x,\cdot)$ *is Lipschitz*

$$\forall t \in [t_0, t_f],\ \forall v \in \mathbf{R},\ \forall (x,y) \in U \times U,\ |\widetilde{f}_1(t,v,x) - \widetilde{f}_1(t,v,y)| \leq K_3|x-y|,$$

then, for all $x \in \mathcal{A}_k^{h,d}$*,*

$$0 \leq J_k^{h,d}(x) - J_k^h(x) \leq L'_k d,$$

where $L'_k = (K_3 + L_0 h)(N_T - k)$.

The main case of interest for Proposition 3.6 is when $k = 0$. In such a case,

$$0 \leq J_0^{h,d}(x) - J_0^h(x) \leq L'_0 d,$$

where,

$$L'_0 = (K_3 + L_0 h)N_T = (K_3 + K_1 N_T h + wK_2)N_T.$$

Knowing that $N_T h = t_f - t_0$, we finally obtain:

$$0 \leq J_0^{h,d}(x) - J_0^h(x) \leq (K_3 + K_1(t_f - t_0) + wK_2)(t_f - t_0)d/h = Cd/h. \tag{3.40}$$

A similar discussion about the error estimate obtained in Proposition 3.6 as in Section 3.2.5 can be made. The main consequence of the inequality (3.40) is that $J_0^{h,d}(v_0)$ converges towards $J_0^h(v_0)$ when d converges towards zero.

3.2.8 Numerical Experiments for PCTS_{2,s_2}

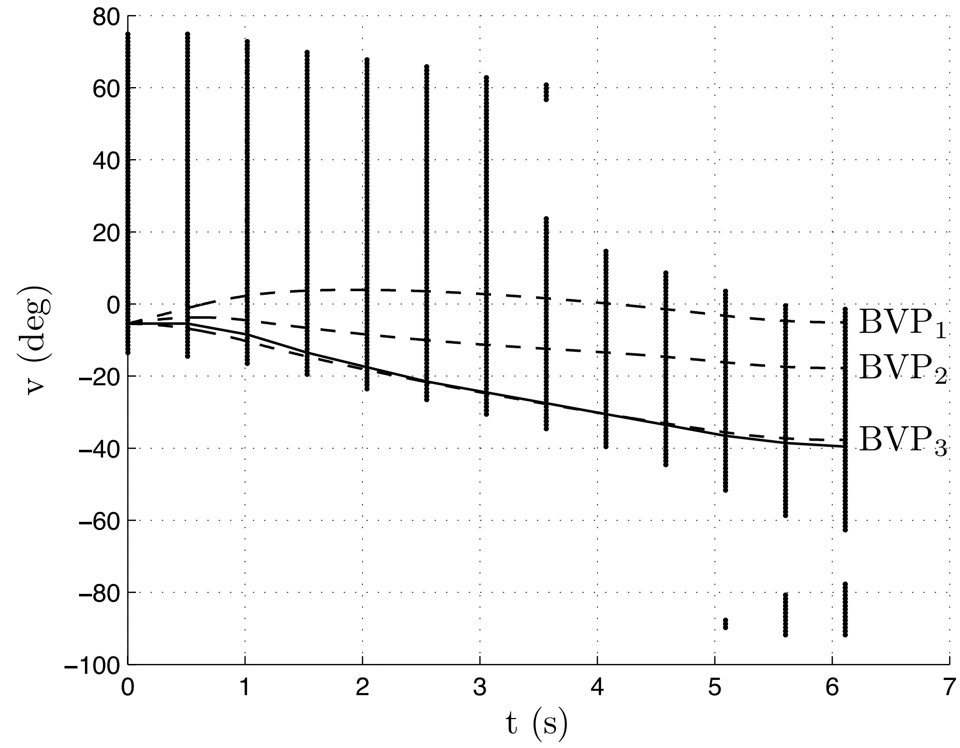
The same three experiments as in Section 3.2.6 are conducted for PCTS_{2,s_2}. Note that the results obtained for PCTS_{2,s_1} and PCTS_{2,s_2} are very similar, particularly for the third experiment. This similarity is just arising from our choice of examples.

For the first experiment, the values of $J_0^{h,d}(v_0)$ and $J_0(\mathbf{q_0})$ are provided in Table 3.5. $F_2(\mathbf{q}^*(\cdot))$ and $F_2^{h,d}$ are now respectively equal to $f_2(\mathbf{q}^*(t_f))$ and $\widetilde{f}_2(N_T, v^*_{N_T})$.

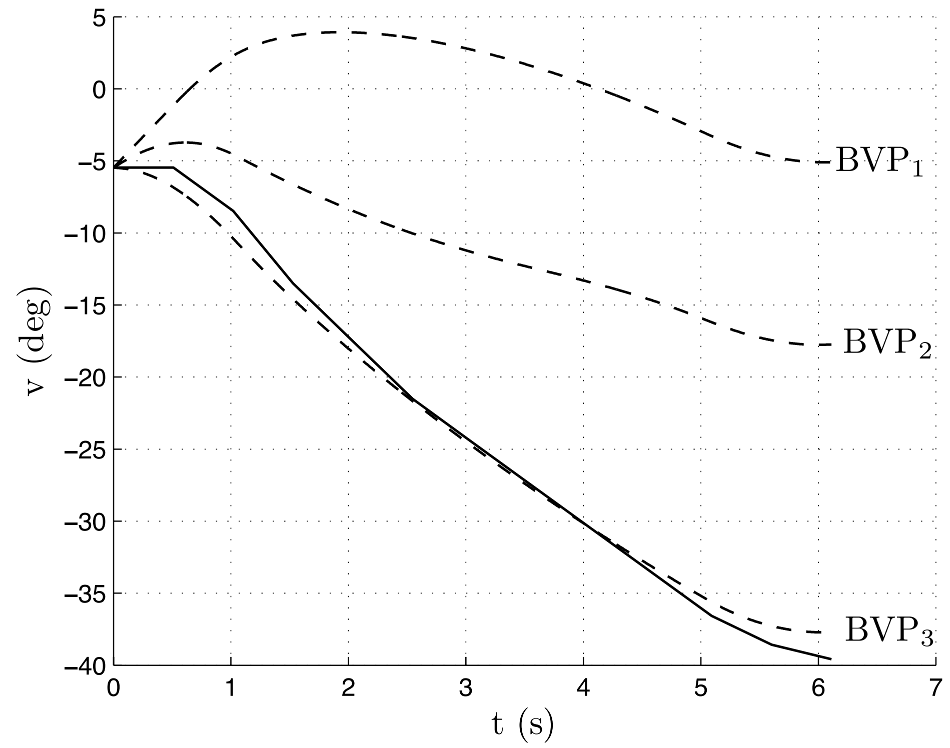
Table 3.5: Values of the objective functions for $(N_T, N_X) = (13, 290)$.

$F_1^{h,d}$	$F_1(\mathbf{q}^*(\cdot))$	$F_2^{h,d}$	$F_2(\mathbf{q}^*(\cdot))$	$J_0^{h,d}(v_0)$	$J_0(\mathbf{q_0})$
0.127024	0.077434	0.114659	0.104485	0.184353	0.129676

In Figures 3.14(a) and 3.14(b), the three trajectories $v(\cdot)$ obtained from the three joint trajectories solution to the BVP (3.7) and (3.9), denoted BVP_1, BVP_2, and BVP_3, and the optimal discrete trajectory are plotted.



(a) With the grid resulting from the discretization.



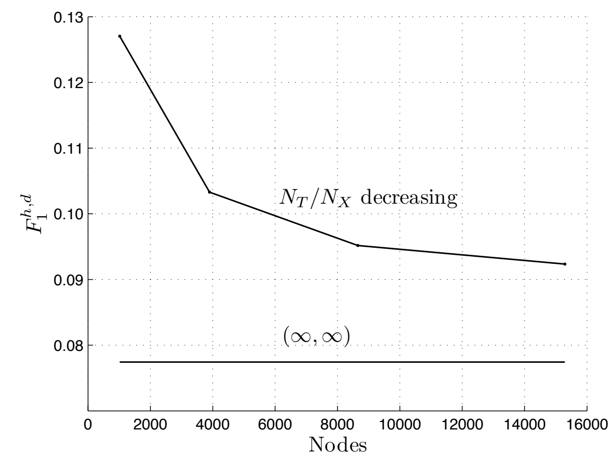
(b) Without the grid resulting from the discretization.

Figure 3.14: The three solutions, BVP$_1$, BVP$_2$, and BVP$_3$, to the BVP (3.7) and (3.9) (dotted lines), and $v^*_{h,d}(\cdot)$ (plain line) when f_2 is taken as terminal cost.

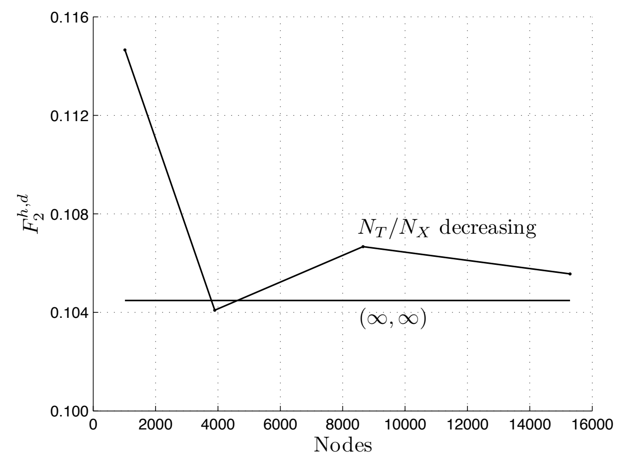
For the second experiment, both h and d are allowed to converge towards zero with a ratio d/h also converging towards zero, or a ratio N_T/N_X converging towards zero. The resulting values for $J_0^{h,d}(v_0)$ are provided in Table 3.6. The results from Table 3.6 are illustrated in Figures 3.15(a), 3.15(b), and 3.15(c).

Table 3.6: Values of the objective functions when increasing both N_T and N_X with a N_T/N_X ratio converging towards zero.

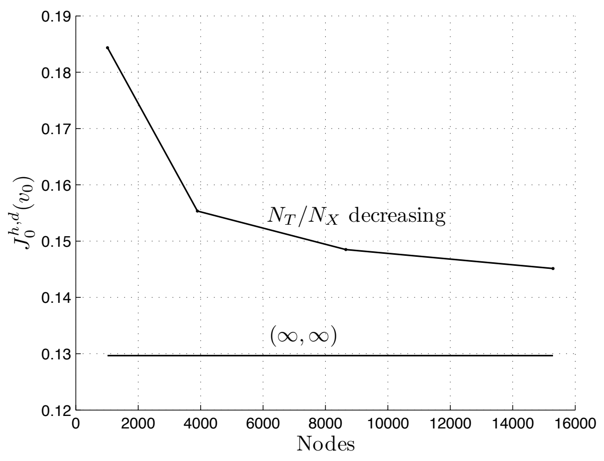
(N_T, N_X)	(13,290)	(20,731)	(26,1255)	(31,1842)	(∞, ∞)
# nodes	1018	3935	8798	15366	-
F_1	0.127024	0.103298	0.095174	0.092355	0.077434
F_2	0.114659	0.104089	0.106673	0.105563	0.104485
$J_0^{h,d}(v_0)$	0.184353	0.155343	0.148510	0.145137	0.129676



(a) The first objective function, $F_1^{h,d}$.



(b) The second objective function, $F_2^{h,d}$.



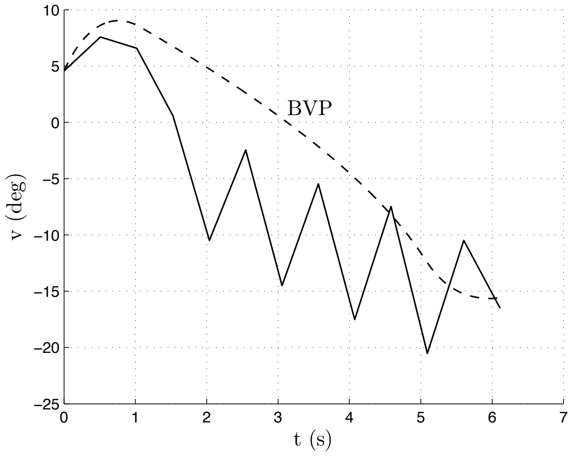
(c) The objective function for $J_0^{h,d}(v_0) = F_1^{h,d} + wF_2^{h,d}$.

Figure 3.15: Visualization of the convergence results from Table 3.6.

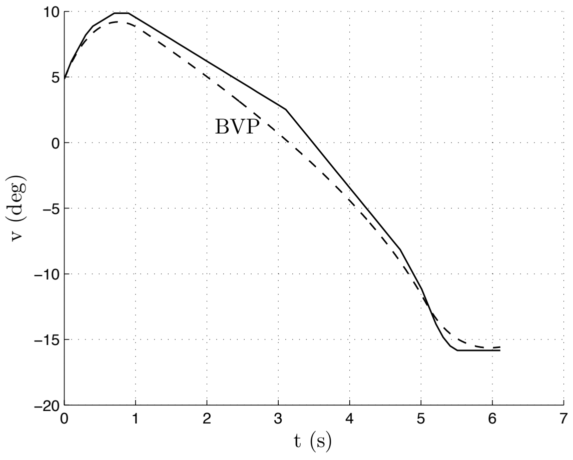
For the third experiment, oscillations for the optimal discrete trajectory are also obtained in Figure 3.16(a). These oscillations can be eliminated by increasing both N_T and N_X, as shown for $(N_T, N_X) = (78, 731)$ in Table 3.7 and as illustrated in Figure 3.16(b). The heuristic approach also eliminates the oscillations as shown with $m = 1$ in Table 3.7 and as illustrated in Figure 3.16(c).

Table 3.7: Values of the objective functions for a second instance of PCTS$_2$.

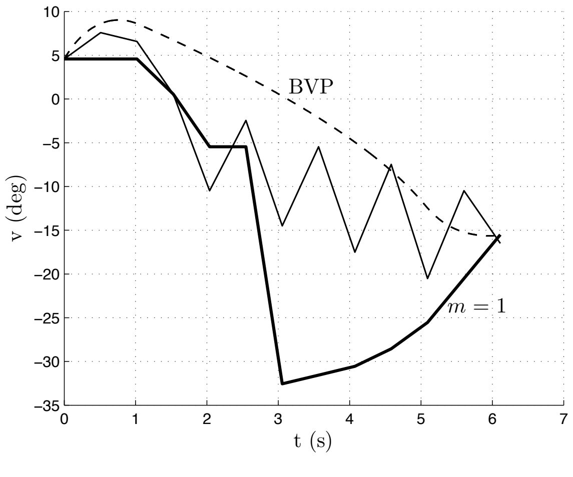
(N_T, N_X)	(13,290), $m = 1$	(13,290)	(78,731)	(∞, ∞)
# nodes	1475	1475	22254	-
F_1	4.399287	3.218412	7.313932	7.457426
F_2	0.018189	0.020603	0.018973	0.018370
$J_0^{h,d}(v_0)$	4.408382	3.228713	7.323418	7.466611



(a) $v^*_{h,d}(\cdot)$ with $(N_T, N_X) = (13, 290)$ (plain line).



(b) $v^*_{h,d}(\cdot)$ with $(N_T, N_X) = (78, 731)$ (plain line).



(c) $v^*_{h,d}(\cdot)$ with $(N_T, N_X) = (13, 290)$ (plain line), and $v^*_{h,d}(\cdot)$ with $(N_T, N_X) = (13, 290)$ and $m = 1$ (bold line).

Figure 3.16: Eliminating the oscillations by limiting the number of modes when f_2 is taken as a terminal cost. For the three figures, the unique, and therefore optimal, solution to the BVP (3.6) and (3.8) is represented with a dotted line.

3.3 A DDP Approximation for a Multiple Objective Function Problem

3.3.1 Presentation

The DDP approximation proposed method in Section 3.2 for PCTS_{2,s_1} can be generalized to PCTS_2. For clarity, PCTS_2, after reformulation with the redundancy parameter as performed in Section 3.2.2 for PCTS_{2,s_1}, is now presented.

Problem PCTS_2: find the Pareto optimal set $\mathcal{E}(\mathrm{cl}(\widetilde{\mathbf{F}}(\mathcal{T})), \mathbf{R}^2_+)$, where $\widetilde{\mathbf{F}}(\cdot)$ is the real vector-valued objective function defined by

$$\widetilde{\mathbf{F}}(\cdot):\ v(\cdot) \in C^1([t_0, t_f], \mathbf{R}) \to \left(\int_{t_0}^{t_f} \widetilde{f}_1(t, v(t), \dot{v}(t))\mathrm{d}t, \int_{t_0}^{t_f} \widetilde{f}_2(t, v(t))\mathrm{d}t \right),$$

where $\mathcal{T}$ is the subset of $C^1([t_0, t_f], \mathbf{R})$ such that the following constraints are satisfied:

$$v(t) \in \mathcal{A}(t),$$

$$\dot{v}(t) \in \mathcal{B}(t, v(t)).$$

The same first-order discretization in time as performed in Section 3.2.2 yields the following multiobjective nonlinear programming problem: find the Pareto optimal set $\mathcal{E}(\mathrm{cl}(\widetilde{\mathbf{F}}^h(\mathcal{T}^h_0(v_0))), \mathbf{R}^2_+)$, where $\widetilde{\mathbf{F}}^h(\cdot)$ is the real vector-valued objective function defined by

$$\widetilde{\mathbf{F}}^h(\cdot):\ \{v_i,\ i = 1, \dots, N_T\} \in \mathcal{T}^h_0(v_0) \to \left(h \sum_{i=0}^{N_T-1} \widetilde{f}_1(i, v_i, \dot{v}_i), h \sum_{i=0}^{N_T-1} \widetilde{f}_2(i, v_i) \right),$$

v_0 being the known initial condition.

The dynamic programming equation (3.28) with the terminal data condition (3.29) obtained in Section 3.2.3 can also be generalized to take into account multiple objective functions. Let $J_k^h(\cdot)$ be the set-valued return function that associates, for each $v_k \in \mathcal{A}_k^h$, the Pareto optimal set $\mathcal{E}(\mathrm{cl}(\widetilde{\mathbf{F}}_h(\mathcal{T}_k^h(v_k))), \mathbf{R}_+^2)$:

$$J_k^h(\cdot): \ v_k \in \mathcal{A}_k^h \to J_k^h(v_k) = \mathcal{E}(\mathrm{cl}(\widetilde{\mathbf{F}}^h(\mathcal{T}_k^h(v_k))), \mathbf{R}_+^2). \tag{3.41}$$

Note that setting $k = 0$ in (3.41) yields exactly the multiobjective nonlinear programming problem described above. The set-valued return function $J_k^h(\cdot)$ can be shown to satisfy the dynamic programming equation

$$J_k^h(v_k) = \mathcal{E}(\mathrm{cl}(\{h(\widetilde{f}_1(k, v_k, \dot{v}_k), \widetilde{f}_2(k, v_k)) + J_{k+1}^h(v_k + h\dot{v}_k), \ \dot{v}_k \in \mathcal{B}_k^h(v_k)\}), \mathbf{R}_+^2), \tag{3.42}$$

with terminal data condition

$$J_{N_T}^h(v_{N_T}) = \{(0,0)\}. \tag{3.43}$$

Finally, the same discretization in the redundancy parameter as performed in Section 3.2.4 leads to the introduction of the approximate return set-valued function $J_k^{h,d}(\cdot)$ defined as the solution to the dynamic programming equation (3.42) with the terminal data condition (3.43) where again the redundancy parameter is restricted to take on values only in $\mathcal{A}_k^{h,d}$, i.e., $\forall v_k \in \mathcal{A}_k^{h,d}$,

$$J_k^{h,d}(v_k) = \mathcal{E}(\{h(\widetilde{f}_1(k, v_k, \dot{v}_k), \widetilde{f}_2(k, v_k)) + J_{k+1}^{h,d}(v_k + h\dot{v}_k), \ \dot{v}_k \in \mathcal{B}_k^{h,d}(v_k)\}, \mathbf{R}_+^2), \tag{3.44}$$

with terminal data condition

$$\forall v_{N_T} \in \mathcal{A}_{N_T}^{h,d}, \ J_{N_T}^{h,d}(v_{N_T}) = \{(0,0)\}. \tag{3.45}$$

Note that the closure has been removed in the dynamic programming equation (3.44), as all the sets involved are finite. The approximate dynamic programming equation (3.44) with terminal data condition (3.45) is straightforward to solve. Indeed, let $v_k \in \mathcal{A}_k^{h,d}$ and assume that the approximate set-valued return function $J_{k+1}^{h,d}(\cdot)$ is known. Knowing $\dot{v}_k \in \mathcal{B}_k^{h,d}(v_k)$, the term $h(\widetilde{f}_1(k, v_k, \dot{v}_k), \widetilde{f}_2(k, v_k))$ can be calculated, from which the set $\{h(\widetilde{f}_1(k, v_k, \dot{v}_k), \widetilde{f}_2(k, v_k)) + J_{k+1}^{h,d}(v_k + h\dot{v}_k)\}$ can be determined. The set $\mathcal{B}_k^{h,d}(v_k)$ being finite, the set $\{h(\widetilde{f}_1(k, v_k, \dot{v}_k), \widetilde{f}_2(k, v_k)) + J_{k+1}^{h,d}(v_k + h\dot{v}_k),\ \dot{v}_k \in \mathcal{B}_k^{h,d}(v_k)\}$ is also finite. Therefore, to determine the Pareto optimal set of $\{h(\widetilde{f}_1(k, v_k, \dot{v}_k), \widetilde{f}_2(k, v_k)) + J_{k+1}^{h,d}(v_k + h\dot{v}_k),\ \dot{v}_k \in \mathcal{B}_k^{h,d}(v_k)\}$, which is precisely $J_k^{h,d}(v_k)$, a finite number of comparisons are needed. Repeating this procedure for every $v_k \in \mathcal{A}_k^{h,d}$ yields the approximate set-valued return function $J_k^{h,d}(\cdot)$. Therefore, starting from the terminal data condition (3.44), the approximate set-valued return function $J_0^{h,d}(\cdot)$ can be recursively obtained.

3.3.2 Application to the Variant of PCTS_2

The developments in Section 3.3.1 can be easily generalized when f_2 is taken as a terminal cost. The dynamic programming equation satisfied by the set-valued return function $J_k^h(\cdot)$ (3.42) becomes:

$$J_k^h(v_k) = \mathcal{E}(\mathrm{cl}(\{h(\widetilde{f}_1(k, v_k, \dot{v}_k), 0) + J_{k+1}^h(v_k + h\dot{v}_k),\ \dot{v}_k \in \mathcal{B}_k^h(v_k)\}), \mathbf{R}_+^2). \tag{3.46}$$

The terminal data condition (3.43) involves now $\widetilde{f}_2$:

$$J_{N_T}^h(v_{N_T}) = \{(0, \widetilde{f}_2(N_T, v_{N_T}))\}. \tag{3.47}$$

Accordingly, the approximate set-valued return function $J_k^{h,d}(\cdot)$ solves:

$$\forall v_k \in \mathcal{A}_k^{h,d},\ J_k^{h,d}(v_k) = \mathcal{E}(\{h(\widetilde{f}_1(k, v_k, \dot{v}_k), 0) + J_{k+1}^{h,d}(v_k + h\dot{v}_k),\ \dot{v}_k \in \mathcal{B}_k^{h,d}(v_k)\}, \mathbf{R}_+^2), \tag{3.48}$$

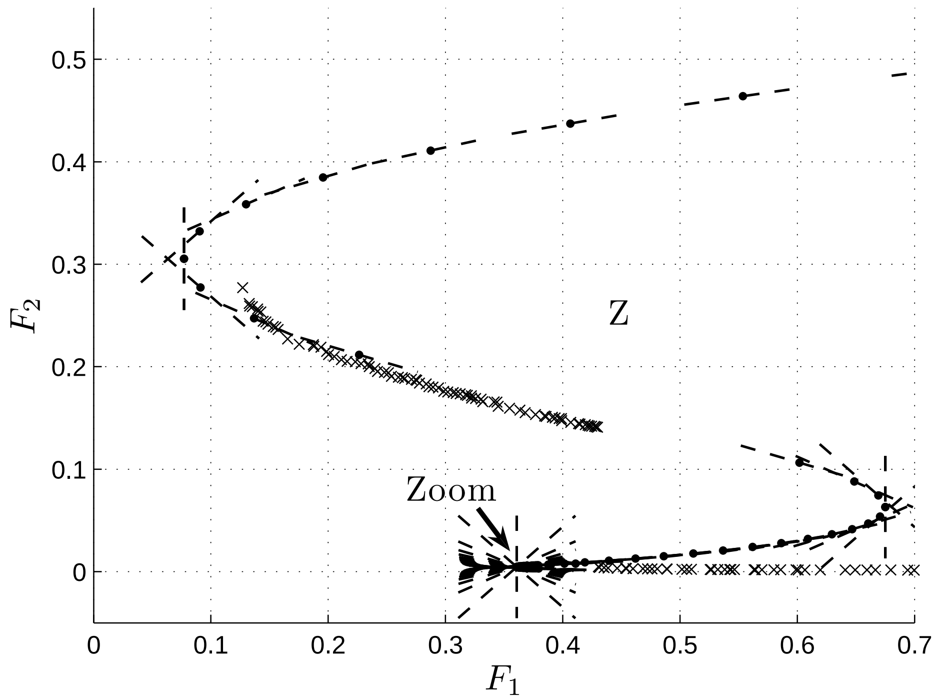
with terminal data condition

$$\forall v_{N_T} \in \mathcal{A}^{h,d}_{N_T},\ J^{h,d}_{N_T}(v_{N_T}) = \{(0, \widetilde{f}_2(N_T, v_{N_T}))\}. \tag{3.49}$$

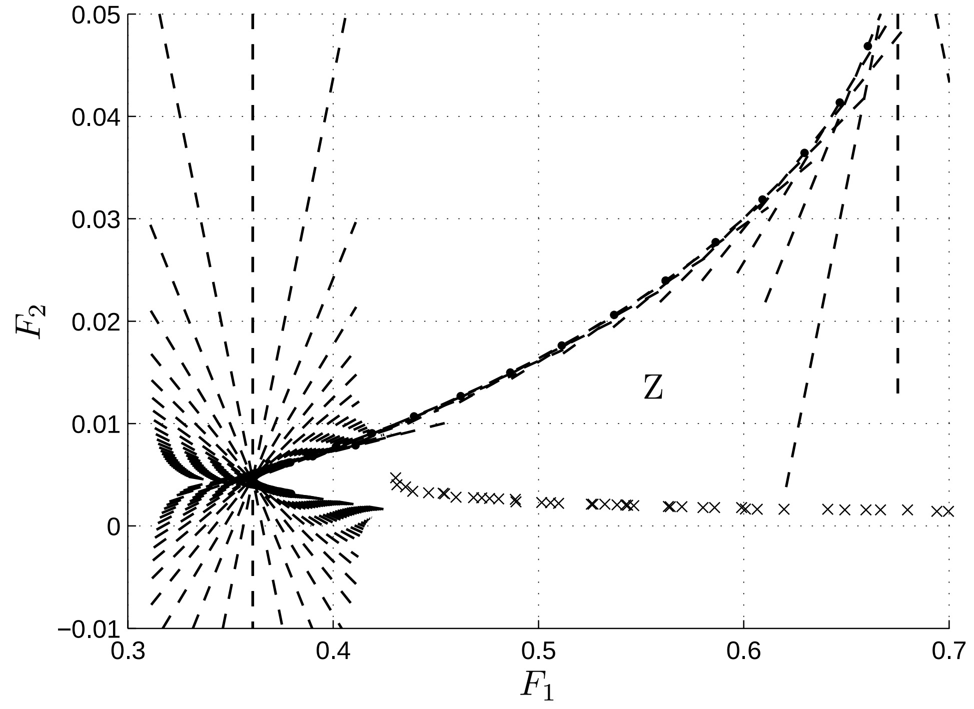
3.3.3 Numerical Experiments

The numerical experiments presented next use the instance of PCTS$_2$ from Sections 3.1.3 and 3.1.4. The first objective of these experiments is to compare the objective spaces that were obtained in Section 3.1.4 with the set $J^{h,d}_0(v_0)$ obtained from the resolution of the approximate dynamic programming equations (3.44) and (3.48) with their respective terminal data conditions (3.45) and (3.49). The second objective of these experiments is to compare $J^{h,d}_0(v_0)$ with the Pareto objective vectors obtained with the weighting method. This comparison has to be done carefully as the Pareto objective vectors obtained with the weighting method in Section 3.1.3 are "exact", whereas the elements of $J^{h,d}_0(v_0)$ are approximate Pareto objective vectors. However, qualitatively, it will be seen that approximate Pareto objective vectors can be obtained in regions where the weighting method fails to generate Pareto objective vectors. Detailed quantitative investigations on this subject will be performed in Chapter 5.

Let h and d be chosen as in Section 3.2.6, and the number of modes be unlimited. Therefore, $(N_T, N_X) = (13, 290)$. In Figures 3.17(a) and 3.17(b), the objective space obtained in Section 3.1.4 is plotted together with the set of approximate Pareto objective vectors $J^{h,d}_0(v_0)$. Note that $|J^{h,d}_0(v_0)| = 138$. The main observation from these two figures is that $J^{h,d}_0(v_0)$ is fairly close to the boundary of the objective space, except the subset of $J^{h,d}_0(v_0)$ in Figure 3.17(b). In this region of the objective space, first, the value of F_1 is higher, therefore, as observed in Section 3.2.6, the accuracy of the DDP approximation is less. Second, and more importantly, not all the joint trajectories corresponding to boundary points satisfy the joint mechanical limits whereas all the optimal discrete trajectories do satisfy this constraint. Therefore, the comparison cannot really be performed.



(a) Boundary points (dots), tangents at each boundary point (dotted lines), and approximate Pareto objective vectors (crosses).



(b) Zoom.

Figure 3.17: Comparing the set of approximate Pareto objective vectors $J_0^{h,d}(v_0)$ and the objective space when f_2 is taken as an integral cost.

It would be interesting to perform an experimental convergence study of $J_0^{h,d}(v_0)$ towards the Pareto optimal set for PCTS$_2$. However, this is not practically feasible. Indeed, as discussed in Chapter 5, the algorithmic complexity for the resolution of the approximate dynamic programming equation (3.44) increases exponentially with the grid size. As an example, it took 24,628 s, or more than 6 h, to obtain the set $J_0^{h,d}(v_0)$ above. In Chapter 5, it will be shown how the time required to solve the approximate dynamic programming equation (3.44) can be drastically reduced without too much loss of optimality.

In Figure 3.18, the same information as in Figure 3.7(a) is reproduced, except for the Pareto objective vectors obtained from the boundary points (this is only for clarity). Figure 3.18 also includes the set $J_0^{h,d}(v_0)$. Three remarks can be made about $J_0^{h,d}(v_0)$.

- The fact that the Pareto optimal set is disconnected, or at least composed of two clear subsets, is captured well.
- It was mentioned in Section 3.1.4 that the weighting method fails to generate Pareto objective vectors in the set $\mathbf{z_2} - \mathrm{int}(\mathbf{R}^2_+)$. Clearly, $J_0^{h,d}(v_0) \cap \mathbf{z_2} - \mathrm{int}(\mathbf{R}^2_+) \neq \emptyset$.
- The points in $J_0^{h,d}(v_0)$ seem to be evenly distributed, and therefore provide a good representation of the Pareto optimal set.

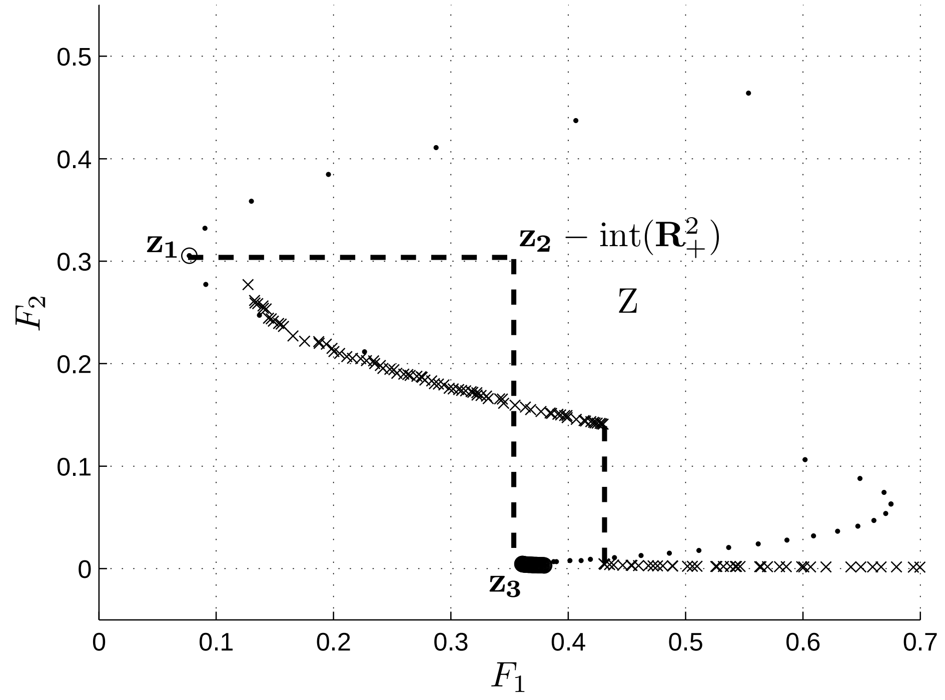
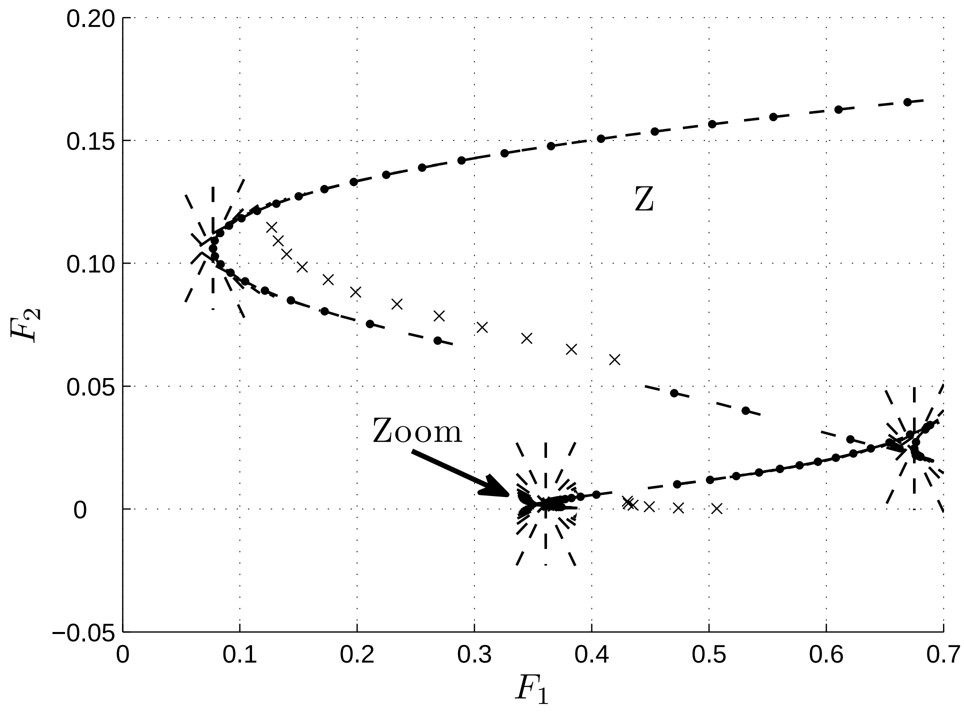


Figure 3.18: When f_2 is taken as an integral cost, as opposed to the weighting method, approximate Pareto objective vectors can be found in the set $\mathbf{z_2} - \mathrm{int}(\mathbf{R}^2_+)$.

