

A Semi-Lagrangian Approach for Time and Energy Path Planning Optimization in Static Flow Fields

Víctor C. da S. Campos^a, Armando A. Neto^a, Douglas G. Macharet^b

^a*Dept. of Electronics Engineering, Universidade Federal de Minas Gerais, Av. Antônio Carlos, 6627, Pampulha, Belo Horizonte, 31270-901, Minas Gerais, Brazil*

^b*Computer Vision and Robotics Laboratory (VeRLab), Dept. of Computer Science, Universidade Federal de Minas Gerais, Av. Antônio Carlos, 6627, Pampulha, Belo Horizonte, 31270-901, Minas Gerais, Brazil*

Abstract

Efficient path planning for autonomous mobile robots is a critical problem across numerous domains, where optimizing both *time* and *energy* consumption is paramount. This paper introduces a novel methodology that considers the dynamic influence of an environmental flow field and geometric constraints, including obstacles and forbidden zones, enriching the complexity of the planning problem. Here, we formulate it as a multi-objective optimal control problem, and propose a novel transformation called *Harmonic Transformation*, applying a semi-Lagrangian scheme to solve it. The set of Pareto efficient solutions is obtained considering two distinct approaches: *i*) a deterministic method referred to as **Concurrent Policy Iteration (CPI)**; and *ii*) an evolutionary-based one, called **Multi-objective Evolutionary Policy Iteration (MEPI)**. Both methods were designed to make use of the proposed Harmonic Transformation. Through an extensive analysis of these approaches, comparing them with the state-of-the-art literature, we demonstrate their efficacy in finding optimized paths. Generally speaking, the Pareto Set of solutions found in our experiments indicates that the **CPI** demonstrated better performance in finding solutions close to the time-optimal one, whereas the **MEPI** was most successful in finding solutions close to the energy-optimal solution.

Keywords: Motion and Path Planning, Collision Avoidance, Optimization and Optimal Control, Multi-objective Optimal Control

1. Introduction

The continual research and development of new and more advanced path-planning approaches play a pivotal role in Robotics [1]. Such techniques enable autonomous mobile robots to navigate efficiently and safely in complex and dynamic environments, making them essential for diverse applications, from logistics to monitoring and exploration. In this context, new challenges arise when robotic systems address singular and multiple objectives and often conflicting goals. These objectives range from minimizing *travel time* and *energy consumption* to optimizing factors like safety and resource allocation [2]. Furthermore, it is also imperative to acknowledge that, in several domains, environmental dynamics substantially influence the trajectories and behaviors of the vehicles. This is particularly evident in fields such as aerospace, where factors like air density, wind patterns, and gravitational forces intricately shape the aircraft flight paths [3].

Similarly, in maritime environments, the varying properties of water, including currents and turbulence, substantially impact the maneuverability of underwater or surface vehicles [4]. Hence, an adequate navigation strategy holds the potential to generate paths that optimize robot movement according to the surrounding flow field resulting from atmospheric and ocean currents. This synergy between path planning and environmental dynamics enhances the efficiency and speed of vehicle navigation and bolsters adaptability, ensuring that robots can navigate seamlessly through environments where flow dynamics are significant. Ultimately, this approach fosters improved resource utilization and reduced energy consumption, increasing the system's performance across a broad spectrum of robotic applications.

In this paper, we introduce an innovative approach to deal with time and energy-efficient path planning within environments characterized by static flow fields and stationary obstacles. This problem poses a multi-objective optimal control problem with forbidden zones in the state space, and we solve it with the following contributions:

- A *novel* transformation, called *Harmonic Transformation*, is employed to map values onto the $[0, 1]$ range to deal with the forbidden zones and avoid possible numerical problems in the computation of the value functions. We show that, given that the corresponding assumptions hold, a semi-Lagrangian approach converges to the unique viscosity solution of the corresponding transformed partial differential equations;

- Considering that time and energy are usually conflicting costs, we propose two approaches to find the set of *Pareto efficient solutions* in a multi-objective manner: a *deterministic* one, based on solving multiple single-objective optimizations concurrently; and a *metaheuristic* approach, based on a proper multi-objective evolutionary algorithm.

Semi-Lagrangian methods [5] are advantageous for path planning in real-time, dynamic environments due to their stability with larger time steps and computational efficiency. Their formulation allows for integrating dynamic constraints, like motion and energy limits, into cost functions, enhancing adaptability to real-world scenarios. Their support for unstructured and adaptive grids, suitable for high-dimensional and/or cluttered spaces, is resource-efficient and ideal for complex, real-world applications such as robotic navigation and obstacle avoidance. In addition, the fact that they find a closed-loop policy for the planning problem ensures greater robustness whenever they are employed on real systems.

The remainder of this paper is structured as follows: Section 2 presents the state-of-art related work; Section 3 delineates the problem formulation and introduces the Harmonic Transformation, demonstrating its application within a dynamic programming framework; Section 4 elaborates on the two distinct approaches we propose to address the multi-objective problem; Section 5 presents numerical results, showcasing the efficacy of our proposed approaches; Lastly, in Section 6, we provide concluding remarks and outline potential avenues for future research.

2. Related work

In single-objective path planning approaches, the most commonly prioritized factors are path length [6, 7] and travel time [8, 9]. However, enhancing solution quality and applicability can often be achieved by incorporating additional attributes, such as path safety or vulnerability, and smoothness [10, 11].

However, problems are complex in the real world, and multi-objective formulations have emerged as a noteworthy approach to providing solutions in challenging scenarios. For instance, [12] introduces a particle swarm optimization-based algorithm that considers multiple objectives, including travel length, path smoothness, economic cost, and path safety. Regarding the more general class of routing problems, where a sequence of visits is

required, a multi-objective version of the [Orienteering Problem](#) ([OP](#)) was introduced in [13], aiming to maximize cumulative reward while simultaneously minimizing exposure to sensors deployed in the environment. The [OP](#) has also been studied in environments with flows [14, 15]. In this multi-objective formulation, the aim is not only to maximize the collected reward but also to minimize energy expenditure by utilizing the surrounding environmental dynamics. In another recent study, [16] employs a level set method to ascertain energy-time optimal solutions within dynamic flow environments.

The authors of [17] introduce the DSFMO algorithm, a multi-objective evolutionary approach designed to solve optimization problems (MaOPs) with numerous objectives. DSFMO prioritizes diversity over convergence, using a global diversity measure and a conditional convergence measure to ensure a balanced evolution process. Similarly, [18] presents DSFMO for general MaOPs, focusing on improving evolutionary algorithms, particularly for complex Pareto fronts. While that method uses a graph-based representation, our approach emphasizes the influence of the flow field on the agent's dynamics.

More specifically to the problem we address here, the literature offers a variety of approaches to tackling environmental flow dynamics. These methods include graph-based methods [19] and evolutionary algorithms [20], which prioritize energy-efficient path planning, as well as sampling-based planners [21], which focus on achieving time-optimal paths. It is important to highlight that in scenarios with constant thrust (velocity of the vehicle concerning the flow), minimizing energy is equivalent to reducing time. In contrast, the travel time is directly proportional to the path length in constant net speed situations.

A comprehensive overview of trajectory planning and obstacle avoidance techniques for Autonomous Underwater Vehicles (AUVs) is presented in [22]. It focuses on algorithms that address the unique constraints and characteristics of AUVs, as well as the influence of marine environments. The article categorizes trajectory planning methods into two groups: global planning with known static obstacles, and local planning with unknown and dynamic obstacles.

The work [23] aims to develop a path-planning framework for marine robots that is robust to the inherent uncertainty in ocean current predictions. The focus is on using ensemble forecasts, which provide a distribution of possible flow fields, to generate path plans that minimize overall trajectory error. In contrast, our work presents a more general methodology for

optimizing time and energy in static flow fields.

The paper [24] describes an algorithm for estimating three-dimensional ocean flow fields using ensemble forecast data and online measurements. The proposed methodology leverages the property of negligible vertical velocity to produce highly accurate results, improving the trajectory planning capability of underwater gliders. It uses singular value decomposition (SVD) and a Kalman filter for online updates, focusing on underwater applications. At the same time, we propose a more general approach to trajectory planning in static flow fields, employing a semi-Lagrangian method based on dynamic programming and genetic algorithms.

In [25], the authors develop a reinforcement learning method for path planning in Autonomous Underwater Vehicles (AUVs) that maximizes data collection, such as temperature and salinity, while considering energy constraints and the influence of ocean currents. The focus is on balancing the acquisition of valuable information and minimizing energy consumption.

We propose a multi-objective approach that accounts for both the dynamic influence of environmental forces and the constraints imposed by fixed obstacles, making it suitable for applications where resource efficiency and obstacle avoidance are critical. We model the problem as an optimal control problem and introduce a novel technique called Harmonic Transformation, which is solved using a semi-Lagrangian scheme.

3. Problem formulation

Given a compact region of interest in the state space, we consider an agent described by:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}), \quad (1)$$

$$\dot{\mathbf{x}} = \mathbf{f}_1(\mathbf{x}) + \mathbf{F}_2(\mathbf{x})\mathbf{u}, \quad (2)$$

with $\mathbf{x} \in \mathbb{R}^n$ being the agent's states, and $\mathbf{u} \in \mathcal{U}$ the control inputs in the allowable control inputs set $\mathcal{U} \subset \mathbb{R}^m$. Also, $\mathbf{f}(\mathbf{x}, \mathbf{u}) : \mathbb{R}^{n+m} \rightarrow \mathbb{R}^n$ represents the agent's dynamics, which can be decomposed into $\mathbf{f}_1(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}^n$ (the *flow* vector field) and $\mathbf{F}_2(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}^{n \times m}$ (the *steering* matrix).

In the context of optimal control, we consider a cost function, $\ell(\mathbf{x}, \mathbf{u}) : \mathbb{R}^{n+m} \rightarrow \mathbb{R}$, that attributes a cost to every pair (\mathbf{x}, \mathbf{u}) whose \mathbf{x} is not a *target state*. We can also define a *value function* $v(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}$ describing the

minimum value for each point, \mathbf{x} , in state space (with a trajectory starting at this point), such that:

$$v(\mathbf{x}) = \inf_{\mathbf{u} \in \mathcal{U}} \int_0^\infty \ell(\mathbf{x}, \mathbf{u}) dt. \quad (3)$$

Since any path starting from a point along the optimal path should be optimal, we can use a Dynamic Programming Principle for $v(\mathbf{x})$ of the form:

$$v(\mathbf{x}) = \inf_{\mathbf{u} \in \mathcal{U}} \left\{ v(\mathbf{y}_x(\Delta t, \mathbf{u})) + \int_0^{\Delta t} \ell(\mathbf{y}_x(t, \mathbf{u}), \mathbf{u}) dt \right\},$$

with $\mathbf{y}_x(t, \mathbf{u})$ representing the point at time t along the path, taken when considering the control input defined by $\mathbf{u}(t)$, for the system dynamics in (1).

To deal with constraints on \mathbf{x} , such as forbidden or dangerous zones, we consider that the value function must be infinite in these locations. In addition, assuming that $\ell(\mathbf{x}, \mathbf{u})$ is always non-negative, the value of the target location must always be null. When considered together, these constraints lead to the boundary conditions:

$$v(\mathbf{x}) = \begin{cases} 0 & \text{for } \mathbf{x}(t) = \mathbf{x}_g, \\ \infty & \text{for } \mathbf{x}(t) \in \partial\mathcal{O}, \end{cases}$$

with \mathbf{x}_g representing a desired target location, and $\partial\mathcal{O}$ representing the boundaries of forbidden regions \mathcal{O} .

3.1. Harmonic Transformation and Dynamic Programming

To properly deal with obstacles/forbidden zones, we transform the value function image from $[0, \infty)$ to $[0, 1]$. Although the *Kruzkov Transformation* is usually employed in these cases, it can lead to numerical problems when the value function assumes large values [5]. This problem can be somewhat mitigated by employing a scalar multiplying $v(\mathbf{x})$, but finding a suitable scalar can be a tiresome process in some cases. In this sense, we consider a transformation of the form:

$$\bar{v}(\mathbf{x}) = \mathcal{H}(v(\mathbf{x})) = \frac{v(\mathbf{x})}{1 + v(\mathbf{x})} = 1 - \frac{1}{1 + v(\mathbf{x})},$$

hereinafter referred to as *Harmonic Transformation*, $\mathcal{H}(.)$.

As we shall detail subsequently, to derive a suitable Dynamic Programming Principle for the transformed problem, the following property of this transformation is essential:

$$\begin{aligned}
\mathcal{H}(x_1 + x_2) &= \frac{x_1 + x_2}{1 + x_1 + x_2} \frac{\frac{1}{1+x_1}}{\frac{1}{1+x_1}}, \\
&= \frac{\mathcal{H}(x_1) + \frac{x_2}{1+x_1} + x_2 - x_2}{\mathcal{H}(x_1) + \frac{1+x_2}{1+x_1} + (1+x_2) - (1+x_2)}, \\
&= \frac{\mathcal{H}(x_1) + x_2 - x_2 \left(1 - \frac{1}{1+x_1}\right)}{1 + \mathcal{H}(x_1) + x_2 - (1+x_2) \left(1 - \frac{1}{1+x_1}\right)}, \\
&= \frac{\mathcal{H}(x_1) + (1 - \mathcal{H}(x_1)) x_2}{1 + \mathcal{H}(x_1) + x_2 - (1+x_2) \mathcal{H}(x_1)}, \\
\mathcal{H}(x_1 + x_2) &= \frac{\mathcal{H}(x_1) + (1 - \mathcal{H}(x_1)) x_2}{1 + (1 - \mathcal{H}(x_1)) x_2}.
\end{aligned} \tag{4}$$

3.1.1. Numerical approximation

By applying property (4), a Dynamic Programming Principle can be found for the transformed value function:

$$\begin{aligned}
\bar{v}(\mathbf{x}(t)) &= \inf_{\mathbf{u} \in \mathcal{U}} \left\{ \frac{\bar{v}(\mathbf{y}_x(\Delta t, \mathbf{u})) + q(\mathbf{x}(t), \mathbf{u})}{1 + q(\mathbf{x}(t), \mathbf{u})} \right\}, \\
q(\mathbf{x}(t), \mathbf{u}) &= (1 - \bar{v}(\mathbf{y}_x(\Delta t, \mathbf{u}))) \int_0^{\Delta t} \ell(\mathbf{y}_x(t, \mathbf{u}), \mathbf{u}) dt,
\end{aligned} \tag{5}$$

and a **semi-Lagrangian (SL)** numerical scheme can approximate the solution by employing a time discretization, followed by a space discretization.