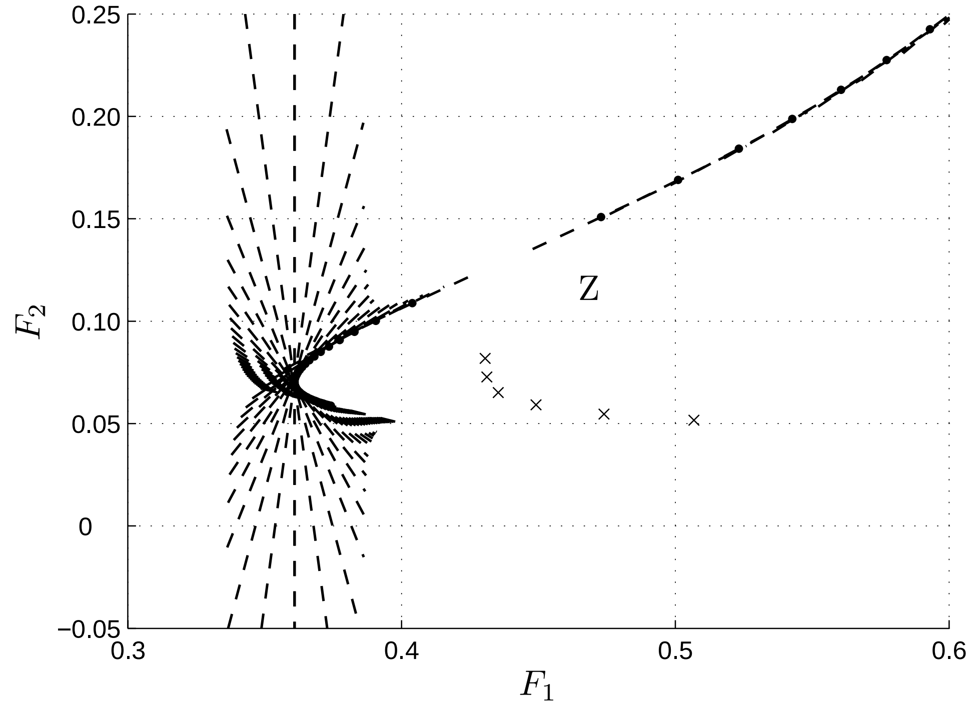
The results obtained when f_2 is taken as a terminal cost will now be presented. The same discussion as above can be made. However, two extra interesting remarks are worth mentioning:

- $|J_0^{h,d}(v_0)| = 18$. This is much smaller than 138, as obtained above. When f_2 is taken as a terminal cost, it is easy to see that there is at most one optimal discrete trajectory that has $v_{N_T} \in \mathcal{A}_{N_T}^{h,d}$ as terminal data condition. Therefore, $|J_0^{h,d}(v_0)|$ is necessarily bounded above by $|\mathcal{A}_{N_T}^{h,d}|$.

- The time needed to solve the approximate dynamic programming equation (3.48) is only 77 s. The explanation for this is similar to the explanation provided for the cardinality of $J_0^{h,d}(v_0)$. For each $v_k \in \mathcal{A}_k^{h,d}$, $|J_k^{h,d}(v_k)|$ is necessarily bounded above by $|\mathcal{A}_{N_T}^{h,d}|$. Therefore, the algorithmic complexity for the resolution of the approximate dynamic programming equation (3.48) is in fact polynomial in the grid size instead of exponential.



(a) Boundary points (dots), tangents at each boundary point (dotted lines), and approximate Pareto objective vectors (crosses).



(b) Zoom.

Figure 3.19: Comparing the set of approximate Pareto objective vectors $J_0^{h,d}(v_0)$ and the objective space when f_2 is taken as a terminal cost.

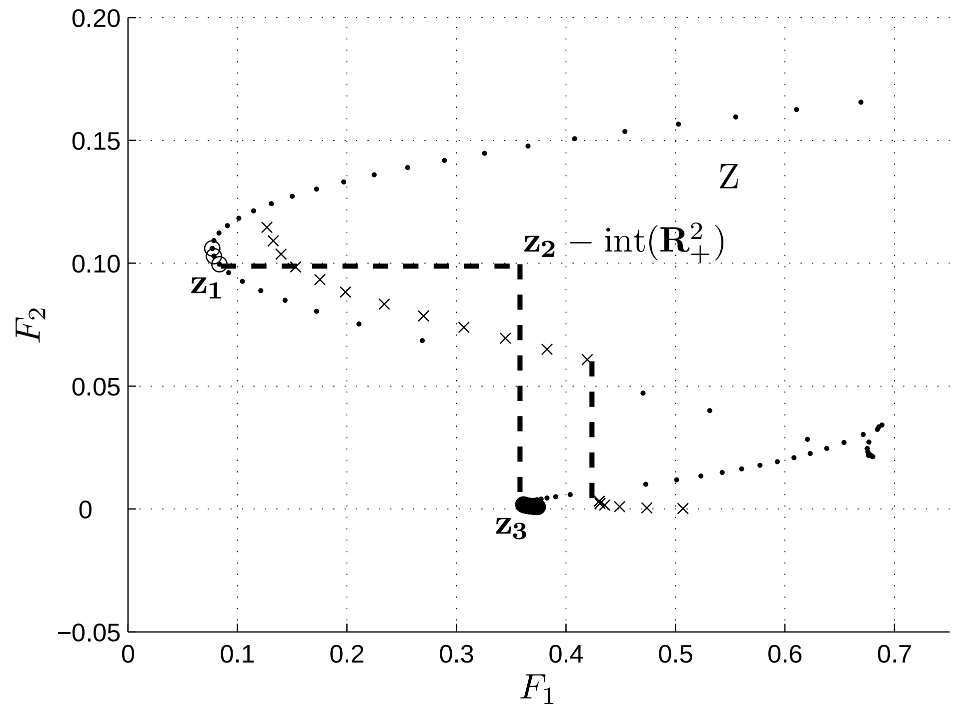


Figure 3.20: When f_2 is taken as a terminal cost, as opposed to the weighting method, approximate Pareto objective vectors can be found in the set $\mathbf{z_2} - \mathrm{int}(\mathbf{R}^2_+)$.

3.4 Summary of Results

The objective of this chapter was to solve PCTS$_2$, i.e., to obtain a good representation of the Pareto optimal set for PCTS$_2$. The first approach was to use the traditional weighting method. The weighting method was shown to only be able to identify a small subset of the Pareto optimal set. As an alternative to the weighting method, a DDP approximation method was proposed. An interesting advantage of this approximation method is that the constraints for PCTS$_2$ can be easily handled. Constraints have the impact of reducing the size of the sets $\mathcal{A}_k^{h,d}$ and $\mathcal{B}_k^{h,d}(v_k)$, which in turn makes the resolution of the approximate dynamic programming equation faster. More importantly, it was shown that a good representation of the Pareto optimal set can be obtained with the proposed DDP approximation method. Similar conclusions were obtained with a variant to PCTS$_2$ where the aerodynamic interference function was taken as a terminal cost instead of an integral cost.

Some work remains to make the proposed DDP approximation method practically applicable. First, sufficiently smooth joint trajectories need to be built from the optimal discrete trajectories $\{v_i^*,\ i = 0, \ldots, N_T\}$. Second, as already mentioned, for practical reasons, PCTS$_2$ must be able to be solved very efficiently. Therefore, the time required to solve the approximate dynamic programming equation needs to be reduced dramatically. These two questions are addressed in Chapter 5. In Chapter 5, it is also attempted to compare quantitatively the Pareto objective vectors obtained with the weighting method and the approximate Pareto objective vectors obtained with the DDP approximation method.

However, first, Chapter 4 investigates whether the DDP approximation method proposed for PCTS$_2$ can be applied to a more general class of problems.

Chapter 4

A Multiobjective Optimal Control Problem

In Chapter 3, PCTS$_2$ was reformulated using the redundancy parameter, v, as follows: find the Pareto optimal set $\mathcal{E}(\mathrm{cl}(\widetilde{\mathbf{F}}(\mathcal{T})), \mathbf{R}^2_+)$, where $\widetilde{\mathbf{F}}(\cdot)$ is the real vector-valued objective function defined by

$$\widetilde{\mathbf{F}}(\cdot):\ v(\cdot) \in C^1([t_0, t_f], \mathbf{R}) \to \left(\int_{t_0}^{t_f} \widetilde{f}_1(t, v(t), \dot{v}(t))\mathrm{d}t, \int_{t_0}^{t_f} \widetilde{f}_2(t, v(t))\mathrm{d}t \right),$$

where $\mathcal{T}$ is the subset of $C^1([t_0, t_f], \mathbf{R})$ such that the following constraints are satisfied:

$$v(t) \in \mathcal{A}(t),$$

$$\dot{v}(t) \in \mathcal{B}(t, v(t)),$$

with $\mathcal{A}(t) \subset [v_{\min}, v_{\max}]$, $\mathcal{B}(t, v(t)) \subset U$, and initial redundancy parameter v_0. By retaining only the constraint $\dot{v}(t) \in U$, and defining the control $u(\cdot)$ as $u(t) = \dot{v}(t)$, PCTS$_2$ can be seen as a particular case of the following multiobjective deterministic finite horizon optimal control problem: find the minimal element set $\mathcal{E}(\text{cl}(\mathbf{F}(x_0,\mathcal{U})), D)$, where $\mathbf{F}(\cdot)$ is a real vector-valued objective function defined by

$$\mathbf{F}(x_0, u(\cdot)) = \Big(\int_{t_0}^{t_f} L_1(t, x(t), u(t))\mathrm{d}t, \int_{t_0}^{t_f} L_2(t, x(t), u(t))\mathrm{d}t) \Big),$$

$\mathcal{U}$ is a set of controls $u(\cdot): [t_0, t_f] \to U$, and U is a compact set. Given a control in $\mathcal{U}$, the trajectory $x(\cdot)$ is the solution to the differential equation

$$\dot{x}(s) = f(s, x(s), u(s)),\ t_0 \leq s \leq t_f,$$

with initial conditions

$$x(t_0) = x_0.$$

The objective of this chapter is to apply the DDP approximation method proposed in Chapter 3 to the above general multiobjective deterministic finite horizon optimal control problem, and to study the convergence of the approximation. This achievement, published in [31], constitutes an advance in optimal control theory.

The multiobjective optimal control problem studied in this chapter, as well as the assumptions required for the convergence study, are detailed in Section 4.1. Some mathematical preliminaries follow in Section 4.2. In particular, a topology on the family of compact sets of $\mathbf{R}^p$ defined from the Hausdorff distance [1] is introduced. With this topology, the minimal element map, which is the map that associates its minimal element set with each compact set, is shown to be continuous. The external stability property ([27, p. 53], [26, p. 59]) for compact sets is also stated in Section 4.2. As in Chapter 3, the proposed approximation method starts with a first-order discretization in time detailed in Section 4.3. This discretization yields a discrete multiobjective optimal control problem, called the *discrete* *problem*. In Section 4.4, it is shown that by choosing a particular sequence of time steps and using the external stability property, convergent sequences of minimal elements of the corresponding discrete problems can be constructed. In Section 4.5, using the dynamic programming principle [54, 15], a discrete multiobjective dynamic programming equation with respect to the ordering cone D is obtained. The solution to this equation is shown to be the minimal element set of the discrete problem. The second step of the approximation method, presented in Section 4.6, consists of a state-space discretization of the above-mentioned discrete multiobjective dynamic programming equation. Using the continuity of the minimal element map, the solution to the resulting approximate dynamic programming equation is shown to converge towards the minimal element set of the discrete problem in the sense of Hausdorff. This result concludes the presentation of the proposed approximation method. The conditions needed for the developments to remain valid for more general classes of multiobjective optimal control problems are discussed in Section 4.7. Finally, further mathematical results extending those obtained in [31] are presented in Section 4.8.

4.1 The Multiobjective Deterministic Finite-Time Horizon Optimal Control Problem

Consider the evolution over a fixed finite-time interval $I = [t_0, t_1]$ ($t_0 < t_1$) of a dynamical system whose n-dimensional state dynamics are given by a continuous function $\mathbf{f}(\cdot,\cdot,\cdot) : I \times \mathbf{R}^n \times U \to \mathbf{R}^n$, where the control space U is a nonempty compact subset of $\mathbf{R}^m$ [15]. The function $\mathbf{f}(t,\cdot,\mathbf{u})$ is assumed to be Lipschitz:

$$\forall \mathbf{u} \in U,\ \forall t \in I,\ \forall (\mathbf{x},\mathbf{y}) \in \mathbf{R}^{n\times n},\ \|\mathbf{f}(t,\mathbf{x},\mathbf{u}) - \mathbf{f}(t,\mathbf{y},\mathbf{u})\| \le K_{\mathbf{f}} \|\mathbf{x}-\mathbf{y}\|, \tag{4.1}$$

where $\| \cdot \|$ denotes the Euclidian norm. A control $\mathbf{u}(\cdot) : [t, t_1] \subset I \to U$ is a bounded, Lebesgue measurable function. The set of such controls $\mathbf{u}(\cdot)$ is denoted by $\mathcal{U}(t)$, which is nonempty for any t. The Lipschitz condition (4.1) guarantees that, given any control $\mathbf{u}(\cdot)$, the system of differential equations governing the dynamical system

$$\dot{\mathbf{x}}(s) = \mathbf{f}(s, \mathbf{x}(s), \mathbf{u}(s)), \ t \le s \le t_1,$$

with initial conditions

$$\mathbf{x}(t) = \mathbf{x_t},$$

has a unique solution $\tilde{\mathbf{x}}(\cdot) : [t, t_1] \to \mathbf{R}^n$ [14, pp. 467–492], called a trajectory of the dynamical system, $\tilde{\mathbf{x}}(s)$ being the state of the system at time s. The cost of each trajectory $\mathbf{x}(\cdot)$ is evaluated by a p-dimensional vector function $\mathbf{J}(\cdot, \cdot, \cdot) : I \times \mathbf{R}^n \times \mathcal{U}(t) \to \mathbf{R}^p$,

$$\mathbf{J}(t, \mathbf{x_t}, \mathbf{u}(\cdot)) = \int_t^{t_1} \mathbf{L}(s, \mathbf{x}(s), \mathbf{u}(s)) \mathrm{d}s, \tag{4.2}$$

where the p-dimensional vector function $\mathbf{L}(\cdot, \cdot, \cdot) : I \times \mathbf{R}^n \times U \to \mathbf{R}^p$, usually called the *running cost* function [15], is assumed to be continuous. The objective space $Y(t, \mathbf{x_t})$ is defined as the set of all possible costs (4.2):

$$Y(t, \mathbf{x_t}) = \{\mathbf{J}(t, \mathbf{x_t}, \mathbf{u}(\cdot)), \mathbf{u}(\cdot) \in \mathcal{U}(t)\}.$$

For simplicity, no terminal cost [15] has been included in (4.2). Moreover, the dynamical system is assumed to be autonomous, $\partial \mathbf{f} / \partial t = 0$, and the running cost function independent of the time, $\partial \mathbf{L} / \partial t = 0$. These simplifications will be discussed later in Section 4.7. Consequently, let $t_0 = 0$, $T = t_1 - t_0$, $I = [0, T]$, $\mathcal{U} = \mathcal{U}(0)$, and $Y(\mathbf{x_0}) = Y(0, \mathbf{x_0})$. Moreover, throughout this chapter, the following additional assumptions on the functions $\mathbf{f}$ and $\mathbf{L}$ are made.

- The function $\mathbf{f}$ is uniformly bounded:

$$\forall \mathbf{x} \in \mathbf{R}^n,\ \forall \mathbf{u} \in U,\ \|\mathbf{f}(\mathbf{x}, \mathbf{u})\| \leq M_{\mathbf{f}}. \tag{4.3}$$

- The function $\mathbf{L}(\cdot, \mathbf{u})$ is Lipschitz:

$$\forall \mathbf{u} \in U,\ \forall (\mathbf{x}, \mathbf{y}) \in \mathbf{R}^n \times \mathbf{R}^n,\ \|\mathbf{L}(\mathbf{x}, \mathbf{u}) - \mathbf{L}(\mathbf{y}, \mathbf{u})\| \leq K_{\mathbf{L}} \|\mathbf{x} - \mathbf{y}\|. \tag{4.4}$$

- The function $\mathbf{L}$ is uniformly bounded:

$$\forall \mathbf{x} \in \mathbf{R}^n,\ \forall \mathbf{u} \in U,\ \|\mathbf{L}(\mathbf{x}, \mathbf{u})\| \leq M_{\mathbf{L}}. \tag{4.5}$$

From the definition of optimality for multiobjective optimization problems provided in Section 2.3.3, the multiobjective deterministic finite-time horizon optimal control problem denoted (P) can now be defined. In this chapter, the ordering cone D is additionally assumed to be closed.

Problem (P): determine the minimal element set $V(\mathbf{x_0})$,

$$V(\mathbf{x_0}) = \mathcal{E}(\mathrm{cl}(Y(\mathbf{x_0})), D), \tag{4.6}$$

and the corresponding optimal controls $\mathbf{u}^*(\cdot)$, if they exist, for which these minimal elements are reached.

Considering the closure of the objective space, $\mathrm{cl}(Y(\mathbf{x_0}))$, instead of the objective space, $Y(\mathbf{x_0})$, in (4.6) guarantees the existence of minimal elements as shown in Proposition 2.1. A special case of interest occurs when $p = 1$ and setting $D = \mathbf{R}_+$ in (4.6). The problem (P) then reduces to the single objective deterministic finite-time horizon optimal control problem. The minimal element set $V(\mathbf{x_0})$ becomes a singleton that can be identified with the so-called *value function* [15, p. 9], defined for nonautonomous problems by

$$V(\mathbf{t}, \mathbf{x_t}) = \inf\{J(t, \mathbf{x_t}, \mathbf{u}(\cdot)),\ \mathbf{u}(\cdot) \in \mathcal{U}(t)\}.$$

4.2 Mathematical Preliminaries

Since the minimal elements of the objective space $Y(\mathbf{x_0})$ form a set, the approximation method proposed in this chapter requires careful attention to the problem of convergence of sequences of sets. In this perspective, the pseudometric space $(\mathcal{M}, \mathcal{H})$ and the metric space $(\mathcal{K}, \mathcal{H})$, where $\mathcal{M} = \{M \subset \mathbf{R}^p,\ M \neq \emptyset,\ M \text{ bounded}\}$, $\mathcal{K} = \{K \subset \mathbf{R}^p,\ K \neq \emptyset,\ K \text{ compact}\}$, and $\mathcal{H}(\cdot, \cdot)$ is the Hausdorff distance, are considered. Section 4.2.1 presents some topological properties of the spaces $(\mathcal{M}, \mathcal{H})$ and $(\mathcal{K}, \mathcal{H})$. In Section 4.2.2, three important results related to minimal elements are then provided. In particular, it is shown that for compact sets, the existence of minimal elements is guaranteed (Proposition 2.1), and the *external stability* or *domination* property holds ([27, p. 53], [26, p. 59]) (Proposition 4.4). Proposition 4.4, together with the more convenient equivalent definition of the convergence of a sequence of sets in $\mathcal{M}$ in terms of the convergence of sequences of elements of these sets provided by Proposition 4.2, allows to state the continuity of the *minimal element map* $E(\cdot) : K \in \mathcal{K} \to \mathcal{E}(K, D) \in \mathcal{M}$ (Proposition 4.5). This key result is used in Section 4.6 to prove the convergence of the state-space approximation. In the following, B is the unit closed ball in $\mathbf{R}^p$.

4.2.1 Topological Properties of $(\mathcal{M}, \mathcal{H})$ and $(\mathcal{K}, \mathcal{H})$ [1]

Let $(M_1, M_2) \in \mathcal{M} \times \mathcal{M}$; the Hausdorff distance [55, p. 365] $\mathcal{H}(\cdot, \cdot)$ between M_1 and M_2 is

$$\mathcal{H}(M_1, M_2) = \max\{\sup_{m_1 \in M_1} d(m_1, M_2), \sup_{m_2 \in M_2} d(m_2, M_1)\},$$

where, for $M \in \mathcal{M}$,

$$d(x, M) = \inf_{m \in M} \|x - m\|.$$

It is easy to check that the Hausdorff distance $\mathcal{H}(\cdot, \cdot)$ defines a pseudometric on $\mathcal{M}$ (since $\mathcal{H}(M_1, M_2) = 0 \Leftrightarrow \text{cl}(M_1) = \text{cl}(M_2)$) and a metric on $\mathcal{K}$. An equivalent definition for the Hausdorff distance $\mathcal{H}(\cdot, \cdot)$ is introduced in Proposition 4.1.

Proposition 4.1.

$$\mathcal{H}(M_1, M_2) = \inf_l \{l \geq 0, \ M_1 \subset M_2 + lB \textit{ and } M_2 \subset M_1 + lB\}.$$

Proof. Let $\mathcal{L} = \{l \geq 0, \ M_1 \subset M_2 + lB \text{ and } M_2 \subset M_1 + lB\}$ and $l^* = \inf \mathcal{L}$; it is proved below that $\mathcal{H}(M_1, M_2) = l^*$.

First, note that l^* is well defined as M_1 and M_2 belong to $\mathcal{M}$. Let $l \in \mathcal{L}$; then by definition, $M_1 \subset M_2 + lB$. Hence, $\forall m_1 \in M_1, \ \exists m_2 \in M_2, \ \|m_1 - m_2\| \leq l$, which implies $\forall m_1, \ d(m_1, M_2) \leq l$ and $\sup\{d(m_1, M_2), \ m_1 \in M_1\} \leq l$. Similarly, $\sup\{d(m_2, M_1), \ m_2 \in M_2\} \leq l$. Hence, $\mathcal{H}(M_1, M_2) \leq l$. Since the inequality holds for any $l \in \mathcal{L}$, it follows that $\mathcal{H}(M_1, M_2) \leq l^*$.

Conversely, $\mathcal{H}(M_1, M_2) \geq \sup\{d(m_2, M_1), \ m_2 \in M_2\} \geq d(m_2, M_1), \ \forall m_2 \in M_2$. Hence, $\forall \epsilon > 0$, $\forall m_2 \in M_2$, $\exists m_1 \in M_1$, $\mathcal{H}(M_1, M_2) > \|m_1 - m_2\| - \epsilon$, and $M_2 \subset M_1 + (\mathcal{H}(M_1, M_2) + \epsilon)B$. By symmetry, $M_1 \subset M_2 + (\mathcal{H}(M_1, M_2) + \epsilon)B$. Hence, $\forall \epsilon > 0, \ \mathcal{H}(M_1, M_2) + \epsilon \in \mathcal{L}$, which implies $l^* \leq \mathcal{H}(M_1, M_2)$.

Combining the two inequalities yields $\mathcal{H}(M_1, M_2) = l^*$. □

Let $(M_n)_{n\in\mathbf{N}}$ be a sequence in $\mathcal{M}$ and $M \in \mathcal{M}$. The sequence (M_n) is said to converge towards M in the sense of Hausdorff if and only if

$$\lim_{n\to\infty} \mathcal{H}(M_n, M) = 0.$$

A more convenient equivalent definition of the convergence of a sequence of sets in terms of the convergence of *samples* of these sets, where a sample is defined as a sequence (m_n) such that $\forall n \in \mathbf{N},\ m_n \in M_n$, is introduced in Proposition 4.2.

Proposition 4.2. *The sequence (M_n) converges towards M in the sense of Hausdorff if and only if the two conditions S_1 and S_2 are satisfied:*

1. Condition S_1: *For all $m \in M$, there exists a sample (m_n) of the sequence (M_n) such that*

$$\lim_{n\to\infty} m_n = m.$$

2. Condition S_2: *For any sample (m_n) of the sequence (M_n), there exists a sequence (x_n) in M such that*

$$\lim_{n\to\infty} (m_n - x_n) = 0.$$

Proof. From Proposition 4.1, we have

$$\lim_{n\to\infty} \mathcal{H}(M_n, M) = 0 \Leftrightarrow \forall \epsilon > 0,\ \exists n_0,\ \forall n \geq n_0,\ M \subset M_n + \epsilon B \text{ and, } M_n \subset M + \epsilon B.$$

This equivalence, together with

1. $\forall n \geq n_0,\ M \subset M_n + \epsilon B \Leftrightarrow \forall m \in M,\ \forall n \geq n_0,\ \exists m_n \in M_n,\ \|m_n - m\| < \epsilon \Leftrightarrow$ condition S_1 holds, and

2. $\forall n \geq n_0,\ M_n \subset M + \epsilon B \Leftrightarrow \forall n \geq n_0,\ \forall m_n \in M_n,\ \exists x_n \in M,\ \|m_n - x_n\| < \epsilon \Leftrightarrow$ condition S_2 holds,

yields the result. □

Corollary 4.1. *If the sequence (M_n) converges towards M, then the following holds:*

1. $\bigcup M_n$ is bounded;

2. Each sample of the sequence (M_n) is bounded;

3. If $M \in \mathcal{K}$, any convergent subsequence of a sample of the sequence (M_n) has its limit in M.

Proof. Part (i) follows from the boundedness of the sets M_n and M and the condition S_2 from Proposition 4.2. Part (ii) is a consequence of Part (i). Part (iii) follows from the closure of the set M and the condition S_2 from Proposition 4.2. □

Proposition 4.3 describes the relation between the set convergence in the sense of Hausdorff and the well-known Kuratowski–Painlevé limit of sets [55, p. 16].

Proposition 4.3. *Let $(K_n)_{n\in\mathbf{N}}$ be a sequence of sets in $\mathcal{K}$ and $K \in \mathcal{K}$; the sequence (K_n) converges towards K in the sense of Hausdorff if and only if $\bigcup K_n$ is bounded and the sequence (K_n) converges towards K in the sense of Kuratowski–Painlevé.*

4.2.2 Existence of Minimal Elements and External Stability for Compact Sets

Two important results related to minimal elements of compact sets are established below.

1. Proposition 4.4 shows that compactness yields the external stability or domination property. This property states that for each element $k \in K$, there exists a minimal element of K that is preferred to k.

2. For a given compact set $K_0 \in \mathcal{K}$, and under the assumption that the minimal elements of the set K_0 with respect to the ordering cone D is equal to the minimal elements of the set K_0 with respect to the ordering cone $\text{int}(D)'$, where $\text{int}(D)' = \text{int}(D) \cup \{0\}$, Proposition 4.5 shows that the minimal element map $E(\cdot) : K \in \mathcal{K} \to \mathcal{E}(K, D) \in \mathcal{M}$ is continuous at K_0. Note that in the definition of the minimal element map, as $\mathcal{E}(K, D)$ is bounded but not necessarily closed, it is only true that $\mathcal{E}(K, D) \in \mathcal{M}$.

Proposition 4.4. *Let $K \in \mathcal{K}$; then for each element $k \in K$, there exists a minimal element of K that is preferred to k, or equivalently, $\mathcal{E}(K, D) \cap (k - D) \neq \emptyset$.*

Proof. The proof is divided into two steps. Let $k \in K$. First, it is proved that $\mathcal{E}(K \cap (k - D), D) \neq \emptyset$, and then that $\mathcal{E}(K \cap (k - D), D) \subset \mathcal{E}(K, D)$. These two facts ensure that $\mathcal{E}(K, D) \cap (k - D) \neq \emptyset$. The first part is a consequence of Proposition 2.1 as $K \cap (k - D) \in \mathcal{K}$ from the assumptions on K and D. For the second part, let $k' \in \mathcal{E}(K \cap (k - D), D)$, then $K \cap (k - D) \cap (k' - D) = \{k'\}$. As $k' \in (k - D)$, $(k' - D) \subset (k - D)$. Hence, $K \cap (k' - D) \subset K \cap (k - D) \cap (k' - D) = \{k'\}$, which proves that k' is a minimal element of K. □

Lemma 4.1. *Let $Y \subset \mathbf{R}^p$, $Y \neq \emptyset$, and then $\text{cl}(\mathcal{E}(Y, D)) \subset \mathcal{E}(Y, \text{int}(D)')$, where $\text{int}(D)' = \text{int}(D) \cup \{0\}$.*

Proof. If $\text{int}(D) = \emptyset$, then the result is obvious. Otherwise, let $y \in \text{cl}(\mathcal{E}(Y, D))$, and then there exists a sequence (y_n) in $\mathcal{E}(Y, D)$ converging towards y. Assume $y \notin \mathcal{E}(Y, \text{int}(D)')$;

then there exists $y' \in Y,\ y' \neq y$ such that $y \in y' + \mathrm{int}(D)'$. We have

$$\lim_{n\to\infty} y_n - y' = y - y' \in \mathrm{int}(D)'.$$

Hence, $y_n - y' \in \mathrm{int}(D)' \subset D$ for large enough n, which contradicts the fact that $y_n \in \mathcal{E}(Y, D)$. □

Corollary 4.2. *Let $K \in \mathcal{K}$, and assume that $\mathcal{E}(K, D) = \mathcal{E}(K, \mathrm{int}(D)')$, then $\mathcal{E}(K, D)$ is compact.*

Proof. It is enough to show that $\mathcal{E}(K, D)$ is closed, which is a consequence of Lemma 4.1. Indeed, $\mathrm{cl}(\mathcal{E}(K, D)) \subset \mathcal{E}(K, \mathrm{int}(D)') \subset \mathcal{E}(K, D)$; hence $\mathrm{cl}(\mathcal{E}(K, D)) = \mathcal{E}(K, D)$. □

Proposition 4.5. *Let $K \in \mathcal{K}$, and assume that $\mathcal{E}(K, D) = \mathcal{E}(K, \mathrm{int}(D)')$; then $E(\cdot)$ is continuous at K.*

Proof. Consider a sequence of sets (K_n) in $\mathcal{K}$ converging towards K in the sense of Hausdorff. Proposition 4.2 is used below to prove that the sequence $(E(K_n)) = (\mathcal{E}(K_n, D))$ converges towards $E(K) = \mathcal{E}(K, D)$ in the sense of Hausdorff. As a result, the proof is divided into two parts.

Part 1 (proof of S_1): Let $k \in \mathcal{E}(K, D)$; it is needed to find a sample of $(\mathcal{E}(K_n, D))$ that converges towards k. Knowing that $k \in K$ and from S_1, there exists a sample (k'_n) of (K_n) such that the sequence (k'_n) converges towards k. From the external stability property (Proposition 4.4), for all k'_n, there exists $k_n \in \mathcal{E}(K_n, D)$ such that $k'_n \in k_n + D$. The sequence (k_n) can be shown to converge towards k. From Corollary 4.1, the sequence (k_n) is bounded. Therefore, it is only needed only to show that any of its convergent subsequences converges towards k. Let $(k_{\psi(n)})$ be such a convergent subsequence

$$\lim_{n\to\infty} k_{\psi(n)} = a.$$

Since D is closed, $k - a \in D$. By assumption, $k \in \mathcal{E}(K, D)$ and from Corollary 4.1, $a \in K$, which implies $a = k$.

Part 2 (proof of S_2): Let (k_n) be a sample of $(\mathcal{E}(K_n, D))$, hence of (K_n). From S_2, there exists a sequence (x_n) in K such that

$$\lim_{n\to\infty} (k_n - x_n) = 0.$$

From the external stability property (Proposition 4.4), for all x_n, there exists $y_n \in \mathcal{E}(K, D)$ such that $x_n \in y_n + D$. The sequence $(k_n - y_n)$ can be shown to converge towards zero. From the boundedness of the sequence (k_n) (Corollary 4.1) and knowing that the sequence (y_n) is in K, the sequence $(k_n - y_n)$ is bounded. Therefore, it is only needed to show that any of its convergent subsequences converges towards zero. It is possible to find convergent subsequences $(k_{\psi(n)} - y_{\psi(n)})$, $(k_{\psi(n)})$, and $(y_{\psi(n)})$ such that

$$\lim_{n\to\infty} k_{\psi(n)} - y_{\psi(n)} = a, \ \lim_{n\to\infty} k_{\psi(n)} = k, \text{ and } \lim_{n\to\infty} y_{\psi(n)} = y.$$

Hence, $a = k - y$. It is additionally true that

$$\lim_{n\to\infty} x_{\psi(n)} = k.$$

D being closed, it follows that $a \in D$. Now, it is claimed that $k \in \mathcal{E}(K, D)$. Otherwise, from the assumption of the proposition, $k \notin \mathcal{E}(K, D)$ implies $k \notin \mathcal{E}(K, \mathrm{int}(D)')$. Hence, there exists $v \in K, \ v \neq k$ such that $k \in v + \mathrm{int}(D)'$. Applying S_1 to v, there exists a sample (v_n) of (K_n) that converges towards v, and

$$\lim_{n\to\infty} k_{\psi(n)} - v_{\psi(n)} = k - v \in \mathrm{int}(D)'.$$

Hence, $k_{\psi(n)} - v_{\psi(n)} \in \operatorname{int}(D)' \subset D$ for large enough n, which contradicts the fact that $k_{\psi(n)} \in \mathcal{E}(K_{\psi(n)}, D)$. Finally, from Corollary 4.2, $\mathcal{E}(K, D)$ is compact; hence $y \in \mathcal{E}(K, D)$. To summarize, it has been shown that $a = k - y$ with $a \in D$ and both k and y in $\mathcal{E}(K, D)$, which implies $a = 0$. □

Note that the assumption $\mathcal{E}(K, D) = \mathcal{E}(K, \operatorname{int}(D)')$ in Proposition 4.5 is only used in the proof of S_2.

4.3 A First-Order Discretization in Time

In this section, a first-order discretization in time with a fixed step h of the problem (P) is performed. This discretization yields a discrete multiobjective optimal control problem denoted by (P$_h$). It is shown, in Section 4.4, how to generate convergent samples of the sequence of sets $(\mathcal{E}(\operatorname{cl}(Y_h(\mathbf{x_0})), D))$ as h converges towards zero, where the set $Y_h(\mathbf{x_0})$ is defined as the objective space for the problem (P$_h$).

Consider a division of I into N intervals of equal length $h = T/N$ and the instants $(t_i)_{i=0,\ldots,N}$, where $t_i = ih$. The discrete multiobjective optimal control problem (P$_h$) is built by considering that the controls $\mathbf{u}(\cdot)$, the dynamics $\mathbf{f}(\cdot, \cdot)$, and the running cost $\mathbf{L}(\cdot, \cdot)$ remain constant in any time interval $[t_i, t_{i+1})$. Hence, the discrete control $(\mathbf{u_i})_{i=0,\ldots,N}$ for the problem (P$_h$) is defined by

$$\mathbf{u_i} = \mathbf{u}(t_i),\ \mathbf{u}(\cdot) \in \mathcal{U}.$$

The discrete trajectory $(\mathbf{x_i})_{i=1,\ldots,N}$ is obtained by the recursion

$$\mathbf{x_{i+1}} = \mathbf{x_i} + h\mathbf{f}(\mathbf{x_i}, \mathbf{u_i}) \tag{4.7}$$

with initial conditions $\mathbf{x_0}$. And finally, the discrete cost $\mathbf{J_h}(\mathbf{x_0}, \mathbf{u}(\cdot))$ is given by the series

$$\mathbf{J_h}(\mathbf{x_0}, \mathbf{u}(\cdot)) = h \sum_{i=0}^{N-1} \mathbf{L}(\mathbf{x_i}, \mathbf{u_i}). \tag{4.8}$$

Therefore, the *discrete objective space* $Y_h(\mathbf{x_0})$ is defined by

$$Y_h(\mathbf{x_0}) = \{\mathbf{J_h}(\mathbf{x_0}, \mathbf{u}(\cdot)), \mathbf{u}(\cdot) \in \mathcal{U}\},$$

and the set of minimal elements of the discrete objective space, $V_h(\mathbf{x_0})$, hereinafter referred to as the *discrete minimal element set*, is

$$V_h(\mathbf{x_0}) = \mathcal{E}(\mathrm{cl}(Y_h(\mathbf{x_0})), D).$$

Note that the final value of the trajectory $\mathbf{x_N}$ and the final control $\mathbf{u_N}$ do not play any role in the proposed discretization as they do not appear in (4.8).

For the error estimates that will follow in Section 4.4.1, it is convenient to consider the piecewise constant extension $\mathbf{u_h}(\cdot)$ to I of the discrete control:

$$\forall t \in I,\ \mathbf{u_h}(t) = \mathbf{u_i},\ i = [\frac{t}{h}], \tag{4.9}$$

and similarly, the piecewise constant extension $\mathbf{x_h}(\cdot)$ to I of the discrete trajectory:

$$\forall t \in I,\ \mathbf{x_h}(t) = \mathbf{x_i},\ i = [\frac{t}{h}].$$

The piecewise constant extensions $\mathbf{u_h}(\cdot)$ and $\mathbf{x_h}(\cdot)$ are also referred to as discrete control and discrete trajectory. If $\mathcal{U}_h \subset \mathcal{U}$ denotes the set of discrete controls (4.9), the discrete objective space $Y_h(\mathbf{x_0})$ is equivalently defined by

$$Y_h(\mathbf{x_0}) = \{\mathbf{J_h}(\mathbf{x_0}, \mathbf{u_h}(\cdot)), \mathbf{u_h}(\cdot) \in \mathcal{U}_h\}.$$

Evidently, the definition of the discrete minimal element set $V_h(\mathbf{x_0})$ remains the same.

The existence of minimal elements (Proposition 2.1) and the external stability property (Proposition 4.4) for the discrete objective space $Y_h(\mathbf{x_0})$ are needed in Section 4.4 to build convergent samples of the sequence of discrete minimal element sets $V_h(\mathbf{x_0})$ as the time step h converges towards zero. For this purpose, it is stated in Proposition 4.6 the compactness of the discrete objective space $Y_h(\mathbf{x_0})$, from which follows that $V_h(\mathbf{x_0}) = \mathcal{E}(Y_h(\mathbf{x_0}), D)$.

Proposition 4.6. *The discrete objective space $Y_h(\mathbf{x_0})$ is a compact set.*

Proof. Let $\mathbf{g}(\cdot) : \mathbf{R}^N \to \mathbf{R}^N$ be the function that associates a discrete control to the discrete trajectory and $\mathbf{h}(\cdot,\cdot) : \mathbf{R}^N \times \mathbf{R}^N \to \mathbf{R}^p$ be the function that associates a discrete control and the discrete trajectory to the cost. Both these functions are continuous as $\mathbf{f}(\cdot,\cdot)$ and $\mathbf{L}(\cdot,\cdot)$ are continuous by assumption. The discrete objective space $Y_h(\mathbf{x_0})$ can be viewed as the image of the compact set U^N by the continuous function $\mathbf{h}(\mathbf{g}(\cdot),\cdot) : \mathbf{R}^N \to \mathbf{R}^p$. Hence, it is itself compact. □

4.4 A Direct Convergence Proof

In this section, a recursive procedure which generates convergent samples of the sequence of discrete minimal element sets $V_h(\mathbf{x_0})$ as the time step h converges towards zero is proposed. It is worth mentioning that under certain assumptions, the discrete objective space $Y_h(\mathbf{x_0})$ can be shown to converge towards the objective space $Y(\mathbf{x_0})$ in the sense of Hausdorff. It then follows from the continuity of the minimal element map (Proposition 4.5) that the discrete minimal element set $V_h(\mathbf{x_0})$ converges towards the minimal element set $V(\mathbf{x_0})$. This guarantees that the samples generated by the recursive procedure converge in the minimal element set $V(\mathbf{x_0})$. The key idea behind this procedure is to use the sequence (h_r), $h_r = T/2^r$, $r \in \mathbf{N}$, for the time step [56]. Using this sequence, it is possible to obtain an error estimate between the minimal elements of the discrete objective space $Y_{2h}(\mathbf{x_0})$ and elements of the discrete objective space $Y_h(\mathbf{x_0})$ as any discrete control $\mathbf{u_{2h}}(\cdot)$ in $\mathcal{U}_{2h}$ can always be viewed as a discrete control $\mathbf{u_h}(\cdot) \in \mathcal{U}_h$ satisfying $\mathbf{u_h}(t_{2i}) = \mathbf{u_h}(t_{2i+1})$, $i = 0, \ldots, N-1$. A minimal element of the discrete objective space $Y_h(\mathbf{x_0})$ can finally be obtained using the external stability property (Proposition 4.4). The error estimate between the elements of the discrete objective space $Y_{2h}(\mathbf{x_0})$ and elements of the discrete objective space $Y_h(\mathbf{x_0})$ is derived in Section 4.4.1, while Section 4.4.2 contains the proposed procedure and the proof of convergence for the samples generated by the procedure.

4.4.1 Error Estimation

Let $\mathbf{u_{2h}}(\cdot)$ be a discrete control in $\mathcal{U}_{2h}$ and choose the discrete control $\mathbf{u_h}(\cdot)$ in $\mathcal{U}_h$ such that $\mathbf{u_{2h}}(t) = \mathbf{u_h}(t)$, $\forall t \in I$. An intermediate step in the derivation of an error estimate between the discrete costs $\mathbf{J_h}(\mathbf{x_0}, \mathbf{u_h}(\cdot))$ and $\mathbf{J_{2h}}(\mathbf{x_0}, \mathbf{u_{2h}}(\cdot))$ is the derivation of an error estimate between the discrete trajectories $\mathbf{x_h}(\cdot)$ and $\mathbf{x_{2h}}(\cdot)$. This step will be achieved in Proposition 4.7 using the Gronwall–Bellman inequality. The derivation of the error estimate between the discrete costs $\mathbf{J_h}(\mathbf{x_0}, \mathbf{u_h}(\cdot))$ and $\mathbf{J_{2h}}(\mathbf{x_0}, \mathbf{u_{2h}}(\cdot))$ will follow in Proposition 4.8. Note that both the error estimates obtained in Proposition 4.7 and in Proposition 4.8 are of order h and uniform in $\mathbf{x_0}$.

Proposition 4.7 (Error estimate for the discrete trajectories)**.** *Under the assumption* (4.3) *that the function* $\mathbf{f}$ *is uniformly bounded, and for two discrete controls* $\mathbf{u_h}(\cdot) \in \mathcal{U}_h$ *and* $\mathbf{u_{2h}}(\cdot) \in \mathcal{U}_{2h}$ *satisfying* $\mathbf{u_h}(t) = \mathbf{u_{2h}}(t)$, $\forall t \in I$, *the following error estimate between the* *discrete trajectories* $\mathbf{x_h}(\cdot)$ *and* $\mathbf{x_{2h}}(\cdot)$ *at time* $t \in I$ *holds:*

$$\|\mathbf{x}_h(t) - \mathbf{x}_{2h}(t)\| \leq C_1 h \exp^{K_{\mathbf{f}} t},$$

where C_1 *is a constant.*

Proof. Equation (4.7) can be rewritten as

$$\mathbf{x_h}(t) = \int_0^{[\frac{t}{h}]h} \mathbf{f}(\mathbf{x_h}(s), \mathbf{u_h}(s))\mathrm{d}s + \mathbf{x}_0.$$

Similarly,

$$\mathbf{x_{2h}}(t) = \int_0^{[\frac{t}{2h}]2h} \mathbf{f}(\mathbf{x_{2h}}(s), \mathbf{u_{2h}}(s))\mathrm{d}s + \mathbf{x}_0.$$

Now, we have

$$\|\mathbf{x}_h(t) - \mathbf{x}_{2h}(t)\| \leq I_1 + I_2 + I_3,$$

where

$$\begin{aligned}
I_1 &= \int_0^t \|\mathbf{f}(\mathbf{x_h}(s), \mathbf{u_h}(s)) - \mathbf{f}(\mathbf{x_{2h}}(s), \mathbf{u_{2h}}(s))\|\mathrm{d}s, \\
I_2 &= \int_t^{[\frac{t}{h}]h} \|\mathbf{f}(\mathbf{x_h}(s), \mathbf{u_h}(s))\|\mathrm{d}s, \\
I_3 &= \int_t^{[\frac{t}{2h}]2h} \|\mathbf{f}(\mathbf{x_{2h}}(s), \mathbf{u_{2h}}(s))\|\mathrm{d}s.
\end{aligned}$$

The uniform boundedness assumption on $\mathbf{f}$ leads directly to

$$\begin{aligned}
I_2 &\leq hM_{\mathbf{f}}, \\
I_3 &\leq 2hM_{\mathbf{f}}.
\end{aligned}$$

Knowing that $\mathbf{u_{2h}}(s) = \mathbf{u_h}(s),\ \forall s \in I$, and using the Lipschitz condition on $\mathbf{f}$, we obtain

$$I_1 \le K_{\mathbf{f}} \int_0^t \|\mathbf{x_h}(s) - \mathbf{x_{2h}}(s)\| \mathrm{d}s.$$

Finally,

$$\|\mathbf{x}_h(t) - \mathbf{x}_{2h}(t)\| \le 3hM_{\mathbf{f}} + K_{\mathbf{f}} \int_0^t \|\mathbf{x}_h(s) - \mathbf{x}_{2h}(s)\| \mathrm{d}s.$$

Applying the Gronwall–Bellman inequality [57] yields

$$\|\mathbf{x}_h(t) - \mathbf{x}_{2h}(t)\| \le 3M_{\mathbf{f}} h \exp^{K_{\mathbf{f}} t} = C_1 h \exp^{K_{\mathbf{f}} t},$$

where $C_1 = 3M_{\mathbf{f}}$. □

Proposition 4.8 (Error estimate for the discrete costs)**.** *Under the same assumptions as in Proposition* 4.7 *and the assumption* (4.4) *that the function* $\mathbf{L}(\cdot, \mathbf{u})$ *is Lipschitz, the following error estimate between the discrete costs* $\mathbf{J_h}(\mathbf{x_0}, \mathbf{u_h}(\cdot))$ *and* $\mathbf{J_{2h}}(\mathbf{x_0}, \mathbf{u_{2h}}(\cdot))$ *holds:*

$$\|\mathbf{J_h}(\mathbf{x_0}, \mathbf{u_h}(\cdot)) - \mathbf{J_{2h}}(\mathbf{x_0}, \mathbf{u_{2h}}(\cdot))\| \le C_2 h,$$

where C_2 *is a constant.*

Proof. Equation (4.8) can be rewritten as

$$\mathbf{J_h}(\mathbf{x_0}, \mathbf{u_h}(\cdot)) = \int_0^T \mathbf{L}(\mathbf{x_h}(s), \mathbf{u_h}(s)) \mathrm{d}s.$$

Knowing that $\mathbf{u_{2h}}(s) = \mathbf{u_h}(s),\ \forall s \in I$, and using the Lipschitz condition on $\mathbf{L}$, we obtain

$$\|\mathbf{J_h}(\mathbf{x_0}, \mathbf{u_h}(\cdot)) - \mathbf{J_{2h}}(\mathbf{x_0}, \mathbf{u_{2h}}(\cdot))\| \le K_{\mathbf{L}} \int_0^T \|\mathbf{x_h}(s) - \mathbf{x_{2h}}(s)\| \mathrm{d}s.$$

Substituting the error estimate for the discrete trajectories obtained in Proposition 4.7

and integrating yields

$$\|\mathbf{J_h}(\mathbf{x_0}, \mathbf{u_h}(\cdot)) - \mathbf{J_{2h}}(\mathbf{x_0}, \mathbf{u_{2h}}(\cdot))\| \leq \frac{3M_{\mathbf{f}}K_{\mathbf{L}}}{K_{\mathbf{f}}}(\exp(K_{\mathbf{f}}T) - 1)h = C_2 h,$$

where $C_2 = \frac{3M_{\mathbf{f}}K_{\mathbf{L}}}{K_{\mathbf{f}}}(\exp(K_{\mathbf{f}}T) - 1)$. $\square$

4.4.2 Generating Convergent Samples of the Sequence $(V_h(\mathbf{x_0}))$

The proposed procedure to generate samples of the sequence $V_h(\mathbf{x_0})$, detailed below as Procedure 1, is a recursive procedure consisting of two main steps. Starting from an element $\mathbf{J_{2h}}(\mathbf{x_0}, \mathbf{u_h}(\cdot))$ of the minimal element set $V_{2h}(\mathbf{x_0})$, the discrete cost $\mathbf{J_h}(\mathbf{x_0}, \mathbf{u_h}(\cdot))$ corresponding to the discrete control $\mathbf{u_h}(\cdot)$ satisfying $\mathbf{u_h}(t) = \mathbf{u_{2h}}(t), \forall t \in I$, is calculated. Then, the application of the external stability property (Proposition 4.4) to the discrete objective set $Y_h(\mathbf{x_0})$ yields an element in the minimal element set $V_h(\mathbf{x_0})$ preferred to the discrete cost $\mathbf{J_h}(\mathbf{x_0}, \mathbf{u_h}(\cdot))$.

Procedure 1: let $h \to 0$ through the sequence (h_r) with $h_r = T/2^r$, $r \in N$. Let the set Z_r be defined by $Z_r = V_{h_r}(\mathbf{x_0})$ ($\mathbf{x_0}$ is dropped for brevity), and let $(\mathbf{z_r})$ be any sample of the sequence (Z_r) built recursively as follows.

Step 1. Let $r = 0$. From Propositions 2.1 and 4.6, the set Z_0 is nonempty; hence let $\mathbf{z_0} \in Z_0$. Note that it is possible to initiate Procedure 1 at any $r = r_0 > 0$.

Step 2. Proposition 4.8 yields an element $\mathbf{y_{r+1}}$ in the discrete objective space $Y_{h_{r+1}}(\mathbf{x_0})$ such that

$$\|\mathbf{z_r} - \mathbf{y_{r+1}}\| \leq C_2 h_{r+1}.$$

Step 3. From the external stability property (Proposition 4.4) and Proposition 4.6, there exists $\mathbf{z_{r+1}} \in Z_{r+1}$ such that $\mathbf{y_{r+1}} \in \mathbf{z_{r+1}} + D$.

Step 4. Repeat Step 2.

The convergence of the sequence $(\mathbf{z_r})$ built from Procedure 1 will be proved in Proposition 4.10. The main idea behind this proof is to show that every element of the sequence $(\mathbf{z_r})$ is in an appropriate neighborhood of elements of any converging subsequence, from which the convergence of the whole sequence can be concluded. This proof requires the following preliminaries.

Lemma 4.2 (Lemma 3.2.3, [26, p. 52])**.** *Let $K \in \mathcal{K}$, and $Y \subset \mathbf{R}^p$ be a closed set; then the set $K + Y$ is closed.*

Define $\widehat{D} = \{d \in D,\ \|d\| = 1\}$ and $\widetilde{D}(d, D) = D \cap (d - D),\ \forall d \in D$. Let also $\alpha(D) = \inf\{\|x - y\|,\ x \in \widehat{D},\ y \in -D\}$.

Lemma 4.3. *$\alpha(D) > 0$.*

Proof. Assume $\alpha(D) = 0$; then there exists a sequence (x_n) in $\widehat{D}$ and a sequence (y_n) in $-D$ such that

$$\lim_{n\to\infty} \|x_n - y_n\| = 0.$$

$\widehat{D}$ being compact, there exists a convergent subsequence $(x_{\psi(n)})$ such that

$$\lim_{n\to\infty} x_{\psi(n)} = x \in \widehat{D}.$$

It follows that the sequence $(y_{\psi(n)})$ is bounded. Hence, there exists a convergent subsequence $(y_{\phi\circ\psi(n)})$ such that

$$\lim_{n\to\infty} y_{\phi\circ\psi(n)} = y \in -D.$$

Finally, $x + (-y) = 0$ with both x and $-y$ in D. Knowing that D is pointed implies that $x = y = 0$, which is a contradiction. □

Lemma 4.4. *$\forall y \in \widetilde{D}(d, D),\ \|y\| \le \|d\|/\alpha(D)$.*

Proof. If $y = 0$, then the result is obvious. Let $y \in \widetilde{D}(d, D),\ y \neq 0$, and $y \in d - D$; hence there exists $x \in D$ such that $y + x = d$. Divide this last expression by $\|y\|$, and rewrite it as

$$\frac{y}{\|y\|} - \frac{-x}{\|y\|} = \frac{d}{\|y\|},$$

$y/\|y\| \in \widehat{D}$, and $-x/\|y\| \in -D$. The proof is completed by taking the norm on both sides and using the definition of $\alpha(D)$. □

Proposition 4.9. *Consider* $(K_1, K_2) \in \mathcal{K} \times \mathcal{K}$ *such that* $(K_1 + D) \cap K_2 \neq \emptyset$ *and define* $K = (K_1 + D) \cap (K_2 - D)$. *Then,* $K \in \mathcal{K}$ *and* $\forall (k, \widetilde{k}) \in K \times K$,

$$\|k - \widetilde{k}\| \leq \frac{2}{\alpha(D)} \sup\{\|k_1 - k_2\|,\ (k_1, k_2) \in K_1 \times K_2\} + diam(K_1),$$

where $diam(K_1)$ *denotes the diameter of the set* K_1.

Proof. Let $k \in K$; then there exists $k_1 \in K_1,\ k_2 \in K_2,\ (d_1, d_2) \in D \times D$ such that $k = k_1 + d_1$ and $k = k_2 - d_2$. Hence, $k_2 - k_1 = d_1 + d_2$, and therefore $k_2 - k_1 \in D$. Now, it can be written $k = k_1 + (k_2 - k_1) - d_2$. Hence, $k - k_1 \in (k_2 - k_1) - D$. Knowing that $k - k_1 \in D$ yields $k - k_1 \in \widetilde{D}(k_2 - k_1, D)$. Similarly, let $\tilde{k} \in K$; then there exists $\widetilde{k_1} \in K_1$ and $\widetilde{k_2} \in K_2$ such that $\widetilde{k} - \widetilde{k_1} \in \widetilde{D}(\widetilde{k_2} - \widetilde{k_1}, D)$. We have

$$\|k - \widetilde{k}\| \leq \|k - k_1\| + \|k_1 - \widetilde{k_1}\| + \|\widetilde{k_1} - \widetilde{k}\|.$$

From Lemma 4.4, $\|k - k_1\| \leq \|k_2 - k_1\|/\alpha(D)$ and $\|\widetilde{k_1} - \widetilde{k}\| \leq \|\widetilde{k_2} - \widetilde{k_1}\|/\alpha(D)$. Hence,

$$\|k - k_1\| + \|k_1 - \widetilde{k_1}\| + \|\widetilde{k_1} - \widetilde{k}\| \leq \frac{1}{\alpha(D)}\|k_2 - k_1\| + \|k_1 - \widetilde{k_1}\| + \frac{1}{\alpha(D)}\|\widetilde{k_2} - \widetilde{k_1}\|.$$

Finally,

$$\|k-\widetilde{k}\| \leq \frac{2}{\alpha(D)} \sup\{\|k_1-k_2\|,\ (k_1,k_2) \in K_1 \times K_2\} + \operatorname{diam}(K_1),$$

which shows that K is bounded. The closure is a consequence of Lemma 4.2. Hence, K is compact. □

Proposition 4.10. *Under the assumption* (4.5) *that the function* $\mathbf{L}$ *is uniformly bounded, the sequence* $(\mathbf{z_r})$ *converges.*

Proof. The uniform boundedness assumption on $\mathbf{L}$ implies that

$$\forall \mathbf{x_0} \in \mathbf{R}^n,\ \forall \mathbf{u}(\cdot) \in \mathcal{U}_h,\ \|\mathbf{J_h}(\mathbf{x_0}, \mathbf{u}(\cdot))\| \leq TM_{\mathbf{L}},$$

which shows that the sets $Y_h(\mathbf{x_0})$, and consequently Z_r, are uniformly bounded. The sequence $(\mathbf{z_r})$ being bounded has at least one accumulation point $\mathbf{z}$. From the definition of $\mathbf{z}$,

$$\forall \epsilon > 0,\ \forall r_0,\ \exists r > r_0,\ \|\mathbf{z_r} - \mathbf{z}\| < \epsilon.$$

Repeatedly applying this definition with $\epsilon_p = 1/p,\ p = 1, \ldots, \infty$ yields an increasing sequence (r_p) such that $\|\mathbf{z_{r_p}} - \mathbf{z}\| < 1/p$. Setting $\psi(p) = r_p$, a subsequence $(\mathbf{z_{\psi(p)}})$ which converges towards $\mathbf{z}$ and satisfies $\|\mathbf{z_{\psi(p)}} - \mathbf{z}\| < 1/p,\ \forall p \geq 1$ is obtained. The key point is to observe that necessarily, from the construction of the sequence $(\mathbf{z_r})$,

$$\mathbf{z_r} \in K,\ \forall r \in [\psi(p), \psi(p+1)],$$

where $B(\mathbf{x}, r)$ denotes the closed ball centered at $\mathbf{x}$ with radius r,

$$K = (B(\mathbf{z_{\psi(p+1)}}, l) + D) \bigcap (B(\mathbf{z_{\psi(p)}}, l) - D),$$

and
$$l = C_2 \sum_{r=\psi(p)+1}^{\psi(p+1)} h_r \leq C_2 \frac{T}{2^{\psi(p)}}.$$

Applying Proposition 4.9 with $K_1 = B(\mathbf{z}_{\psi(\mathbf{p+1})}, l)$ and $K_2 = B(\mathbf{z}_{\psi(\mathbf{p})}, l)$ leads to

$$\|\mathbf{z_r} - \mathbf{z}_{\psi(\mathbf{p})}\| \leq \frac{2}{\alpha(D)}\left(2l + \|\mathbf{z}_{\psi(\mathbf{p+1})} - \mathbf{z}_{\psi(\mathbf{p})}\|\right) + l.$$

By applying the triangular inequality,

$$\|\mathbf{z}_{\psi(\mathbf{p})} - \mathbf{z}_{\psi(\mathbf{p+1})}\| < \|\mathbf{z}_{\psi(\mathbf{p})} - \mathbf{z}\| + \|\mathbf{z} - \mathbf{z}_{\psi(\mathbf{p+1})}\| < \frac{1}{p} + \frac{1}{p+1},$$

which implies that $\|\mathbf{z_r} - \mathbf{z}_{\psi(\mathbf{p})}\| < \beta_p$, where the sequence (β_p) converges towards zero.

Let now $r \in \mathbf{N}$; then there exists a unique p such that $r \in [\psi(p), \psi(p+1))$. By applying the triangular inequality,

$$\|\mathbf{z_r} - \mathbf{z}\| \leq \|\mathbf{z_r} - \mathbf{z}_{\psi(\mathbf{p})}\| + \|\mathbf{z}_{\psi(\mathbf{p})} - \mathbf{z}\| \leq \beta_p + \frac{1}{p},$$

which shows that the sequence $(\mathbf{z_r})$ converges towards $\mathbf{z}$. □

4.5 A Discrete Dynamic Programming Formulation

After having performed the time discretization, the problem is to determine the discrete minimal element set $V_h(\mathbf{x_0})$. This is done in two steps. First, it is proved in Corollary 4.3, using the dynamic programming principle [15], that the discrete minimal element set $V_h(\mathbf{x_0})$ is also the solution to a discrete multiobjective dynamic programming equation with respect to the ordering cone D denoted by (HJ$_h$). Second, in Section 4.6, a discretization in the state space of the discrete multiobjective dynamic programming equation (HJ$_h$)

is performed, yielding an approximate multiobjective dynamic programming equation denoted by (HJ_h^d). The solution to this approximate equation is shown in Corollary 4.4 to converge towards the discrete minimal element set $V_h(\mathbf{x_0})$ in the sense of Hausdorff.

The statement of the discrete multiobjective dynamic programming equation (HJ_h) in Proposition 4.11 requires the introduction of a few additional notations, which are introduced below. For $0 \leq k \leq m \leq N-1$, the discrete objective space $Y_h^{k,m}(\mathbf{x_k})$ is defined by

$$Y_h^{k,m}(\mathbf{x_k}) = \{\mathbf{J}_h^{k,m}(\mathbf{x_k}, \mathbf{u}(\cdot)) = h \sum_{i=k}^{m} \mathbf{L}(\mathbf{x_i}, \mathbf{u_i}), \mathbf{u}(\cdot) \in \mathcal{U}_h\},$$

and the corresponding discrete minimal element set by

$$V_h^{k,m}(\mathbf{x_k}) = \mathcal{E}(\mathrm{cl}(Y_h^{k,m}(\mathbf{x_k})), D).$$

With this notation, it can be seen that the discrete objective space $Y_h(\mathbf{x_0})$ is the same as the set $Y_h^{0,N-1}(\mathbf{x_0})$ and the discrete minimal element set $V_h(\mathbf{x_0})$ is the same as the set $V_h^{0,N-1}(\mathbf{x_0})$. The set $\widetilde{Y}_h^{k,m}(\mathbf{x_k})$ that will also be called discrete objective space is defined by

$$\widetilde{Y}_h^{k,m}(\mathbf{x_k}) = \{\mathbf{J}_h^{k,m}(\mathbf{x_k}, \mathbf{u}(\cdot)) + V_h^{m+1,N-1}(\mathbf{x_{m+1}}), \mathbf{u}(\cdot) \in \mathcal{U}_h\}, \tag{4.10}$$

and the corresponding discrete minimal element set by

$$\widetilde{V}_h^{k,m}(\mathbf{x_k}) = \mathcal{E}(\mathrm{cl}(\widetilde{Y}_h^{k,m}(\mathbf{x_k})), D).$$

In the definition of the discrete objective space $\widetilde{Y}_h^{k,m}(\mathbf{x_k})$, the particular case $m = N-1$ yields the discrete minimal element set $V_h^{N,N-1}(\mathbf{x_N})$, which is usually referred to as the terminal data condition [15], and is set to

$$V_h^{N,N-1}(\mathbf{x_N}) = \{0\}. \tag{4.11}$$

The definition of the discrete objective space $\widetilde{Y}_h^{k,m}(\mathbf{x_k})$ can be justified by noting that the discrete objective space $Y_h^{m+1,N-1}(\mathbf{x_{m+1}})$ is bounded from the uniform boundedness of the running cost function $\mathbf{L}(\cdot,\cdot)$, which guarantees the existence of minimal elements from Proposition 2.1. The existence of minimal elements for the discrete objective space $\widetilde{Y}_h^{k,m}(\mathbf{x_k})$ can be justified using a similar argument.

Proposition 4.11. *For $0 \leq k \leq m \leq N-1$, the discrete multiobjective dynamic programming equation denoted by* (HJ_h) *is*

$$V_h^{k,N-1}(\mathbf{x_k}) = \widetilde{V}_h^{k,m}(\mathbf{x_k}),$$

which yields, together with (4.10),

$$\widetilde{V}_h^{k,m}(\mathbf{x_k}) = \mathcal{E}(\mathrm{cl}(\{\mathbf{J}_h^{k,m}(\mathbf{x_k},\mathbf{u}(\cdot)) + \widetilde{V}_h^{m+1,m'}(\mathbf{x_{m+1}}), \mathbf{u}(\cdot) \in \mathcal{U}_h\}), D),$$

with $m+1 \leq m' \leq N-1$. From (4.11)*, the terminal data condition is*

$$\widetilde{V}_h^{N,N}(\mathbf{x_N}) = \{0\}.$$

Proof. For clarity, the following three lemmas to be used in the proof of Proposition 4.11 are first proved and this proof is postponed to the end of this section. □

Lemma 4.5. $\widetilde{Y}_h^{k,m}(\mathbf{x_k}) \subset \mathrm{cl}(Y_h^{k,N-1}(\mathbf{x_k}))$.

Proof. Let $\widetilde{\mathbf{y}} \in \widetilde{Y}_h^{k,m}(\mathbf{x_k})$ and $\epsilon > 0$; then there exists $\widetilde{\mathbf{u}}(\cdot) \in \mathcal{U}_h$ and $\mathbf{y_{m+1}} \in Y_h^{m+1,N-1}(\mathbf{x_{m+1}})$

such that

$$\|\widetilde{\mathbf{y}} - \mathbf{J}_h^{k,m}(\mathbf{x_k}, \widetilde{\mathbf{u}}(\cdot)) - \mathbf{y_{m+1}}\| \leq \epsilon,$$

with $\mathbf{y_{m+1}} = \mathbf{J}_h^{m+1,N-1}(\mathbf{x_{m+1}}, \widehat{\mathbf{u}}(\cdot))$ for some $\widehat{\mathbf{u}}(\cdot) \in \mathcal{U}_h$. Define the new control $\mathbf{u}(\cdot) \in \mathcal{U}_h$ as

$$\mathbf{u} = \begin{cases} \widetilde{\mathbf{u}}, & j = 1, \ldots, m, \\ \widehat{\mathbf{u}}, & j = m+1, \ldots, N-1. \end{cases}$$

Then, $\mathbf{J}_h^{k,m}(\mathbf{x_k}, \widetilde{\mathbf{u}}(\cdot)) + \mathbf{y_{m+1}} = \mathbf{J}_h^{k,N-1}(\mathbf{x_k}, \mathbf{u}(\cdot)) = \mathbf{y} \in Y_h^{k,N-1}(\mathbf{x_k})$, and $\mathbf{y}$ verifies

$$\|\widetilde{\mathbf{y}} - \mathbf{y}\| \leq \epsilon.$$

Hence, $\widetilde{\mathbf{y}} \in \mathrm{cl}(Y_h^{k,N-1}(\mathbf{x_k}))$. □

Lemma 4.6. $\mathrm{cl}(Y_h^{k,N-1}(\mathbf{x_k})) \subset \mathrm{cl}(\widetilde{Y}_h^{k,m}(\mathbf{x_k}) + D)$.

Proof. Let $\overline{\mathbf{y}} \in \mathrm{cl}(Y_h^{k,N-1}(\mathbf{x_k}))$ and $\epsilon > 0$; then there exists $\mathbf{y} \in Y_h^{k,N-1}(\mathbf{x_k})$ such that $\|\overline{\mathbf{y}} - \mathbf{y}\| \leq \epsilon$. Writing $\mathbf{y} = \mathbf{J}_h^{k,m}(\mathbf{x_k}, \mathbf{u}(\cdot)) + \mathbf{y_{m+1}}$ with $\mathbf{y_{m+1}} \in Y_h^{m+1,N-1}(\mathbf{x_{m+1}})$ yields

$$\|\overline{\mathbf{y}} - \mathbf{J}_h^{k,m}(\mathbf{x}_k, \mathbf{u}(\cdot)) - \mathbf{y_{m+1}}\| \leq \epsilon.$$

We also have $\mathbf{y_{m+1}} \in \mathrm{cl}(Y_h^{m+1,N-1}(\mathbf{x_{m+1}}))$. From the external stability property, there exists $\mathbf{y_{m+1}}^* \in V_h^{m+1,N-1}(\mathbf{x_{m+1}})$ such that $\mathbf{y_{m+1}} \in \mathbf{y_{m+1}}^* + D$. Therefore,

$$\|\overline{\mathbf{y}} - \mathbf{J}_h^{k,m}(\mathbf{x_k}, \mathbf{u}(\cdot)) - \mathbf{y_{m+1}}^* - d\| \leq \epsilon,$$

which leads to $\mathbf{J}_h^{k,m}(\mathbf{x_k}, \mathbf{u}(\cdot)) + \mathbf{y_{m+1}}^* \in \widetilde{Y}_h^{k,m}(\mathbf{x_k})$. Hence, $\overline{\mathbf{y}} \in \mathrm{cl}(\widetilde{Y}_h^{k,m}(\mathbf{x_k}) + D)$. □

Lemma 4.7. $\mathrm{cl}(\widetilde{Y}_h^{k,m}(\mathbf{x_k}) + D) \subset \mathrm{cl}(\widetilde{Y}_h^{k,m}(\mathbf{x_k})) + D$.

Proof. This is a consequence of the facts that the discrete objective space $\widetilde{Y}_h^{k,m}(\mathbf{x_k})$ is bounded and D is closed. □

Lemma 4.8. *Let $K_1 \subset \mathcal{K}$, $K_2 \subset \mathcal{K}$ satisfying $K_1 \subset K_2$ and $K_2 \subset K_1 + D$; then $\mathcal{E}(K_1, D) = \mathcal{E}(K_2, D)$.*

Proof. From Proposition 2.1, $\mathcal{E}(K_1, D) \neq \emptyset$. Let $k_1 \in \mathcal{E}(K_1, D)$, and hence $k_1 \in K_1 \subset K_2$. Assume that $k_1 \notin \mathcal{E}(K_2, D)$, and hence there exists $k_2 \in K_2$, $d \in D$, $d \neq 0$ such that $k_1 = k_2 + d$. By assumption, $k_2 \in K_1 + D$; hence there exists $k_1' \in K_1$, $d' \in D$ such that $k_2 = k_1' + d'$, which yields $k_1 = k_1' + d + d'$. D being convex, $d + d' \in D$ with $d + d' \neq 0$, which contradicts the fact that $k_1 \in \mathcal{E}(K_1, D)$. Therefore, $\mathcal{E}(K_1, D) \subset \mathcal{E}(K_2, D)$.