We consider the time discretization of Eq. (5), with step Δt , by applying a trapezoidal approximation for the integral term

$$\begin{aligned}
\int_0^{\Delta t} \ell(\mathbf{y}_x(t, \mathbf{u}), \mathbf{u}) dt &\approx \frac{\Delta t}{2} (\ell(\mathbf{x}_{k+1}, \mathbf{u}_k) + \ell(\mathbf{x}_k, \mathbf{u}_k)), \\
g(\mathbf{x}_k, \mathbf{u}_k) &= \frac{\Delta t}{2} (\ell(\mathbf{x}_{k+1}, \mathbf{u}_k) + \ell(\mathbf{x}_k, \mathbf{u}_k)),
\end{aligned} \tag{6}$$

and a trapezoidal method to solve the system of equations composed of (1) (assuming that the control input is held constant between the two samples),

leading to

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \frac{\Delta t}{2} (\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{f}(\mathbf{x}_{k+1}, \mathbf{u}_k)), \quad (7)$$

$$\bar{v}_k(\mathbf{x}_k) = \inf_{\mathbf{u} \in \mathcal{U}} \left\{ \frac{\bar{v}_{k+1}(\mathbf{x}_{k+1}) + (1 - \bar{v}_{k+1}(\mathbf{x}_{k+1})) g(\mathbf{x}_k, \mathbf{u}_k)}{1 + (1 - \bar{v}_{k+1}(\mathbf{x}_{k+1})) g(\mathbf{x}_k, \mathbf{u}_k)} \right\}. \quad (8)$$

Next, we perform the space discretization of \bar{v} by considering an unstructured grid of points covering the state space. Once these points represent \bar{v} , they are the only points over the space for which the value is updated. And since $\bar{v}(\mathbf{x}_{k+1})$ might not be a part of the grid, it is replaced by a finite element *linear* interpolation over the grid. One way of doing this linear interpolation is by employing a Delaunay triangulation on the unstructured grid points to find a triangulation of the space. We consider these triangles as our finite elements and represent the interpolation of $\bar{v}(\mathbf{x}_{k+1})$ as $\mathcal{I}_{\bar{v}_{k+1}}[\mathbf{x}_{k+1}]$.

Taken together, both discretization (time and space) lead to an **SL** approximation scheme of (5), in the form:

$$\bar{v}_k(\mathbf{x}_k) = \inf_{\mathbf{u} \in \mathcal{U}} \left\{ \frac{\mathcal{I}_{\bar{v}_{k+1}}[\mathbf{x}_{k+1}] + (1 - \mathcal{I}_{\bar{v}_{k+1}}[\mathbf{x}_{k+1}]) g(\mathbf{x}_k, \mathbf{u}_k)}{1 + (1 - \mathcal{I}_{\bar{v}_{k+1}}[\mathbf{x}_{k+1}]) g(\mathbf{x}_k, \mathbf{u}_k)} \right\}, \quad (9)$$

with boundary conditions

$$\bar{v}_k(\mathbf{x}_k) = \begin{cases} 0 & \text{if } \mathbf{x}(t) = \mathbf{x}_g, \\ 1 & \text{if } \mathbf{x}(t) \in \partial\mathcal{O}. \end{cases} \quad (10)$$

As in most approaches based on **Dynamic Programming** (DP), Eq. (9) can be solved backward in time using a *value iteration* algorithm. Since we have formulated our problem as a stationary/infinite horizon optimal control problem, an acceleration technique known as *policy iteration* [5, Section 8.4.7] can also be employed.

At every grid point, the optimal policy is:

$$\mathbf{u}_k = \underset{\mathbf{u}_k \in \mathcal{U}}{\operatorname{argmin}} \left\{ \frac{\mathcal{I}_{\bar{v}}[\mathbf{x}_{k+1}] + (1 - \mathcal{I}_{\bar{v}}[\mathbf{x}_{k+1}]) g(\mathbf{x}_k, \mathbf{u}_k)}{1 + (1 - \mathcal{I}_{\bar{v}}[\mathbf{x}_{k+1}]) g(\mathbf{x}_k, \mathbf{u}_k)} \right\},$$

and fixed for this iteration. Afterward, the value function is updated according to conditions (10) and

$$\begin{aligned} g(\mathbf{x}_k, \mathbf{u}_k) &= \bar{v}(\mathbf{x}_k) (1 + (1 - \mathcal{I}_{\bar{v}}[\mathbf{x}_{k+1}]) g(\mathbf{x}_k, \mathbf{u}_k)) \\ &\quad - (1 - g(\mathbf{x}_k, \mathbf{u}_k)) \mathcal{I}_{\bar{v}}[\mathbf{x}_{k+1}]. \end{aligned} \quad (11)$$

The algorithm iterates until it converges to the minimum of the value function. Finally, the original value function (3) can be recovered by:

$$v(\mathbf{x}_k) = \frac{\bar{v}(\mathbf{x}_k)}{1 - \bar{v}(\mathbf{x}_k)}.$$

One of the main problems in this case is that Eq. (11) defines a set of nonlinear equations. However, it is important to note that this is a highly sparse problem, for which specialized solvers drastically increase performance.

The DP Principle in Eq. (5) can be recast as a partial differential equation, known as the [Hamilton-Jacobi-Bellman \(HJB\)](#) equation for the transformed value function of the form:

$$\sup_{\mathbf{u} \in \mathcal{U}} \left\{ -\nabla \bar{v}(\mathbf{x}) \cdot \mathbf{f}(\mathbf{x}, \mathbf{u}) - (1 - \bar{v}(\mathbf{x}))^2 \ell(\mathbf{x}, \mathbf{u}) \right\} = 0. \quad (12)$$

With this representation in mind, considering the usual regularity conditions in the literature to ensure that the value function is continuous [26], we can state the following result regarding the optimal control problem and the proposed [SL](#) approximation scheme.

Theorem 1. *Consider the optimal control problem represented by the HJB equation (12). As long as \mathbf{f} and ℓ are Lipschitz, and ℓ is positive definite, with regards to the states, there exists a unique viscosity solution to this equation, representing the transformed value function of the optimal control problem. In addition to this, if $\ell > 0$, for every \mathbf{x} different from the target state, ℓ is convex with regards to \mathbf{u} , \mathbf{u} is bounded, and \mathbf{f} is bounded, the proposed numerical scheme converges to this unique solution as the time step, Δt , and the maximum distance between points on the grid, Δx , tend to zero, so long as Δx tends faster than Δt .*

Proof. The proof of this theorem is presented in [Appendix A](#). \square

Remark 1. *Theorem 1 ensures that, as $\Delta x \rightarrow 0$ and $\Delta t \rightarrow 0$ (with $\Delta t > \Delta x$), the Semi-Lagrangian numerical solution converges to the unique viscosity solution of the corresponding HJB. In addition to this, its proof shows that, so long as $\ell > 0$, for every \mathbf{x} different from the target state, the numerical scheme is monotone and a contraction mapping (even if the other assumptions of the theorem are not met, or the viscosity solution is discontinuous). This ensures that the method always converges to its unique fixed point, from*

the Banach Fixed Point Theorem, uniformly (since it is monotone). If value iteration is employed directly, by solving equation (9) backward in time, the distance between the current value function to the optimal one can be shown to decay, in the best scenario, as

$$\|\bar{v}_k - \bar{v}^*\| \leq \left(\frac{1}{1 + \bar{g}\Delta t} \right)^2 \|\bar{v}_{k+1} - \bar{v}^*\|,$$

and in the worst scenario, as

$$\|\bar{v}_k - \bar{v}^*\| \leq \frac{1}{1 + \varepsilon\bar{g}\Delta t} \|\bar{v}_{k+1} - \bar{v}^*\|,$$

with $1 - \varepsilon$ the largest value, smaller than 1, that the optimal value assumes. In either case, we have a decay rate that inversely depends on the size of the time step (Δt). Policy iteration, on the other hand, can usually be shown to converge superlinearly (with a convergence rate between 1.5 and 2) with a decay rate that inversely depends on the maximum distance between points on the grid (Δx) [27].

3.2. Energy and time value functions

Even though we have shown that an **SL** scheme can be employed with the *Harmonic Transformation* with any cost function ℓ that satisfies the conditions of Thm. 1, in this work, we will focus our attention on two specific costs, and the multi-objective problem defined by them.

Assuming that

$$\ell_T(\mathbf{x}, \mathbf{u}) = \begin{cases} 1 & \text{if } \mathbf{x} \neq \mathbf{x}_g, \\ 0 & \text{if } \mathbf{x} = \mathbf{x}_g, \end{cases} \quad (13)$$

its value function in Eq. (3) can be written as:

$$v_T(\mathbf{x}) = \inf_{\mathbf{u} \in \mathcal{U}} \int_0^\infty \ell_T(\mathbf{x}, \mathbf{u}) dt = \inf_{\mathbf{u} \in \mathcal{U}} \int_0^{T(\mathbf{x})} 1 dt = T(\mathbf{x}), \quad (14)$$

which corresponds to the *time taken to reach the target state from the current state*, hereinafter referred to as *time value function*.

We aim to examine the cost incurred in the following form:

$$\ell(\mathbf{x}, \mathbf{u}) = \begin{cases} \mathbf{u}^T \mathbf{u}, & \text{if } \mathbf{x} \neq \mathbf{x}_g, \\ 0, & \text{if } \mathbf{x} = \mathbf{x}_g, \end{cases}$$

so that its value function represents the *energy required to reach the target state from the current state* (assuming the energy is given by the squared \mathcal{L}_2 norm of the control inputs), it could lead to an *ill-posed* optimal control problem since the agent could decide to stop *indefinitely* at any *equilibrium point* of its dynamics, without being penalized.

In that regard, we consider the cost:

$$\ell_E(\mathbf{x}, \mathbf{u}) = \begin{cases} \varepsilon + \mathbf{u}^T \mathbf{u} & \text{if } \mathbf{x} \neq \mathbf{x}_g, \\ 0 & \text{if } \mathbf{x} = \mathbf{x}_g, \end{cases} \quad (15)$$

with ε being a small scalar that penalizes the time spent to reach the target state. Consequently, its value function in Eq. (3) can be written as:

$$\begin{aligned} v_E(\mathbf{x}) &= \inf_{\mathbf{u} \in \mathcal{U}} \int_0^\infty \ell_E(\mathbf{x}, \mathbf{u}) dt = \inf_{\mathbf{u} \in \mathcal{U}} \int_0^{T(\mathbf{x})} (\varepsilon + \mathbf{u}^T \mathbf{u}) dt, \\ &= \varepsilon T(\mathbf{x}) + \inf_{\mathbf{u} \in \mathcal{U}} \int_0^{T(\mathbf{x})} \mathbf{u}^T \mathbf{u} dt, \end{aligned} \quad (16)$$

and can be interpreted as *the energy used to reach the target state plus a small penalization of the time taken to reach it from the current state*. We will refer to this value function as the *energy value function*.

Since these two cost functions (and respective value functions) are antagonistic, in this work we try to find a set of *Pareto efficient solutions* (described either by the corresponding value function or policy defined for the agent), in a multi-objective manner. In that regard, we present two approaches to this multi-objective optimization: a *deterministic* one (based on solving multiple single-objective optimizations concurrently), and an *evolutionary* one (based on a proper multi-objective evolutionary algorithm).

4. Proposed approaches

4.1. Concurrent Policy Iteration

A simple and direct way of dealing with this multi-objective optimal control problem would be to perform a scalarization of the cost function:

$$\ell_\alpha(\mathbf{x}, \mathbf{u}) = \alpha \ell_T(\mathbf{x}, \mathbf{u}) + (1-\alpha) \ell_E(\mathbf{x}, \mathbf{u}), \quad \alpha \in [0, 1], \quad (17)$$

which leads to the suitable value function in Eq. (3). A grid could be performed on this *convex scalar weight* α , which would lead to different mono-objective problems that could be solved separately and correspond to different efficient solutions on the Pareto set of optimal solutions for this problem.

One way to improve this proposal is by noting that, despite solving for different value functions, all these optimal control problems explore the same system (described by the agent's dynamics, target states, and obstacles/forbidden regions). In that regard, the policies could be evaluated under all of the single-objective costs/value functions simultaneously leading to a higher *exploration* of the optimal control problem.

In this setting, a *naive* implementation, considering policy iteration and a grid of n_p scalar points in α , would consist of updating a policy for each grid point, but calculating all n_p value functions for each of these points. The main advantage of this approach is that it allows, in a sense, the different policies to share information employing the many value functions considered. The main drawback is that it requires the update of n_p^2 value functions at each time step, and this is the most costly step in many cases.

If we were not dealing with transformed value functions, it would suffice to consider the time and energy value functions in (14) and (16), respectively, since the rest of the value functions could be composed by taking the convex combination of them using α . Since we are dealing with value functions transformed by the proposed *Harmonic Transformation*, we need to define a relationship between them, and the transformation of a convex combination of these two value functions. In that regard, we can write

$$\begin{aligned} \mathcal{H}(\alpha v_T + (1-\alpha)v_E) &= \frac{\alpha v_T + (1-\alpha)v_E}{1 + \alpha v_T + (1-\alpha)v_E} \\ &= \frac{\alpha \frac{\mathcal{H}(v_T)}{1-\mathcal{H}(v_T)} + (1-\alpha) \frac{\mathcal{H}(v_E)}{1-\mathcal{H}(v_E)}}{1 + \alpha \frac{\mathcal{H}(v_T)}{1-\mathcal{H}(v_T)} + (1-\alpha) \frac{\mathcal{H}(v_E)}{1-\mathcal{H}(v_E)}} \\ &= \frac{\alpha \mathcal{H}(v_T) + (1-\alpha) \mathcal{H}(v_E) - \mathcal{H}(v_T) \mathcal{H}(v_E)}{1 - (1-\alpha) \mathcal{H}(v_T) - \alpha \mathcal{H}(v_E)}. \end{aligned} \quad (18)$$

By applying Eq. (18), we can find any of the transformed value functions (for a given policy) as long as we have the transformed time and energy value functions. So, only two value functions are updated and stored for every policy, and only $2n_p$ value functions are updated at each step, leading to the [Concurrent Policy Iteration](#) procedure presented in Algorithm 1.

Remark 2. Policy Iteration *schemes ensure that, at every iteration, a better policy is found. Therefore, if a discretization of the set of allowable controls is considered (such that the minimum can be found from an inspection over the set), they are assured to converge in a finite number of steps (since there*

will only be a finite number of allowable policies). In the worst case, the number of iterations taken to converge can be exponential, with $\mathcal{O}\left(\frac{n_u^{n_x}}{n_x}\right)$, n_u the number of points in our allowable controls set, and n_x the number of points discretizing our state space [28]. As discussed in Remark 1, though, these methods exhibit superlinear convergence and usually converge with a small number of iterations [27]. In comparison to the single objective policy iteration, **Concurrent Policy Iteration** require $2n_p$ times more policy evaluations (lines 4, 5, 11 and 12 in Algorithm 1) and n_p times more policy improvement (line 10 in Algorithm 1) steps.

4.2. Genetic Algorithm and Evolutionary Policy Iteration

While the proposed **Concurrent Policy Iteration** (CPI) algorithm exchanges information about the single-objective problems being solved to obtain better and faster results, it still consists of solving n_p single-objective optimal control problems. In addition to this, to find the optimal policies, it is common to consider a finite set of allowable control inputs (so that the minimum can be found from a simple verification of all possible elements).

To overcome both problems, here we propose a multi-objective genetic algorithm. To deal with the problem in a multi-objective approach directly, we employ the NSGA-II [29] (specifically its *fast non-dominated sorting* and *crowding distance* procedures), whereas the Evolutionary Policy Iteration [30, 31] approach is adopted to deal with a large/continuous set of allowable controls.

In this paper, we use policies \mathbf{u} to represent the individuals in the population. Equation (11) is employed with costs (13) and (15) to associate time and energy value functions to these policies, respectively. Note that the optimal policy, when considering any value function, would lead to a smaller value over the grid of unstructured points (since it needs to be the best policy everywhere inside of our region of interest). In that regard, we use the average of the value function over the unstructured grid as a proxy for how good that policy is, and end up with a bi-objective optimization problem with average time value and average energy value as our costs. Since we are employing an *Harmonic Transformation* on both value functions, they assume values in $[0, 1]$ and, as such, so will their average values.

In a single-objective setting, *Evolutionary Policy Iteration* [30] makes use of the concept of *Policy Switching* to generate an elite individual which is

Algorithm 1 Concurrent Policy Iteration (CPI)

Input: $\mathbf{x}_g, \boldsymbol{\alpha}, \varepsilon$

- 1: $k \leftarrow 0$
- 2: **for each** $\alpha_i \in \boldsymbol{\alpha}$ **do**
- 3: Assign a corresponding initial policy $\mathbf{u}_0^{(i)}$
- 4: Calculate the transformed time value $\bar{v}_{T0}^{(i)}$ of using the $\mathbf{u}_0^{(i)}$ policy, the cost in (13), and (7), (6) and (11)
- 5: Calculate the transformed energy value $\bar{v}_{E0}^{(i)}$ of using the $\mathbf{u}_0^{(i)}$ policy, the cost in (15), and (7), (6) and (11)
- 6: **while** policies $\mathbf{u}_k^{(i)}$ have not converged **do**
- 7: $k \leftarrow k + 1$
- 8: **for each** $\alpha_i \in \boldsymbol{\alpha}$ **do**
- 9: Find the *best* current value for α_i by

$$\tilde{v}^{(i)}(\mathbf{x}) = \min_s \frac{\alpha_i \bar{v}_{T(k-1)}^{(s)} + (1 - \alpha_i) \bar{v}_{E(k-1)}^{(s)} - \bar{v}_{T(k-1)}^{(s)} \bar{v}_{E(k-1)}^{(s)}}{1 - (1 - \alpha_i) \bar{v}_{T(k-1)}^{(s)} - \alpha_i \bar{v}_{E(k-1)}^{(s)}}$$

- 10: Updates the policy by using the cost (17) defined by α_i using (7), (6) and

$$\mathbf{u}_k^{(i)} = \operatorname{argmin}_{\mathbf{u}_k \in \mathcal{U}} \left\{ \frac{\mathcal{I}_{\tilde{v}}[\mathbf{x}_{k+1}] + (1 - \mathcal{I}_{\tilde{v}}[\mathbf{x}_{k+1}]) g(\mathbf{x}_k, \mathbf{u}_k)}{1 + (1 - \mathcal{I}_{\tilde{v}}[\mathbf{x}_{k+1}]) g(\mathbf{x}_k, \mathbf{u}_k)} \right\}$$

- 11: Calculate the transformed time value $\bar{v}_{Tk}^{(i)}$ using the $\mathbf{u}_k^{(i)}$ policy, cost in (13), and (7), (6) and (11)
 - 12: Calculate the transformed energy value $\bar{v}_{Ek}^{(i)}$ using the $\mathbf{u}_k^{(i)}$ policy, cost in (15), and (7), (6) and (11)
 - 13: **return** Policies $\mathbf{u}_k^{(i)}$, and value functions $\bar{v}_{Tk}^{(i)}$ and $\bar{v}_{Ek}^{(i)}$
-

guaranteed to be an improvement over the policies in the current population. In our setting, *Policy Switching* can be written, with regards to a specific

value function, as

$$\tilde{v}(\mathbf{x}_k) = \min_{i \in \text{population}} v_i(\mathbf{x}_k),$$

$$\mathbf{u}_{ps}(\mathbf{x}_k) = \operatorname{argmin}_{\mathbf{u}_k \in \mathcal{U}_{pop}(\mathbf{x}_k)} \left\{ \frac{\mathcal{I}_{\tilde{v}}[\mathbf{x}_{k+1}] + (1 - \mathcal{I}_{\tilde{v}}[\mathbf{x}_{k+1}]) g(\mathbf{x}_k, \mathbf{u}_k)}{1 + (1 - \mathcal{I}_{\tilde{v}}[\mathbf{x}_{k+1}]) g(\mathbf{x}_k, \mathbf{u}_k)} \right\},$$

with $\tilde{v}(\mathbf{x}_k)$ representing the value function obtained by taking the minimum value over the individuals of the population, $\mathcal{U}_{pop}(\mathbf{x}_k)$ the set of controls defined by the policies in the population at point \mathbf{x}_k , and $\mathbf{u}_{ps}(\mathbf{x}_k)$ the policy defined by taking the best control available in $\mathcal{U}_{pop}(\mathbf{x}_k)$ according to $\tilde{v}(\mathbf{x}_k)$. *Policy Switching* can also be employed as a *cross-over* operator that can generate an offspring given a set of parents, in which case a subset of the population is usually employed as the parents. In a single-objective setting, it can be shown that when paired with a *mutation* operator to ensure the exploration of the policy space, *policy switching* ensures the elite policy convergence to the optimal policy with probability one, when the state space is finite [31].

In our multi-objective setting, we can still employ *policy switching* to generate elite individuals. Since every individual already possesses time and energy value functions associated with its policy, it is straightforward to generate elite individuals concerning time and energy. In addition to this, considering a scalar parameter $\alpha \in [0, 1]$, we can employ (18) to recover elite individuals which are not on the extreme points of the Pareto set. In that regard, in our proposed algorithm, we generate three elite individuals using *policy switching* at every generation, the two extreme policies (associated with minimum time and minimum energy), and a third one using a random α at every generation. Even though the value of α could be fixed, at $\alpha = 0.5$ for instance, we employ a random α to increase exploration of the Pareto Set. To increase the convergence of the method, as well as ensure some diversity in the population, this elite individual with a random α also considers some control possibilities from a fixed set of control actions (aside from the ones on the individuals on the population) when searching for the value of the best action.

Similarly, *policy switching* is also employed as a *cross-over* operator, with a random $\alpha \in [0, 1]$, to generate other new offspring in the population. In this case, however, only a small number of individuals (usually 2 or 3) are employed as parents. To create new individuals, part of the offspring is also created by employing a simple *cross-over* between two policies. A random

scalar value λ is sampled from a uniform distribution over $[0, 1]$ and given policies $\mathbf{u}_1(\mathbf{x}_k)$ and $\mathbf{u}_2(\mathbf{x}_k)$, a new policy is generated by

$$\mathbf{u}_{\text{off}} = \lambda \mathbf{u}_1(\mathbf{x}_k) + (1 - \lambda) \mathbf{u}_2(\mathbf{x}_k). \quad (19)$$

With both *cross-over* operators, a simple *mutation* with a Gaussian noise (but modified to respect the bounds of the allowable control set) is employed in these new individuals to ensure exploration of the policy space.

As in a standard NSGA-II algorithm, the *fast non-dominated sorting* procedure is employed to rank solutions according to dominance in the objective functions space, whereas the *crowding distance* is employed to rank solutions on the same level (which do not dominate one another). These ranking solutions are employed when selecting individuals from the population (both for choosing parents for the *cross-over* operators and for selecting survivors for the next generation). Parents are chosen according to a binary tournament, whereas survivors (from the combined original and offspring population) are chosen according to their level of dominance. For the last positions available in the surviving population, a roulette strategy is employed, in which the probability of selection is proportional to the *crowding-distance* of an individual. This approach is summarized in Algorithm 2.

Remark 3. *Similarly to other stochastic optimal path planning methods in the literature, such as RRT* [32], Evolutionary Policy Iteration can be shown to be asymptotically optimal (converge to the optimal value with probability 1) [31]. In the same fashion, Multi-objective Evolutionary Policy Iteration can also be shown to be asymptotically optimal, since over time Policy Switching will ensure better policies are found for the elite individuals (lines 8 and 9 in Algorithm 2). In addition to this, the use of the crowding distance and fast non-dominated sorting will ensure that the points found on the Pareto set of efficient solutions are well spread out. Unlike, CPI, the additional cost for running MEPI over the single-objective equivalent is not that high (only doubling the required Policy Evaluation steps). Note that, since we only have asymptotical optimality, it can be hard to define a stopping criterion for MEPI, and the number of generations is usually employed.*

5. Numerical simulations

To illustrate the methods proposed in this paper, this section presents 5 numerical examples. All examples were run on a Ryzen 7 2700 CPU with

Algorithm 2 MEPI

Input: \mathbf{x}_g , n_{pop} , n_{ger} , n_{cp} , n_{par} , ε

- 1: $k \leftarrow 0$
- 2: **for each** individual i in the population **do**
- 3: Assign a random initial policy $\mathbf{u}^{(i)}$
- 4: Calculate the transformed time value $\bar{v}_T^{(i)}$ and transformed energy value $\bar{v}_E^{(i)}$ using the $\mathbf{u}^{(i)}$ policy, as well as their average values over the state space
- 5: **while** $k < n_{\text{ger}}$ **do**
- 6: $k \leftarrow k + 1$
- 7: Calculate *crowding distance* of the current population
- 8: Use *Policy Switching* over all population to generate elite individuals with regards to the $\bar{v}_T^{(i)}$ and $\bar{v}_E^{(i)}$ value functions respectively
- 9: Considering a random α , use *Policy Switching* over all the population to generate an elite individual with regards to the value functions

$$\tilde{v}^{(i)}(\mathbf{x}) = \frac{\alpha_i \bar{v}_T^{(i)} + (1-\alpha_i) \bar{v}_E^{(i)} - \bar{v}_T^{(i)} \bar{v}_E^{(i)}}{1 - (1-\alpha_i) \bar{v}_T^{(i)} - \alpha_i \bar{v}_E^{(i)}}$$

considering an *additional* set of fixed control actions.

- 10: Generate n_{cp} new offspring individuals using *Policy Switching* with n_{par} parents chosen using a binary tournament (decided by *dominance* and *crowding distance*) and a Gaussian mutation.
 - 11: Generate $n_{\text{pop}} - n_{\text{cp}} - 3$ individuals using the simple *cross-over* in (19) with 2 parents chosen using a binary tournament (decided by *dominance* and *crowding distance*) and a Gaussian mutation.
 - 12: Considering the individuals from the original population and the offspring population, sort them using the *fast non-dominated sorting* and calculate their *crowding distance*
 - 13: Select individuals to form the next generation of the population according to their *dominance level*. Employ a roulette (using the *crowding distance*) to fill the available spots when the remaining individuals cannot be chosen with dominance only.
 - 14: **return** Population $\mathbf{u}^{(i)}$, and value functions $\bar{v}_T^{(i)}$ and $\bar{v}_E^{(i)}$
-

16GB of RAM on Windows 11 with MATLAB R2023a. To solve (11), we employed the *fsolve* function with the ‘*SpecifyObjectiveGradient*’ option and made use of sparse matrices to describe the Jacobian matrix.

Throughout this section, whenever we refer to the *Average Time Cost* and *Average Energy Cost* as a way to compare the value functions, we will employ an average over the points of the CPI/MEPI grid of the transformed time, \bar{v}_T , and transformed energy, \bar{v}_E . These are interesting metrics, as using the Harmonic Transformation guarantees that they always assume values in $[0, 1]$. Note that, the use of the Harmonic Transformation implies that these are dimensionless quantities, and as such do not possess a corresponding unit.

5.1. Example 1 - comparing Harmonic and Kruzkov transformations

In this first example, we consider a simple scenario to compare the numerical behavior of the Harmonic transformation against the Kruzkov transformation. We consider an agent without drift dynamics (*i.e.* without a flow field affecting its velocities), whose behavior can be described by

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \end{bmatrix},$$

with x and $y \in [-10, 10]$ representing the position on the plane, and v_x and $v_y \in [-0.2, 0.2]$ the agent’s velocities, used as control inputs. The control objective is to drive the agent from any state (x, y) to the origin while avoiding the obstacles.

Since the main purpose of this example is to compare the numerical properties of both transformations, in this single example, only the minimum time problem is considered, and both implementations employ only a value iteration scheme (*i.e.* the Harmonic transformation scheme is implemented by solving (9), whereas the Kruzkov transformation scheme is implemented by solving [5, Eq. (8.69)]) with no acceleration schemes in either case. A structured grid with 141 points in X and 141 points in y , leading to 19881 points, was employed, with a time step of $\Delta t = 1s$.

To illustrate the numerical behavior of each transformation, the value functions found by each method are converted back to the *time to reach the origin* value function. This can be done by $v_T = \frac{\bar{v}}{1 - \bar{v}}$ for the Harmonics transformation, and $v_T = -\ln(1 - \bar{v})$ for the Kruzkov transformation. These are presented in Figures 1 and 2. As can be seen in these figures, the Harmonic transformation was capable of representing the value function

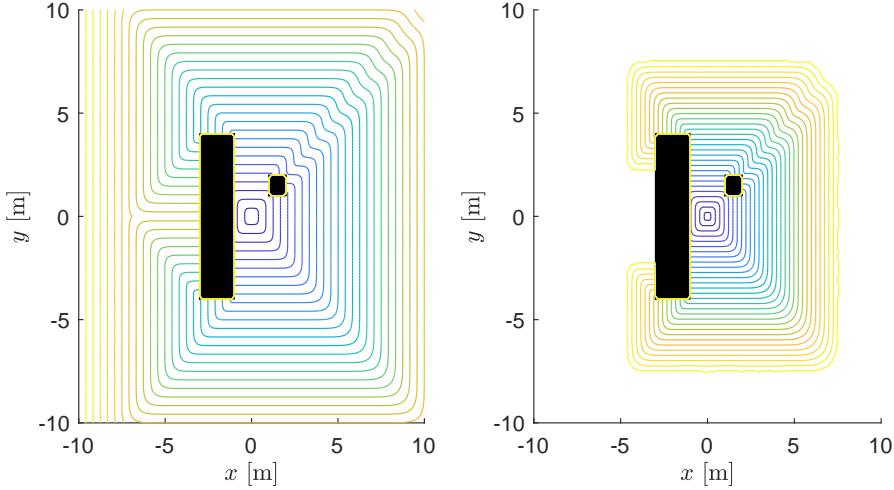


Figure 1: Time to reach the origin employing the *Harmonic* transformation and the *Kruzkov* transformation in Example 1. The obstacles are presented in black whereas the level curves of the *time to reach the origin* value functions are presented as the colored curves. The left plot represents the solution found by the *Harmonic* transformation while the right plot represents the solution found by the *Kruzkov* transformation.

throughout the whole domain, whereas the Kruzkov transformation gets numerically unstable once the time to reach the origin exceeds 40s (being unable to cover the whole domain). This illustrates the problem discussed in Section 3.1, justifying the proposition of the Harmonic transformation.