From Proposition 2.1, $\mathcal{E}(K_2, D) \neq \emptyset$. Let $k_2 \in \mathcal{E}(K_2, D)$; then $k_2 \in K_2$. By assumption, $k_2 \in K_1 + D$; hence there exists $k_1 \in K_1$, $d \in D$ such that $k_2 = k_1 + d$. By assumption, $k_1 \in K_2$. Hence, necessarily, as $k_2 \in \mathcal{E}(K_2, D)$, we have $d = 0$, $k_2 = k_1$, and $k_2 \in K_1$. From the assumption $K_1 \subset K_2$ follows that $k_2 \in \mathcal{E}(K_1, D)$. Therefore, $\mathcal{E}(K_2, D) \subset \mathcal{E}(K_1, D)$.

Combining the two inclusions yields $\mathcal{E}(K_1, D) = \mathcal{E}(K_2, D)$. □

Proposition 4.11 can now be proved.

Proof. Apply Lemma 4.8 with $K_1 = \mathrm{cl}(\widetilde{Y}_h^{k,m}(\mathbf{x_k}))$ and $K_2 = \mathrm{cl}(Y_h^{k,N-1}(\mathbf{x_k}))$. The sets $\mathrm{cl}(\widetilde{Y}_h^{k,m}(\mathbf{x_k}))$ and $\mathrm{cl}(Y_h^{k,N-1}(\mathbf{x_k}))$ are compact. The inclusion $K_1 \subset K_2$ comes from Lemma 4.5, while the inclusion $K_2 \subset K_1 + D$ comes from Lemmas 4.6 and 4.7. □

Corollary 4.3. *The discrete minimal element set $V_h(\mathbf{x_0})$ is equal to the discrete minimal element set $\widetilde{V}_h^{0,0}(\mathbf{x_0})$, which can be obtained from the discrete multiobjective dynamic programming equation* (HJ_h)

$$\widetilde{V}_h^{k,k}(\mathbf{x_k}) = \mathcal{E}(\mathrm{cl}(\{h\mathbf{L}(\mathbf{x_k}, \mathbf{u}) + \widetilde{V}_h^{k+1,k+1}(\mathbf{x_k} + hf(\mathbf{x_k}, \mathbf{u})), \mathbf{u} \in U\}), D), \tag{4.12}$$

with terminal data condition

$$\widetilde{V}_h^{N,N}(\mathbf{x}_N) = \{0\}. \tag{4.13}$$

Proof. Recalling that the discrete element set $V_h(\mathbf{x_0})$ is the same as the set $V_h^{0,N-1}(\mathbf{x_0})$, this result follows directly from Proposition 4.11. $\square$

4.6 A Discretization in the State Space

This section describes the last step of the approximation method proposed in this chapter. This step consists of a discretization in the state space of the discrete multiobjective dynamic programming equation (4.12) of Corollary 4.3, which yields an approximate multiobjective dynamic programming equation (HJ_h^d). Using the continuity of the minimal element map (Proposition 4.5), the solution to the approximate multiobjective dynamic programming equation (HJ_h^d) is shown in Corollary 4.4 to converge towards the discrete minimal element set $V_h(\mathbf{x_0})$ in the sense of Hausdorff as the state-space discretization converges towards zero.

The discretization of the state space is first performed. If $X_k,\ k = 0, \ldots, N$ denotes the set of possible values for the state at the instant t_k, assuming that the set X_0 is compact, then it can be observed, using the continuity of the function $\mathbf{f}(\cdot,\cdot)$, that each set $X_k,\ k = 1, \ldots, N$ is compact. By compactness, each set $X_k,\ k = 0, \ldots, N$ can be covered by a finite number M_k of closed balls $B(\mathbf{x_{k,j}}, \epsilon_k),\ j = 1, \ldots, M_k$. Define $d = \max\{\epsilon_k,\ k = 0, \ldots, N\}$ as the size of the state-space discretization.

Let $k = 0, \ldots, N-1$ and $\mathbf{x_k} \in X_k$; then there always exists j such that $\mathbf{x_k} \in B(\mathbf{x_{k,j}}, \epsilon_k)$. The set $\widehat{Y}_h^{k,k}(\mathbf{x_k})$, representing the state-space approximation to the discrete objective space $\widetilde{Y}_h^{k,k}(\mathbf{x_k})$, is defined by

$$\widehat{Y}_h^{k,k}(\mathbf{x_k}) = \{h\mathbf{L}(\mathbf{x_{k,j}}, \mathbf{u}) + \mathcal{E}(\mathrm{cl}(\widehat{Y}_h^{k+1,k+1}(\mathbf{x_k} + hf(\mathbf{x_k}, \mathbf{u}))), D), \mathbf{u} \in U\}, \tag{4.14}$$

while the set $\widehat{Y}_h^{N,N}(\mathbf{x_N})$ is defined by

$$\widehat{Y}_h^{N,N}(\mathbf{x_N}) = \{0\}. \tag{4.15}$$

Proposition 2.1, together with the boundedness assumption on the running cost function $\mathbf{L}(\cdot,\cdot)$, justify the definition of the set $\widehat{Y}_h^{k,k}(\mathbf{x_k})$. The minimal element set $\widehat{V}_h^{k,k}(\mathbf{x_k})$, representing the state-space approximation to the minimal element set $\widetilde{V}_h^{k,k}(\mathbf{x_k})$, is defined by

$$\widehat{V}_h^{k,k}(\mathbf{x_k}) = \mathcal{E}(\mathrm{cl}(\widehat{Y}_h^{k,k}(\mathbf{x_k})), D).$$

From (4.14)-(4.15), it can be written

$$\widehat{V}_h^{k,k}(\mathbf{x_k}) = \mathcal{E}(\mathrm{cl}(\{h\mathbf{L}(\mathbf{x_{k,j}}, \mathbf{u}) + \widehat{V}_h^{k+1,k+1}(\mathbf{x_k} + hf(\mathbf{x_k}, \mathbf{u})), \mathbf{u} \in U\}), D), \tag{4.16}$$

and

$$\widehat{V}_h^{N,N}(\mathbf{x_N}) = \{0\}. \tag{4.17}$$

The multiobjective dynamic programming equation (4.16) denoted by (HJ_h^d), together with the terminal data condition (4.17), is an approximation to the multiobjective dynamic programming equation (4.12), together with the terminal data condition (4.13). Using the continuity of the minimal element map, it is shown in Proposition 4.12 that the minimal element set $\widehat{V}_h^{k,k}(\mathbf{x_k})$ converges towards the discrete minimal element set $\widetilde{V}_h^{k,k}(\mathbf{x_k})$ in the sense of Hausdorff as the state-space discretization d converges towards zero.

Proposition 4.12. *With the assumption (see Proposition* 4.5*)*

$$\widetilde{V}_h^{k,k}(\mathbf{x_k}) = \mathcal{E}(\mathrm{cl}(\widetilde{Y}_h^{k,k}(\mathbf{x_k})), \mathrm{int}(D)'), \ \forall k = 0, \dots, N-1, \ \forall \mathbf{x_k} \in X_k,$$

we have

$$\lim_{d \to 0} \mathcal{H}(\widehat{V}_h^{k,k}(\mathbf{x_k}), \widetilde{V}_h^{k,k}(\mathbf{x_k})) = 0, \ \forall \mathbf{x_k} \in X_k.$$

Proof. The proposed proof is a proof by induction.

Step 1. Proof for the case $k = N-1$. Let $\mathbf{x_{N-1}} \in X_{N-1}$. We have

$$\widetilde{Y}_h^{N-1,N-1}(\mathbf{x_{N-1}}) = \{h\mathbf{L}(\mathbf{x_{N-1}}, \mathbf{u}), \mathbf{u} \in U\},$$

and

$$\widehat{Y}_h^{N-1,N-1}(\mathbf{x_{N-1}}) = \{h\mathbf{L}(\mathbf{x_{N-1,j}}, \mathbf{u}), \mathbf{u} \in U\}.$$

To be able to apply Proposition 4.5, it must be verified that the sets $\mathrm{cl}(\widetilde{Y}_h^{N-1,N-1}(\mathbf{x_{N-1}}))$ and $\mathrm{cl}(\widehat{Y}_h^{N-1,N-1}(\mathbf{x_{N-1}}))$ are compact, and they are as both these sets are bounded. It is also required that

$$\lim_{\epsilon_{N-1} \to 0} \mathcal{H}(\mathrm{cl}(\widehat{Y}_h^{N-1,N-1}(\mathbf{x_{N-1}})), \mathrm{cl}(\widetilde{Y}_h^{N-1,N-1}(\mathbf{x_{N-1}}))) = 0.$$

This follows from the Lipschitz assumption on the running cost $\mathbf{L}(\cdot,\cdot)$.

Step 2. Proof for the case $k-1$ assuming that the result is true for $k > 1$, i. e.,

$$\lim_{(\epsilon_k, \dots, \epsilon_{N-1}) \to 0} \mathcal{H}(\widehat{V}_h^{k,k}(\mathbf{x_k}), \widetilde{V}_h^{k,k}(\mathbf{x_k})) = 0, \forall \mathbf{x_k} \in X_k. \tag{4.18}$$

Let $\mathbf{x_{k-1}} \in X_{k-1}$. To be able to apply Proposition 4.5, it must be verified that the sets $\mathrm{cl}(\widetilde{Y}_h^{k-1,k-1}(\mathbf{x_{k-1}}))$ and $\mathrm{cl}(\widehat{Y}_h^{k-1,k-1}(\mathbf{x_{k-1}}))$ are compact, and they are as both these sets are bounded. It is also required that

$$\lim_{(\epsilon_{k-1},\ldots,\epsilon_{N-1})\to 0} \mathcal{H}(\mathrm{cl}(\widehat{Y}_h^{k-1,k-1}(\mathbf{x_{k-1}})), \mathrm{cl}(\widetilde{Y}_h^{k-1,k-1}(\mathbf{x_{k-1}}))) = 0.$$

This follows from comparing (4.10), which can be rewritten, using Proposition 4.11,

$$\widetilde{Y}_h^{k-1,k-1}(\mathbf{x_{k-1}}) = \{h\mathbf{L}(\mathbf{x_{k-1}}, \mathbf{u}) + \widetilde{V}_h^{k,k}(\mathbf{x_{k-1}} + hf(\mathbf{x_{k-1}}, \mathbf{u})), \mathbf{u} \in U\},$$

with (4.14), which can be rewritten,

$$\widehat{Y}_h^{k-1,k-1}(\mathbf{x_{k-1}}) = \{h\mathbf{L}(\mathbf{x_{k-1,j}}, \mathbf{u}) + \widehat{V}_h^{k,k}(\mathbf{x_{k-1}} + hf(\mathbf{x_{k-1}}, \mathbf{u})), \mathbf{u} \in U\},$$

and using both the induction assumption (4.18) and the Lipschitz assumption on the running cost $\mathbf{L}(\cdot,\cdot)$.

□

Corollary 4.4. *The minimal element set $\widehat{V}_h^{0,0}(\mathbf{x_0})$, solution to the multiobjective dynamic programming equation approximation* (4.16) *with terminal condition* (4.17)*, converges towards the discrete minimal element set $V_h(\mathbf{x_0})$ in the sense of Hausdorff as the state-space discretization d converges towards zero.*

Proof. From Proposition 4.12, it is known that the minimal element set $\widehat{V}_h^{0,0}(\mathbf{x_0})$ converges towards the discrete minimal element set $\widetilde{V}_h^{0,0}(\mathbf{x_0})$. From Corollary 4.3, the discrete minimal element set $\widetilde{V}_h^{0,0}(\mathbf{x_0})$ has been shown to be equal to the discrete minimal element set $V_h(\mathbf{x_0})$, which completes the proof. □

4.7 Extensions

The results obtained in this chapter remain valid, with minimal changes, for nonautonomous problems, including terminal cost. At the expense of adding constraints on the terminal state, a terminal cost can be reformulated as an integral cost [54, pp. 25–26]. However, such constraints are not considered in this chapter. Another alternative is to directly include the terminal cost in the formulation of the multiobjective problem. In this case, the terminal cost function is required to be Lipschitz for the error estimate (Proposition 4.8), and uniformly bounded for the proof of convergence (Proposition 4.10). Another important modification concerning the terminal data condition (4.13) and (4.17) is to include the terminal cost evaluated at the terminal state $\mathbf{x_N}$. For nonautonomous problems, the time variable appears explicitly at each step of the approximation method. In this case, the Lipschitz assumption in time is also required [58, 59].

4.8 Further Results: Minimal Element Map Continuity

The continuity of the minimal element map at $K \in \mathcal{K}$ was established in Proposition 4.5 under the condition that $\mathcal{E}(K, D) = \mathcal{E}(K, \text{int}(D)')$. It will be shown in Proposition 4.14 that the continuity of the minimal element map remains valid under the weaker condition $\text{cl}(\mathcal{E}(K, D)) = \mathcal{E}(K, \text{int}(D)')$. This condition is in fact also necessary as shown in Proposition 4.13.

Proposition 4.13. *If the minimal element map is continuous at $K \in \mathcal{K}$, then $\text{cl}(\mathcal{E}(K, D)) = \mathcal{E}(K, \text{int}(D)')$.*

Proof. Recall that from Lemma 4.1, $\text{cl}(\mathcal{E}(K, D)) \subset \mathcal{E}(K, \text{int}(D)')$. Therefore, to prove the proposition, it is only needed to establish that $\mathcal{E}(K, \text{int}(D)') \subset \text{cl}(\mathcal{E}(K, D))$. Let

$y \in \mathcal{E}(K, \mathrm{int}(D)')$; consider a sequence (y_n) such that $y_n \in y - \mathrm{int}(D)$ and $\lim_{n\to\infty} y_n = y$, and define the sequence of compact sets $K_n = K \cup \{y_n\}$. By construction, $y_n \in \mathcal{E}(K_n, D)$. Moreover, from $\lim_{n\to\infty} y_n = y$, it follows that the sequence (K_n) converges towards K in the sense of Hausdorff. By continuity of the minimal element map at K, it follows that the sequence $(\mathcal{E}(K_n, D))$ converges towards $\mathcal{E}(K, D)$ in the sense of Hausdorff. Hence, $y \in \mathrm{cl}(\mathcal{E}(K, D))$ and finally $\mathrm{cl}(\mathcal{E}(K, D)) \subset \mathcal{E}(K, \mathrm{int}(D)')$. □

The following lemma will prove useful for the proof of Proposition 4.14.

Lemma 4.9. Let (k_n) be a bounded sequence in $\mathbf{R}^n$, and A be the set of cluster points of the sequence (k_n), then there exists a sequence $(x_n),\ x_n \in A$ such that

$$\lim_{n\to\infty} k_n - x_n = 0. \tag{4.19}$$

Proof. Given that the sequence (k_n) is bounded, it follows that A is compact. Let $(x_n),\ x_n \in A$ be a sequence such that:

$$\forall n,\ \|k_n - x_n\| = d(k_n, A).$$

It remains to be proven that (4.19) holds with (x_n) as defined above. Let $\epsilon > 0$, A being compact, there is a finite number of open balls $B(x_i^*, \epsilon),\ i = 1, \dots, p,\ x_i^* \in A$ such that $A \subset \cup_{i=1,\cdots,p} B(x_i^*, \epsilon)$. It is now proved that there exists n_0 such that $\forall n \geq n_0$, $k_n \in \cup_{i=1,\dots,p} B(x_i^*, \epsilon)$. Assume the contrary, then it is possible to find a subsequence $(k_{\psi(n)})$ such that $\forall n,\ k_{\psi(n)} \notin \cup_{i=1,\dots,p} B(x_i^*, \epsilon)$. Let $(k_{\phi\circ\psi(n)})$ be a subsequence of $(k_{\psi(n)})$ converging towards k^*. By the definition of A, $k^* \in A$, and therefore $k^* \in \cup_{i=1,\dots,p} B(x_i^*, \epsilon)$, where $\cup_{i=1,\dots,p} B(x_i^*, \epsilon)$ is an open set. It follows that $(k_{\phi\circ\psi(n)}) \in \cup_{i=1,\dots,p} B(x_i^*, \epsilon)$ for large enough n, which is a contradiction. Now, given $n \geq n_0$, there exists an index $i(n) \in \{1, \dots, p\}$ such that $k_n \in B(x_{i(n)}^*, \epsilon)$. Therefore, $\|x_n - k_n\| = d(k_n, A) \leq \|x_{i(n)}^* - k_n\| = \epsilon$, which completes the proof. $\square$

Proposition 4.14. If $\mathrm{cl}(\mathcal{E}(K,D)) = \mathcal{E}(K, \mathrm{int}(D)')$, then the minimal element map is continuous at $K \in \mathcal{K}$.

Proof. It is needed to prove the two conditions S_1 and S_2 as defined in Proposition 4.2.

Part 1 (proof of S_1): In Part 1 of the proof of Proposition 4.5, it was shown that every element in $\mathcal{E}(K,D)$ is the limit of a sample of the sequence $(\mathcal{E}(K_n,D))$. Hence, it can be deduced that every element in $\mathrm{cl}(\mathcal{E}(K,D))$ is the limit of a sample of the sequence $(\mathcal{E}(K_n,D))$.

Part 2 (proof of S_2): Let (k_n) be a sample of the sequence $(\mathcal{E}(K_n,D))$, it is needed to show that there exists a sequence (x_n) in $\mathrm{cl}(\mathcal{E}(K,D))$ such that

$$\lim_{n\to\infty} k_n - x_n = 0.$$

As $\bigcup_{n\in N} K_n$ is bounded, it follows that the sequence (k_n) is bounded. It can simply be shown that each cluster point of (k_n) belongs to $\mathcal{E}(K, \mathrm{int}(D)')$; therefore, by assumption, it also belongs to $\mathrm{cl}(\mathcal{E}(K,D))$. Then, the sequence (x_n) can be directly obtained from Lemma 4.9. $\square$

Corollary 4.5. The minimal element map is continuous at K if and only if $\mathrm{cl}(\mathcal{E}(K,D)) = \mathcal{E}(K, \mathrm{int}(D)')$.

Proof. This follows directly from Proposition 4.13 and 4.14. $\square$

Chapter 5

Practical Considerations

As described in Chapter 3, the major issue with using DDP to solve PCTS_2 is the resolution time. As shown in Section 5.1, this issue arises from the fact that the complexity of the DDP approximation method is exponential with the grid size. By clustering [32, pp. 325–329] the Pareto optimal set at each step of the resolution of the approximate multiobjective dynamic programming equation, this complexity can be reduced to polynomial. In Section 5.2, a measure indicating how well a given set of points may represent the Pareto optimal set is introduced. This measure is applied to the Pareto objective vectors and the approximate Pareto objective vectors as obtained in Chapter 3 with the weighting method and the DDP approximation method respectively. This measure is also applied to the set of points obtained when solving the approximate dynamic programming equation with clustering. Finally, in Section 5.3, it is shown how continuous joint trajectories can be generated from the discrete joint trajectories obtained with the DDP approximation method.

5.1 Resolution Time Reduction via Clustering

5.1.1 Complexity of the DDP Approximation Method

For clarity, the approximate dynamic programming equation (3.44) resulting from the DDP approximation method is

$$J_k^{h,d}(v_k) = \mathcal{E}(\{h(\widetilde{f}_1(k, v_k, \dot{v}_k), \widetilde{f}_2(k, v_k)) + J_{k+1}^{h,d}(v_k + h\dot{v}_k),\ \dot{v}_k \in \mathcal{B}_k^{h,d}(v_k)\}, \mathbf{R}_+^2),$$

with (3.45) as terminal data condition:

$$J_{N_T}^{h,d}(v_{N_T}) = \{(0,0)\}.$$

The complexity of the resolution of the approximate dynamic programming equation is directly related to the cardinality of the sets $J_k^{h,d}(v_k)$. Indeed, the set $J_k^{h,d}(v_k)$ is determined by performing comparisons between the elements of the set $\{h(\widetilde{f}_1(k, v_k, \dot{v}_k), \widetilde{f}_2(k, v_k)) + J_{k+1}^{h,d}(v_k + h\dot{v}_k),\ \dot{v}_k \in \mathcal{B}_k^{h,d}(v_k)\}$ with cardinality

$$\sum_{\dot{v}_k \in \mathcal{B}_k^{h,d}(v_k)} |J_{k+1}^{h,d}(v_k + h\dot{v}_k)|.$$

Therefore, using a simple element by element comparison method, the complexity to obtain the set $J_k^{h,d}(v_k)$ is

$$O\Big(\Big(\sum_{\dot{v}_k \in \mathcal{B}_k^{h,d}(v_k)} |J_{k+1}^{h,d}(v_k + h\dot{v}_k)|\Big)^2\Big).$$

Recalling that $|\mathcal{B}_k^{h,d}(v_k)| = O(N_X)$, we obtain

$$O\Big(\Big(\sum_{\dot{v}_k \in \mathcal{B}_k^{h,d}(v_k)} |J_{k+1}^{h,d}(v_k + h\dot{v}_k)|\Big)^2\Big) = O(N_X^2 |J_{k+1}^{h,d}(v_{k+1})|^2), \tag{5.1}$$

All the sets $J_k^{h,d}(v_k)$ at t_k can therefore be determined in $O(N_X^3|J_{k+1}^{h,d}(v_{k+1})|^2)$, and the sets $J_0^{h,d}(v_0)$ in

$$\sum_{k=0}^{N_T-1} O(N_X^3|J_{k+1}^{h,d}(v_{k+1})|^2). \tag{5.2}$$

To express (5.2) as a function of N_T and N_X only, $|J_{k+1}^{h,d}(v_{k+1})|$ must be determined. From the equality $|\mathcal{B}_k^{h,d}(v_k)| = O(N_X)$, it can be deduced that the cardinality of the sets $J_k^{h,d}(v_k)$ increases, in the worst case, by a factor of N_X between each time step. It is therefore a simple induction to show that $|J_k^{h,d}(v_k)| = O(N_X^{N_T-k})$, which substituted in (5.2), yields

$$\sum_{k=0}^{N_T-1} O(N_X^3|J_{k+1}^{h,d}(v_{k+1})|^2) = \sum_{k=0}^{N_T-1} O(N_X^{2N_T-2k+3}) = O(N_X^{2N_T+3}). \tag{5.3}$$

The worst case for the resolution of the approximate dynamic programming equation (3.44) with terminal data condition (3.45) is therefore exponential in the grid size, which explains why the resolution time to obtain the set $J_0^{h,d}(v_0)$ in Section 3.3.3 was as large as 24,628 s. It also justifies a posteriori why no experimental convergence study for the DDP approximation method was performed in Chapter 3.

The clustering problem [32, p. 325–329] is defined as the problem of determining the "best" partition of a given set into subsets, or *clusters*. As already noted, the complexity of the resolution of the approximate dynamic programming equation directly relates to the cardinality of the sets $J_k^{h,d}(v_k)$. Therefore, once the set $J_k^{h,d}(v_k)$ is obtained, it is proposed to determine a cluster of $J_k^{h,d}(v_k)$ with cardinality N_C and use this cluster for the subsequent iteration of the approximate dynamic programming equation. As a result, the cardinality of the sets $J_{k+1}^{h,d}(v_k + h\dot{v}_k)$ is at most N_C, the complexity (5.1) to obtain the set $J_k^{h,d}(v_k)$ becomes $O(N_X^2 N_C^2)$, and the cardinality of the set $J_k^{h,d}(v_k)$ is $O(N_X N_C)$. Accordingly, the complexity (5.2) to obtain the sets $J_0^{h,d}(v_0)$ becomes $O(N_T N_X^3 N_C^2)$. Therefore, the complexity of the resolution of the approximate dynamic programming equation becomes polynomial in the grid size.

However, to be precise, the effect of the clustering complexity $f(m,n)$, where m is the cardinality of the cluster and n the cardinality of the original set, must also be taken into account. As seen above, with clustering, the complexity to obtain the set $J_k^{h,d}(v_k)$ is $O(N_X^2 N_C^2)$ and the cardinality of the set $J_k^{h,d}(v_k)$ is $O(N_X N_C)$. Therefore, the complexity to obtain a cluster of the set $J_k^{h,d}(v_k)$ is $O(N_X^2 N_C^2)+f(N_C, O(N_X N_C))$. Hence, the clusters for all the sets $J_k^{h,d}(v_k)$ at t_k can be determined in $O(N_X^3 N_C^2)+N_X f(N_C, O(N_X N_C))$, and finally the clusters for the sets $J_0^{h,d}(v_0)$ in

$$\sum_{k=0}^{N_T-1} O(N_X^3 N_C^2)+N_X f(N_C, O(N_X N_C)) = O(N_T N_X^3 N_C^2)+N_T N_X f(N_C, O(N_X N_C)). \quad (5.4)$$

Equation (5.4) shows that, for the complexity of the resolution of the approximate dynamic programming equation with clustering to be polynomial, it is necessary that the clustering complexity is itself polynomial. In Section 5.1.2, attention will therefore be restricted to clustering methods with polynomial complexity.

5.1.2 Clustering Methods

As PCTS$_2$ has only two objective functions, the sets $J_k^{h,d}(v_k)$ are subsets of $\mathbf{R}^2$. Therefore, the clustering problem for PCTS$_2$ can be formally stated as follows. Let $A = \{x_i,\ i = 1,\dots,n\}$ be a set of distinct points in the plane. Given m ($m < n$), the clustering problem for PCTS$_2$ is the problem of selecting m points in A, or equivalently, choosing the cluster as an element $\widetilde{A}^*$ of the set $\mathcal{A}_m$, where $\mathcal{A}_m = \{\widetilde{A} \subset A,\ |\widetilde{A}| = m\}$ is the set of subsets of A with cardinality m. Three clustering methods are proposed below. For each of the clustering methods, the complexity f is provided. The functions f that are provided are only indicative of the actual implementation of the clustering methods, and might not correspond to the “best” complexity or the “best” implementation for these methods. For each of the clustering methods, the resulting complexity for the resolution of the approximate dynamic programming equation, calculated using (5.4), is shown to be $O(N_T N_X^3 N_C^2)$.

For the description of the clustering methods, the concept of *extreme points* needs to be introduced. The extreme points are the two points with the lowest x and y coordinates and will be assumed to correspond to x_1 and x_n respectively. As explained below, each of the clustering methods has some freedom, which, in particular, allows for the inclusion of the extreme points in the construction of the cluster $\widetilde{A}^*$. The impact of such a choice over a random point in A will be illustrated in Section 5.1.3. It also explains why variants are introduced for each clustering method.

Random clustering

The first clustering method, denoted M_5, consists of selecting the m points randomly. A variant of this clustering method, denoted M_4, is to include the two extreme points among the m points. For M_5, $f(m,n) = O(m)$. Therefore, substituting in (5.4) yields

$$O(N_T N_X^3 N_C^2) + O(N_T N_X N_C) = O(N_T N_X^3 N_C^2).$$

For M_4, $O(n)$ is required to find the extreme points, therefore $f(m,n) = O(m+n)$. Substituting in (5.4) yields

$$O(N_T N_X^3 N_C^2) + N_T N_X O(N_C + N_X N_C) = O(N_T N_X^3 N_C^2) + O(N_T N_X^2 N_C) = O(N_T N_X^3 N_C^2).$$

Clustering Based on the Hausdorff Distance

The Hausdorff distance, introduced in Chapter 4, is a distance function between compact sets. Therefore, it is natural to suggest, as a second clustering method, to select a cluster $\widetilde{A}^*$ that minimizes the Hausdorff distance to A. From Chapter 4, the Hausdorff distance between the sets $\widetilde{A}$ and A is:

$$\mathcal{H}(\widetilde{A}, A) = \max\{\max_{x_i \in \widetilde{A}} \min_{x_j \in A} \|x_i - x_j\|, \max_{x_i \in A} \min_{x_j \in \widetilde{A}} \|x_i - x_j\|\}.$$

By definition, $\widetilde{A} \subset A$, therefore:

$$\forall x_i \in \widetilde{A},\ \min_{x_j \in A} \|x_i - x_j\| = 0.$$

Hence,

$$\mathcal{H}(\widetilde{A}, A) = \max_{x_i \in A} \min_{x_j \in \widetilde{A}} \|x_i - x_j\| = \max_{x_i \in A \backslash \widetilde{A}} \min_{x_j \in \widetilde{A}} \|x_i - x_j\|.$$

The cluster $\widetilde{A}^*$ therefore satisfies

$$\mathcal{H}(\widetilde{A}^*, A) = \min_{\widetilde{A} \in \mathcal{A}_m} \mathcal{H}(\widetilde{A}, A) = \min_{\widetilde{A} \in \mathcal{A}_m} \max_{x_i \in A \backslash \widetilde{A}} \min_{x_j \in \widetilde{A}} \|x_i - x_j\|.$$

In Proposition 5.1, it is shown that $\widetilde{A}^*$ can be obtained by solving the *restricted central m-clustering problem* for the set A. The restricted central m-clustering problem for a given set is a well-known problem in geometric location theory [33], [34, pp. 325–327], which can be described as follows. Call a partition of A into m clusters, $A_i,\ i = 1, \ldots, m$, an *m-clustering.* The *central cluster size* of an m-clustering is defined as the least value d for which all the points in each cluster A_i are within the distance d of some cluster centre which can be any point in space. The *central clustering problem* is to find, for a given set A and integer m, an m-clustering for A with minimum central cluster size d^*. When the cluster centres are restricted to belong to A, the central m-clustering problem is referred to as the *restricted central m-clustering problem* [33].

Proposition 5.1. Let d^* be the minimum central cluster size for A and $A_i^*,\ i = 1, \dots, m$ be an m-clustering with central cluster size d^*. If $\widehat{A}$ is the set of the m cluster centres, then

$$\mathcal{H}(\widehat{A}, A) = \min_{\widetilde{A} \in \mathcal{A}_m} \mathcal{H}(\widetilde{A}, A),$$

$d^* = \mathcal{H}(\widehat{A}, A)$, and hence, the cluster $\widetilde{A}^*$ can be chosen to be $\widehat{A}$.

Proof. The proof of this result uses the following characterization of the Hausdorff distance provided in Chapter 4:

$$\mathcal{H}(\widetilde{A}, A) = \min_{l \geq 0} \{\widetilde{A} \subset A + lB \text{ and } A \subset \widetilde{A} + lB\}.$$

By definition, $\widetilde{A} \subset A$, therefore:

$$\mathcal{H}(\widetilde{A}, A) = \min_{l \geq 0} \{A \subset \widetilde{A} + lB\}.$$

First, we prove that $\forall \widetilde{A} \in \mathcal{A}_m,\ \mathcal{H}(\widetilde{A}, A) \geq d^*$ by building an m-clustering for each $\widetilde{A} \in \mathcal{A}_m$. Without loss of generality, it can be assumed that $\widetilde{A} = \{x_1, \dots, x_m\}$. First, define the set A_1 which contains x_1 and all the elements in $A \backslash \widetilde{A}$ within a distance of $\mathcal{H}(\widetilde{A}, A)$ from x_1:

$$A_1 = \{x_1\} \cup \{x_i \in B(x_1, \mathcal{H}(\widetilde{A}, A)) \cap (A \backslash \widetilde{A})\}.$$

Then, define, recursively from $j = 2$ to m, the set A_j, which contains x_j and all the elements in $A \backslash \widetilde{A}$ within a distance of $\mathcal{H}(\widetilde{A}, A)$ from x_j that have not already been included in the sets $A_i,\ i < j$:

$$A_j = \{x_j\} \cup \{x_i \in B(x_j, \mathcal{H}(\widetilde{A}, A)) \cap (A \backslash (\widetilde{A} \cup_{i=1}^{j-1} A_i))\}.$$

It follows, from the above characterization of the Hausdorff distance, that $\cup_1^m A_j = A$. Hence, the sets $A_j,\ j = 1, \ldots, m$ form an m-clustering whose size is by construction smaller than or equal in size to $\mathcal{H}(\widetilde{A}, A)$. Therefore, $\forall \widetilde{A} \in \mathcal{A}_m,\ d^* \leq \mathcal{H}(\widetilde{A}, A)$, which yields

$$d^* \leq \min_{\widetilde{A} \in \mathcal{A}_m} \mathcal{H}(\widetilde{A}, A).$$

To prove the reverse inequality, from its definition, $\widehat{A}$ is the set of the m cluster centres. Therefore, by definition of the central cluster size, $A \subset \widehat{A} + d^* B$. From the above characterization of the Hausdorff distance, it follows that

$$\mathcal{H}(\widehat{A}, A) \leq d^*.$$

Hence, combining the two above inequalities yields

$$d^* = \mathcal{H}(\widehat{A}, A) = \min_{\widetilde{A} \in \mathcal{A}_m} \mathcal{H}(\widetilde{A}, A).$$

Hence, the cluster $\widetilde{A}^*$ can be chosen to be $\widehat{A}$. □

Unfortunately, for dimension two and above, the restricted central m-clustering problem is very difficult to solve: it is *NP-hard* [35, pp. 114–145] for approximation factors of δ in the central cluster size if $\delta < \sqrt{3}$ ([33], Theorem 12). In other words, unless P=NP, there does not exist any deterministic polynomial algorithm that can solve the approximate restricted central m-clustering problem with an approximation factor less than $\sqrt{3}$. The best known algorithms in terms of algorithmic complexity for the resolution of the restricted central m-clustering problem and its $(1+\epsilon)$-approximation ($\epsilon > 0$) are the ones from [60] with an algorithmic complexity respectively of $n^{O(\sqrt{m})}$ and $O(n\log(m) + (m/\epsilon)^{O(\sqrt{m})})$. However, these two algorithms do not satisfy the requirement of polynomial complexity. The "best" known polynomial algorithm for the resolution of the restricted central m-clustering problem is the farthest-point clustering from [34, p. 326] and [36]. This $O(nm)$ algorithm guarantees a solution within at most two times the minimum central cluster size. The $O(nm)$ algorithmic complexity was improved to $O(n\log(m))$ in [33]. However, due to simplicity of its implementation, the $O(nm)$ algorithm from [36] is preferred and used here. Substituting $f(m,n) = O(nm)$ in (5.4) yields

$$O(N_T N_X^3 N_C^2) + N_T N_X f(N_C, O(N_X N_C) = O(N_T N_X^3 N_C^2) + O(N_T N_X^2 N_C^2) = O(N_T N_X^3 N_C^2).$$

For the farthest-point clustering, the choice of the initial point is arbitrary. Therefore, for the method M_6, the initial point is chosen randomly, and for the variant M_7, the initial point is chosen to be the extreme point x_1.