5.2. Example 2 - simple linear drift dynamics

In this second example, we consider an agent with simple linear drift dynamics

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} -1 & 1.2094 & 0.6937 \\ -1.2094 & -1 & 2.6564 \\ -0.6937 & -2.6564 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} -0.2415 & 0.3971 & 0.8855 \\ -0.9701 & -0.0744 & -0.2312 \\ -0.0259 & -0.9148 & 0.4031 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix},$$

with x_1, x_2 and $x_3 \in [-1, 1]$ being the position on the space, and v_1, v_2 and $v_3 \in [-2, 2]$ the control inputs. The control objective is to drive the agent

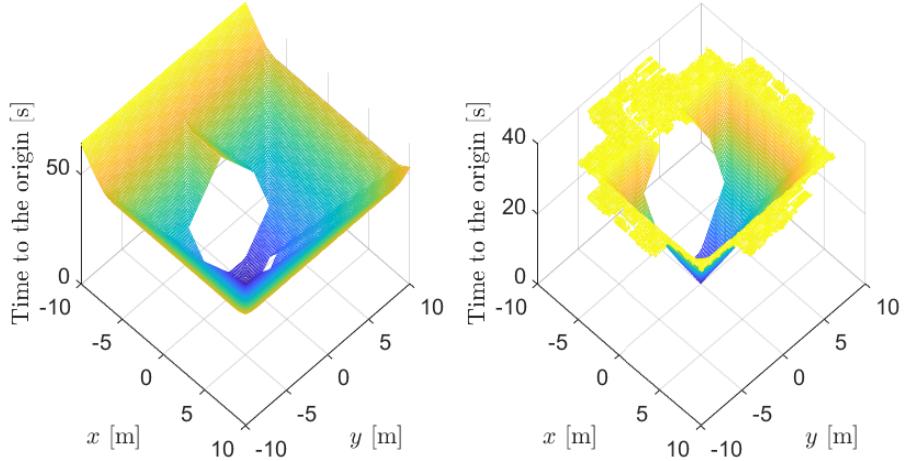


Figure 2: Time to reach the origin employing the *Harmonic* transformation and the *Kruzkov* transformation in Example 1. The left plot represents the solution found by the *Harmonic* transformation while the right plot represents the solution found by the *Kruzkov* transformation.

from any state (x_1, x_2, x_3) to the goal position $(-0.2, 0.2, 0)$. Note that, from the system dynamics, the flow vector field is described as

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} -1 & 1.2094 & 0.6937 \\ -1.2094 & -1 & 2.6564 \\ -0.6937 & -2.6564 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

and represent the *drift* dynamics, whereas the term

$$\begin{bmatrix} -0.2415 & 0.3971 & 0.8855 \\ -0.9701 & -0.0744 & -0.2312 \\ -0.0259 & -0.9148 & 0.4031 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

represent the *steering* terms for this linear system.

This problem was solved using the **CPI** (Alg. 1) and the **MEPI** (Alg. 2), and in both cases, the space was discretized by a structured grid of 1332 points (with the grid points equally distributed in space, and with the goal added to the grid). The step size used for the Trapezoidal method was $\Delta t = 0.1$.

For the **CPI**, we considered 14 logarithmically spaced values of α_i ranging from 0.01 to 1.0, and the allowable inputs were discretized considering 9

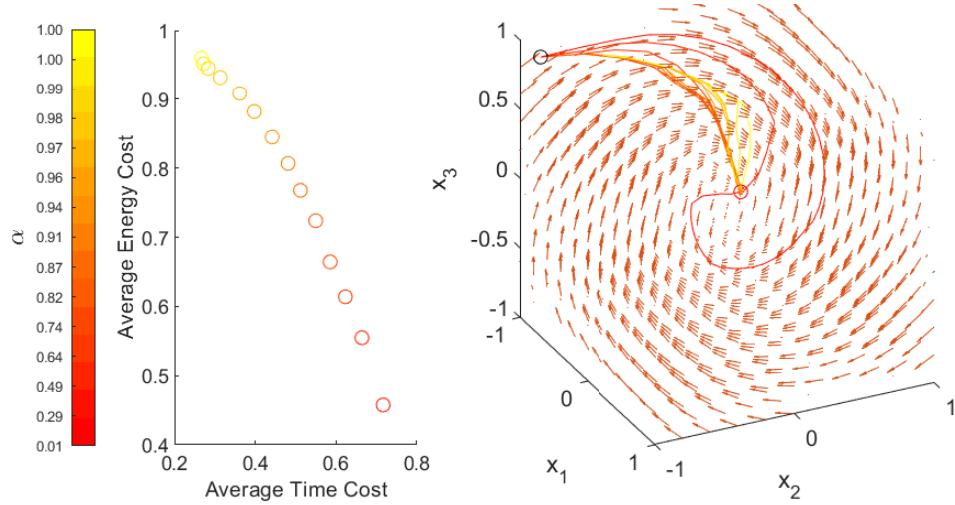


Figure 3: Result running the *Concurrent Policy Iteration* for Example 2. The left plot represents the solution set found, while the right plot shows how each solution corresponds to a different path in state space, for initial point $(-0.9, -0.9, 0.9)$. The red arrows represent the flow vector field, and the solutions are color-coded to match each trajectory in state space with a point on the objective function space, with yellow being the fastest trajectory and red the trajectory that spends the least amount of energy.

points in each direction (v_1, v_2 and v_3), leading to 729 possible control actions. The algorithm ran 15 iterations to find the result presented in Fig. 3.

For the MEPI, we considered a population of 20 individuals, with $n_{cp} = 15$, $n_{par} = 3$ and $\sigma = 0.2$ as the standard deviation for the Gaussian mutation. The initial population was randomly chosen using a uniform distribution over the allowable controls. The algorithm ran for 60 generations to find the result presented in Figure 4.

For this example, both algorithms were able to find similar solutions. It is important to note that, even though for this example the CPI was able to find a set of solutions that seem well-distributed in the objective space, that is not always the case. Finding a suitable set of α_i values can take some time. This problem does not happen with MEPI since it ranks solutions based on the *crowding distance* metric.

5.3. Example 3 - scenario with obstacles

As a third example, we consider an agent whose drift dynamics approximate a vortex with constant flow velocity centered at $(0.5, 0)$, and complete

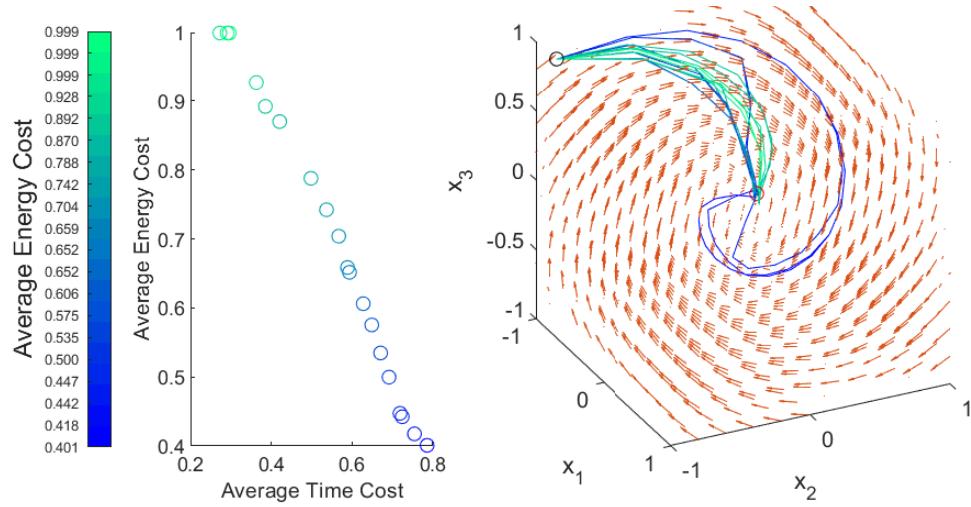


Figure 4: Result running the *Multi-objective Evolutionary Policy Iteration* for Example 2. The left plot represents the final population found while the right plot shows how each solution corresponds to a different path in state space, for initial point $(-0.9, -0.9, 0.9)$. The red arrows represent the flow vector field, and the solutions are color-coded to match each trajectory in state space with a point on the objective function space, with green being the fastest trajectory and blue the trajectory that spends the least amount of energy.

actuation. The dynamics can be described by:

$$\bar{\mathbf{f}}(x, y) = \begin{bmatrix} -1 & 3 \\ -3 & -1 \end{bmatrix} \begin{bmatrix} x - 0.5 \\ y \end{bmatrix},$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \frac{1}{0.01 + \|\bar{\mathbf{f}}(x, y)\|_2} \bar{\mathbf{f}}(x, y) + \begin{bmatrix} v_x \\ v_y \end{bmatrix},$$

with x and $y \in [-1, 1]$ representing the position on the plane, and v_x and $v_y \in [-2, 2]$ the agent's velocities relative to the drift field, used as control inputs. The control objective is to drive the agent from any state (x, y) to the goal position $(-0.5, 0.6)$ while avoiding the obstacle. Once again this problem was solved by employing both, **CPI** and **MEPI**. The space was originally discretized by 596 points, however, with this discretization, the **CPI** was not capable of finding suitable solutions. Therefore, in this example, the **MEPI** used 596 points to discretize the state space, while **CPI** used 796 points (in both cases these points include the desired goal position and 95 points describing the boundaries of the obstacle). The step size of the trapezoidal method used was $\Delta t = 0.05$ for both methods.

For the **CPI**, we considered 15 logarithmically spaced values of α_i ranging from 0.01 to 1, and the allowable inputs were discretized considering 15 points in each direction (v_x and v_y), leading to 225 possible control actions. The algorithm ran for 30 iterations to find the result presented in Figure 5.

For **MEPI**, we considered a population with 20 individuals, with $n_{cp} = 15$, $n_{par} = 3$, and $\sigma = 0.2$ as the standard deviation for the Gaussian mutation. The initial population was randomly chosen using a uniform distribution over the allowable controls. The algorithm ran for 100 generations to find the result presented in Figure 6.

For this example, unlike with Example 1, the solutions were considerably different (looking at the objective function space) and the solutions found with MEPI have a substantially smaller energy cost than the ones from CPI. This can be explained by the fact that CPI uses a discrete set of control actions, while MEPI can use a continuous set. It is also interesting to note that, with both methods, the optimal time paths chose the shortest path to the goal, while the energy optimal path tries to make the most use of the drift dynamics possible.

5.3.1. Comparison with RRT*

Additionally, we have also employed another optimal path planning algorithm from the literature, RRT* [32, 33, 34]. In this example, to generate

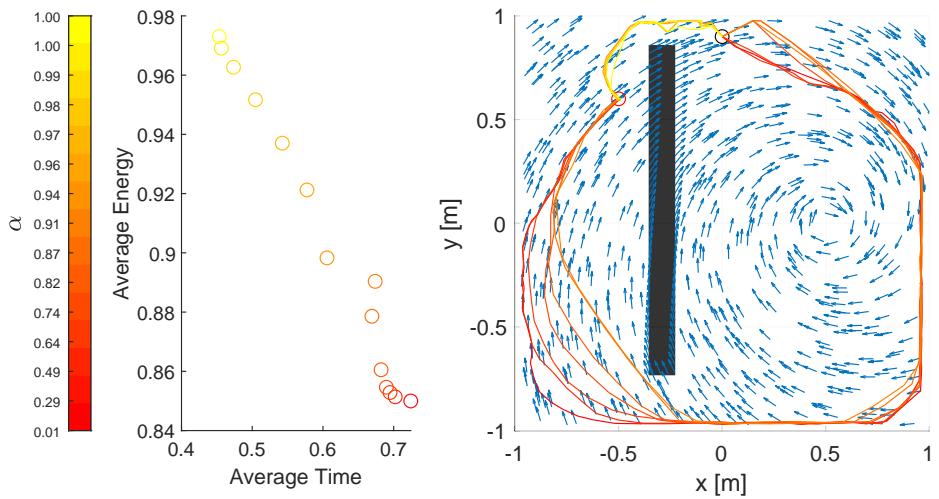


Figure 5: Result found when running *Concurrent Policy Iteration* for Example 3. The left plot represents the solutions found while the right plot shows how each solution corresponds to a different path in state space, for initial point $(0, 0.9)$. The **blue arrows** represent the flow vector field, the **black region** represent an obstacle, and the solutions are color-coded to match each trajectory in state space with a point on the objective function space, with **yellow** being the fastest trajectory and **red** the trajectory that spends the least amount of energy.

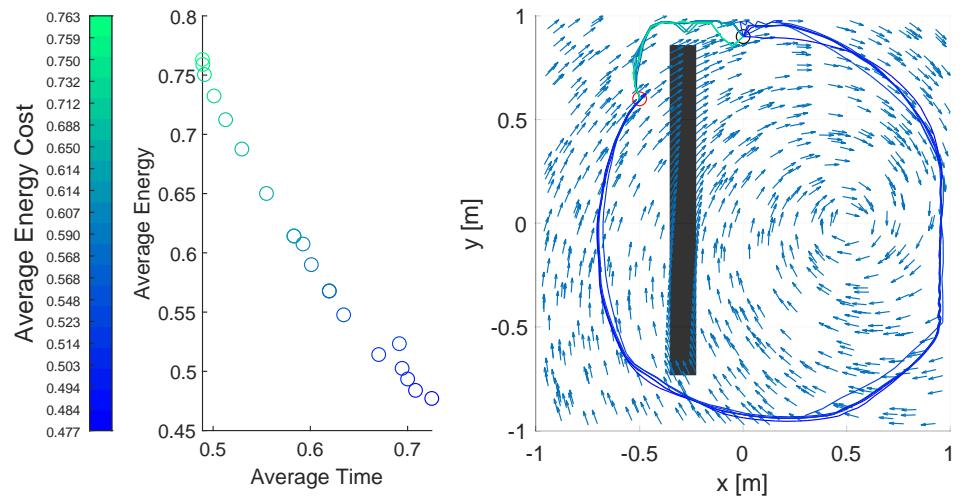


Figure 6: Result found when running *Multi-objective Evolutionary Policy Iteration* for Example 3. The left plot represents the final population found while the right plot shows how each solution corresponds to a different path in state space, for initial point $(0, 0.9)$. The **blue arrows** represent the flow vector field, the **black region** represent an obstacle, and the solutions are color-coded to match each trajectory in state space with a point on the objective function space, with **green** being the fastest trajectory and **blue** the trajectory that spends the least amount of energy.

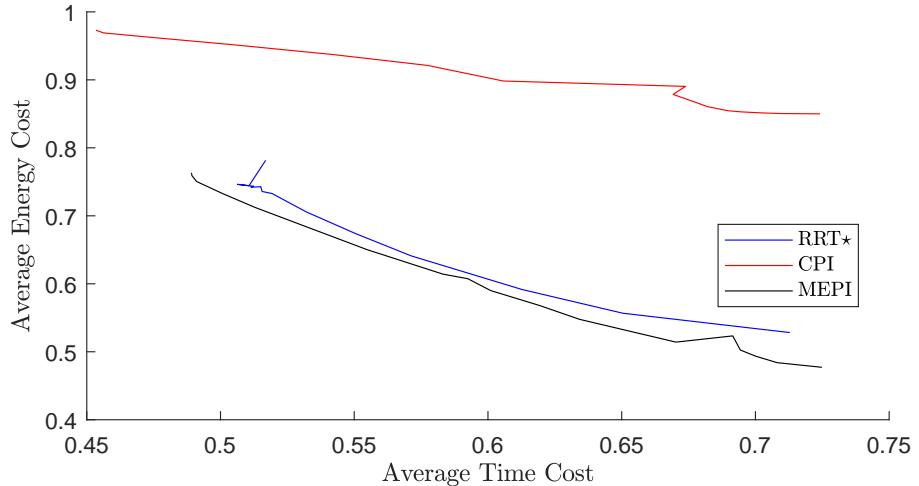


Figure 7: Comparison of the Pareto set found in Example 3 by employing an RRT*, *Concurrent Policy Iteration* and *Multi-objective Evolutionary Policy Iteration*. The blue line indicates the RRT* solutions, the red line indicates the CPI solutions and the black line indicates the MEPI solutions.

paths from every point on the state space to the desired target (similarly to what our algorithms do), we employed a single tree starting from the desired target and evolving backwards in time. We ran the algorithm 15 times, with the same α_i values as in the CPI, corresponding to different scalar cost functions for each case. We considered an Euler time-discretization and the time step was reduced to $\Delta t = 0.025$ to compensate (concerning our use of a trapezoidal method in our methods). With this time-discretization employed, the neighborhood sets were calculated by the reachable sets over a single time step (forward reachable sets for the rewiring step, and backward reachable sets for choosing optimal parents) so that the RRT* steering could be done via quadratic programming. To achieve similar accuracy as the methods presented in this paper, the RRT* ran until the tree had 6000 points.

A comparison among the Pareto Sets found in this Example is presented in Figure 7. It can be seen that, for this particular example, the Pareto Set found using MEPI and RRT* are quite similar, with the MEPI dominating the RRT* solutions by a small margin. The CPI solutions are almost completely dominated (aside from the solutions near the minimum time solution) by the other approaches. As previously discussed, this can be explained by the fact that CPI uses a discrete set of control actions whereas the other two

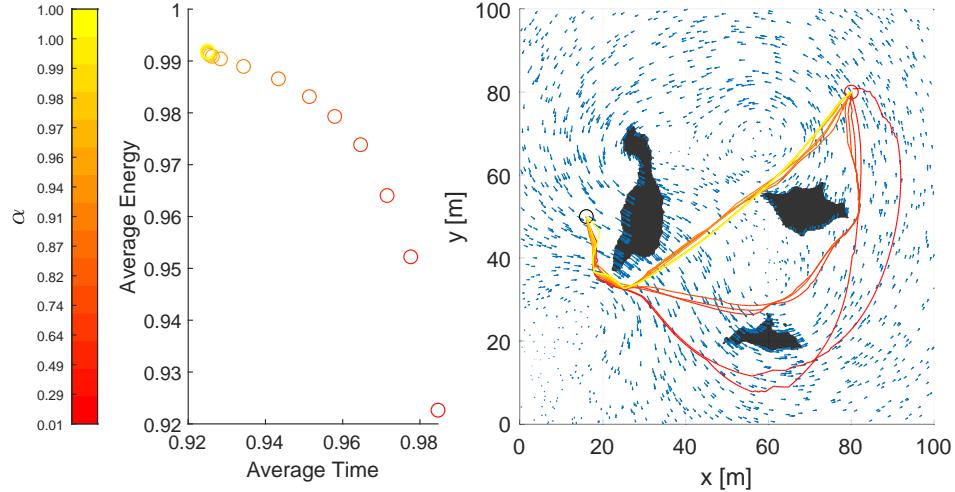


Figure 8: Result found when running *Concurrent Policy Iteration* for Example 4. The left plot represents the solutions found while the right plot shows how each solution corresponds to a different path in state space, for the initial point $(16, 50)$. The **blue arrows** represent the flow vector field, the **black regions** represent obstacles, and the solutions are color-coded to match each trajectory in state space (on the right) with a point on the objective function space (on the left), with **yellow** being the fastest trajectory and **red** the trajectory that spends the least amount of energy.

approaches employ a continuous set of control actions, which seems to be a key aspect in finding minimum energy solutions for this particular example.

5.4. Example 4 - marine navigation problem

In this fourth example, we consider a problem in which the agent can be regarded as a marine vessel moving through an ocean environment. In this case, the flow vector field describes the ocean currents, whereas the steering matrix describes the allowable input velocity directions for the vehicle.

Inspired by [35, 14] we consider an ocean current model in \mathbb{R}^2 given by the superposition of different one-point vortex solutions called *viscous Lamb vortices*, given by

$$\mathbf{f}_i(\mathbf{x}) = \Gamma_i \begin{bmatrix} -\frac{y - c_{2i}}{2\pi(\mathbf{x} - \mathbf{c}_i)^T(\mathbf{x} - \mathbf{c}_i)} \left(1 - e^{-\frac{(\mathbf{x} - \mathbf{c}_i)^T(\mathbf{x} - \mathbf{c}_i)}{\delta_i^2}} \right) \\ \frac{x - c_{1i}}{2\pi(\mathbf{x} - \mathbf{c}_i)^T(\mathbf{x} - \mathbf{c}_i)} \left(1 - e^{-\frac{(\mathbf{x} - \mathbf{c}_i)^T(\mathbf{x} - \mathbf{c}_i)}{\delta_i^2}} \right) \end{bmatrix},$$

in which, $\mathbf{x} = (x, y)$ is the vessel position, $\mathbf{c}_i = (c_{1i}, c_{2i})$ is the center of the

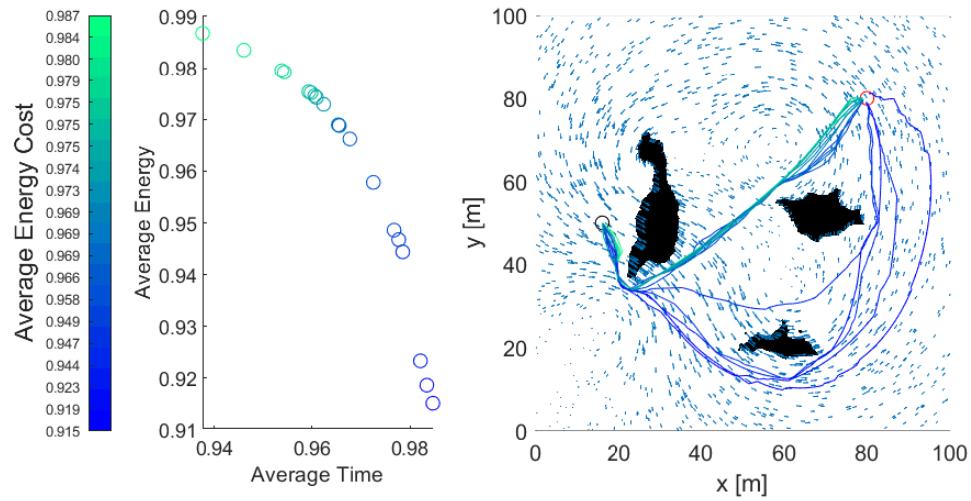


Figure 9: Result found when running *Multi-objective Evolutionary Policy Iteration* for Example 4. The left plot represents the final population found while the right plot shows how each solution corresponds to a different path in state space, for the initial point $(16, 50)$. The **blue arrows** represent the flow vector field, the **black regions** represent the obstacles, and the solutions are color-coded to match each trajectory in state space with a point on the objective function space, with **green** being the fastest trajectory and **blue** the trajectory that spends the least amount of energy.

i -th vortex, and δ_i and Γ_i are parameters related to its radius and strength. In this example, we consider that four vortices are used, so that the agent dynamics can be described by

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \left(\sum_{i=1}^4 \mathbf{f}_i(\mathbf{x}) \right) + \begin{bmatrix} v_x \\ v_y \end{bmatrix},$$

with $v_x \in [-3, 3]$ and $v_y \in [-3, 3]$ being the input velocities, and parameters $\Gamma_1 = -50$, $\Gamma_2 = \Gamma_3 = \Gamma_4 = 50$, $\delta_1 = \delta_2 = \delta_3 = \delta_4 = 10$, $\mathbf{c}_1 = (20, 30)$, $\mathbf{c}_2 = (60, 70)$, $\mathbf{c}_3 = (27, 65)$, and $\mathbf{c}_4 = (60, 30)$. The environment is a 100m x 100m square ($x \in [0, 100]$, $y \in [0, 100]$) with island shaped objects and is depicted in Figures 8 and 9. The goal is to drive the vessel from any point (x, y) in this area to the goal position (80, 80) while avoiding the obstacles.

Similarly to the other examples, this problem was solved by employing CPI and MEPI. In both cases, the space was discretized by an unstructured grid of 3412 points and the step size used for the Trapezoidal method was $\Delta t = 1$.

For CPI, we considered 15 logarithmically spaced values of α_i ranging from 0.01 to 1, and the allowable inputs were discretized considering 15 points in each direction (v_x and v_y), leading to 225 possible control actions. The algorithm ran for 30 iterations to find the result presented in Figure 8.

For MEPI, we considered a population with 20 individuals, with $n_{cp} = 15$, $n_{par} = 3$ and $\sigma = 0.2$ as the standard deviation for the Gaussian mutation. The initial population was randomly chosen using a uniform distribution over the allowable controls. The algorithm ran for 60 generations to find the result presented in Figure 9.

Similarly to Example 1, the solutions found in this example were comparable for both methods (though the concentration of points in the objective space was a little different in both cases). When considering the extremes of the Pareto Set of solutions found, CPI was able to find the best time-optimal solution, whereas MEPI was able to find the best energy-optimal solution.

It is also interesting to note that, with both methods, when moving along the efficient solutions they can be *visually* divided into 3 groups of paths (considering the trajectories taken in a closed loop).

5.5. Example 5 - time-varying periodic flow field

Finally, In this last example, we consider a problem with a time-varying flow field, to illustrate that, even though the proposed approach was developed for static flow fields, it can still be employed in this case through a

simple transformation of the state space. In this case, we consider that the flow field is a time-varying periodic double-gyre, given by

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -\theta\pi \sin(\pi(a(t)x_1^2 + b(t)x_1)) \cos(\pi x_2) \\ \theta\pi(2a(t)x_1 + b(t)) \cos(\pi(a(t)x_1^2 + b(t)x_1)) \sin(\pi x_2) \end{bmatrix} + \begin{bmatrix} v_{x_1} \\ v_{x_2} \end{bmatrix}, \quad (20)$$

with $x_1 \in [0, 2]$ and $x_2 \in [0, 1]$ the state space coordinates, $v_{x_1} \in [-0.8, 0.8]$ and $v_{x_2} \in [-0.8, 0.8]$ being the input velocities, $a(t)$ and $b(t)$ periodic signals given by

$$\begin{aligned} a(t) &= \varepsilon \sin(\omega t), \\ b(t) &= 1 - 2\varepsilon \sin(\omega t), \end{aligned}$$

and parameters $\theta = 0.1$, $\varepsilon = 0.25$ and $\omega = \frac{2\pi}{5}$. The control objective in this example is to drive the agent from any state (x_1, x_2) to the goal position $(1.5, 0.5)$. The time-varying flow field and the goal position are illustrated in Fig. 10.

Even though the methods presented in this paper were not directly developed to deal with time-varying flow fields, note that they can still deal with these problems if we augment the system's state space description to include the time t as a state (therefore guaranteeing that all of the system's dynamics can be directly defined simply by the states and the control inputs). In that regard, for our implementation, we consider a dynamics defined as

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{t} \end{bmatrix} = \begin{bmatrix} -\theta\pi \sin(\pi(a(t)x_1^2 + b(t)x_1)) \cos(\pi x_2) \\ \theta\pi(2a(t)x_1 + b(t)) \cos(\pi(a(t)x_1^2 + b(t)x_1)) \sin(\pi x_2) \\ 1 \end{bmatrix} + \begin{bmatrix} v_{x_1} \\ v_{x_2} \\ 0 \end{bmatrix}.$$

Like the other examples before it, this problem was solved using the [CPI](#) (Alg. 1) and the [MEPI](#) (Alg. 2), and in both cases, the space was discretized by a structured grid of 5625 points (with a $15 \times 15 \times 25$ grid over x_1 , x_2 and t). The step size used for the Trapezoidal method was $\Delta t = 0.2$ (matching the step size on the grid for t). Since the flow field employed is periodic, time loops on itself every 5 seconds, and as such we only represent time over $[0, 5]$.

For CPI, we considered 12 logarithmically spaced values of α_i ranging from 0.01 to 1, and the allowable inputs were discretized considering 15 points in each direction (v_{x_1} and v_{x_2}), leading to 225 possible control actions. The algorithm ran 14 iterations to find the result presented in Figure 11.