Clustering based on the Normal-Boundary Intersection (NBI) method [61]

The third clustering method is inspired by the NBI method developed to solve multiobjective nonlinear optimization problem [61], and aims at selecting points that are "uniformly distributed". This clustering method is based on the following essential observation. From the definition of the extreme points x_1 and x_n, it follows that the set A is completely contained in a rectangle with x_1 as the upper left corner, and x_n as the lower right corner. From there, trace the diagonal between x_1 and x_n, and divide this diagonal into $m-2$ identical segments. By tracing lines perpendicular to the diagonal at the extremities of these segments, a partition A_i, $1 \leq i \leq m-2$ of the set $A\backslash\{x_1, x_n\}$ is obtained. This procedure is illustrated in Figure 5.1.

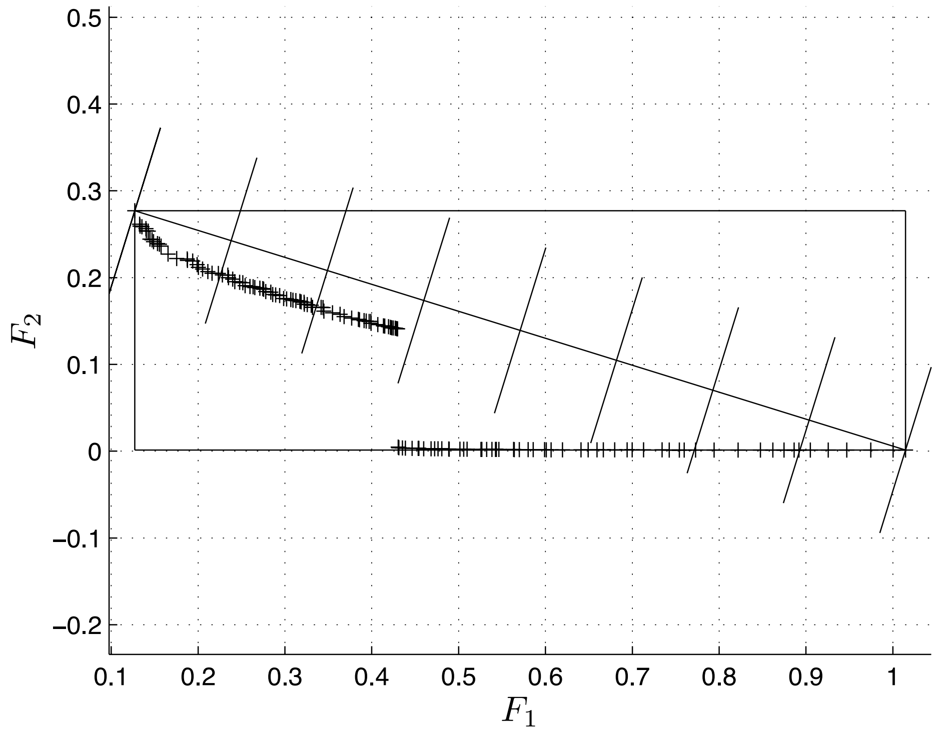


Figure 5.1: Performing the partition based on the NBI method for a given set and for $m = 10$.

The final step of this clustering method consists of selecting one point in each set A_i, which can be done as follows:

- Method M_1: take the most "centred" point in each set A_i, i.e., the closest point to the line perpendicular to the diagonal dividing the set A_i evenly.
- Method M_2: take the furthest point from the diagonal in the direction of the origin.
- Method M_3: take the point randomly.

Once one point in each A_i is selected, the extreme points are added. Note that it is possible that a set A_i does not contain any point. In such a case, this clustering method provides a subset with a cardinality less than m.

Table 5.1: Description of the clustering methods. (E) indicates that the cluster contains at least one extreme point by construction.

NBI			Random		Hausdorff	
M_1	M_2	M_3	M_4	M_5	M_6	M_7
(E)	(E)	(E)	(E)	-	-	(E)

For all the three methods, it is required to find the extreme points and select a point in A_i. Therefore, the complexity $f(m,n)$ is $O(m+n)$. Substituting in (5.4) yields

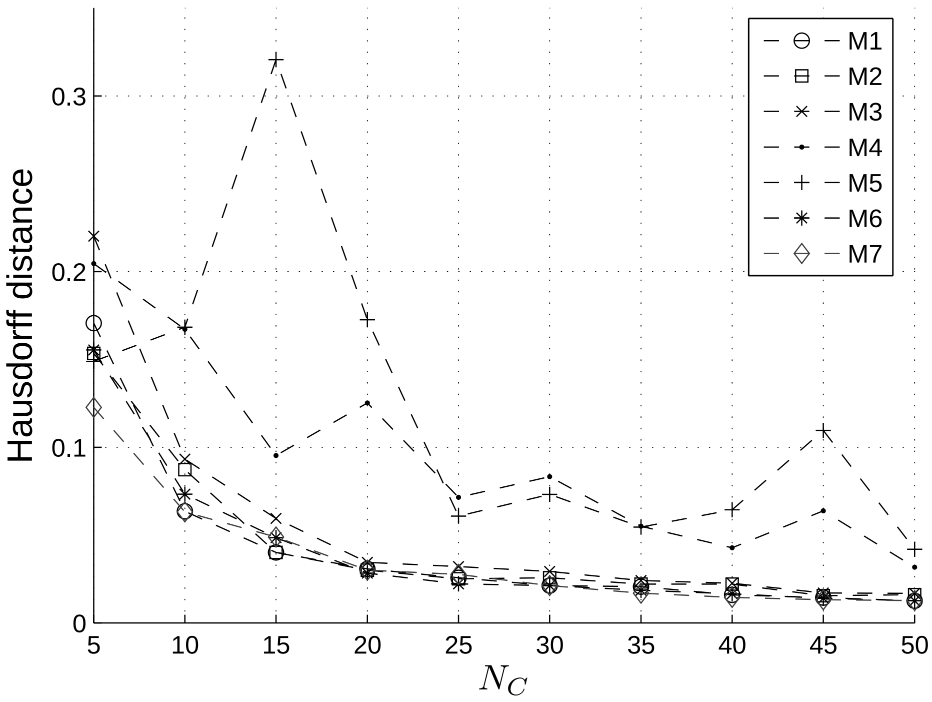
$$O(N_T N_X^3 N_C^2) + N_T N_X O(N_C + N_X N_C) = O(N_T N_X^3 N_C^2) + O(N_T N_X^2 N_C) = O(N_T N_X^3 N_C^2).$$

The three proposed clustering methods therefore yield the same complexity $O(N_T N_X^3 N_C^2)$ for the resolution of the approximate dynamic programming equation. A summary of the methods is provided Table 5.1.

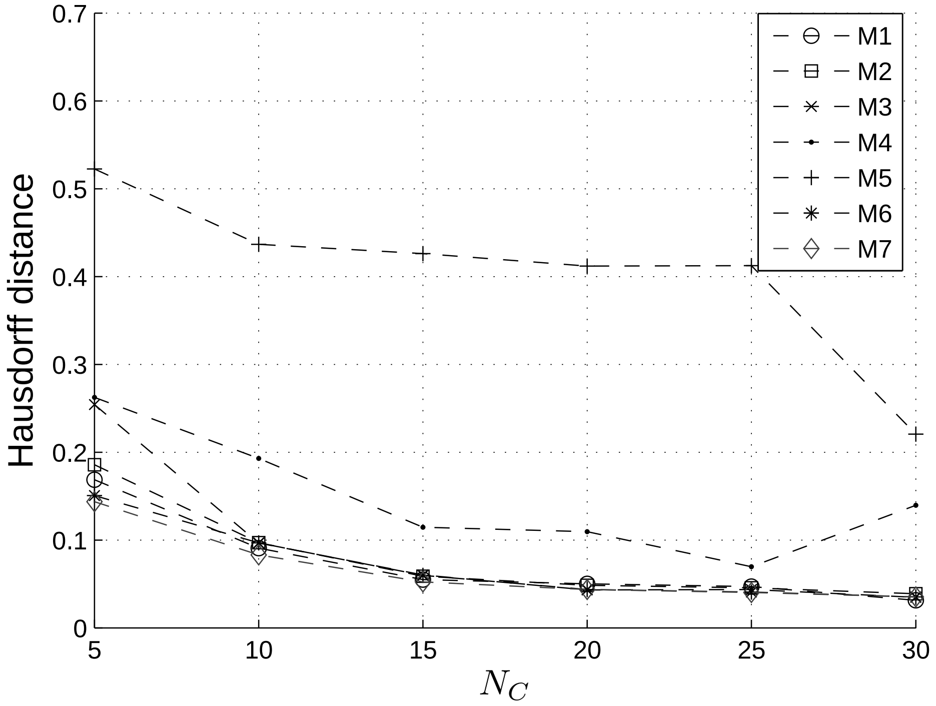
5.1.3 Numerical Results

The same problem PCTS$_2$ as solved in Section 3.3.3 is also used in this section. Therefore, N_T and N_X are fixed. Two series of tests are performed and reported below. For the first series of tests in Figure 5.2(a), clustering is applied directly to the set $J_0^{h,d}(v_0)$ whose cardinality in Section 3.3.3 was found to be 138. With the notation from Section 5.1.2, this corresponds to $A = J_0^{h,d}(v_0)$. For the second series of tests in Figure 5.2(b), the clustering is applied to the resolution of the approximate dynamic programming equation, i.e., $A = J_k^{h,d}(v_k)$. The objective of these two series of tests is to find which clustering method performs the best, to illustrate the importance of including the extreme points, and finally to verify that clustering indeed reduces the resolution time of the approximate dynamic programming equation. As a measure to compare the clustering methods, the Hausdorff distance between the final set obtained with clustering and the original set $J_0^{h,d}(v_0)$ is used.

For the clustering method involving randomness, an average of ten executions was found to provide consistent results.



(a) First series of tests: clustering applied to $J_0^{h,d}(v_0)$.



(b) Second series of tests: clustering applied to the approximate dynamic programming equation.

Figure 5.2: Variation of the Hausdorff distance as a function of N_C for the clustering methods M_1 to M_7.

From Figures 5.2(a) and 5.2(b), for each of the clustering methods, the trend is that the Hausdorff distance decreases as N_C increases, which is expected. It is clear that clustering based on the NBI method, i.e., M_1, M_2, and M_3, and clustering based on the Hausdorff distance, i.e., M_6 and M_7, perform much better than random clustering, i.e., M_4 and M_5. The comparison between clustering based on the NBI method and clustering based on the Hausdorff distance is less obvious, therefore, it is proposed to count the number of times these methods provide the best Hausdorff distance. When different methods provide the same Hausdorff distance, the counter for each of these methods is incremented. The results are provided in Tables 5.2 and 5.3, which indicate that clustering based on the Hausdorff distance performs the best.

Table 5.2: First series of tests: comparison based on the smallest Hausdorff distance.

NBI			Random		Hausdorff	
4			0		10	
M_1	M_2	M_3	M_4	M_5	M_6	M_7
3	1	0	0	0	4	6

Table 5.3: Second series of tests: comparison based on the smallest Hausdorff distance.

NBI			Random		Hausdorff	
2			0		4	
M_1	M_2	M_3	M_4	M_5	M_6	M_7
1	1	0	0	0	0	4

When comparing M_6 and M_7 from Tables 5.2 and 5.3, the same conclusion as when comparing M_4 and M_5 from Figures 5.2(a) and 5.2(b) can be made. The variants of the clustering methods which include the extreme points by construction, i.e., M_4 and M_7, perform better. This can be explained by the fact that including the extreme points in the clustering method guarantees that the extreme points at each iteration of the approximate dynamic programming equation are the same with and without clustering. In other words, the rectangle defined from the extreme points in Figure 5.1 is the same at each iteration with and without clustering. On the other hand, if the extreme points are not included, then, at each iteration, the rectangle with clustering becomes smaller and smaller compared to the rectangle without clustering. From the above discussion, M_7 is retained as the clustering method for the rest of this section.

The approximate dynamic programming equation is now solved with M_7 and $N_C = 30$. The resulting set is shown, together with $J_0^{h,d}(v_0)$, in Figure 5.3. The Hausdorff distance between these two sets is 0.0351. The resolution time drops considerably from 24,628 s to 159 s. As expected, note that the resulting set includes the extreme points without clustering. This shows that it is possible to obtain an approximate set to $J_0^{h,d}(v_0)$ in a very short time which is very close to $J_0^{h,d}(v_0)$ in terms of the Hausdorff distance.

It is also interesting to compare the set obtained above with the set obtained when applying M_7 directly with $N_C = 30$ to $J_0^{h,d}(v_0)$. Indeed, the Hausdorff distance obtained when directly applying M_7 to $J_0^{h,d}(v_0)$ represents the "best" Hausdorff distance that can be expected when applying clustering to the resolution of the approximate dynamic programming equation. In this case, a Hausdorff distance of 0.0213 was obtained, which is close to the value 0.0351 previously obtained. The two resulting sets are represented in Figure 5.4.

5.1.4 Conclusion and Extensions

Clustering has been shown to address very efficiently the resolution time issue associated with using DDP by reducing the complexity from exponential to polynomial in the grid

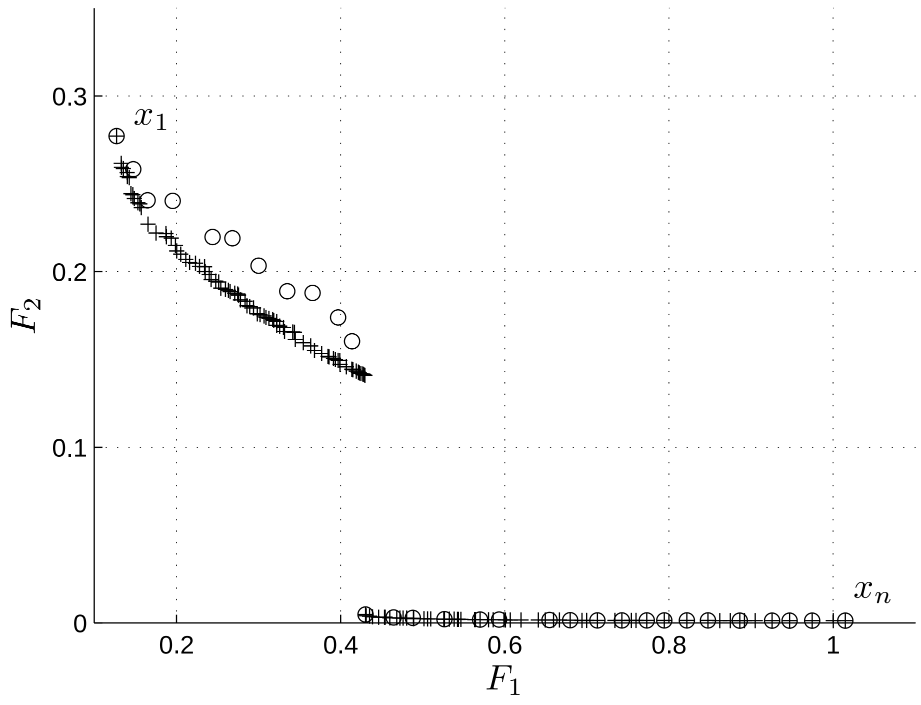


Figure 5.3: Comparison between the set obtained from the resolution of the approximate dynamic programming equation with M_7 and $N_C = 30$ (circles) and the set $J_0^{h,d}(v_0)$ (plus signs).

size. The resolution time increases by increasing the clustering parameter N_C, but reduces the Hausdorff distance with the set $J_0^{h,d}(v_0)$ of approximate Pareto objective vectors. The choice of N_C will be discussed in Section 5.2.

The clustering methods proposed in Section 5.1.2 have been developed for two objective functions. However, the clustering method based on the NBI method cannot be extended to more than two objective functions. Indeed, the construction of the rectangle containing the Pareto objective vectors does not extend to dimension three and above as shown by the following example. Let $x_1 = (1, 0, 0)$ be the Pareto objective vector with the smallest x component, $x_2 = (3, 2, -2)$ is another Pareto objective vector which happens to have both the smallest y and z components. Any point $x_n = (2, n, -1),\ n > 0$ could be another

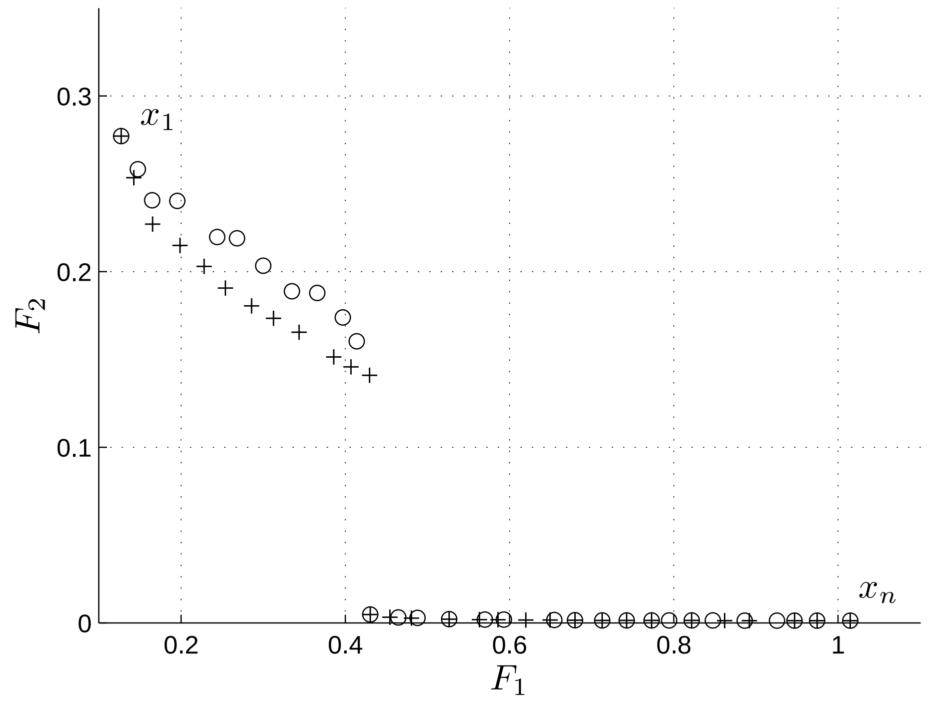


Figure 5.4: Comparison between the set obtained from the resolution of the approximate dynamic programming equation with M_7 and $N_C = 30$ (circles) and the set obtained when applying directly M_7 with $N_C = 30$ to $J_0^{h,d}(v_0)$ (plus signs).

Pareto objective vector. Therefore, the set of Pareto objective vectors is not necessarily bounded by the extreme points. However, the farthest point algorithm used in the clustering method based on the Hausdorff distance is valid in any dimension. Interestingly, its complexity does not depend on the dimension of the space. Therefore, the complexity of the resolution of the approximate dynamic programming equation with clustering based on the Hausdorff distance will remain polynomial for dimension three and above.

5.2 A Quantitative Comparison between the Weighting Method and DDP

In Chapter 3, the comparison between the approximate Pareto objective vectors obtained with the DDP approximation method and the Pareto objective vectors obtained with the weighting method was only qualitative. This section aims at providing a performance measure [62, pp. 187–211] of how well these two sets represent the Pareto optimal set. Naturally, the Hausdorff distance is chosen for this measure. However, as the Pareto optimal set is not known, the Hausdorff distance cannot be applied directly. Nevertheless, lower and upper bounds can still be derived, which are sufficient to conclude that the DDP approximation method outperforms the weighting method.

5.2.1 An Upper Bound for the Performance Measure

Let $A^* \subset \mathbf{R}^2$ be a Pareto optimal set, and $A = \{x_1, \ldots, x_n\}$ be a subset of A^*, where x_1 and x_n are assumed to be the extreme points of A^*. The objective of this section is to derive an upper bound for $\mathcal{H}(A, A^*)$ when A^* is supposed to be not known and A is known.

As already explained in Section 5.1.2, A^* is included in the rectangle R with x_1 as the upper left corner and x_n as the lower right corner. Also, from the definition of a Pareto objective vector, it follows that there cannot be any Pareto objective vector in any of the sets $(x_i + \mathbf{R}^2_+) \cup (x_i - \mathbf{R}^2_+),\ i = 1, \ldots, n$. Therefore, $A^* \subset B$, where the open set B, illustrated in Figure 5.5, is defined as follows:

$$B = \bigcup_{1}^{n} R \backslash ((x_i + \mathbf{R}^2_+) \cup (x_i - \mathbf{R}^2_+)).$$

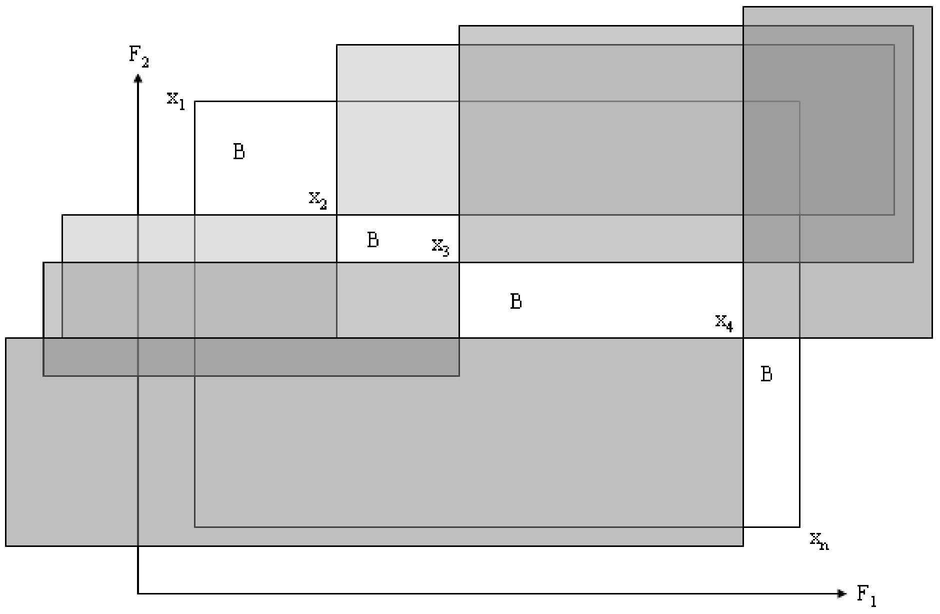


Figure 5.5: Illustration of the set B.

Hence, an upper bound for $\mathcal{H}(A, A^*)$ is given by

$$\sup\{\mathcal{H}(A, C),\ A \subset C \subset B,\ C \text{ compact}\},$$

which corresponds to the worst-case Hausdorff distance knowing that the Pareto optimal set A^* includes A, which is known, and is included in B, which can easily be built from A. As shown in Proposition 5.2, the problem of finding this worst-case distance amounts to searching for the point in B with the largest distance from A.

Proposition 5.2.

$$\sup\{\mathcal{H}(A,C),\ A \subset C \subset B,\ C \text{ compact}\} = \sup_{x\in B} d(x,A).$$

Proof. Let $x \in B$, the set $A \cup \{x\}$ is compact and satisfies $A \subset A \cup \{x\} \subset B$, hence

$$\mathcal{H}(A, A\cup\{x\}) \le \sup\{\mathcal{H}(A,C),\ A \subset C \subset B,\ C \text{ compact}\}.$$

Or,

$$\mathcal{H}(A, A\cup\{x\}) = d(x,A).$$

Therefore,

$$d(x,A) \le \sup\{\mathcal{H}(A,C),\ A \subset C \subset B,\ C \text{ compact}\}, \tag{5.5}$$

and finally,

$$\sup_{x\in B} d(x,A) \le \sup\{\mathcal{H}(A,C),\ A \subset C \subset B,\ C \text{ compact}\}.$$

Conversely, let C be a compact set such as $A \subset C \subset B$, as $A \subset C$, by definition of the Hausdorff distance, it must be that:

$$\mathcal{H}(A,C) = \sup_{x\in C} d(x,A).$$

As $C \subset B$,

$$\sup_{x\in C} d(x,A) \le \sup_{x\in B} d(x,A).$$

Hence,

$$\sup\{\mathcal{H}(A,C),\ A \subset C \subset B,\ C \text{ compact}\} \le \sup_{x\in B} d(x,A), \tag{5.6}$$

Combining (5.5) and (5.6) completes the proof. □

To find $\sup_{x \in B} d(x, A)$, assume that the points x_i have been ordered from the lowest to the highest x coordinate, and define the open rectangle R_i with upper left corner x_i and lower right corner x_{i+1}. Hence:

$$\sup_{x \in B} d(x, A) = \max_{i=1,\ldots,n-1} \sup_{x \in R_i} d(x, A). \tag{5.7}$$

Let $x \in R_i$, it is clear that

$$\sup_{x \in R_i} d(x, A) = \min\{d(x, x_i), d(x, x_{i+1})\}.$$

The solution of $\min\{d(x, x_i), d(x, x_{i+1})\}$ is given by the distance $d(x_i, y_i)$ as illustrated in Figure 5.6, where the point y_i is defined by the intersection of the median of the segment defined by the two points x_i and x_{i+1} with the rectangle R_i. A simple geometric argument gives

$$d(x_i, y_i) = \frac{d(x_i, x_{i+1})^2}{2L_i},$$

where L_i is the length of the rectangle R_i. Substituting in (5.7) yields,

$$\sup_{x \in B} d(x, A) = \max_{i=1,\ldots,n-1} \frac{d(x_i, x_{i+1})^2}{2L_i},$$

which, from Proposition 5.2, corresponds to an upper bound for $\mathcal{H}(A, A^*)$:

$$\mathcal{H}(A, A^*) \leq \max_{i=1,\ldots,n-1} \frac{d(x_i, x_{i+1})^2}{2L_i}.$$

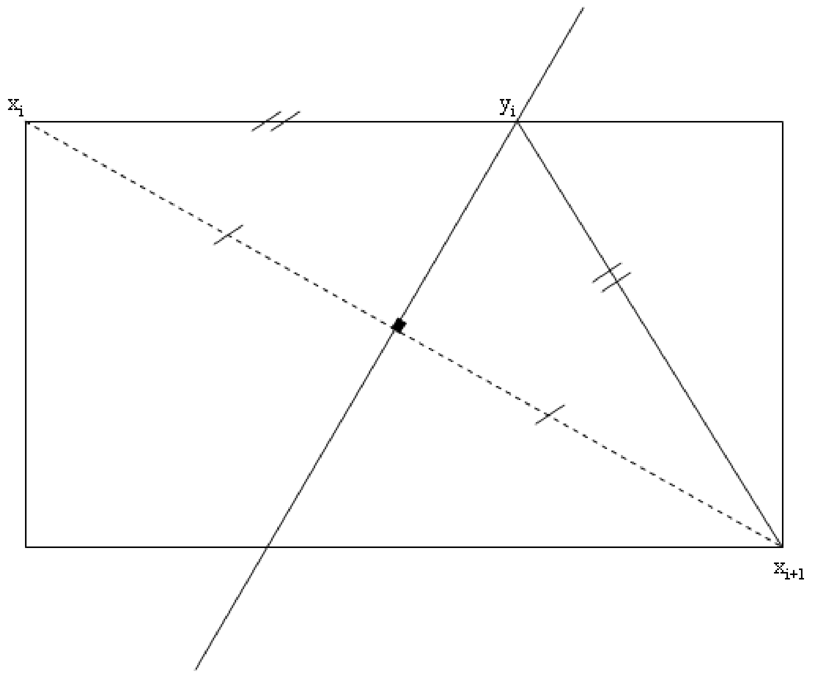


Figure 5.6: Geometric determination of y_i.

5.2.2 A Lower Bound for the Performance Measure

Let A_1 and A_2 be two subsets of A^*, as $A_1 \cup A_2 \subset A^*$, it follows that

$$\mathcal{H}(A_1, A_1 \cup A_2) \leq \mathcal{H}(A_1, A^*),$$

and

$$\mathcal{H}(A_2, A_1 \cup A_2) \leq \mathcal{H}(A_2, A^*).$$

Therefore, $\mathcal{H}(A_1, A_1 \cup A_2)$ and $\mathcal{H}(A_2, A_1 \cup A_2)$ can be seen as lower bounds for the Hausdorff distances $\mathcal{H}(A_1, A^*)$ and $\mathcal{H}(A_2, A^*)$.

5.2.3 Bounding the Performance Measure

Combining the upper and lower bounds obtained from Sections 5.2.1 and 5.2.2, we obtain:

$$\mathcal{H}(A_1, A_1 \cup A_2) \leq \mathcal{H}(A_1, A^*) \leq \max_{i=1,\ldots,|A_1|-1} \frac{d(x_i, x_{i+1})^2}{2L_i},$$

and

$$\mathcal{H}(A_2, A_1 \cup A_2) \leq \mathcal{H}(A_2, A^*) \leq \max_{i=1,\ldots,|A_2|-1} \frac{d(x_i, x_{i+1})^2}{2L_i},$$

where the sets of rectangles R_i have been determined for both A_1 and A_2 as described in Section 5.2.1.

5.2.4 Numerical Experiments

In this section, the bounds derived in Section 5.2.3 are applied to the Pareto objective vectors A_1 obtained with the weighting method and to the approximate Pareto objective vectors obtained with the DDP approximation, i.e., $A_2 = J_0^{h,d}(v_0)$. Strictly, as only approximate Pareto objective vectors are obtained with the DDP approximation method, A_2 is not a subset of the Pareto optimal set A^*, and the bounds do not apply. Nevertheless, these bounds are determined and their validity will be discussed later in this section. For clarity, the sets A_1 and A_2 are reproduced in Figure 5.7, which is identical to Figure 3.18. The rectangles R_i for determining the upper bounds for A_1 and A_2 are drawn in Figures 5.8(a) and 5.8(b) respectively.

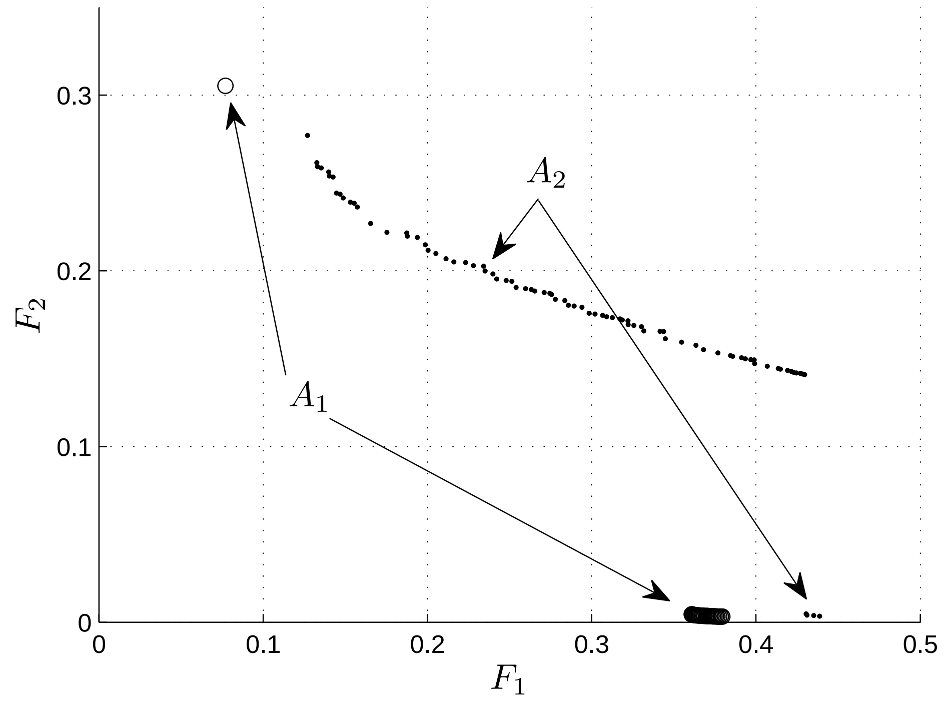
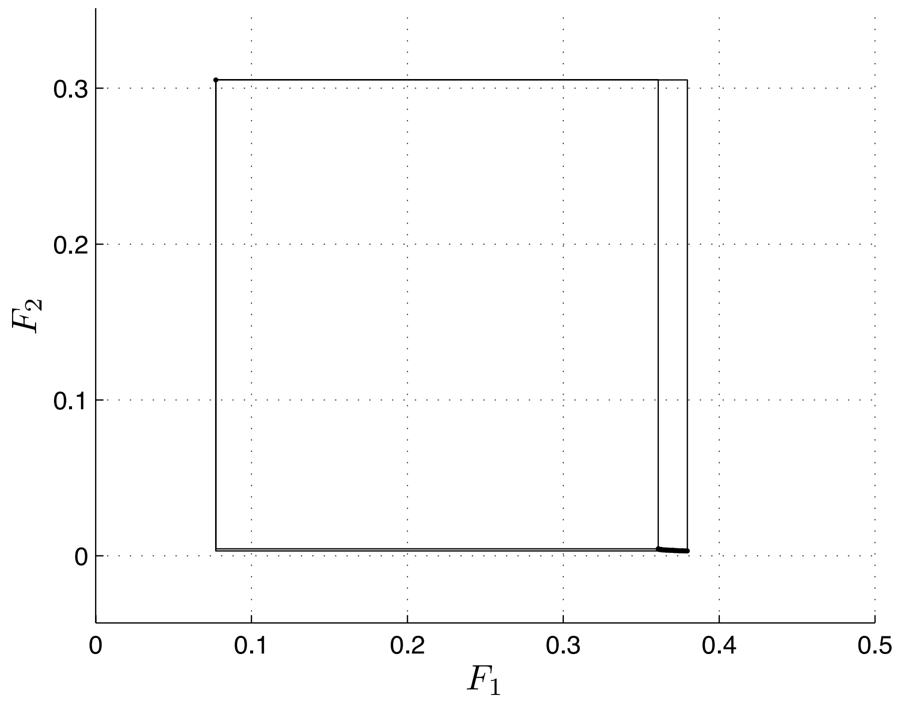
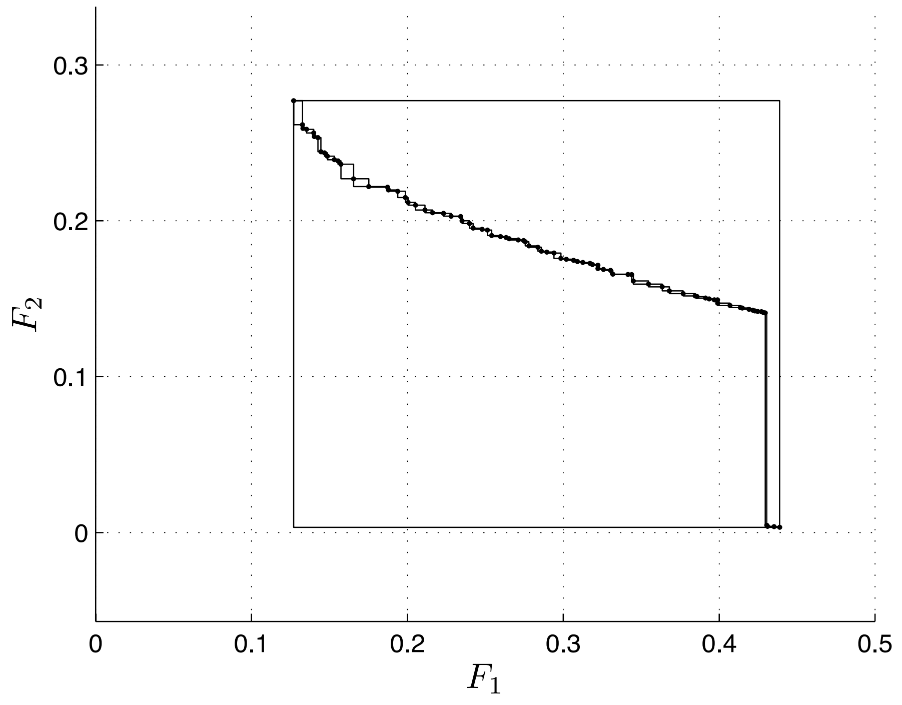


Figure 5.7: The set A_1 of Pareto objective vectors obtained with the weighting method (circles) and $A_2 = J_0^{h,d}(v_0)$ (dots).



(a) For the Pareto objective vectors A_1 obtained with the weighting method.



(b) For the approximate Pareto objective vectors $A_2 = J_0^{h,d}(v_0)$.

Figure 5.8: Building the rectangles to find an upper bound.

The resulting bounds are presented in Table 5.4. It can be concluded from Table 5.4 that the Hausdorff distance between $A_2 = J_0^{h,d}(v_0)$ and the Pareto optimal set is much lower than the Hausdorff distance between the set A_1 of Pareto objective vectors obtained with the weighting method and the Pareto optimal set, which indicates that the DDP approximation method provides a much better representation of the Pareto optimal set than the weighting method.

Table 5.4: Lower and upper bounds for the set A_1 of Pareto objective vectors obtained with the weighting method and $A_2 = J_0^{h,d}(v_0)$.

	Lower bound	Upper bound
A_1	0.21	0.28
A_2	0.069	0.068

It can be noted in Table 5.4 that for the DDP approximation method, the lower bound is slightly greater than the upper bound. This comes from the assumption made above that the approximate Pareto objective vectors are Pareto objective vectors. Therefore, it is proposed to perform another comparison not requiring that assumption. For this comparison, the weighting method is applied to the grid resulting from the DDP approximation. In this way, the weighting method also produces approximate Pareto objective vectors. As in Section 3.1.3, the weighting coefficient w is given the value $0, 1, \ldots, 30$. Only four different approximate Pareto objective vectors result, which are plotted together with $J_0^{h,d}(v_0)$ in Figure 5.9. The rectangles R_i to determine the upper bound for A_1 and A_2 are drawn in Figures 5.10(a) and 5.10(b) respectively.

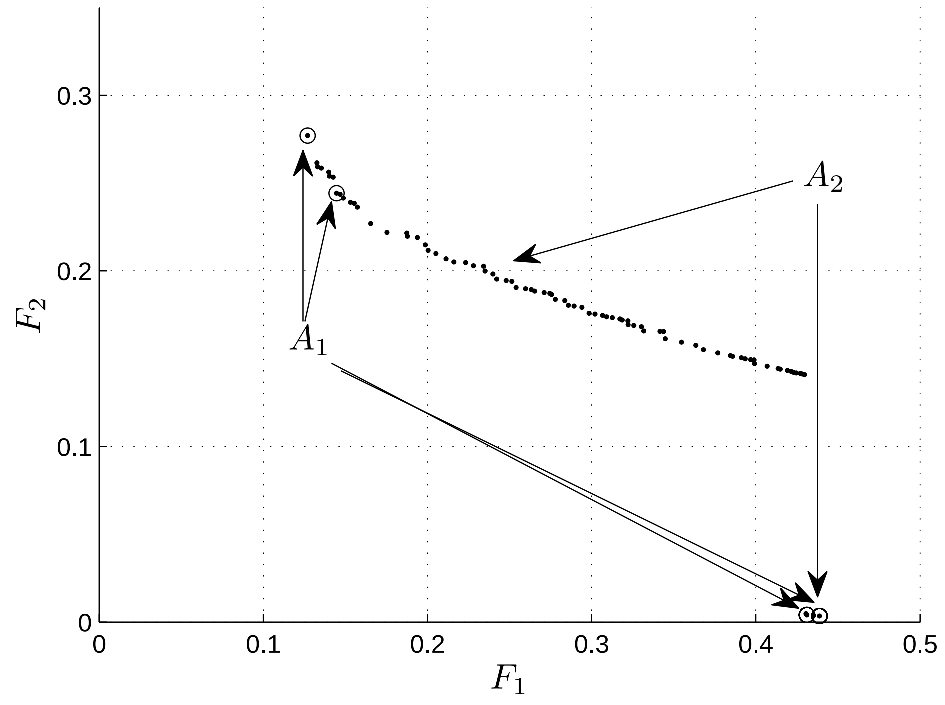
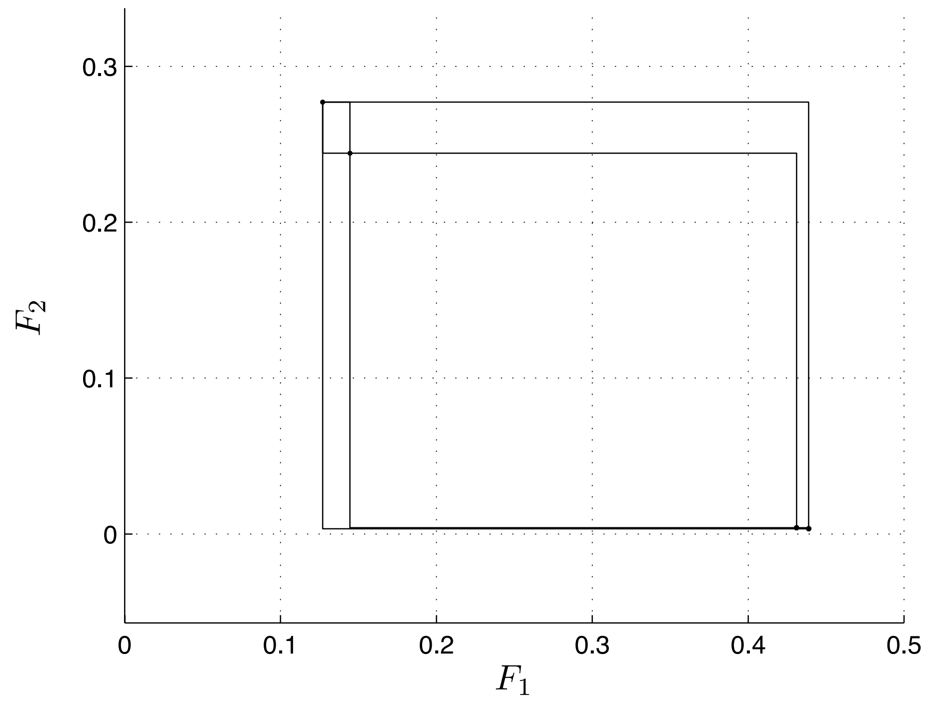
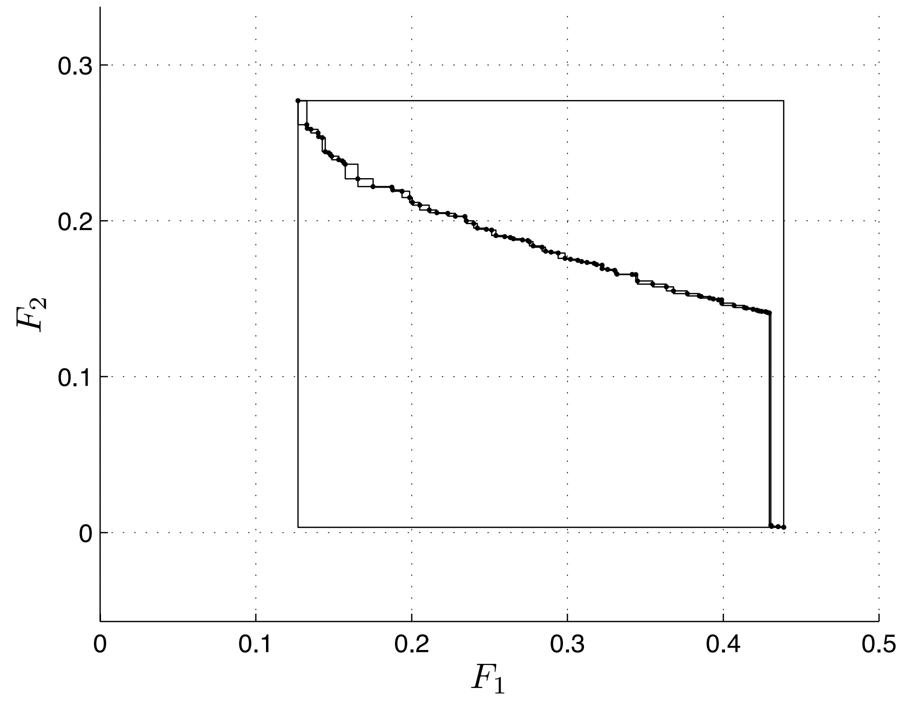


Figure 5.9: The set A_1 of approximate Pareto objective vectors obtained with the weighting method (circles) and $A_2 = J_0^{h,d}(v_0)$ (dots).



(a) For the approximate Pareto objective vectors A_1 obtained with the weighting method.



(b) For the approximate Pareto objective vectors $A_2 = J_0^{h,d}(v_0)$.

Figure 5.10: Building the rectangles to find an upper bound.

The lower and upper bounds are recalculated in Table 5.5. From Figure 5.9, as $A_1 \subset A_2$, it is normal that the lower bound for A_2 is zero. It can be observed that the lower and upper bounds obtained for A_1 in Table 5.5 are very similar to those obtained in Table 5.4, which indicates that our original assumption that the approximate Pareto objective vectors are Pareto objective vectors was reasonable. It can again be concluded that the DDP approximation method provides a much better representation of the Pareto optimal set than the weighting method.

Table 5.5: Lower and upper bounds for the set A_1 of approximate Pareto objective vectors obtained with the weighting method and $A_2 = J_0^{h,d}(v_0)$.

	Lower bound	Upper bound
A_1	0.19	0.24
A_2	0.0	0.068

5.2.5 Conclusion and Extensions

The lower bound that was derived in Section 5.2.2 remains valid for dimension two and above. However, the geometric construction for the derivation of the upper bound cannot be extended to more than two objective functions for the same reason mentioned in Section 5.1.4 for the clustering method based on the NBI method.

Returning to clustering and the choice of the clustering parameter N_C, it can be concluded from Figure 5.2(b) and the results from Tables 5.4 and 5.5 that even a small value for N_C still yields a better representation of the Pareto optimal set from the DDP approximation method. Indeed, the lower bound for the weighting method was found to be around 0.2 in Tables 5.4 and 5.5, whereas, for example, from Figure 5.2(b), the Hausdorff distance for M_7 and $N_C = 5$ is 0.14.

5.3 Generating Continuous Joint Trajectories

Together with the set of approximate Pareto objective vectors, the DDP approximation provides the corresponding discrete trajectories $\{v_i,\ i = 0, \ldots, N_T\}$, as introduced in Section 3.2.3. From these discrete trajectories, it is required to produce continuous trajectories $v(t)$ such that the joint configuration, joint speed, and joint acceleration can be known at any time $t \in [t_0, t_f]$. The common approach is to interpolate with cubic splines [63, pp. 9–17]. The main reason for such a choice is that, by construction, cubic spline interpolation guarantees that the first and second derivatives are continuous. Therefore, from the inverse kinematics equation (2.6), interpolating $\{v_i,\ i = 0, \ldots, N_T\}$ with cubic splines guarantees that the joint speed and joint acceleration will be continuous. The illustration for cubic spline interpolation will be provided in Section 6.6.

As mentioned in 3.1.2, it is desirable to have zero initial and final joint speeds. However, this was never taken into account as a constraint. The two degrees of freedom in cubic spline interpolation can be advantageously used to set the initial and final redundancy parameter first derivative to zero, which yields, from the inverse kinematics equation (2.7), zero initial and final joint speeds.

One drawback of the proposed DDP approximation method is that the constraints are only satisfied at $t_i,\ i = 0, \ldots, N_T$. At other times, it cannot be guaranteed a priori that the constraints will be satisfied due to the discretization. The joint trajectories have to be checked a posteriori against the constraints. This can be done by evaluating the joint configuration and joint speeds for a small time step. The possibility of generating joint trajectories not satisfying the constraints everywhere is aggravated by the fact that, to limit the grid size, the discretization in time should not be chosen to be too small. Nevertheless, first, as already mentioned, the joint speeds are expected to be small, therefore, the variation of the joint configuration is expected to be small between two instants t_i and t_{i+1}. Second, one advantage of the DDP approximation method has been shown to be its ability to produce a desired number N_C of approximate Pareto objective vectors that are well distributed in the objective space. As will be illustrated in Section 6.6, joint trajectories corresponding to these approximate Pareto objective vectors span the configuration space. Therefore, by applying a safety factor on the constraints, it is expected that few joint trajectories will be eliminated from the checking process.

Chapter 6

Resolution of the Problem in the Task Space

For clarity, the definition of PCTS_1 is recalled in Section 6.1, and additional notation is introduced. To obtain fast resolution times, an approximation algorithm is used to solve PCTS_1. To evaluate the performance of this algorithm, a lower bound on the number of store trajectories needed to complete all the subtasks in $\mathcal{S}$ is derived in Section 6.2. To ease the presentation of the algorithm presented in Section 6.4, standard definitions in graph theory are recalled in Section 6.3. The approximation algorithm, detailed in Section 6.4, is based on the resolution of a rural postman problem [37, 64]. Numerical experiments evaluating the performance of this algorithm are presented in Section 6.5. Finally, in Section 6.6, for a given grid survey experiment, PCTS is entirely solved using the developments of Chapters 3, 5, and 6.

6.1 Problem Definition

For clarity, the notation introduced in Section 2.3.1 is recalled. A subtask was defined as the task of moving the store from A_i with an angle of attack β_j to $A_{i'}$ with an angle of attack $\beta_{j'}$, and was represented by the quadruplet (i, j, i', j'). A grid survey experiment consists of completing the set of subtasks $\mathcal{S}$ such that $(A_i, A_{i'}) \in \mathcal{K}$, and $j' = j,\ \beta_j \in \mathcal{B}_{i,i'}$, where $\mathcal{K} = \{(A_i, A_{i'})\}$ is a set of segments, $\mathcal{A} = \{A_i,\ i = 1, \ldots, I\}$ a set of positions defined relative to the parent aircraft, and $\mathcal{B}_{i,i'}$ is a set of store angles of attack. Also,

$$\mathcal{B} = \bigcup_{(A_i, A_{i'}) \in \mathcal{K}} \mathcal{B}_{i,i'}.$$

Some additional notation is introduced now. Let $\mathcal{S}_1$ be the set of subtasks consisting of moving the store along a segment with a constant angle of attack, but that do not belong to $\mathcal{S}$. Subtasks in $\mathcal{S}_1$ are store motions between segments of the grid that are only required to connect segments of the grid together. For subtasks in $\mathcal{S}_1$, either the segment belongs to $\mathcal{K}$ and the angle of attack does not belong to $\mathcal{B}_{i,i'}$, which corresponds to $(A_i, A_{i'}) \in \mathcal{K}$, and $j' = j,\ \beta_j \notin \mathcal{B}_{i,i'},\ \beta_j \in \mathcal{B}$, or the segment does not belong to $\mathcal{K}$ which corresponds to $(A_i, A_{i'}) \notin \mathcal{K}$, and $j' = j,\ \beta_j \in \mathcal{B}$. In this latter case, the angle of attack can be any angle of attack in $\mathcal{B}$. Let $\mathcal{S}_2$ be the set of subtasks consisting of changing the angle of attack from β_j to $\beta_{j'}$ at a position A_i. Subtasks in $\mathcal{S}_2$ are store motions for which the position of the store is fixed but the store angle of attack is changed to another desired value in $\mathcal{B}$. Subtasks in $\mathcal{S}_2$ correspond to $i = i'$, $\beta_j \in \mathcal{B}$, and $\beta_{j'} \in \mathcal{B}$. For example, for the grid survey experiment presented in Figure 2.4, where $\mathcal{B}_{i,i'} = \mathcal{B} = \{0°, -3°, -6°, -9°, -12°, -15°\ -18°\}$,

- The set $\mathcal{S}$ consists of the subtasks $\{(1, j, 2, j), (1, j, 3, j), (1, j, 4, j),\ \beta_j \in \mathcal{B}\}$. Therefore, $|S| = 21$.

- The set $\mathcal{S}_1$ consists of the subtasks $\{(2, j, 3, j), (2, j, 4, j), (3, j, 4, j),\ \beta_j \in \mathcal{B}\}$. Therefore, $|S_1| = 21$.

- The set $\mathcal{S}_2$ consists of the subtasks $\{(1, j, 1, j'), (2, j, 2, j'), (3, j, 3, j'), (4, j, 4, j'),\ \beta_j \in \mathcal{B},\ \beta_{j'} \in \mathcal{B}\}$. Therefore, $|S_2| = 4(6 + 5 + 4 + 3 + 2 + 1) = 84$.

The trajectory for each subtask s in $\mathcal{S} \cup \mathcal{S}_1 \cup \mathcal{S}_2$ is built as discussed in Section 2.3.2. Therefore, the duration for each subtask trajectory is known and is denoted by t_s. Finally, recall from Section 2.3.2 that a store trajectory is obtained by arranging sequentially subtask trajectories.

In Section 2.3.1, PCTS$_1$ was defined as follows.

Problem PCTS$_1$: Find the minimum number of store trajectories such that all the subtasks in $\mathcal{S}$ are completed under the constraint that the duration of each store trajectory is less than the run time of the wind tunnel T_{tunnel}.

Recall from Section 2.1 that T_{tunnel} is typically 30 s. For safety reasons with respect to the large aerodynamic loading at the beginning of the wind-tunnel run, the additional constraint that the store trajectories all start at the "safe" position A_0 is imposed. Let $\mathcal{S}_0$ be the set of subtasks consisting of moving the store from the safe position A_0 with zero angle of attack to any position $A_{i'}$ with angle of attack $\beta_{j'} \in \mathcal{B}$. By convention, these subtasks will be given the value zero for i and j, and will correspond to $(0, 0, i', j')$. For example, for the grid survey experiment presented in Section 2.1.2, there are four positions and for each position, seven possible angles of attack, therefore $|S_0| = 28$. The trajectory for each subtask in $\mathcal{S}_0$ is also built as discussed in Section 2.3.2. Therefore, the duration for each subtask trajectory in $\mathcal{S}_0$ is also known.

For clarity, the assumptions made for the resolution of PCTS_1 are reviewed:

1. The duration of each subtask in $\mathcal{S} \cup \mathcal{S}_0 \cup \mathcal{S}_1 \cup \mathcal{S}_2$ is assumed to be known and less than T_{tunnel}.
2. When the wind-tunnel starts, the store is supposed to be located at the "safe" position A_0. Therefore, all the store trajectories start at A_0.
3. The store trajectories do not need to end at A_0. The store can be brought back to A_0 between two consecutive wind-tunnel runs.
4. No segment can be broken into two parts and performed in two different wind-tunnel runs.

6.2 A Lower Bound on the Number of Store Trajectories

The duration of a subtask in $\mathcal{S}$ is t_s, therefore the total duration needed to complete all the subtasks in $\mathcal{S}$ is simply

$$T_{\mathcal{S}} = \sum_{s=1}^{|S|} t_s.$$

Let t_0 be the minimum duration among all the subtasks in $\mathcal{S}_0$, N be the minimum number of store trajectories needed to complete $\mathcal{S}$, and T_{traj} be the total duration of these N trajectories. As each store trajectory must start at the safe position A_0, it takes at least t_0 for each store trajectory to get to the first position A_i, and the N store trajectories complete $\mathcal{S}$, it follows that

$$T_{\mathcal{S}} + Nt_0 \leq T_{\text{traj}}. \tag{6.1}$$

Moreover, the duration of each store trajectory is less than the run time of the wind tunnel. Hence,

$$T_{\text{traj}} \leq NT_{\text{tunnel}}. \tag{6.2}$$

Combining (6.1) and (6.2) yields

$$T_{\mathcal{S}} + Nt_0 \leq NT_{\text{tunnel}}. \tag{6.3}$$

Hence, the minimum number of store trajectories needed to complete $\mathcal{S}$ is bounded below by

$$N_{\text{lower}} = \left\lceil \frac{T_{\mathcal{S}}}{(T_{\text{tunnel}} - t_0)} \right\rceil \leq N. \tag{6.4}$$

The duration T_{lower} calculated from N_{lower} is therefore a lower bound for T_{traj}:

$$T_{\text{lower}} = T_{\mathcal{S}} + t_0 N_{\text{lower}} \leq T_{\text{traj}}.$$

The quantities N_{lower} and T_{lower} assume that no subtasks from neither S_1 nor S_2 need to be added to build the store trajectories and that all the store trajectories start by the subtask(s) in S_0 corresponding to t_0. Therefore, practically, it is expected that either N_{lower} is reached but T_{lower} is not, or N_{lower} is not reached. The first case corresponds to some subtasks that need to be added, and $\mathcal{S}$ and the added subtasks can still be completed with N_{lower} store trajectories. The second case corresponds to some subtasks that need to be added, but $\mathcal{S}$ and the added subtasks cannot be completed with only N_{lower} store trajectories.

6.3 Some Definitions in Graph Theory

Let $G = (V, E)$, where V is a set of vertices, and E a set of undirected edges, be an undirected graph.

- The *degree* [64, p. 3] of a vertex is the number of edges incident to this vertex.
- A *path* [64, p. 12] is a sequence of vertices such that each pair of consecutive vertices is connected by an edge.
- G is *connected* [64, p. 12] if there is a path connecting each pair of vertices.
- A *cycle* [64, p. 12] is a path whose initial and terminal vertices coincide.
- An *Eulerian cycle* is a cycle traversing each edge only once.
- An *Eulerian graph* is a connected graph for which all vertices have even degrees. It can be shown that a graph has an Eulerian cycle if and only if it is Eulerian [64, p. 312]. An Eulerian cycle can be determined very efficiently [64, p. 313].

6.4 An Approximation Algorithm Based on the Resolution of a Rural Postman Problem

From a given grid survey experiment, build the following undirected graph $G = (V, E)$.

- A vertex is defined for each position A_i and each angle of attack $\beta_j \in \mathcal{B}$. Therefore, $V = \{N_{i,j},\ i = 1, \ldots, I,\ j = 1, \ldots, J\}$, and $|V| = IJ$.
- An edge connects two vertices $N_{i,j}$ and $N_{i',j'}$ if and only if the subtask (i, j, i', j') belongs to $\mathcal{S} \cup \mathcal{S}_1 \cup \mathcal{S}_2$. Therefore, $E = \{(N_{i,j}, N_{i',j'}),\ (i, j, i', j') \in \mathcal{S} \cup \mathcal{S}_1 \cup \mathcal{S}_2\}$, and $|E| = |\mathcal{S} \cup \mathcal{S}_1 \cup \mathcal{S}_2|$. The cost for an edge is the duration t_s of the corresponding subtask.

Using subtasks from $\mathcal{S}$, $\mathcal{S}_1$, and $\mathcal{S}_2$, it is always possible to move the store from a position $A_{i'}$ with angle of attack β_j to the position A_i with angle of attack $\beta_{j'}$. Therefore, the graph G is connected. From the graph G, the undirected graph $G_0 = (V_0, E_0)$ is defined by adding the safe position A_0 and the subtasks in $\mathcal{S}_0$. Therefore:

- Let $N_{0,0}$ be the node corresponding to the safe position, then $V_0 = V \cup \{N_{0,0}\}$ and $|V_0| = |V| + 1 = IJ + 1$.
- From the definition of $\mathcal{S}_0$, an edge connects $N_{0,0}$ with each node of the graph G. Therefore, $E_0 = E \cup \{(N_{0,0}, N_{i,j}),\ i = 1, \ldots, I,\ j = 1, \ldots, J\}$, and $|E_0| = |E| + |\mathcal{S}_0| = |\mathcal{S} \cup \mathcal{S}_1 \cup \mathcal{S}_2| + IJ$. The cost for an edge is the duration t_s of the corresponding subtask.

Hence, the graph G_0 is such that to any path starting at $N_{0,0}$ corresponds a possible store trajectory, and conversely, to any store trajectory corresponds a path starting at $N_{0,0}$.

Assume for now that the store trajectories $\mathbf{p_n}(\cdot),\ \mathbf{n} = 1, \ldots, N$ are known. Remove the first subtask (which is in $\mathcal{S}_0$) for each store trajectory $\mathbf{p_n}(\cdot)$ and call the resulting trajectories $\mathbf{p'_n}(\cdot)$. To each $\mathbf{p'_n}(\cdot)$ corresponds a path in G. Consider two such paths, assume that the terminal vertex of the first path is $N_{i,j}$, and that the initial vertex of the second path is $N_{i',j'}$. Knowing that G is connected, there exists a path connecting $N_{i,j}$ to $N_{i',j'}$. Therefore, by combining these three paths together, a single path in G can be built. If this operation is repeated for each pair $(\mathbf{p'_n}(\cdot), \mathbf{p'_{n+1}}(\cdot)),\ \mathbf{n} = 1, \ldots, N-1$ and the pair $(\mathbf{p'_N}(\cdot), \mathbf{p'_1}(\cdot))$, a cycle in G can be built. Because the store trajectories $\mathbf{p_n}(\cdot),\ \mathbf{n} = 1, \ldots, N$ complete all the subtasks in $\mathcal{S}$, it follows that this cycle necessarily traverses every edge in G corresponding to the subtasks in $\mathcal{S}$. Note that the cycle could traverse some of these edges more than once. Therefore, to the store trajectories $\mathbf{p_n}(\cdot),\ \mathbf{n} = 1, \ldots, N$, corresponds a cycle in G traversing every edge in G corresponding to the subtasks in $\mathcal{S}$. The idea behind the proposed approximation algorithm to solve PCTS$_1$ follows from this observation. It is first proposed to find the minimum-time cycle in G that traverses every edge in G corresponding to the subtasks in $\mathcal{S}$, and second, to build the minimum number of store trajectories from this cycle such that all the subtasks in $\mathcal{S}$ are completed and the duration of each store trajectory is less than T_{tunnel}.

For an undirected graph $G = (V, E)$ and a subset R of E, assuming that each edge in E is given a cost, the problem of finding the minimum-cost cycle traversing at least once each edge in R is known in the literature as the *rural postman problem* [37, 64]. Therefore, the proposed approximation algorithm to solve PCTS_1 can be described as follows:

STEP 1: Solve the rural postman problem for the graph G built from a grid survey experiment as described above. The set R is the set of edges corresponding to the subtasks in $\mathcal{S}$.

STEP 2: From the cycle obtained in STEP 1, build the minimum number of paths in G_0 starting at A_0 with cost less than T_{tunnel} such all the subtasks in $\mathcal{S}$ are completed. To each of these paths corresponds a store trajectory with duration less than T_{tunnel}, thereby solving PCTS_1.

The number of paths obtained from this approximation algorithm gives an upper bound N_{upper} on the number of store trajectories required to complete S:

$$N_{\text{lower}} \leq N \leq N_{\text{upper}}.$$

If $N_{\text{upper}} = N_{\text{lower}}$, then it is possible to conclude that the approximation algorithm has provided the optimal solution, and $N = N_{\text{lower}} = N_{\text{upper}}$. Otherwise, nothing can be concluded and N can assume any integer value between N_{lower} and N_{upper}.

6.4.1 The Rural Postman Problem

The rural postman problem is a difficult problem to solve, which has been shown to be NP-complete [37]. Let V_R be the sets of vertices incident to an edge in R, and denote $G_R = (V_R, R)$ the resulting graph. There exists an exact recursive algorithm that is exponential in the number of disconnected components in G_R [37]. However, for the graphs G_R built from grid survey experiments, the number of disconnected components is not known in advance and depends entirely on the choice of the subtasks in $\mathcal{S}$. For example, for the grid survey experiment presented in Section 2.1.2, the number of disconnected components is seven. Therefore, to obtain fast resolution times, it is preferred to use the approximation algorithm proposed in [37] that is polynomial in the size of the graph. If G_R happens to be connected, then the solution provided by the approximation algorithm is in fact optimal [37]. On the other hand, if G_R is not connected, but the costs for G satisfy the triangular inequality, which is the case for the graphs built from grid survey experiments as described above, then the approximation algorithm provides a 3/2-approximation [37]. The idea behind the approximation algorithm is to build an Eulerian graph from G_R. The cycle obtained from the resulting graph will traverse, in particular, every edge in R.

STEP 1.1: Find the disconnected components in G_R [64, p. 17]. Build the graph H as follows. Each disconnected component in G_R defines a vertex for H. The cost between two vertices x and y in H is given by the shortest path between x and y in G. The minimum cost path between every two pairs of vertices in G can be efficiently determined [64, p. 52].

STEP 1.2: Find the minimum spanning tree for H [64, p. 126]. Let A be the subset of edges in G corresponding to this minimum spanning tree.

STEP 1.3: Consider the graph $G' = (V, R \cup A)$. The cost between two vertices x and y in G' is given by the shortest path between x and y in G. Find the minimum weight matching [64, p. 289] for the graph composed of the vertices of G' that have odd degree. Let M be the subset of edges corresponding to this matching.

STEP 1.4: Find an Eulerian cycle in the graph $G' = (V, R \cup A \cup M)$.

6.4.2 Generating Paths from the Solution to the Rural Postman Problem

From the Eulerian cycle obtained from STEP 1, we propose a greedy algorithm to generate paths starting from $N_{0,0}$ such that the set of all paths traverses R, and the duration of each path is less than T_{tunnel}. The idea behind this algorithm is to cut the cycle into paths such that, for each path, the duration of the path plus the duration of the subtask between $N_{0,0}$ and the first node of the path is less than T_{tunnel}. The proposed algorithm is greedy because all the different ways of cutting the cycle as just described are investigated, and the one providing the best solution in terms of number of paths is retained. Let

$$N_{i_1,j_1}, N_{i_2,j_2}, \ldots, N_{i_l,j_l}, \ldots, N_{i_1,j_1}$$

be the Eulerian cycle obtained from STEP 1. Assume that this cycle contains $L = |R \cup A \cup M|$ edges, and let $v_l,\ l = 1, \ldots, L-1$, be the edge connecting N_{i_l,j_l} and $N_{i_{l+1},j_{l+1}}$, and v_L be the last edge connecting N_{i_L,j_L} and N_{i_1,j_1}. By construction, $R \subset C$, where $C = \{v_l,\ l = 1, \ldots, L\}$. However, note that some edges in V and therefore in particular in R might be present more than once in C, which means that some subtasks in $\mathcal{S}$ might be completed twice. For the presentation of the greedy algorithm, it is convenient to associate a flag f_l to each edge $v_l \in C$, which is set to one if the edge belongs to R, and zero otherwise.

STEP 2.1: Start at index i_1. Find the first index i_l in C such that $f_l = 1$. Create the path $\{N_{0,0}, N_{i_l,j_l}\}$. Add vertices from C to the path until the duration of the path exceeds T_{tunnel}, or all the edges in C have been considered. Let $i_{l'}$ be the index of the last added vertex. Note that each time an edge v_l with a flag equal to one is added to the path, then the flag for all the edges in C corresponding to the same subtask as v_l is changed to zero. This is to avoid completing a subtask in $\mathcal{S}$ more than once if possible.

STEP 2.2: If all the edges in C have been considered, then we have obtained N_{i_1} paths starting from $N_{0,0}$ such that the set of all paths traverses R, and the duration of each path is less than T_{tunnel}. Go to STEP 2.3. Otherwise, repeat STEP 2.1 starting at index $i_{l'}$.

STEP 2.3: Repeat STEP 2.1 for all indices $i_l,\ i = 2, \ldots, L$. Finally,

$$N_{\text{upper}} = \min\{N_{i_l},\ l = 1, \ldots, L\}.$$

For a path generated in STEP 2.1, it is possible that the last edge has a flag equal to zero. In such a case, this edge can be removed from the path. This operation can be repeated until an edge with a flag equal to one is encountered (note that this will always happen). This "cleaning" procedure does not change the value of N_{i_l}, but avoids performing unnecessary subtasks. Finally, to each of the N_{upper} paths corresponds a store trajectory that can be built as discussed in Section 2.3.2.

6.5 Numerical Experiments

For the first set of numerical experiments, the approximation algorithm is applied to six grid survey experiments derived from the grid survey experiment presented in Section 2.1.2. First, only two angles of attack: $\mathcal{B}_{i,i'} = \{0°, -3°\}$ are considered; then three: $\mathcal{B}_{i,i'} = \{0°, -3°, -6°\}$; and so on until $\mathcal{B}_{i,i'} = \{0°, -3°, -6°, -9°, -12°, -15°, -18°\}$; which corresponds to the grid survey experiment presented in Section 2.1.2. The run time of the wind tunnel T_{tunnel} is assumed to be 30 s. Tables 6.1, 6.2, and 6.3 contain the durations for the subtasks in $\mathcal{S}$, $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_0$, which have been obtained as discussed in Section 2.3.2.

Table 6.1: Duration for the subtasks in $\mathcal{S}$ and in $\mathcal{S}_1$. The duration is the same regardless the store angle of attack.

	$N_{1,1}$	$N_{2,1}$	$N_{3,1}$	$N_{4,1}$
$N_{1,1}$	-	7.11	7.11	7.11
$N_{2,1}$	7.11	-	2.85	2.10
$N_{3,1}$	7.11	2.85	-	3.84
$N_{4,1}$	7.11	2.10	3.84	-

Table 6.2: Duration for the subtasks in $\mathcal{S}_2$. The duration is independent from the position, and only depends on the difference $|\Delta\beta|$ between the two store angles of attack.

$\|\Delta\beta\|$ (deg)	3	6	9	12	15	18
Duration (s)	2.61	4.11	5.61	7.11	8.61	10.11

Table 6.3: Duration for the subtasks in $\mathcal{S}_0$.

	$0°$	$-3°$	$-6°$	$-9°$	$-12°$	$-15°$	$-18°$
1	3.91	3.91	4.11	5.61	7.11	8.61	10.11
2	4.30	4.30	4.30	5.61	7.11	8.61	10.11
3	4.30	4.30	4.30	5.61	7.11	8.61	10.11
4	4.30	4.30	4.30	5.61	7.11	8.61	10.11

Table 6.4: Comparing the lower bound on the number of paths with the number of paths provided by the approximation algorithm.