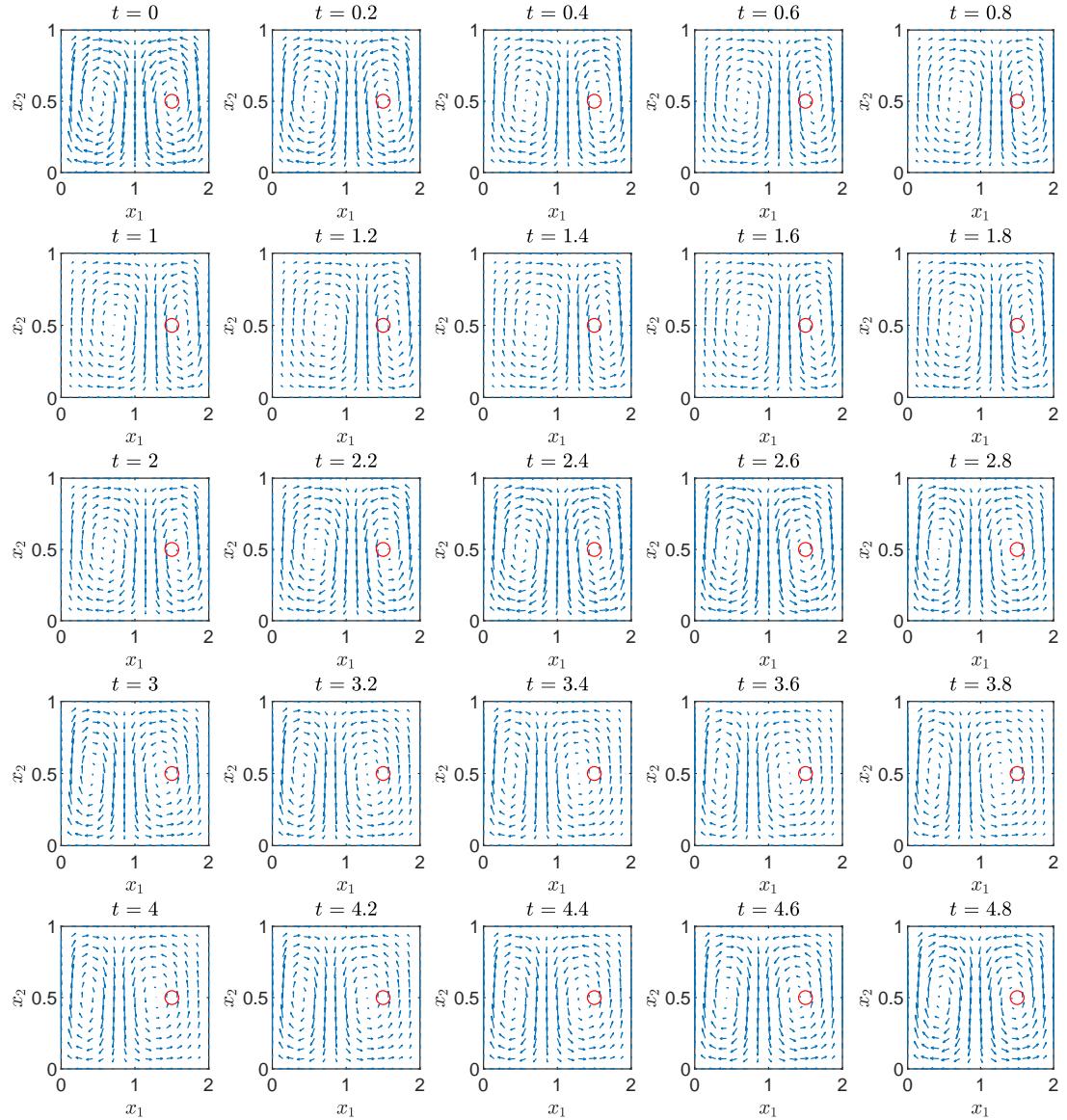


Figure 10: Time-varying flow field for (20) in Example 5. Each plot represents a time snapshot of the flow field, and the red circle represents the desired goal in this example. Note that, even though the flow field is time-varying, it is periodic with a period of 5, therefore the snapshots represent one period. Video: <https://youtu.be/ycnHhr4KtrQ>

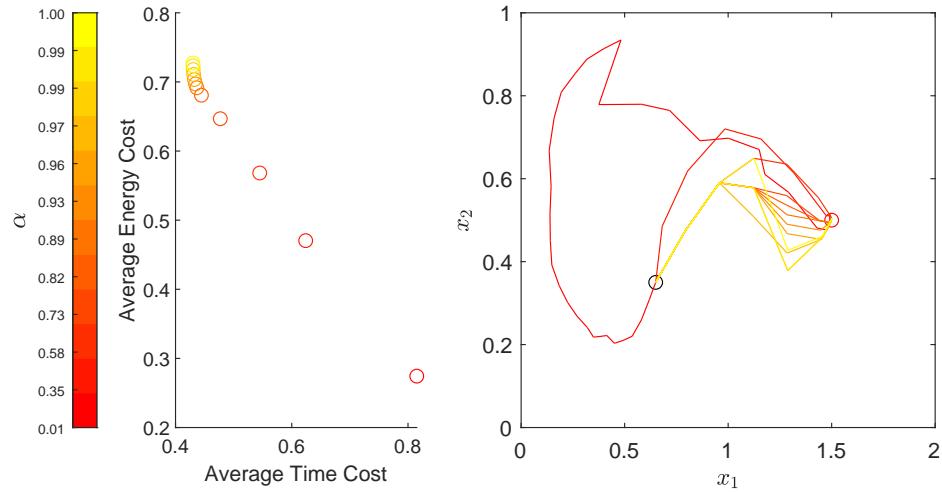


Figure 11: Result found when running *Concurrent Policy Iteration* for Example 5. The left plot represents the solutions found while the right plot shows how each solution corresponds to a different path in state space, for initial point $(0.65, 0.35)$. The solutions are color-coded to match each trajectory in state space (on the right) with a point on the objective function space (on the left), with yellow being the fastest trajectory and red the trajectory that spends the least amount of energy.

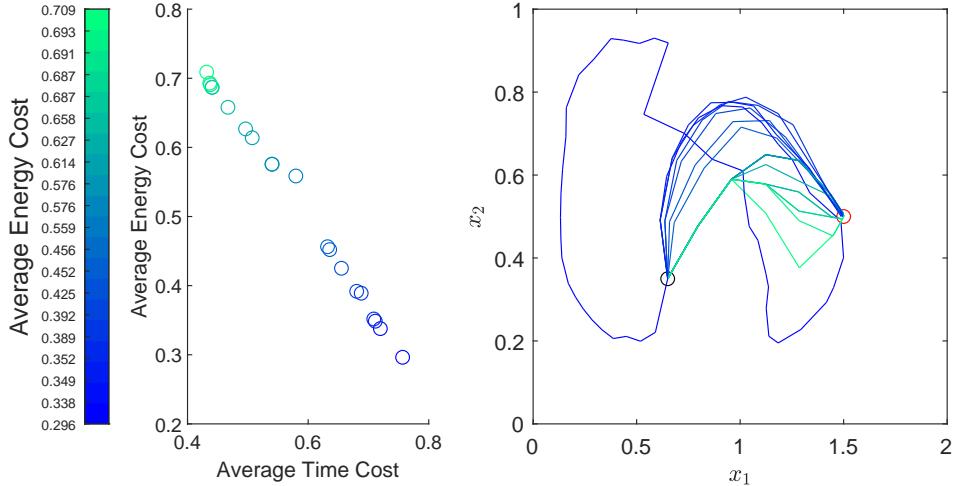


Figure 12: Result found when running *Multi-objective Evolutionary Policy Iteration* for Example 5. The left plot represents the final population found while the right plot shows how each solution corresponds to a different path in state space, for the initial point $(0.65, 0.35)$. The solutions are color-coded to match each trajectory in state space with a point on the objective function space, with green being the fastest trajectory and blue the trajectory that spends the least amount of energy.

For MEPI, we considered a population with 20 individuals, with $n_{cp} = 20$, $n_{par} = 3$ and $\sigma = 0.2$ as the standard deviation for the Gaussian mutation. The initial population was randomly chosen using a uniform distribution over the allowable controls. The algorithm ran for 30 generations to find the result presented in Figure 12.

Aside from the minimal energy solution (which looks a bit different for both methods), the trajectories found for both methods are quite similar. Unlike the previous examples, though, it is hard to see those trajectories over the flow field in a single plot. In that regard, Figure 13 illustrates one of the trajectories found over several time snapshots of the flow field.

6. Conclusion and Future Work

Path planning approaches are of paramount importance in Robotics, as they enable autonomous robots to navigate complex and dynamic environments with efficiency and safety. This paper proposes a multi-objective path planning formulation to determine paths that simultaneously optimize travel time and energy consumption.

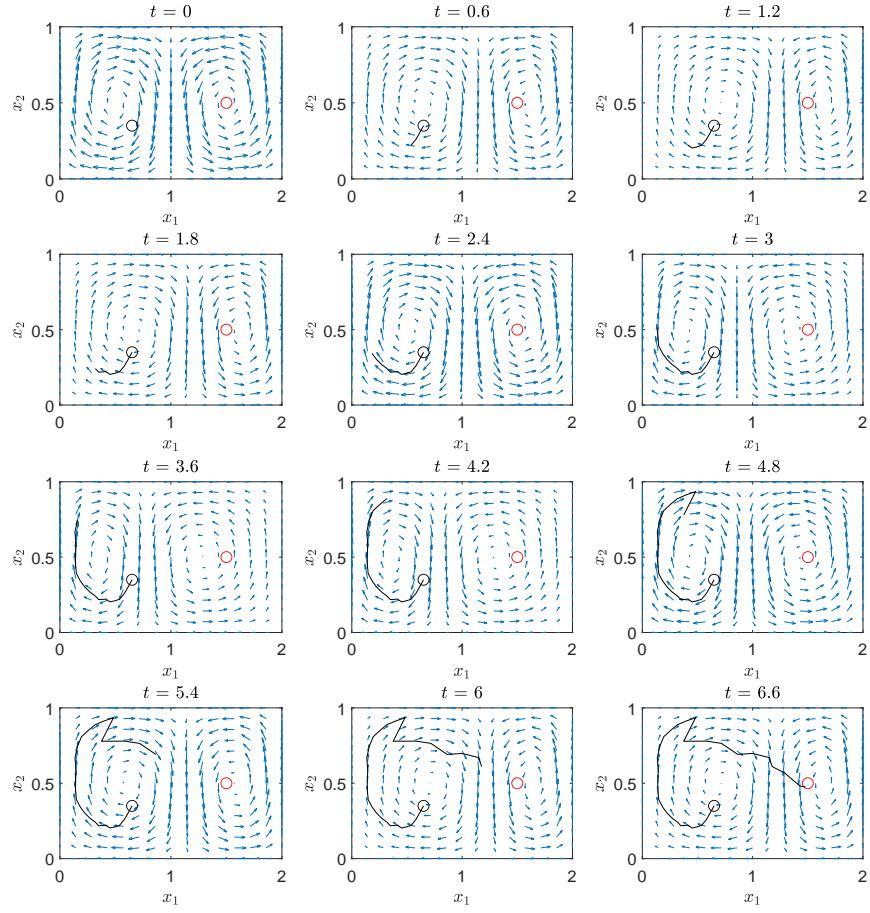


Figure 13: Time *snapshots* of one of the trajectory solutions found by *Concurrent Policy Iteration* in Example 5. Each plot represents a time snapshot of the time-varying flow field, in blue, the trajectory is displayed in black, and the red circle represents the desired goal in this example. Video: https://youtu.be/p0eKqx_r4u0

The presented methodology incorporates the dynamic influence of environmental flow fields and considers obstacles and forbidden zones. Our approach relies on the proposed Harmonic Transformation, which maps values onto a specific range, effectively mitigating potential numerical issues.

Here we presented two distinct approaches to determine the set of Pareto efficient solutions within the context of multi-objective optimization. The first approach is deterministic, involving the simultaneous solution of multiple single-objective optimizations. This deterministic method capitalizes on the parallel resolution of individual objectives to achieve Pareto efficiency. The second approach takes on an evolutionary perspective. By employing principles of evolution and selection, this evolutionary approach explores the solution space to uncover more adaptive and comprehensive Pareto optimal solutions.

As demonstrated in the examples, both methods were capable of solving the multi-objective optimal planning problem, though with different characteristics. [CPI](#) usually demonstrated a faster convergence, in practice, and was better at finding solutions close to the time-optimal solution. However, since the optimization (in line 10 of Algorithm 1) is done from a discretization of the set of allowable controls, its results were worse than [MEPI](#) when considering the energy cost (especially in Example 2). [MEPI](#), on the other hand, was better suited to finding solutions close to the energy-optimal solution. Since it is only asymptotically optimal (meaning that it converges to the optimal value with probability 1), determining a suitable stop criterion is harder (a premature stopping of the method usually leads to suboptimal solutions).

Even though the proposed approaches are capable of numerically solving the examples presented in this paper, they still suffer from the *curse of dimensionality* since an increase in the number of dimensions usually demands an exponential increase in the number of points discretizing the state-space. As such, trying to overcome some of the computational limitations of the proposed approach, in future work, we intend to study: the use of sparse [36] and adaptive grids [37, 38], to be capable of discretizing the state space with a reduced number of grid points; domain decomposition methods [39], to allow for faster parallel implementations as well as higher dimensional spaces; the use of *Physics-Informed Neural Networks* as an alternative to the solution of our transformed HJB equations [40, 41], avoiding the need for the space discretization step; and Reinforcement Learning approaches [42], in an actor-critic setting, which can be very interesting to deal with a continuous

set of control actions.

Acknowledgments

This work has been financed by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) through the Academic Excellence Program (PROEX), Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) - grant numbers 310446/2021-0, 306286/2020-3, 407063/2021-8 and 308597/2023-0, and Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG) - grant number APQ-02228-22.

References

- [1] S. M. LaValle, Planning Algorithms, Cambridge University Press, New York, NY, USA, 2006.
- [2] N. Hohmann, M. Bujny, J. Adamy, M. Olhofer, Hybrid Evolutionary Approach to Multi-objective Path Planning for UAVs, in: IEEE Symp. Series on Computational Intelligence (SSCI), 2021, pp. 1–8.
- [3] K. Cole, A. M. Wickenheiser, Reactive trajectory generation for multiple vehicles in unknown environments with wind disturbances, IEEE Trans. on Robotics 34 (5) (2018) 1333–1348.
- [4] K. Weekly, A. Tinka, L. Anderson, A. M. Bayen, Autonomous river navigation using the hamilton–jacobi framework for underactuated vehicles, IEEE Trans. on Robotics 30 (5) (2014) 1250–1255.
- [5] M. Falcone, R. Ferretti, Semi-Lagrangian approximation schemes for linear and Hamilton–Jacobi equations, SIAM, 2013.
- [6] M. N. Zafar, J. Mohanta, Methodology for Path Planning and Optimization of Mobile Robots: A Review, Procedia Computer Science 133 (2018) 141–152, international Conference on Robotics and Smart Manufacturing (RoSMA2018).
- [7] E. S. Low, P. Ong, K. C. Cheah, Solving the optimal path planning of a mobile robot using improved Q-learning, Robotics and Autonomous Systems 115 (2019) 143–161.

- [8] A. Gasparetto, P. Boscaroli, A. Lanzutti, R. Vidoni, Path Planning and Trajectory Planning Algorithms: A General Overview, Springer International Publishing, Cham, 2015, pp. 3–27.
- [9] P. Foehn, A. Romero, D. Scaramuzza, Time-optimal planning for quadrotor waypoint flight, *Science Robotics* 6 (56) (2021) eabh1221.
- [10] F. Ahmed, K. Deb, Multi-objective optimal path planning using elitist non-dominated sorting genetic algorithms, *Soft Computing* 17 (7) (2013) 1283–1299.
- [11] Y. Xue, J.-Q. Sun, Solving the Path Planning Problem in Mobile Robotics with the Multi-Objective Evolutionary Algorithm, *Applied Sciences* 8 (9) (2018).
- [12] Y. Ma, M. Hu, X. Yan, Multi-objective path planning for unmanned surface vehicle with currents effects, *ISA Transactions* 75 (2018) 137–156.
- [13] D. G. Macharet, A. Alves Neto, D. Shishika, Minimal Exposure Dubins Orienteering Problem, *IEEE Robotics and Automation Letters* 6 (2) (2021) 2280–2287.
- [14] A. Mansfield, D. G. Macharet, M. A. Hsieh, Energy-efficient orienteering problem in the presence of ocean currents, in: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2022, pp. 10081–10087.
- [15] A. Mansfield, D. G. Macharet, M. A. Hsieh, Energy-efficient team orienteering problem in the presence of time-varying ocean currents, in: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2023.
- [16] M. M. Doshi, M. S. Bhabra, P. F. Lermusiaux, Energy-time optimal path planning in dynamic flows: Theory and schemes, *Computer Methods in Applied Mechanics and Engineering* 405 (2023) 115865.
- [17] W. Zhang, J. Liu, Y. Liu, J. Liu, S. Tan, A many-objective evolutionary algorithm under diversity-first selection based framework, *Expert Systems with Applications* 250 (2024) 123949.
- [18] J. Weise, S. Mostaghim, A scalable many-objective pathfinding benchmark suite, *IEEE Transactions on Evolutionary Computation* 26 (1) (2021) 188–194.