$\lvert\mathcal{B}_{i,i'}\rvert$	T_{lower} (s)	N_{lower}	T_{upper} (s)	N_{upper}	Computation time (s)
2	50.49	2	55.09	2	2.71
3	75.74	3	85.15	4	3.18
4	100.99	4	111.68	4	3.12
5	126.24	5	148.61	6	3.99
6	147.58	5	183.99	7	3.93
7	172.83	6	221.35	9	5.89

The results for each of the six grid survey experiments are presented in Table 6.4. This table provides the total duration T_{lower} needed to complete S, the lower bound N_{lower} on the number of paths, the number of paths N_{upper} provided by the approximation algorithm, the total duration for these paths, denoted T_{upper}, and finally the computation time for the approximation algorithm. As explained in Section 6.4, if $N_{\text{upper}} = N_{\text{lower}}$, then it is possible to conclude that the approximation algorithm has provided the optimal solution, and $N = N_{\text{lower}} = N_{\text{upper}}$. Otherwise, nothing can be concluded and N can assume any integer value between N_{lower} and N_{upper}. Note that as the number $\lvert\mathcal{B}_{i,i'}\rvert$ of possible store angles of attack increases, the difference between N_{upper} and N_{lower} increases. This can be explained as follows. As $\lvert\mathcal{B}_{i,i'}\rvert$ increases, N_{lower} tends to underestimate the minimum number of required store trajectories. Indeed, for the calculation of N_{lower}, it is supposed that all the store trajectories start with the minimum-duration subtask t_0 in $\mathcal{S}_0$, which is 3.91 s from Table 6.3. However, in practice, the store trajectories might start with other subtasks in $\mathcal{S}_0$, whose duration could be much larger, as illustrated in Table 6.3.

For the second set of numerical experiments, N_{upper} as obtained with the approximation algorithm is compared with the minimum number of store trajectories N'_{upper} required to complete $\mathcal{S}$ with the articulated sting. As it is not possible to change the store angle

Table 6.5: Comparing the two upper bounds.

$\lvert\mathcal{B}_{i,i'}\rvert$	T'_{upper} (s)	N'_{upper}	T_{upper} (s)	N_{upper}
2	54.70	2	55.09	2
3	82.24	3	85.15	4
4	111.29	4	111.68	4
5	146.85	6	148.61	6
6	185.40	8	183.99	7
7	226.96	10	221.35	9

of attack during a wind-tunnel run with the articulated sting, N'_{upper} also constitutes an upper bound for N. However, for the sake of comparison, it will be assumed that, with the articulated sting, the angle of attack can change from the safe position to the first position of the store trajectory. With this assumption, N'_{upper} is straightforward to obtain. Indeed, from Tables 6.1 and 6.3, for each store angle of attack, the shortest-duration path is

$$N_{0,0}, N_{1,j}, N_{4,j}, N_{2,j}, N_{1,j}, N_{3,j}.$$

This follows from the facts that, from Table 6.3, the position 1 is always the closest in terms of duration from $N_{0,0}$, and that, from Table 6.1, the duration between $N_{2,j}$ and $N_{4,j}$ is less than between $N_{2,j}$ and $N_{3,j}$. However, the duration of the shortest-duration path might be greater than T_{tunnel}, which happens when $\beta_j \leq -12°$. In such a case, the two paths

$$N_{0,0}, N_{2,j}, N_{1,j}, N_{4,j} \text{ and } N_{0,0}, N_{1,j}, N_{3,j}$$

represent the best alternative. The total duration for the N'_{upper} paths is denoted T'_{upper}. N'_{upper} and T'_{upper} are provided for each of the six grid survey experiments in Table 6.5. Two conclusions can be drawn from Table 6.5. First, for grid survey experiments with a large number of subtasks in $\mathcal{S}$, i.e., $\lvert\mathcal{B}_{i,i'}\rvert$ large, N_{upper} is lower than N'_{upper}, which indicates that the proposed approximation algorithm provides a better solution. Second, for such experiments, as $N \leq N_{\text{upper}}$, it follows that $N < N'_{\text{upper}}$, which means that adding the possibility for the CTS system to change the angle of attack during a wind-tunnel run can allow, among other benefits, reducing the number of wind-tunnel runs needed to complete a grid survey experiment.

6.6 An Example of a Complete Resolution of PCTS

For clarity, the grid survey experiment presented in Section 2.1.2 is reproduced in Figure 6.1. However, for the grid survey experiment considered in this section, it will only be required to move the store along each segment (1,2), (1,3), and (1,4) with the store angles of attack $\{0°, -3°, -6°, -9°, -12°\}$. Therefore, $\mathcal{S}$ consists of the subtasks $\{(1, j, 2, j), (1, j, 3, j), (1, j, 4, j),\ \beta_j \in \mathcal{B}\}$, where $\mathcal{B} = \{0°, -3°, -6°, -9°, -12°\}$. The run time of the wind tunnel T_{tunnel} is assumed to be 30 s.

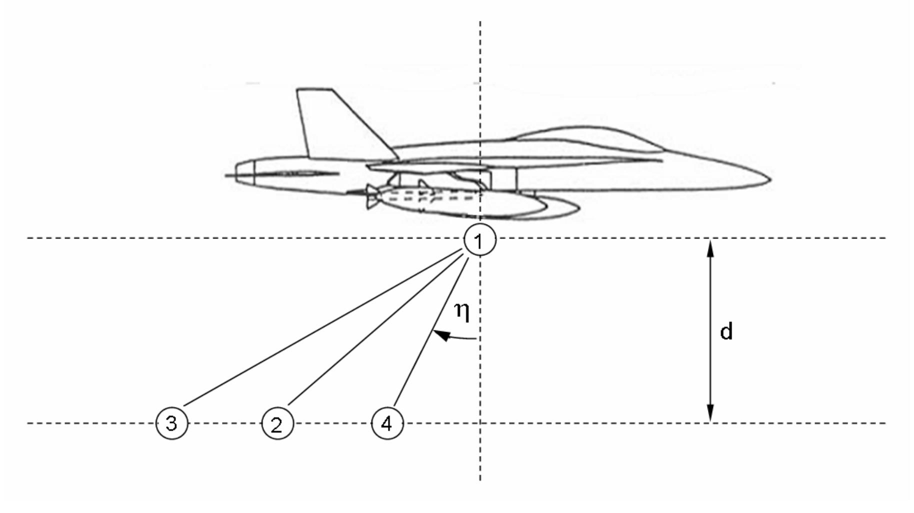


Figure 6.1: An example of a grid survey experiment.

The resolution of PCTS for the grid survey experiment considered above consists of solving sequentially the task space CTS trajectory planning problem PCTS_1 followed by the joint space CTS trajectory planning problem PCTS_2. The problem PCTS_1 for this grid survey experiment has already been solved in Section 6.5. From Table 6.4, it can be seen that, from the approximation algorithm described in Section 6.4, six store trajectories are needed to complete $\mathcal{S}$. In the graph G_0 as built in Section 6.4 from the grid survey experiment, the corresponding store paths are described as follows. For clarity, the index j for the store angle of attack is replaced by the actual absolute value of the angle.

- Store path 1:

$$N_{0,0}, N_{1,0^\circ}, N_{2,0^\circ}, N_{3,0^\circ}, N_{1,0^\circ}, N_{4,0^\circ},$$

- Store path 2:

$$N_{0,0}, N_{4,3^\circ}, N_{1,3^\circ}, N_{2,3^\circ},$$

- Store path 3:

$$N_{0,0}, N_{1,3^\circ}, N_{3,3^\circ}, N_{3,6^\circ}, N_{1,6^\circ}, N_{2,6^\circ},$$

- Store path 4:

$$N_{0,0}, N_{4,6^\circ}, N_{1,6^\circ}, N_{1,9^\circ}, N_{2,9^\circ},$$

- Store path 5:

$$N_{0,0}, N_{4,9^\circ}, N_{1,9^\circ}, N_{3,9^\circ}, N_{3,12^\circ}, N_{1,12^\circ},$$

- Store path 6:

$$N_{0,0}, N_{1,12^\circ}, N_{2,12^\circ}, N_{4,12^\circ}, N_{1,12^\circ}.$$

The trajectory along each of these paths can then be determined as described in Section 2.3.2. The resulting duration for these trajectories, provided in Table 6.6, can be easily retrieved from Tables 6.1, 6.2, and 6.3 and the description of the store paths given above. It can also be verified that the sum of these durations is 148.61 s, which corresponds

Table 6.6: The store trajectory durations.

Store path #	1	2	3	4	5	6
Duration (s)	28.10	18.52	27.86	21.13	29.55	23.43

Table 6.7: List of subtasks completed by the store trajectories.

Store path #	1	2	3
$\mathcal{S}$	$(1, 0°, 2, 0°)$ $(1, 0°, 3, 0°)$ $(1, 0°, 4, 0°)$	$(1, 3°, 2, 3°)$ $(1, 3°, 4, 3°)$	$(1, 3°, 3, 3°)$ $(1, 6°, 2, 6°)$ $(1, 6°, 3, 6°)$
$\mathcal{S}_1$	$(2, 0°, 3, 0°)$	-	-
$\mathcal{S}_2$	-	-	$(3, 3°, 3, 6°)$
Store path #	**4**	**5**	**6**
$\mathcal{S}$	$(1, 6°, 4, 6°)$ $(1, 9°, 2, 9°)$	$(1, 9°, 4, 9°)$ $(1, 9°, 3, 9°)$ $(1, 12°, 3, 12°)$	$(1, 12°, 2, 12°)$ $(1, 12°, 4, 12°)$
$\mathcal{S}_1$	-	-	$(2, 12°, 4, 12°)$
$\mathcal{S}_2$	$(1, 6°, 1, 9°)$	$(3, 9°, 3, 12°)$	-

to T_{upper} in Table 6.4. Note that each duration is less than T_{tunnel}.

The subtasks completed by the store trajectories are detailed in Table 6.7. It can be verified that all the subtasks in $\mathcal{S}$ are completed. It can also be noted that, to build the store paths, two subtasks from $\mathcal{S}_1$ and three subtasks from $\mathcal{S}_2$ have to be added.

Once the store trajectories are determined, the problem PCTS$_2$ can be solved for each of these trajectories. Consider for example the store path 3. This is an interesting example, because the duration of the corresponding trajectory is large, i.e., very close to T_{tunnel}, and this path involves a change in orientation through the subtask $(3, 3°, 3, 6°)$. For this example, the values for N_T and N_X are set to 56 and 290 respectively, which yields a grid size of 5463 nodes. The clustering method chosen is M_7 with $N_C = 20$.

Finally, the maximum number of modes is set to two. First, the grid resulting from the discretization in the time and redundancy parameter variables is represented in Figure 6.2.

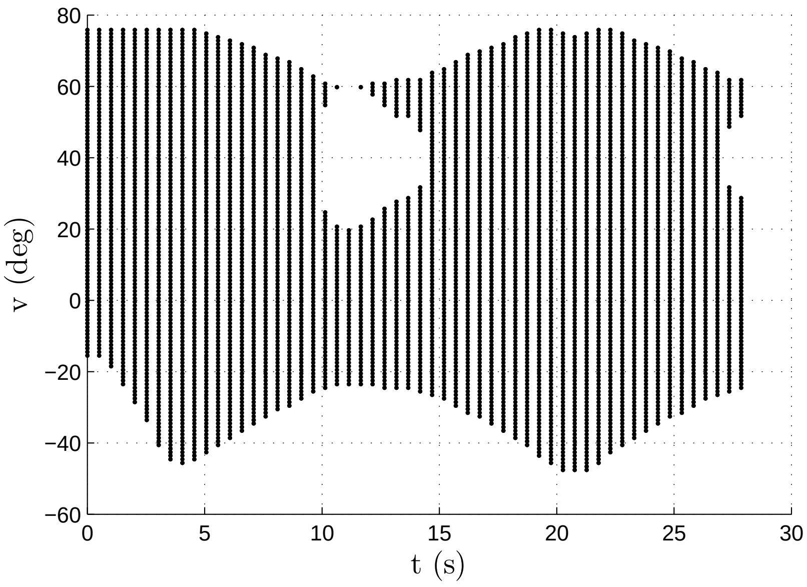


Figure 6.2: The grid resulting from the discretization.

Figure 6.3 confirms the fact, already noted in Section 3.3.3, that the approximate Pareto objective vectors tend to be evenly distributed. It also interesting to observe from Figure 6.4 that the optimal discrete trajectories are also distributed in the configuration space. For clarity, only the boundary of the grid is represented in this figure. Note that, in Figure 6.3, the number of approximate Pareto optimal objective vectors is 18, whereas N_C was set to 20. This difference can be explained by the fact that two optimal discrete trajectories were eliminated by the checking process, as discussed in Section 5.3. Finally, the optimal discrete trajectories and their corresponding continuous joint trajectories obtained as described in Section 5.3 are represented in Figure 6.5.

The resolution time for PCTS_2 is 523 s, which added to the 4 s needed to solve PCTS_1

as indicated in Table 6.4, yields a total resolution time of 527 s for PCTS. This resolution time satisfies to a large extent the requirement to solve PCTS in less time than the time between two consecutive wind-tunnel runs, which, as discussed in Section 2.1, is typically 30 min.

As mentioned above, the approximate Pareto objective vectors obtained from the resolution of PCTS$_2$, represented in Figure 6.3, are well distributed and cover a wide range of values for the two objective functions. This is very useful for the application. Indeed, if the store trajectory for PCTS$_2$ has to be performed at large Mach numbers, then it is desirable to have a joint trajectory with low joint speeds. Therefore, a joint trajectory corresponding to an approximate Pareto objective vector with a low value for F_1 can be chosen for the final joint trajectory. On the other hand, if the store trajectory for PCTS$_2$ has to be performed at low Mach numbers, then having low joint speeds for the joint trajectory is less critical. Therefore, a joint trajectory corresponding to an approximate Pareto objective vector with a low value for F_2 can be chosen for the final joint trajectory, which will increase the reliability of the measurement of the aerodynamic loads acting on the store model.

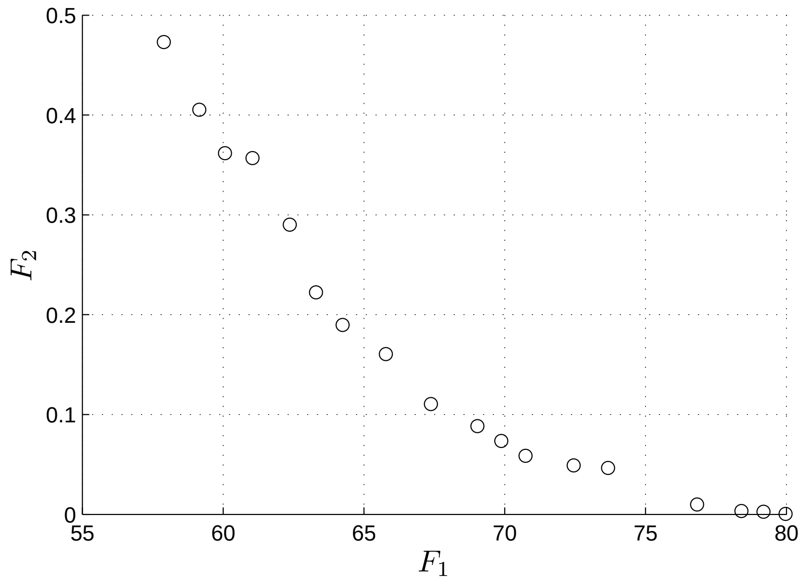


Figure 6.3: Approximate Pareto objective vectors obtained from the resolution of the approximate dynamic programming equation with the clustering method M_7 and $N_C = 20$, and the maximum number of modes $m = 2$.

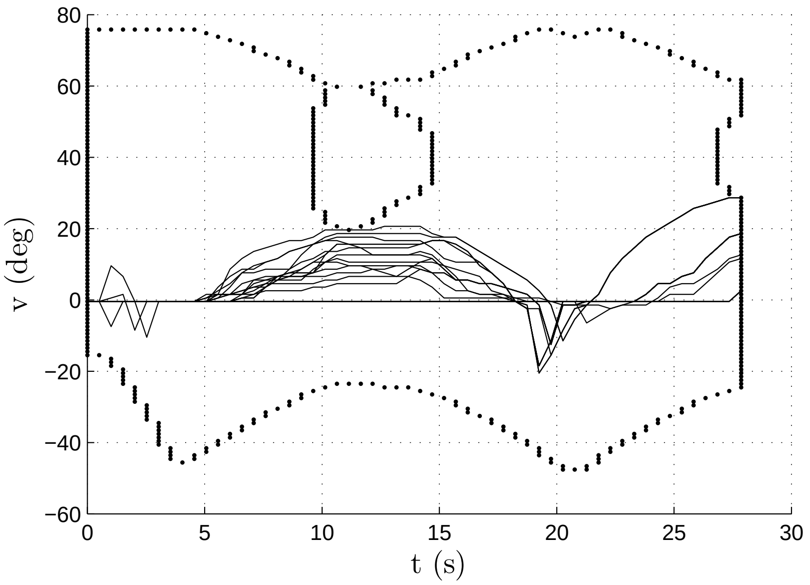


Figure 6.4: The optimal discrete trajectories (plain lines) corresponding to the approximate Pareto objective vectors from Figure 6.3 and the boundary of the grid (dots).

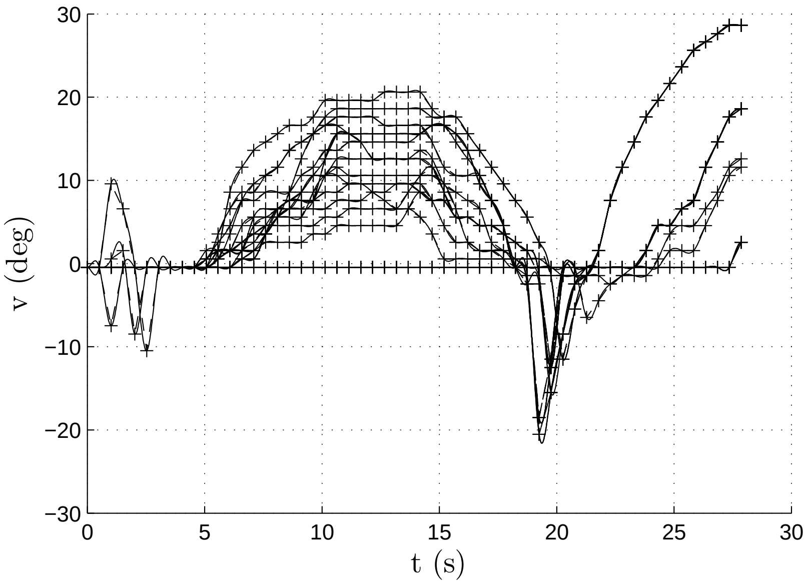


Figure 6.5: The optimal discrete trajectories (plus signs) and the interpolated continuous joint trajectories (plain lines) corresponding to the approximate Pareto objective vectors from Figure 6.3.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this thesis, a new DDP approximation method for the resolution of a general class of multiobjective optimal control problems where the objective space is partially ordered by a closed cone was proposed. The partial convergence results that were obtained using set convergence in the sense of Hausdorff show that this method provides a convergent approximate minimal element set. Therefore, by reducing both the discretization step in the time and the state space variables, a better approximation of the minimal element set of the original problem can always be obtained.

The joint space CTS trajectory planning problem, which is a typical multiobjective trajectory planning problem, was identified as a particular case of this class of multiobjective optimal control problems. The most common approach to solve multiobjective trajectory planning problems is the weighting method. Therefore, the joint space CTS trajectory planning problem was solved with the weighting method and the results were compared to the results obtained with the DDP approximation method. It was shown, using a measure derived from the Hausdorff distance, that the approximate Pareto optimal set provided a superior representation of the Pareto optimal set than the set of Pareto objective vectors obtained with the weighting method. The results obtained with the weighting method also illustrated the well-known weaknesses of this method.

The main weakness associated with the proposed DDP approximation method is its exponential algorithmic complexity, which is a limiting factor for its applicability. To address this issue, the idea of clustering was therefore introduced. With clustering, the algorithmic complexity of the DDP approximation method was shown to reduce to polynomial, and more importantly, without an excessive deterioration of the quality of the approximation. Again, using the measure derived from the Hausdorff distance, it was shown that the approximate Pareto optimal set obtained with clustering provided a superior representation of the Pareto optimal set than the set of Pareto objective vectors obtained with the weighting method.

The trajectory planning problem arising from a grid survey experiment in a CTS system was identified and formulated. For the operation of the CTS system, it is crucial to be able to solve this problem in less than the time between two consecutive wind-tunnel runs, i.e., 30 min. A two-step sequential resolution approach was therefore proposed. The first step consists of the resolution of the task space CTS trajectory planning problem. For this problem, an algorithm based on the resolution of a rural postman problem was developed. The resolution time for this algorithm was shown to be very fast, i.e., 3 s. The second step consists of the resolution of the joint space CTS trajectory planning problem, which takes the store trajectories obtained from the resolution of the task space CTS trajectory planning problem as inputs. As discussed above, with clustering, fast resolution time can also be obtained for the resolution of the joint space CTS trajectory planning problem. Therefore, the requirement to solve the CTS trajectory planning problem in less time than the time between two consecutive wind-tunnel runs was largely satisfied. Finally, the resolution of the CTS trajectory planning problem for a typical grid survey experiment confirmed, as expected, that the upgrade from an articulated sting to an 8-DOF manipulator allowed for a reduction in the number of wind-tunnel runs needed to complete a grid survey experiment, and therefore, in the operating cost of a grid survey experiment.

7.2 Future Work

7.2.1 The Trajectory Planning Problem for a Captive Load Experiment

As mentioned in Section 2.1.2, a captive load experiment is another type of experiment that can be performed with a CTS system. The constraints and the objective functions for this problem are the same as for the trajectory planning problem arising in a grid survey experiment. However, the fundamental difference is that the task, i.e., the store trajectory to be followed, is not known in advance, but is built as the task progresses. Therefore, the trajectory planning problem arising in a captive load experiment is a *local trajectory planning problem* [65, 66], whose resolution produces a sequence of local joint motions. Local multiobjective trajectory planning problems [67, 68] have received much more attention than global multiobjective local trajectory planning problems as discussed in this thesis. However, the main difficulty for local trajectory planning problems remains properly handling the constraints, particularly the constraints depending only on the joint configuration such as the joint mechanical limits and the obstacles. This is very critical for the CTS manipulator as, as illustrated in Figure 6.2, these constraints impose severe restrictions on the allowable values for the redundancy parameter. For example, if the store trajectory yielding Figure 6.2 were to be followed, the resolution of the local trajectory planning problem could yield a constant value for the redundancy parameter, say 40 deg. With this value, locally, all the constraints would be satisfied. However, at approximately $t = 10$ s, it would become impossible to satisfy either the joint mechanical limit or the obstacle constraints, and the CTS system would have to be stopped. This is very undesirable considering the operating cost of the wind tunnel, and knowing that there exists a choice of the redundancy parameter that would have allowed following the store trajectory.

Various generic approaches [69, 70, 71] have been proposed in the literature to address the difficulty of handling the joint mechanical limits or the obstacle constraints in the context of local trajectory planning problems. However, because of the local nature of these problems, none of these approaches can completely remedy this difficulty. The most promising approach seems to be a manipulator dependent approach that uses the knowledge of the topology of the joint configuration space [42, 72]. One might object that the determination of the topology of the joint configuration space, even for a simple 3-DOF planar redundant manipulator, is in general a formidable task. However, this approach can be reasonably conceived for the CTS manipulator because of the existence of a closed-form solution to the inverse kinematics.

7.2.2 Completing the Convergence Proof for the DDP Approximation Method

In Chapter 4, only partial convergence results of the approximate minimal element set towards the original minimal element set were obtained. It would be interesting to extend the convergence proof for the case of a single-valued objective function [22]. For this proof, first, the HJB equation was derived. Second, the return function was shown to be the unique viscosity solution of the HJB equation. Finally, the approximate return function was shown to converge towards a viscosity solution of the HJB equation, and therefore by uniqueness towards the return function.

7.2.3 Extension to the DDP Approximation Method

In Chapter 4, it was mentioned that the proposed DDP approximation could readily be extended to take into account terminal constraints, terminal costs, and nonautonomous systems. However, for the DDP approximation method to be suitable to a wide range of engineering problems, more complex constraints than just simple control constraints must be considered, e.g., state-dependent control constraints which are constraints where the control depends on the state. A nice way to express state-dependent control constraints is through the use of differential inclusions [14, pp. 37–38]:

$$\dot{\mathbf{x}}(t) \in F(t, \mathbf{x}(t)),$$

where $F(\cdot, \cdot)$ is a set-valued function. The difficulty then becomes carrying the discretization both in the time and state space variables in this context.

Two other important extensions could also be considered. The first extension would be to consider variable final time. This extension would allow using the final time of the joint trajectory as an objective function, which is very common in trajectory planning problems [13]. The second extension would be to consider **infinite** horizon problems. Here, the major difficulty would be to solve the multiobjective dynamic programming equation.

Bibliography

[1] F. Hausdorff. *Set Theory.* New York, Chelsea Pub. Co., 1962.

[2] J. J. Craig. *Introduction to Robotics, Mechanics and Control.* 3rd ed., Pearson Prentice Hall, Upper Saddle River, NJ, 2005.

[3] E. S. Conkur and R. Buckingham. Clarifying the definition of redundancy as used in robotics. *Robotica*, 15:583–586, 1997.

[4] M. Ahmadi, M. Jaber, and F. C. Tang. Real-time multi-body collision detection for a captive trajectory simulation system. *Mechatronics and Robotics Conference, Aachen, Germany*, pages 720–725, September 2004.

[5] M. Van der Steen, D. M. Orchard, and F. C. Tang. The application of semi-empirical store release simulation for the optimization of grid survey wind tunnel testing. *International Congress of the Aeronautical Sciences*, pages 1–8, 2008.

[6] M. Ahmadi, A. Guigue, M. Gibeault, and F. C. Tang. Mechatronic design of redundant robotic systems for captive trajectory trajectory simulation applications. *IASTED Conference on Control Applications, Quebec City, Canada*, pages 24–29, May 2008.

[7] A. Guigue, M. Ahmadi, M. J. D. Hayes, R. G. Langlois, and F. C. Tang. A dynamic programming approach to redundancy resolution with multiple criteria. *IEEE International Conference on Robotics and Automation*, pages 1375–1380, 2007.

[8] L.-Y. Jiang. Advances in aircraft/store separation methodologies, LTR-A-001. Technical report, Institute for Aerospace Research, National Research Council Canada, October 1995.

[9] M. Zefran. *Continuous Methods for Motion Planning.* PhD thesis, University of Pennsylvania, 1996.

[10] Y. Shen and K. Huper. Optimal trajectory planning of manipulators subject to motion constraints. *International Conference on Advanced Robotics*, pages 9–16, 2005.

[11] M. Galicki. The planning of robotic optimal motions in the presence of obstacles. *The International Journal of Robotics Research*, 17(3):248–259, 1998.

[12] K. Cleary and D. Tesar. Incorporating multiple criteria in the operation of redundant manipulators. *IEEE International Conference on Robotics and Automation*, pages 618–624, 1990.

[13] A. J. Cahill, M. R. James, J. C. Kieffer, and D. Williamson. Remarks on the application of dynamic programming to the optimal path timing of robot manipulators. *Internat. J. Robust Nonlinear Control*, 8:463–482, 1998.

[14] R. Vinter. *Optimal Control.* Birkauser, Boston, 2000.

[15] W. H. Fleming and H. Mete Soner. *Controlled Markov Processes and Viscosity Solutions.* 2nd ed., Springer-Verlag, New York, 2006.

[16] B. Dacorogna. *Introduction to the Calculus of Variations.* Imperial College Press, London, 2004.

[17] Y. Nakamura and H. Hanafusa. Optimal redundancy control of redundant manipulators. *The International Journal of Robotics Research*, 6(1):32–42, 1987.

[18] D. P. Martin, J. Baillieul, and J. M. Hollerbach. Resolution of kinematic redundancy using optimization techniques. *IEEE Transactions on Robotics and Automation*, 5(4):529–533, August 1989.

[19] K. G. Shin and N. D. McKay. A dynamic programming approach to trajectory planning of robotic manipulators. *IEEE Transactions on Automatic Control*, AC-31(6):491–500, June 1986.

[20] J. Gregory and C. Lin. *Constrained optimization in the calculus of variations and optimal control theory.* Van Nostrand Reinhold, New York, 1992.

[21] H. J. Kushner and P. G. Dupuis. *Numerical Methods for Stochastic Control Problems in Continuous Time.* Springer-Verlag, New York, 1992.

[22] M. Bardi and I. Capuzzo-Dolcetta. *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equations.* Birkhauser, Boston, 1997.

[23] K. D. Malanowski. Convergence of the lagrange-newton method for optimal control problems. *Int. J. Appl. Math. Comput. Sci.*, 14(4):531–540, 2004.

[24] R. Pytlak. *Numerical Methods for Optimal Control Problems with State Constraints.* Springer-Verlag, Berlin, 1999.

[25] K. M. Miettinen. *Nonlinear Multiobjective Optimization.* Kluwer Academic Publishers, Boston, 1999.

[26] Y. Sawaragi, H. Nakayama, and T. Tanino. *Theory of Multiobjective Optimization.* Academic Press, Inc., Orlando, FL, 1985.

[27] D. T. Luc. *Theory of Vector Optimization.* Springer-Verlag, New York, 1989.

[28] I. Das and J. E. Dennis. A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems. *Structural Optimization*, 14:63–69, September 1997.

[29] R. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, NJ, 1957.

[30] A. E. Bryson and Y.-C. Ho. *Applied Optimal Control: optimization, estimation and control.* Taylor & Francis, Levittown, PA, 1975.

[31] A. Guigue, M. Ahmadi, M. J. D. Hayes, and R. G. Langlois. A discrete dynamic programming approximation to the multiobjective deterministic finite horizon optimal control problem. *SIAM J. Control Optim.*, 48:2581–2599, 2009.

[32] D. S. Hochbaum and D. B. Shmoys. A best possible heuristic for the k-center problem. *Math. Oper. Res.*, 10(2):180–184, May 1985.

[33] T. Feder and D. H. Greene. Optimal algorithms for approximate clustering. *Twentieth Annual ACM Symposium on the Theory of Computation*, pages 434–444, 1988.

[34] D. S. Hochbaum. *Approximation algorithms for NP-hard problems.* PWS Publishing Co., Boston, 1997.

[35] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization.* John Wiley & Sons, Inc., New York, 1988.

[36] T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoret. Comput. Sci.*, 38:293–306, 1985.

[37] G. N. Frederickson. Approximation algorithms for some postman problems. *J. ACM*, 26:538–554, 1979.

[38] National Research Council Canada. *NRC Aerospace: 1.5 m Trisonic Blowdown Wind Tunnel, http://iar-ira.nrc-cnrc.gc.ca.*

[39] A. Guigue, M. Ahmadi, and F. C. Tang. On the kinematic analysis and design of a redundant manipulator for a captive trajectory simulation system (CTS). *IEEE Canadian Conference on Electrical and Computer Engineering*, pages 1513–1516, May 2006.

[40] L. Sciavicco and B. Siciliano. *Modeling and Control of Robot Manipulators.* McGraw-Hill, New York, 1996.

[41] Bombardier Aerospace. *Aircraft/Store Separation Course*, 1998.

[42] J. W. Burdick. *Kinematic Analysis and Design of Redundant Robot Manipulators.* PhD thesis, Stanford University, 1988.

[43] D. R. Baker and C. W. Wampler. On the inverse kinematics of redundant manipulators. *The International Journal of Robotic Research*, 7(2):3–21, March/April 1988.

[44] Waterloo Maple Inc. *Maple User Manual*, 1996-2009.

[45] P. L. Yu. Cone convexity, cone extreme points, and nondominated solutions in decision problems with multiobjectives. *J. Optim. Theory Appl.*, 14(3):319–377, 1974.

[46] W. Cheney. *Analysis for Applied Mathematics.* Springer-Verlag, New York, 2001.

[47] H. W. Corley. An existence result for maximizations with respect to cones. *J. Optim. Theory Appl.*, 31(2):277–281, June 1980.

[48] R. Hartley. On cone-efficiency, cone-convexity and cone-compactness. *SIAM J. Appl. Math.*, 34(2):211–222, March 1978.

[49] S. Singh and M. Leu. Manipulator motion planning in the presence of obstacles and dynamic constraints. *The International Journal of Robotics Research*, 10(2):171–187, 1991.

[50] A. Guigue, M. Ahmadi, R. G. Langlois, and M. J. D. Hayes. Generating the pareto optimal set for multiobjective trajectory planning problems using dynamic programming. *Submitted to IEEE Transactions on Robotics*, 2009.

[51] L. F. Shampine, M. W. Reichelt, and J. Kierzenka. *Solving Boundary Value Problems for Ordinary Differential Equations in MATLAB with bvp4c.*

[52] F. H. Clarke. *Optimization and Nonsmooth Analysis.* John Wiley & Sons, Inc., New York, 1983.

[53] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear programming: theory and algorithms.* 2nd ed., John Wiley & Sons, Inc., New York, 1993.

[54] W. H. Fleming and R. W. Rishel. *Deterministic and Stochastic Optimal Control.* Springer-Verlag, New York, 1975.

[55] J.-P. Aubin and H. Frankowska. *Set-Valued Analysis.* Birkhauser, Boston, 1990.

[56] R. Bellman. Functional equations in the theory of dynamic programming-IV, a direct convergence proof. *Annals of Mathematics*, 65(2):215–223, March 1957.

[57] G. S. Jones. Fundamental inequalities for discrete and discontinuous functional equations. *J. Soc. Indust. Appl. Math.*, 12(1):215–223, March 1964.

[58] R. Gonzalez and E. Rofman. On deterministic control problems: An approximation procedure for the optimal cost II the nonstationary case. *SIAM J. Control Optim.*, 23(2):264–285, March 1985.

[59] N. D. McKay. Minimum-cost control of robotic manipulators with geometric path constraints, robot syst. division tech. rep. RSD-TR-16-85. Technical report, Center for Research on Integrated Manufacturing, Univ. Michigan, October 1985.

[60] P. K. Agarwal and C. M. Procopiuc. Exact and approximation algorithms for clustering. *Algorithmica*, 33:201–226, 2002.

[61] I. Das and J. E. Dennis. Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems. *SIAM J. Optim.*, 8:631–657, 1998.

[62] Y. Collette and P. Siarry. *Optimisation Multiobjectif*. Eyrolles, 2002.

[63] R. H. Bartels, J. C. Beatty, and B. A. Barsky. *An Introduction to Splines for Use in Computer Graphics and Geometric Modelling*. Morgan Kaufmann, San Francisco, 1998.

[64] M. Gonfran and M. Minoux. *Graphes et algorithmes*. 3e éd., Eyrolles, 1995.

[65] D. N. Nenchev. Redundancy resolution through local optimization: A review. *Journal of Robotic Systems*, 6(6):769–798, 1989.

[66] B. Siciliano. Kinematic control of redundant robot manipulators: A tutorial. *Journal of Intelligent and Robotic Systems*, 3:201–212, 1990.

[67] F.-T. Cheng, M.-S. Shih, F.-C. Kung, and Y.-Y. Sun. The improved parallel scheme for multiple-goal priority considerations of redundant manipulators. *IEEE International Conference on Robotics and Automation*, pages 2409–2414, April 1997.

[68] C. Pholsiri, C. Kapoor, and D. Tesar. Manipulator task-based performance optimization. *ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, September 2004.

[69] T. F. Chan and R. V. Dubey. A weighted least-norm solution based scheme for avoiding joint limits for redundant joint manipulators. *IEEE Transactions on Robotics and Automation*, 11(2):286–292, April 1995.

[70] B. Allotta, V. Colla, and G. Bioli. Kinematic control of robots with joint constraints. *Journal of Dynamic Systems, Measurement and Control*, 121:433–442, 1999.

[71] L. Zlajpah and B. Nemec. Kinematic control algorithms for on-line obstacle avoidance for redundant manipulators. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1898–1903, October 2002.

[72] C. L. Luck and S. Lee. Global path planning of redundant manipulators based on self-motion topology. *IEEE International Conference on Robotics and Automation*, pages 372–377, 1994.

Appendix A

DH Parameters

Table A.1: Set of DH parameters for the 8-DOF CTS manipulator.

Joint	α_i (deg)	a_i (inch)	θ_i (deg)	d_i (inch)
1	0	0	0	$\mathbf{q}_1$
2	-90	0	$\mathbf{q}_2$	0
3	0	0	0	$\mathbf{q}_3 + \mathbf{q}_{3,0}$
4	90	0	0	$\mathbf{q}_4 + \mathbf{q}_{4,0}$
5	90	0	$\mathbf{q}_5$	l_{45}
6	-90	0	$\mathbf{q}_6 + 90$	h_{56}
7	90	h_{67}	$\mathbf{q}_7 + 90$	0
8	0	0	$\mathbf{q}_8$	l_{78}

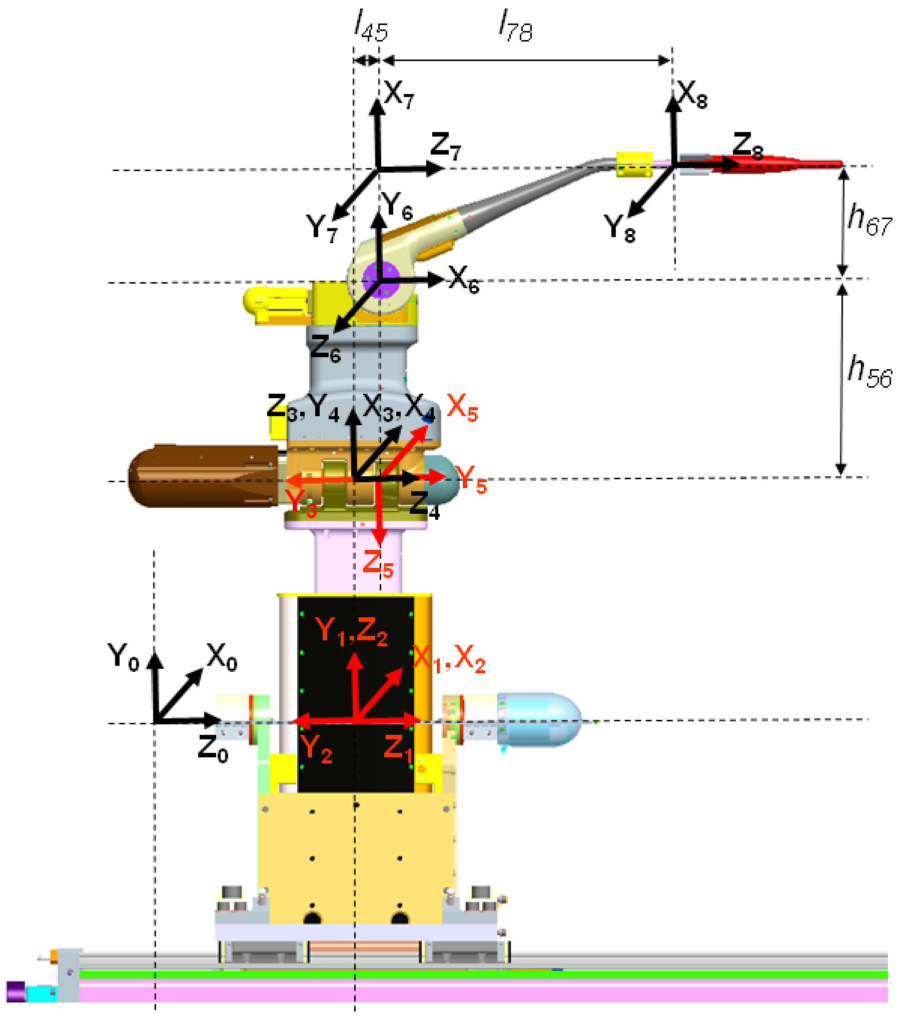


Figure A.1: The frames obtained from the DH convention.

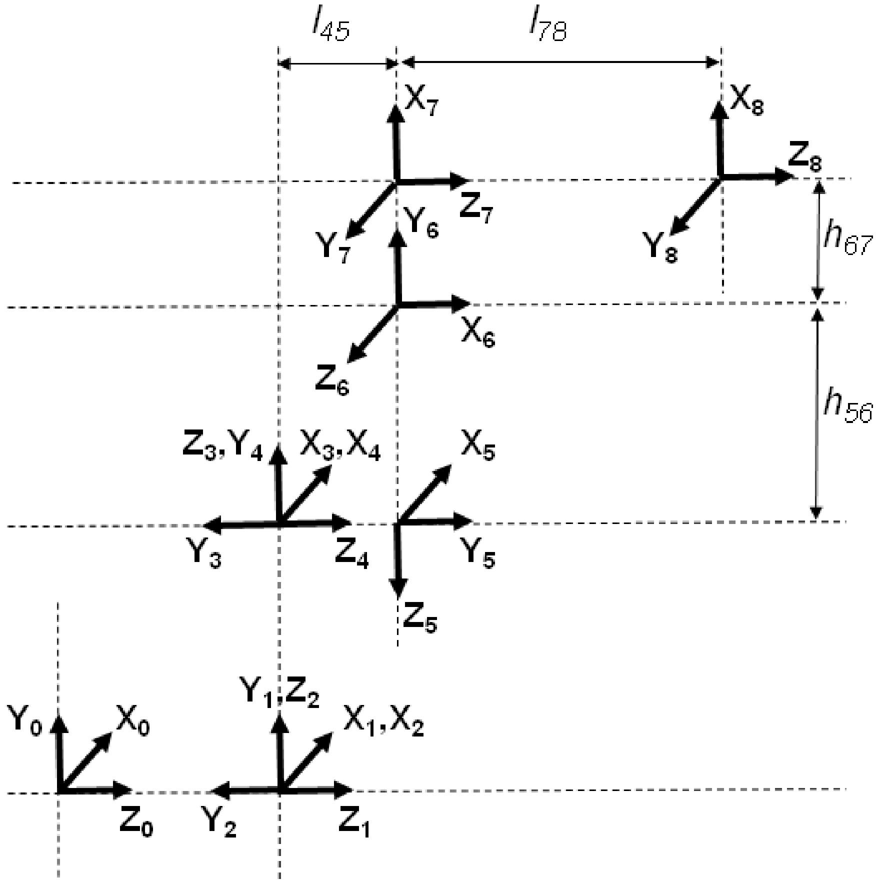


Figure A.2: The frames obtained from the DH convention without the CTS manipulator.

Appendix B

Inverse Kinematics

At the design stage, the joint arrangement of the 8-DOF CTS manipulator was chosen such that a closed-form solution to the inverse kinematics problem in $\mathcal{W}$ could be derived. The inverse kinematics equation at the position level (2.6) details as follows.

Let $\mathbf{A}^{j}_{j+1}(\mathbf{q}_j),\ j = 0, \ldots, 7$ be the transformation matrices representing the position and orientation of the DH frame $j+1$ relative to the DH frame j, as provided in Figure 2.8, and let the position and orientation of the end-effector $\mathbf{P}$ be

$$\mathbf{P} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

We have:

$$\mathbf{P} = \mathbf{A^0_1}(\mathbf{q}_1)\mathbf{A^1_2}(\mathbf{q}_2)\mathbf{A^2_3}(\mathbf{q}_3)\mathbf{A^3_4}(\mathbf{q}_4)\mathbf{A^4_5}(\mathbf{q}_5)\mathbf{A^5_6}(\mathbf{q}_6)\mathbf{A^6_7}(\mathbf{q}_7)\mathbf{A^7_8}(\mathbf{q}_8). \tag{B.1}$$

Denote $c_i = \cos(\mathbf{q}_i)$, $s_i = \sin(\mathbf{q}_i)$, $\mathbf{q}_{25} = \mathbf{q}_2 + \mathbf{q}_5$, and $\mathbf{q}_{34} = \mathbf{q}_3 + \mathbf{q}_4$. It is proposed to derive the solution to the inverse kinematics problem as a function of $\mathbf{P}$, $\mathbf{q}_{25}$ and $\mathbf{q}_3$.

Rearranging (B.1) yields

$$\mathbf{A_8^7}(\mathbf{q}_8)^{-1}\mathbf{A_7^6}(\mathbf{q}_7)^{-1}\mathbf{A_6^5}(\mathbf{q}_6)^{-1} = \mathbf{P}^{-1}\mathbf{A_1^0}(\mathbf{q}_1)\mathbf{A_2^1}(\mathbf{q}_2)\mathbf{A_3^2}(\mathbf{q}_3)\mathbf{A_4^3}(\mathbf{q}_4)\mathbf{A_5^4}(\mathbf{q}_5). \tag{B.2}$$

The third column in (B.2) gives the following equations:

$$\begin{cases} -c_7c_8 &= r_{11}s_{25} - r_{21}c_{25} \\ c_7s_8 &= r_{12}s_{25} - r_{22}c_{25} \\ -s_7 &= r_{13}s_{25} - r_{23}c_{25} \end{cases}. \tag{B.3}$$

Rearranging (B.1) yields

$$\mathbf{P}\mathbf{A_8^7}(\mathbf{q}_8)^{-1}\mathbf{A_7^6}(\mathbf{q}_7)^{-1} = \mathbf{A_1^0}(\mathbf{q}_1)\mathbf{A_2^1}(\mathbf{q}_2)\mathbf{A_3^2}(\mathbf{q}_3)\mathbf{A_4^3}(\mathbf{q}_4)\mathbf{A_5^4}(\mathbf{q}_5)\mathbf{A_6^5}(\mathbf{q}_6). \tag{B.4}$$

The third row in (B.4) gives the following equations:

$$\begin{cases} c_6 &= -r_{31}s_7c_8 + r_{32}s_7s_8 + r_{33}c_7 \\ -s_6 &= r_{31}s_8 + r_{32}c_8 \end{cases}. \tag{B.5}$$

The fourth column in (B.4) gives the following equations:

$$\begin{cases} -s_2(\mathbf{q}_{34} + \mathbf{q}_{34,0}) &= -r_{11}c_8h_{67} + r_{12}s_8h_{67} - r_{13}l_{78} + t_x - s_{25}h_{56} \\ c_2(\mathbf{q}_{34} + \mathbf{q}_{34,0}) &= -r_{21}c_8h_{67} + r_{22}s_8h_{67} - r_{23}l_{78} + t_y + c_{25}h_{56} \\ \mathbf{q}_1 &= -r_{31}c_8h_{67} + r_{32}s_8h_{67} - r_{33}l_{78} + t_z - l_{45} \end{cases}, \tag{B.6}$$

where $\mathbf{q}_{34,0} = \mathbf{q}_{3,0} + \mathbf{q}_{4,0}$. Given (B.3), (B.5), and (B.6), it is possible to calculate the joint configuration $\mathbf{q}$, solution to the inverse kinematics problem, as a function of $\mathbf{P}$, $\mathbf{q}_{25}$ and $\mathbf{q}_3$ as follows. The value of joints 7 and 8 can be obtained from (B.3):

$$\left\{\begin{array}{lcl} \mathbf{q}_8 & = & \arctan((r_{12}s_{25} - r_{22}c_{25})/(-r_{11}s_{25} + r_{21}c_{25})) \\ \mathbf{q}_7 & = & \arctan2(-r_{13}s_{25} + r_{23}c_{25}, (-r_{11}s_{25} + r_{21}c_{25})/c_8) \end{array}\right. .$$

Knowing $\mathbf{q}_7$ and $\mathbf{q}_8$, the value of joint 6 can be obtained from (B.5):

$$\mathbf{q}_6 = \arctan2(-r_{31}s_8 - r_{32}c_8, -r_{31}s_7c_8 + r_{32}s_7s_8 + r_{33}c_7).$$

Knowing $\mathbf{q}_8$, $\mathbf{q}_{34}$ can be calculated using the first two equations in (B.6):

$$\begin{aligned} \mathbf{q}_{34} = & \Big(\big(-r_{11}c_8h_{67} + r_{12}s_8h_{67} - r_{13}l_{78} + t_x - s_{25}h_{56}\big)^2 + \\ & \big(-r_{21}c_8h_{67} + r_{22}s_8h_{67} - r_{23}l_{78} + t_y + c_{25}h_{56}\big)^2\Big)^{1/2} - \mathbf{q}_{34,0}. \end{aligned}$$

From there, the value $\mathbf{q}_2$ of joint 2 follows:

$$\begin{aligned} \mathbf{q}_2 = & \arctan2\big((r_{11}c_8h_{67} - r_{12}s_8h_{67} + r_{13}l_{78} - t_x + s_{25}h_{56})/(\mathbf{q}_{34} + \mathbf{q}_{34,0}), \\ & (-r_{21}c_8h_{67} + r_{22}s_8h_{67} - r_{23}l_{78} + t_y + c_{25}h_{56})/(\mathbf{q}_{34} + \mathbf{q}_{34,0})\big). \end{aligned}$$

The value $\mathbf{q}_1$ of joint 1 is obtained from the third equation in (B.6):

$$\mathbf{q}_1 = -r_{31}c_8h_{67} + r_{32}s_8h_{67} - r_{33}l_{78} + t_z - l_{45}.$$

Finally, the value $\mathbf{q}_4$ and $\mathbf{q}_5$ of joints 4 and 5 are simply:

$$\left\{\begin{array}{lcl} \mathbf{q}_4 & = & \mathbf{q}_{34} - \mathbf{q}_3 \\ \mathbf{q}_5 & = & \mathbf{q}_{25} - \mathbf{q}_2 \end{array}\right. .$$

To be complete, it should be noted that there are in fact three more solutions to the inverse kinematics problem. From (B.3) and (B.5), it can be observed that, if $(\mathbf{q}_6, \mathbf{q}_7, \mathbf{q}_8)$ is a solution, then so is $(\mathbf{q}_6 + \pi, -\mathbf{q}_7 + \pi, \mathbf{q}_8 + \pi)$. Moreover, from (B.6), it can be observed that, if $(\mathbf{q}_2, \mathbf{q}_{34})$ is a solution, then so is $(\mathbf{q}_2 + \pi, -\mathbf{q}_{34} - 2\mathbf{q}_{34,0})$. Therefore, the four possible solutions $\mathbf{q}^0$, $\mathbf{q}^1$, $\mathbf{q}^2$, and $\mathbf{q}^3$ to the inverse kinematics problem are:

$$\begin{cases} \mathbf{q}^0 = (\mathbf{q}_1^0, & \mathbf{q}_2^0 & , \mathbf{q}_3^0 & , \mathbf{q}_4^0 & , \mathbf{q}_5^0 & , q_6^0 & , q_7^0 & , q_8^0 &) \\ \mathbf{q}^1 = (\mathbf{q}_1^1, & \mathbf{q}_2^1 & , \mathbf{q}_3^0 & , \mathbf{q}_4^1 & , \mathbf{q}_5^1 & , q_6^0 + \pi & , \pi - q_7^0 & , q_8^0 + \pi &) \\ \mathbf{q}^2 = (\mathbf{q}_1^0, & \mathbf{q}_2^0 + \pi & , \mathbf{q}_3^0 & , -2\mathbf{q}_3^0 - \mathbf{q}_4^0 - 2\mathbf{q}_{34,0} & , \mathbf{q}_5^0 - \pi & , q_6^0 & , q_7^0 & , q_8^0 &) \\ \mathbf{q}^3 = (\mathbf{q}_1^1, & \mathbf{q}_2^1 + \pi & , \mathbf{q}_3^0 & , -2\mathbf{q}_3^0 - \mathbf{q}_4^1 - 2\mathbf{q}_{34,0} & , \mathbf{q}_5^1 - \pi & , q_6^0 + \pi & , \pi - q_7^0 & , q_8^0 + \pi &) \end{cases} .$$

However, only one of these four solutions is within the joint mechanical limits. In particular, the range for joints 6 and 7 is less that 90 deg and the range for the prismatic joints is nonnegative. Finally, the inverse kinematics equation at the position level (2.6) writes:

$$\mathbf{g}(\mathbf{p}, v) = \mathbf{q}^0,$$

where $v = \mathbf{q}_{25}$ is the redundancy parameter. Recall that, in this thesis, it is assumed that the two prismatic joints 3 and 4 form a single prismatic joint, therefore, $\mathbf{q}_3$ does not appear as another redundancy parameter in the inverse kinematics equation.

Appendix C

Building Subtask Trajectories

C.1 Problem Definition

Formally, a subtask consists of moving the store from an initial position and orientation $\mathbf{p_0}$ at t_0 to a final position and orientation $\mathbf{p_f}$ at the constant velocity $\dot{\mathbf{p}}_{\max}$. As mentioned in Section 2.3.2, it is desired that the acceleration along the trajectory is continuous and that the initial and final velocity and the initial and final acceleration are zero. How to build a one-dimensional trajectory with such requirements is detailed in Section C.2. This approach is then generalized to n-dimensional trajectories in Section C.3. As explained in Section C.4, the subtask trajectories are just a particular case of these n-dimensional trajectories when $n = 6$.

C.2 One-Dimensional Trajectory

The problem stated in Section C.1 can be formulated as follows: Find a C^2 function $x(\cdot)$ such that

$$x(t_0) = x_0,\ \dot{x}(t_0) = 0,\ \ddot{x}(t_0) = 0,$$

$$x(t_f) = x_f,\ \dot{x}(t_f) = 0,\ \ddot{x}(t_f) = 0,$$

where $t_f > 0$ is the final time (not given). The absolute value of the first derivative $\dot{x}(\cdot)$ cannot exceed $v_{\max}$. It is also assumed that the absolute value of the second derivative $\ddot{x}(\cdot)$ cannot exceed $a_{\max}$. For simplicity, let $x_f > x_0$.

The traditional approach in robotics to address such a problem is to use linear functions with parabolic blends [2, p. 210]. However, these functions are not C^2 and the second derivative at t_0 and t_f is not zero. It is therefore proposed to blend the first derivative. Many different ways to blend exist. It is chosen to use a 4^{th} symmetric blend which makes the second derivative not only continuous but also C^1. The second derivative $\ddot{x}(\cdot)$ is therefore as illustrated in Figure C.1. The intervals $[t_0, t_c]$, $[t_c, t_f - t_c]$ and $[t_f - t_c, t_f]$ corresponds to the acceleration phase, the constant velocity phase, and the deceleration phase respectively. The parameter $\alpha \in [0, 1]$ corresponds to the percentage of the interval $[t_0, t_c]$ where the maximum value of the second derivative $a_{\max}$ is reached. When $\alpha = 0$, the maximum value of the second derivative $a_{\max}$ is only reached at the time $(t_c - t_0)/2$. When $\alpha = 1$, the linear function with parabolic blends discussed above is retrieved as a special case.

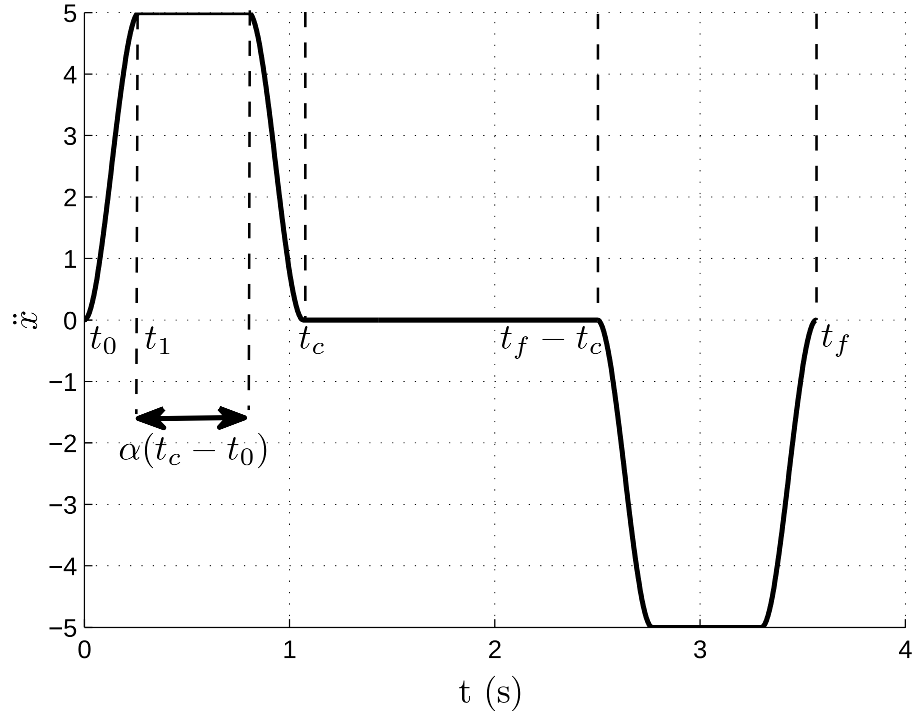


Figure C.1: A C^1 profile for the second derivative

More precisely, the second derivative $\ddot{x}(\cdot)$ can be described as follows, defining $t_1 = t_c(1-\alpha)/2$:

- in $[t_0, t_1]$, $\ddot{x}(\cdot)$ is a 3rd order polynomial,
- $\forall t \in [t_1, t_c - t_1]$, $\ddot{x}(t) = a_{\max}$,
- in $[t_c - t_1, t_c]$, $\ddot{x}(\cdot)$ is a 3rd order polynomial,
- $\forall t \in [t_c, t_f - t_c]$, $\ddot{x}(t) = 0$,
- in $[t_f - t_c, t_f - t_c + t_1]$, $\ddot{x}(\cdot)$ is a 3rd order polynomial,
- $\forall t \in [t_f - t_c + t_1, t_f - t_1]$, $\ddot{x}(t) = -a_{\max}$,
- in $[t_f - t_1, t_f]$, $\ddot{x}(\cdot)$ is a 3rd order polynomial.

The function $x(\cdot)$ must satisfy the continuity conditions for its second and third derivative:

- $\ddot{x}(t_0) = 0,\ \frac{d}{dt}\ddot{x}(t_0) = 0,$
- $\ddot{x}(t_1) = a_{\max},\ \frac{d}{dt}\ddot{x}(t_1) = 0,$
- $\ddot{x}(t_c - t_1) = a_{\max},\ \frac{d}{dt}\ddot{x}(t_c - t_1) = 0,$
- $\ddot{x}(t_c) = 0,\ \frac{d}{dt}\ddot{x}(t_c) = 0,$
- $\ddot{x}(t_f - t_c) = 0,\ \frac{d}{dt}\ddot{x}(t_f - t_c) = 0,$
- $\ddot{x}(t_f - t_c + t_1) = -a_{\max},\ \frac{d}{dt}\ddot{x}(t_f - t_c + t_1) = 0,$
- $\ddot{x}(t_f - t_1) = -a_{\max},\ \frac{d}{dt}\ddot{x}(t_f - t_1) = 0,$
- $\ddot{x}(t_f) = 0,\ \frac{d}{dt}\ddot{x}(t_f) = 0.$

With the above conditions and the parameter α being set, it can easily be shown that the only unknowns remaining to fully determine the function $x(\cdot)$ are t_c and t_f. First, we have

$$\dot{x}(t_c) = \frac{(1+\alpha)}{2} a_{\max} t_c, \tag{C.1}$$

that must satisfy the constraint

$$\dot{x}(t_c) \leq v_{\max}.$$

It is possible that the maximum first derivative $v_{\max}$ is reached or not. Therefore,

- Assume that the maximum first derivative $v_{\max}$ is reached, i.e., $\dot{x}(t_c) = v_{\max}$. Hence, from (C.1),

  $$t_c = \frac{2v_{\max}}{(1+\alpha)a_{\max}}. \tag{C.2}$$

  We have

  $$x_f = x_0 + \dot{x}(t_c)(t_f - t_c).$$

Hence,

$$t_f = t_c + \frac{x_f - x_0}{v_{\max}}. \tag{C.3}$$

We need to check that

$$t_f - 2t_c \geq 0,$$

which corresponds to the condition

$$\frac{1+\alpha}{2} a_{\max}(x_f - x_0) \geq v_{\max}^2. \tag{C.4}$$

- Assume that the maximum first derivative $v_{\max}$ is not reached. Therefore, there is no constant velocity phase, i.e.,

  $$t_f = 2t_c. \tag{C.5}$$

  We have

  $$x_f = x_0 + \dot{x}(t_c)t_c. \tag{C.6}$$

  Substituting (C.1) in (C.6) yields

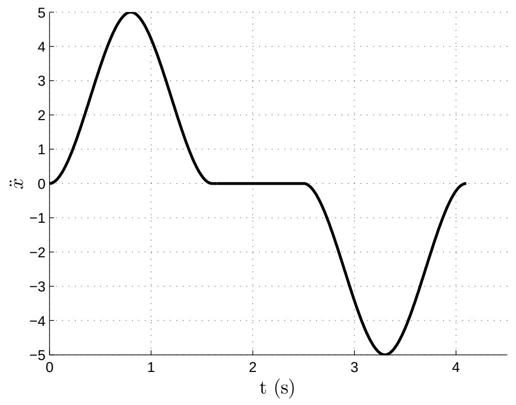
  $$t_c = \sqrt{\frac{2(x_f - x_0)}{(1+\alpha)a_{\max}}}, \tag{C.7}$$

  which corresponds obviously to the opposite condition to (C.4)

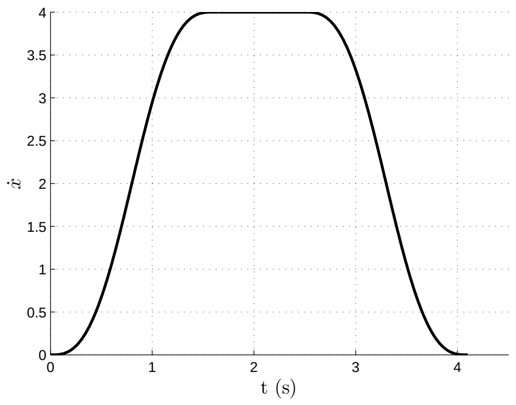
  $$\frac{1+\alpha}{2} a_{\max}(x_f - x_0) < v_{\max}^2. \tag{C.8}$$

Finally, given x_0, x_f, $v_{\max}$, $a_{\max}$, and α, the values of t_c and t_f can be calculated either by (C.2) and (C.3) or (C.5) and (C.7) depending on the sign of the quantity $v_{\max}^2 - a_{\max}(x_f - x_0)(1+\alpha)/2$. Knowing t_f and t_c, the function $x(\cdot)$ can be fully determined. For brevity, the detailed analytical expression for $x(\cdot)$ is not provided.

The function $x(\cdot)$ as determined above is now illustrated. First, let $x_0 = 0$, $x_f = 10$, $v_{\max} = 4$, $a_{\max} = 5$, and $\alpha = 0$, then $v_{\max}^2 - \frac{1+\alpha}{2}a_{\max}(x_f - x_0) = 16 - 25 = -9 < 0$. $t_c = 1.6$ and $t_f = 4.1$ are given by (C.2) and (C.3). The second derivative $\ddot{x}(\cdot)$ and the first derivative $\dot{x}(\cdot)$ are illustrated in Figures C.2(a) and C.2(b).

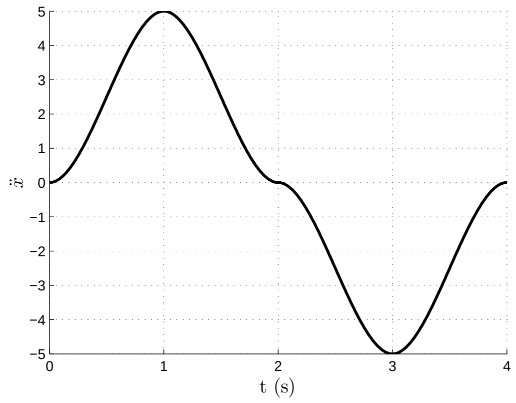


(a) The second derivative.

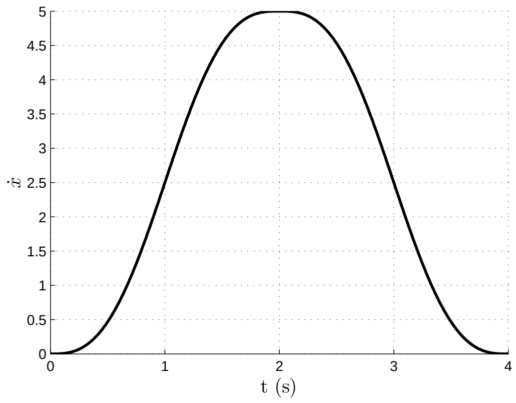


(b) The first derivative.

Figure C.2: The case $v_{\max}^2 - \frac{1+\alpha}{2}a_{\max}(x_f - x_0) < 0$.



(a) The second derivative.



(b) The first derivative.

Figure C.3: The case $v_{\max}^2 - \frac{1+\alpha}{2}a_{\max}(x_f - x_0) = 0$.

For this example, $v_{\max} = 5$ is the limiting case in the sense that the maximum first derivative $v_{\max}$ is only reached at the single instant $t_c = 2$. In such a case, the second derivative $\ddot{x}(\cdot)$ and the first derivative $\dot{x}(\cdot)$ are illustrated in Figures C.3(a) and C.3(b). Finally, when $v_{\max} > 5$, the maximum first derivative $v_{\max}$ is never reached. In such a case, the second derivative $\ddot{x}(\cdot)$ and the first derivative $\dot{x}(\cdot)$ are also illustrated in Figures C.3(a) and C.3(b).

C.3 n-Dimensional Trajectory

The same problem as the one stated in Section C.2 is considered in this section but $\mathbf{x}$ is assumed to be a n-dimensional vector. A simple approach to solve this problem would be to decouple the n dimensions and build for every component a function as in Section C.2. However, with this simple approach, it is not guaranteed that the final time t_f for all the components are the same. Therefore, the problem in this section can be stated as follows: Find a n-dimensional C^2 function $\mathbf{x}(\cdot)$ such that

$$\mathbf{x}(t_0) = \mathbf{x_0},\ \dot{\mathbf{x}}(t_0) = 0,\ \ddot{\mathbf{x}}(t_0) = 0,$$

$$\mathbf{x}(t_f) = \mathbf{x_f},\ \dot{\mathbf{x}}(t_f) = 0,\ \ddot{\mathbf{x}}(t_f) = 0,$$

where $t_f > 0$ is the final time (not given). The absolute value of the first derivative for each component $\dot{\mathbf{x}}_i(\cdot),\ i = 1, \ldots, n$ cannot exceed $\mathbf{v}_{\max i},\ i = 1, \ldots, n$. It is also assumed that the absolute value of the second derivative for each component $\ddot{\mathbf{x}}_i(.)$ cannot exceed $\mathbf{a}_{\max i},\ i = 1, \ldots, n$. For simplicity, let $\mathbf{x_f} \neq \mathbf{x_0}$.

Let $s(\cdot)$ be a function built as in Section C.2:

$$s(t_0) = 0,\ \dot{s}(t_0) = 0,\ \ddot{s}(t_0) = 0,$$

$$s(t_f) = 1,\ \dot{s}(t_f) = 0,\ \ddot{s}(t_f) = 0,$$

and define the n-dimensional function $\mathbf{x}(\cdot)$ as follows:

$$\mathbf{x}(t) = (\mathbf{x_f} - \mathbf{x_0})s(t) + \mathbf{x_0}. \tag{C.9}$$

It is easy to see that the resulting n-dimensional function $\mathbf{x}(.)$ is C^2 and that the initial and final conditions for $\mathbf{x}(.)$ are satisfied. The conditions relative to the maximum first derivative and the maximum second derivative for $\mathbf{x}(.)$ translate into conditions for the maximum first derivative and the maximum second derivative for $s(.)$ as follows. We have

$$|(\mathbf{x_f}_i - \mathbf{x_0}_i)\dot{s}(t)| \leq \mathbf{v}_{\max i}, \; |(\mathbf{x_f}_i - \mathbf{x_0}_i)\ddot{s}(t)| \leq \mathbf{a}_{\max i}, \; i = 1, \ldots, n.$$

Therefore, the maximum value of the first derivative $v_{\max}$ for $s(.)$ is defined by

$$v_{\max} = \min_{i=1,\ldots,n} \left\{ \frac{\mathbf{v}_{\max i}}{|\mathbf{x_f}_i - \mathbf{x_0}_i|} \right\}, \tag{C.10}$$

and the maximum value of the second derivative $a_{\max}$ for $s(.)$ is defined by

$$a_{\max} = \min_{i=1,\ldots,n} \left\{ \frac{\mathbf{a}_{\max i}}{|\mathbf{x_f}_i - \mathbf{x_0}_i|} \right\}, \tag{C.11}$$

With (C.10)-(C.11), the function $s(.)$ can be determined as in Section C.2. The n-dimensional function $\mathbf{x}(.)$ then follows from (C.9).

C.4 Application to the Subtask Trajectories

The problem of generating the subtask trajectory is just a particular case of Section C.3, where

- $n = 6$ and $\mathbf{x} = \mathbf{p}$,

- $\mathbf{x_0} = \mathbf{p_0}$ and $\mathbf{x_f} = \mathbf{p_f}$,
- $\mathbf{v}_{\max} = \dot{\mathbf{p}}_{\max}$ where $\dot{\mathbf{p}}_{\max}$ is provided in Table 2.1.

In PCTS, the maximum end-effector acceleration $\ddot{\mathbf{p}}_{\max}$, corresponding to $\mathbf{a}_{\max}$, is not provided. An admissible value for $\ddot{\mathbf{p}}_{\max}$ must respect the constraint mentioned in Section 2.2 that the end-effector must be able to reach any position and orientation within $\mathcal{W}$ in less time than the run time of the wind tunnel.