- [19] D. Kularatne, S. Bhattacharya, M. A. Hsieh, Going with the flow: a graph based approach to optimal path planning in general flows, *Autonomous Robots* 42 (2018) 1369–1387.
- [20] A. Alvarez, A. Caiti, R. Onken, Evolutionary path planning for autonomous underwater vehicles in a variable ocean, *IEEE J. of Oceanic Engineering* 29 (2) (2004) 418–429.
- [21] K. C. To, K. M. B. Lee, C. Yoo, S. Anstee, R. Fitch, Streamlines for motion planning in underwater currents, in: Int. Conf. on Robotics and Automation (ICRA), IEEE, 2019, pp. 4619–4625.
- [22] C. Cheng, Q. Sha, B. He, G. Li, Path planning and obstacle avoidance for auv: A review, *Ocean Engineering* 235 (2021) 109355.
- [23] C. Yoo, J. J. H. Lee, S. Anstee, R. Fitch, Path planning in uncertain ocean currents using ensemble forecasts, in: IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2021, pp. 8323–8329.
- [24] F. H. Kong, K. C. To, G. Brasington, S. Anstee, R. Fitch, 3d ensemble-based online oceanic flow field estimation for underwater glider path planning, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2021, pp. 4358–4365.
- [25] Y. Yu, H. Zheng, W. Xu, Learning and sampling-based informative path planning for auvs in ocean current fields, *IEEE Transactions on Systems, Man, and Cybernetics: Systems* (2024).
- [26] H. M. Soner, Optimal control with state-space constraint I, *SIAM Journal on Control and Optimization* 24 (3) (1986) 552–561. [doi: 10.1137/0324032](https://doi.org/10.1137/0324032).
- [27] M. S. Santos, J. Rust, Convergence properties of policy iteration, *SIAM J. on Control and Optimization* 42 (6) (2004) 2094–2115.
- [28] R. Hollanders, J.-C. Delvenne, R. M. Jungers, The complexity of policy iteration is exponential for discounted markov decision processes, in: IEEE Conf. on Decision and Control (CDC), 2012, pp. 5997–6002.
- [29] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE Trans. on Evolutionary Computation* 6 (2) (2002) 182–197.

- [30] H. S. Chang, H.-G. Lee, M. Fu, S. Marcus, Evolutionary policy iteration for solving markov decision processes, *IEEE Trans. on Automatic Control* 50 (11) (2005) 1804–1808.
- [31] H. Chang, J. Hu, M. Fu, S. Marcus, *Simulation-Based Algorithms for Markov Decision Processes*, Communications and Control Engineering, Springer London, 2013.
- [32] S. Karaman, E. Frazzoli, Sampling-based algorithms for optimal motion planning, *The International Journal of Robotics Research* 30 (7) (2011) 846–894. [doi:10.1177/0278364911406761](https://doi.org/10.1177/0278364911406761).
- [33] D. J. Webb, J. van den Berg, Kinodynamic RRT*: Optimal motion planning for systems with linear differential constraints (2012). [arXiv: 1205.5088](https://arxiv.org/abs/1205.5088).
- [34] S. Zhang, H. Sang, X. Sun, F. Liu, Y. Zhou, P. Yu, A multi-objective path planning method for the wave glider in the complex marine environment, *Ocean Engineering* 264 (2022) 112481. [doi:https://doi.org/10.1016/j.oceaneng.2022.112481](https://doi.org/10.1016/j.oceaneng.2022.112481).
- [35] B. Garau, A. Alvarez, G. Oliver, AUV navigation through turbulent ocean environments supported by onboard H-ADCP, in: *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2006, pp. 3556–3561.
- [36] W. Kang, L. C. Wilcox, Mitigating the curse of dimensionality: sparse grid characteristics method for optimal feedback control and HJB equations, *Computational Optimization and Applications* 68 (2) (2017) 289–315. [doi:10.1007/s10589-017-9910-0](https://doi.org/10.1007/s10589-017-9910-0).
- [37] R. Munos, A. Moore, Variable resolution discretization in optimal control, *Machine learning* 49 (2002) 291–323.
- [38] L. Grüne, W. Semmler, Using dynamic programming with adaptive grid scheme for optimal control problems in economics, *Journal of Economic Dynamics and Control* 28 (12) (2004) 2427–2456. [doi:https://doi.org/10.1016/j.jedc.2003.11.002](https://doi.org/10.1016/j.jedc.2003.11.002).
- [39] A. Festa, Domain decomposition based parallel howard’s algorithm, *Mathematics and Computers in Simulation* 147 (2018) 121–139, APPLIED SCIENTIFIC COMPUTING XIV FOR CHALLENGING AP-

PLICATIONS. [doi:`https://doi.org/10.1016/j.matcom.2017.04.008`](https://doi.org/10.1016/j.matcom.2017.04.008).

- [40] Y. Meng, R. Zhou, A. Mukherjee, M. Fitzsimmons, C. Song, J. Liu, Physics-informed neural network policy iteration: Algorithms, convergence, and verification (2024). [arXiv:2402.10119](https://arxiv.org/abs/2402.10119).
- [41] A. Shilova, T. Delliaux, P. Preux, B. Raffin, Learning HJB Viscosity Solutions with PINNs for Continuous-Time Reinforcement Learning, Research Report (RR) 9541, Inria Lille - Nord Europe, CRISTAL - Centre de Recherche en Informatique, Signal et Automatique de Lille - UMR 9189, Univ. Lille, CNRS, Centrale Lille, Villeneuve d'Ascq, France; Univ. Grenoble Alps, CNRS, Inria, Grenoble INP, LIG, 38000 Grenoble, France (2024).
- [42] A. Mukherjee, J. Liu, Bridging physics-informed neural networks with reinforcement learning: Hamilton-Jacobi-Bellman Proximal Policy Optimization (HJBPO) (2023). [arXiv:2302.00237](https://arxiv.org/abs/2302.00237).
- [43] G. Barles, P. E. Souganidis, Convergence of approximation schemes for fully nonlinear second order equations, *Asymptotic Analysis* 4 (3) (1991) 271–283.
- [44] J. Butcher, *Numerical Methods for Ordinary Differential Equations*, Wiley, 2008.
- [45] D. Cruz-Uribe, C. Neugebauer, Sharp error bounds for the trapezoidal rule and simpson's rule., *J. of Inequalities in Pure & Applied Mathematics* 3 (4) (2002).

Appendix A. Proof of Theorem 1

The proof of Theorem 1 is divided into two parts. The first one proves that the HJB equation admits a comparison principle and a unique viscosity solution, whereas the second part employs the Barles-Souganidis Theorem [43, Theorem 2.1] to show that the proposed SL numerical scheme converges to this viscosity solution.

Appendix A.1. Viscosity solution - existence and uniqueness

For the HJB equation in (12), we can write the Hamiltonian

$$H(\mathbf{x}, v, \mathbf{p}) = \sup_{\mathbf{u} \in \mathcal{U}} \left\{ -\mathbf{p} \cdot \mathbf{f}(\mathbf{x}, \mathbf{u}) - (1-v)^2 \ell(\mathbf{x}, \mathbf{u}) \right\},$$

which is easily shown to be *uniformly continuous in \mathbf{x}, v and \mathbf{p} , convex in \mathbf{p} and monotone in v* if \mathbf{f} is Lipschitz and ℓ is *positive semidefinite* and Lipschitz.

In addition to this, considering that $f(., \mathbf{u})$ and $\ell(., \mathbf{u})$ are Lipschitz in \mathbf{x} (with *moduli of continuity* L_1 and L_2 independent of \mathbf{u}), it follows that (considering that the sup is attained by \mathbf{u} for $H(\mathbf{y}, v, \mathbf{p})$)

$$\begin{aligned} H(\mathbf{x}, v, \mathbf{p}) - H(\mathbf{y}, v, \mathbf{p}) &\leq \mathbf{p} \cdot (\mathbf{f}(\mathbf{y}, \mathbf{u}) - \mathbf{f}(\mathbf{x}, \mathbf{u})) + (1-v)^2 (\ell(\mathbf{y}, \mathbf{u}) - \ell(\mathbf{x}, \mathbf{u})), \\ H(\mathbf{x}, v, \mathbf{p}) - H(\mathbf{y}, v, \mathbf{p}) &\leq L_1 \|p\| \|\mathbf{x} - \mathbf{y}\| + L_2 \|\mathbf{x} - \mathbf{y}\|, \\ H(\mathbf{x}, v, \mathbf{p}) - H(\mathbf{y}, v, \mathbf{p}) &\leq L_3 (1 + \|p\|) \|\mathbf{x} - \mathbf{y}\|, \end{aligned}$$

with $L_3 = \max(L_1, L_2)$. Since a similar bound can be similarly found for the other difference, it follows that

$$|H(\mathbf{x}, v, \mathbf{p}) - H(\mathbf{y}, v, \mathbf{p})| \leq L_3 (1 + \|p\|) \|\mathbf{x} - \mathbf{y}\|,$$

and, according to [5, Theorem 2.13], the viscosity sub and supersolutions of (12) admit a *comparison principle*, and, as such, equation (12) has a unique *viscosity solution*.

Appendix A.2. Convergence analysis

Having shown that equation (12) admits a unique viscosity solution, as well as a *comparison principle*, from the Barles-Souganidis Theorem [43, Theorem 2.1], it suffices to show that the proposed numerical scheme is *monotone*, a *contraction mapping* and *consistent* to ensure its convergence to the *viscosity solution*.

→Monotonicity

Consider two functions \overline{W} and \overline{V} , with $\overline{W} \leq \overline{V}$ for every point on the grid. Suppose that the inf operator in (9) is attained by $\overline{\mathbf{w}}$ for \overline{W} , and $\overline{\mathbf{u}}$ for

\bar{V} . It follows that:

$$\begin{aligned}\bar{W}_k(\mathbf{x}_k) &\leq \frac{\mathcal{I}_{\bar{W}_{k+1}}[\mathbf{x}_{k+1}] + \left(1 - \mathcal{I}_{\bar{W}_{k+1}}[\mathbf{x}_{k+1}]\right)g(\mathbf{x}_k, \bar{\mathbf{u}})}{1 + \left(1 - \mathcal{I}_{\bar{W}_{k+1}}[\mathbf{x}_{k+1}]\right)g(\mathbf{x}_k, \bar{\mathbf{u}})}, \\ \bar{V}_k(\mathbf{x}_k) - \bar{W}_k(\mathbf{x}_k) &\geq \frac{\mathcal{I}_{\bar{V}_{k+1}}[\mathbf{x}_{k+1}] + \left(1 - \mathcal{I}_{\bar{V}_{k+1}}[\mathbf{x}_{k+1}]\right)g(\mathbf{x}_k, \bar{\mathbf{u}})}{1 + \left(1 - \mathcal{I}_{\bar{V}_{k+1}}[\mathbf{x}_{k+1}]\right)g(\mathbf{x}_k, \bar{\mathbf{u}})} \\ &\quad - \frac{\mathcal{I}_{\bar{W}_{k+1}}[\mathbf{x}_{k+1}] + \left(1 - \mathcal{I}_{\bar{W}_{k+1}}[\mathbf{x}_{k+1}]\right)g(\mathbf{x}_k, \bar{\mathbf{u}})}{1 + \left(1 - \mathcal{I}_{\bar{W}_{k+1}}[\mathbf{x}_{k+1}]\right)g(\mathbf{x}_k, \bar{\mathbf{u}})}, \\ \bar{V}_k(\mathbf{x}_k) - \bar{W}_k(\mathbf{x}_k) &\geq \frac{\mathcal{I}_{\bar{V}_{k+1}}[\mathbf{x}_{k+1}] - \mathcal{I}_{\bar{W}_{k+1}}[\mathbf{x}_{k+1}]}{\Omega_{\bar{V}}(\mathbf{x}_k, \bar{\mathbf{u}})\Omega_{\bar{W}}(\mathbf{x}_k, \bar{\mathbf{u}})}\end{aligned}$$

with

$$\begin{aligned}\Omega_{\bar{V}}(\mathbf{x}_k, \mathbf{u}) &= \left(1 + \left(1 - \mathcal{I}_{\bar{V}_{k+1}}[\mathbf{x}_{k+1}]\right)g(\mathbf{x}_k, \mathbf{u})\right) \\ \Omega_{\bar{W}}(\mathbf{x}_k, \mathbf{u}) &= \left(1 + \left(1 - \mathcal{I}_{\bar{W}_{k+1}}[\mathbf{x}_{k+1}]\right)g(\mathbf{x}_k, \mathbf{u})\right)\end{aligned}\tag{A.1}$$

which implies that $\bar{V}_k(\mathbf{x}_k) - \bar{W}_k(\mathbf{x}_k) \geq 0$ because we employed a linear interpolation, and $\Omega_{\bar{V}}(\mathbf{x}_k, \bar{\mathbf{u}}) > 0$, $\Omega_{\bar{W}}(\mathbf{x}_k, \bar{\mathbf{u}}) > 0$, since $\bar{V} \leq 1$, $\bar{W} \leq 1$, $g(\mathbf{x}_k, \bar{\mathbf{u}}) > 0$.

→ **Contractiveness**

Considering two functions \bar{W} and \bar{V} , with $\bar{\mathbf{w}}$ minimizing \bar{W} . It follows that:

$$\bar{V}_k(\mathbf{x}_k) - \bar{W}_k(\mathbf{x}_k) \leq \frac{\mathcal{I}_{\bar{V}_{k+1}}[\mathbf{x}_{k+1}] - \mathcal{I}_{\bar{W}_{k+1}}[\mathbf{x}_{k+1}]}{\Omega_{\bar{V}}(\mathbf{x}_k, \bar{\mathbf{w}})\Omega_{\bar{W}}(\mathbf{x}_k, \bar{\mathbf{w}})}.$$

Note that, for the case in which both $\mathcal{I}_{\bar{V}_{k+1}}[\mathbf{x}_{k+1}]$ and $\mathcal{I}_{\bar{W}_{k+1}}[\mathbf{x}_{k+1}]$ are equal to 1, the inequality is trivially upper bounded by anything larger than zero. As such, we can consider, as a worst-case bound, the case in which one of them is one and the other is $1 - \varepsilon$, leading to

$$\bar{V}_k(\mathbf{x}_k) - \bar{W}_k(\mathbf{x}_k) \leq \frac{1}{1 + \varepsilon g(\mathbf{x}_k, \bar{\mathbf{w}})} \|\bar{V} - \bar{W}\|_{\infty}^{(k+1)},$$

with $\|\bar{V} - \bar{W}\|_{\infty}^{(k+1)}$ being the maximum of the error between the two functions in the next time step. Since we are assuming that $\ell(\mathbf{x}, \mathbf{u}) > 0$, then

$g(\mathbf{x}_k, \mathbf{u}_k) > \bar{g}\Delta t > 0 \quad \forall \mathbf{x}_k, \mathbf{u}_k$, and a similar bound can be found for $\bar{W}_k(\mathbf{x}_k) - \bar{V}_k(\mathbf{x}_k)$, then:

$$\|\bar{V} - \bar{W}\|_{\infty}^{(k)} \leq \frac{1}{1 + \varepsilon \bar{g} \Delta t} \|\bar{V} - \bar{W}\|_{\infty}^{(k+1)}.$$

As we solve the problem back in time, this shows that our approximation scheme is a contraction mapping. From the Banach fixed-point Theorem, it guarantees that our approximation scheme converges to a unique solution.

→ **Consistency**

We start our consistency analysis by considering the error of time discretization, comparing solutions from (5) and (8). If we consider that the inf operator is attained by $\bar{\mathbf{u}}$ in (8), it follows that:

$$\begin{aligned} \bar{v}(\mathbf{x}(t)) - \bar{v}_k(\mathbf{x}_k) &\leq \\ &\frac{\bar{v}(\mathbf{y}_x(\Delta t, \bar{\mathbf{u}})) + (1 - \bar{v}(\mathbf{y}_x(\Delta t, \bar{\mathbf{u}}))) \int_0^{\Delta t} \ell(\mathbf{y}_x(t, \bar{\mathbf{u}}), \bar{\mathbf{u}}) dt}{1 + (1 - \bar{v}(\mathbf{y}_x(\Delta t, \bar{\mathbf{u}}))) \int_0^{\Delta t} \ell(\mathbf{y}_x(t, \bar{\mathbf{u}}), \bar{\mathbf{u}}) dt} \\ &- \frac{\bar{v}_k(\mathbf{x}_{k+1}) + (1 - \bar{v}_k(\mathbf{x}_{k+1})) g(\mathbf{x}_k, \bar{\mathbf{u}})}{1 + (1 - \bar{v}_k(\mathbf{x}_{k+1})) g(\mathbf{x}_k, \bar{\mathbf{u}})}. \end{aligned}$$

Some algebraic manipulations lead to:

$$\begin{aligned} \bar{v}(\mathbf{x}(t)) - \bar{v}_k(\mathbf{x}_k) &\leq \\ &\frac{(\bar{v}(\mathbf{y}_x(\Delta t, \bar{\mathbf{u}})) - \bar{v}_k(\mathbf{x}_{k+1}))}{(1 + q(\mathbf{x}(t), \bar{\mathbf{u}}))(1 + (1 - \bar{v}_k(\mathbf{x}_{k+1})) g(\mathbf{x}_k, \bar{\mathbf{u}}))} \\ &+ \frac{(1 - \bar{v}(\mathbf{y}_x(\Delta t, \bar{\mathbf{u}})))(1 - \bar{v}_k(\mathbf{x}_{k+1})) \Delta g}{(1 + q(\mathbf{x}(t), \bar{\mathbf{u}}))(1 + (1 - \bar{v}_k(\mathbf{x}_{k+1})) g(\mathbf{x}_k, \bar{\mathbf{u}}))}, \quad (\text{A.2}) \\ q(\mathbf{x}(t), \bar{\mathbf{u}}) &= (1 - \bar{v}(\mathbf{y}_x(\Delta t, \bar{\mathbf{u}}))) \int_0^{\Delta t} \ell(\mathbf{y}_x(t, \bar{\mathbf{u}}), \bar{\mathbf{u}}) dt, \\ \Delta g &= \int_0^{\Delta t} \ell(\mathbf{y}_x(t, \bar{\mathbf{u}}), \bar{\mathbf{u}}) dt - g(\mathbf{x}_k, \bar{\mathbf{u}}). \end{aligned}$$

From [44, Lemma 406B], we know that, using a trapezoidal method to solve the system dynamics in (1) leads to a bound (over one time step)

$$\|\mathbf{y}_x(\Delta t, \bar{\mathbf{u}}) - \mathbf{x}_{k+1}\| \leq LM\Delta t^2,$$

with L the Lipschitz constant of $\mathbf{f}(\mathbf{x}, \mathbf{u})$ with regards to \mathbf{x} , and $\|\mathbf{f}(\mathbf{x}, \mathbf{u})\| \leq M$. Note that tighter bounds are available for the local truncation error

of the trapezoidal method (the error over one time step), but these usually require extra assumptions about the smoothness of \mathbf{f} . In addition to this, from [45, Corollary 1.4], considering that $\ell(t)$ is Lipschitz, we have that

$$\begin{aligned} \left| \int_0^{\Delta t} \ell(\mathbf{y}_x(t, \bar{\mathbf{u}}) dt - g(\mathbf{x}_k, \bar{\mathbf{u}}) \right| &\leq \left| \int_0^{\Delta t} \ell(\mathbf{y}_x(t, \bar{\mathbf{u}}) dt - \frac{(\ell(\mathbf{x}_k, \bar{\mathbf{u}}) + \ell(\mathbf{y}_x(\Delta t, \bar{\mathbf{u}}), \bar{\mathbf{u}}))}{2} \right| \\ &\quad + \left| \frac{(\ell(\mathbf{x}_{k+1}, \bar{\mathbf{u}}) - \ell(\mathbf{y}_x(\Delta t, \bar{\mathbf{u}}), \bar{\mathbf{u}}))}{2} \right|, \\ &\leq \frac{\Delta t^2}{8} \left(\sup \dot{\ell} - \inf \dot{\ell} \right) + KLM\Delta t^2, \end{aligned}$$

with K being the Lipschitz constant of ℓ with regards to \mathbf{x} , and C_1 some constant.

If we consider that, \bar{v} is Lipschitz continuous, it follows that

$$\begin{aligned} \left| \int_0^{\Delta t} \ell(\mathbf{y}_x(t, \bar{\mathbf{u}}), \bar{\mathbf{u}}) dt - g(\mathbf{x}_k, \bar{\mathbf{u}}) \right| &\leq C_1 \Delta t^2, \\ |\bar{v}(\mathbf{y}_x(\Delta t, \bar{\mathbf{u}})) - \bar{v}(\mathbf{x}_{k+1})| &\leq C_2 \Delta t^2, \end{aligned}$$

and that the worst case upper bound for (A.2) happens when one of the value functions is equal to 1, while the other is equal to $1 - \varepsilon$, it follows that

$$\bar{v}(\mathbf{x}(t)) - \bar{v}_k(\mathbf{x}_k) \leq \frac{C_3 \Delta t^2 + \|\bar{v} - \bar{v}_k\|_\infty}{1 + \varepsilon \bar{g} \Delta t}. \quad (\text{A.3})$$

If we consider that the inf operator is attained by \mathbf{u}^* in (5), and that

$$\hat{\mathbf{u}}_k = \frac{1}{\Delta t} \int_0^{\Delta t} \mathbf{u}^*(\tau) d\tau$$

is the control obtained by the mean of the optimal control over a time step, it follows that

$$\begin{aligned} \bar{v}_k(\mathbf{x}_k) - \bar{v}(\mathbf{x}(t)) &\leq \frac{\bar{v}_k(\mathbf{x}_{k+1}) + (1 - \bar{v}_k(\mathbf{x}_{k+1})) g(\mathbf{x}_k, \hat{\mathbf{u}}_k)}{1 + (1 - \bar{v}_k(\mathbf{x}_{k+1})) g(\mathbf{x}_k, \hat{\mathbf{u}}_k)} \\ &\quad - \frac{\bar{v}(\mathbf{y}_x(\Delta t, \mathbf{u}^*)) + (1 - \bar{v}(\mathbf{y}_x(\Delta t, \mathbf{u}^*))) \int_0^{\Delta t} \ell(\mathbf{y}_x(t, \mathbf{u}^*), \mathbf{u}^*) dt}{1 + (1 - \bar{v}(\mathbf{y}_x(\Delta t, \mathbf{u}^*))) \int_0^{\Delta t} \ell(\mathbf{y}_x(t, \mathbf{u}^*), \mathbf{u}^*) dt}. \end{aligned}$$

Some algebraic manipulations lead to:

$$\begin{aligned}
& \bar{v}_k(\mathbf{x}_k) - \bar{v}(\mathbf{x}(t)) \leq \\
& \frac{(\bar{v}_k(\mathbf{x}_{k+1}) - \bar{v}(\mathbf{y}_x(\Delta t, \mathbf{u}^*)))}{(1 + (1 - \bar{v}_k(\mathbf{x}_{k+1})) g(\mathbf{x}_k, \hat{\mathbf{u}}_k)) (1 + \bar{q}(\mathbf{x}(t), \mathbf{u}^*))} \\
& + \frac{(1 - \bar{v}(\mathbf{y}_x(\Delta t, \mathbf{u}^*))) (1 - \bar{v}_k(\mathbf{x}_{k+1})) \Delta \hat{g}}{(1 + (1 - \bar{v}_k(\mathbf{x}_{k+1})) g(\mathbf{x}_k, \hat{\mathbf{u}}_k)) (1 + \bar{q}(\mathbf{x}(t), \mathbf{u}^*))}, \tag{A.4} \\
& \bar{q}(\mathbf{x}(t), \mathbf{u}^*) = (1 - \bar{v}(\mathbf{y}_x(\Delta t, \mathbf{u}^*))) \int_0^{\Delta t} \ell(\mathbf{y}_x(t, \mathbf{u}^*), \mathbf{u}^*) dt, \\
& \Delta \hat{g} = \left(g(\mathbf{x}_k, \hat{\mathbf{u}}_k) - \int_0^{\Delta t} \ell(\mathbf{y}_x(t, \mathbf{u}^*), \mathbf{u}^*) dt \right).
\end{aligned}$$

Consider that

$$\begin{aligned}
\mathbf{y}_x(\Delta t, \mathbf{u}^*) &= \mathbf{x}_k + \int_0^{\Delta t} \mathbf{f}(\mathbf{y}_x(\tau, \mathbf{u}^*(\tau))) d\tau, \\
\mathbf{x}_{k+1} &= \mathbf{x}_k + \frac{\Delta t}{2} (\mathbf{f}(\mathbf{x}_k, \hat{\mathbf{u}}_k) + \mathbf{f}(\mathbf{x}_{k+1}, \hat{\mathbf{u}}_k)),
\end{aligned}$$

since \mathbf{f} can be decomposed, as in (2), it follows that

$$\begin{aligned}
\mathbf{y}_x(\Delta t, \mathbf{u}^*) &= \mathbf{x}_k + \int_0^{\Delta t} \mathbf{f}_1(\mathbf{y}_x(\tau, \mathbf{u}^*(\tau))) d\tau \\
&+ \int_0^{\Delta t} \frac{(F_2(\mathbf{y}_x(\tau, \mathbf{u}^*(\tau)) - F_2(\mathbf{x}_k))}{2} \mathbf{u}^*(\tau) d\tau \\
&+ \int_0^{\Delta t} \frac{(F_2(\mathbf{y}_x(\tau, \mathbf{u}^*(\tau)) - F_2(\mathbf{x}_{k+1}))}{2} \mathbf{u}^*(\tau) d\tau \\
&+ \frac{F_2(\mathbf{x}_k)}{2} \int_0^{\Delta t} \mathbf{u}^*(\tau) d\tau + \frac{F_2(\mathbf{x}_{k+1})}{2} \int_0^{\Delta t} \mathbf{u}^*(\tau) d\tau, \\
\mathbf{y}_x(\Delta t, \mathbf{u}^*) &= \mathbf{x}_k + \int_0^{\Delta t} \mathbf{f}_1(\mathbf{y}_x(\tau, \mathbf{u}^*(\tau))) d\tau \\
&+ \int_0^{\Delta t} \frac{(F_2(\mathbf{y}_x(\tau, \mathbf{u}^*(\tau)) - F_2(\mathbf{x}_k))}{2} \mathbf{u}^*(\tau) d\tau \\
&+ \int_0^{\Delta t} \frac{(F_2(\mathbf{y}_x(\tau, \mathbf{u}^*(\tau)) - F_2(\mathbf{x}_{k+1}))}{2} \mathbf{u}^*(\tau) d\tau \\
&+ \frac{F_2(\mathbf{x}_k) \Delta t}{2} \hat{\mathbf{u}}_k + \frac{F_2(\mathbf{x}_{k+1}) \Delta t}{2} \hat{\mathbf{u}}_k,
\end{aligned}$$

$$\begin{aligned}
\mathbf{y}_x(\Delta t, \mathbf{u}^*) - \mathbf{x}_{k+1} &= \int_0^{\frac{\Delta t}{2}} (\mathbf{f}_1(\mathbf{y}_x(\tau, \mathbf{u}^*(\tau)) - \mathbf{f}_1(\mathbf{x}_k)) d\tau \\
&\quad + \int_{\frac{\Delta t}{2}}^{\Delta t} (\mathbf{f}_1(\mathbf{y}_x(\tau, \mathbf{u}^*(\tau)) - \mathbf{f}_1(\mathbf{x}_{k+1})) d\tau \\
&\quad + \int_0^{\Delta t} \frac{(F_2(\mathbf{y}_x(\tau, \mathbf{u}^*(\tau)) - F_2(\mathbf{x}_k))}{2} \mathbf{u}^*(\tau) d\tau \\
&\quad + \int_0^{\Delta t} \frac{(F_2(\mathbf{y}_x(\tau, \mathbf{u}^*(\tau)) - F_2(\mathbf{x}_{k+1}))}{2} \mathbf{u}^*(\tau) d\tau.
\end{aligned}$$

If we consider that \mathbf{f}_1 and F_2 are Lipschitz in \mathbf{x} , and that \mathbf{u} is bounded, it follows that

$$\begin{aligned}
\|\mathbf{y}_x(\Delta t, \mathbf{u}^*) - \mathbf{x}_{k+1}\| &\leq L_1 \int_0^{\frac{\Delta t}{2}} \|\mathbf{y}_x(\tau, \mathbf{u}^*(\tau)) - \mathbf{x}_k\| d\tau \\
&\quad + L_1 \int_{\frac{\Delta t}{2}}^{\Delta t} \|\mathbf{y}_x(\tau, \mathbf{u}^*(\tau)) - \mathbf{x}_{k+1}\| d\tau \\
&\quad + L_2 \int_0^{\Delta t} \|\mathbf{y}_x(\tau, \mathbf{u}^*(\tau)) - \mathbf{x}_k\| d\tau \\
&\quad + L_2 \int_0^{\Delta t} \|\mathbf{y}_x(\tau, \mathbf{u}^*(\tau)) - \mathbf{x}_{k+1}\| d\tau,
\end{aligned}$$

$$\begin{aligned}
\|\mathbf{y}_x(\Delta t, \mathbf{u}^*) - \mathbf{x}_{k+1}\| &\leq L_1 \int_0^{\frac{\Delta t}{2}} \|\mathbf{y}_x(\tau, \mathbf{u}^*(\tau)) - \mathbf{x}_k\| d\tau \\
&\quad + L_1 \int_0^{\Delta t} \|\mathbf{y}_x(\tau, \mathbf{u}^*(\tau)) - \mathbf{x}_{k+1}\| d\tau \\
&\quad + L_2 \int_0^{\Delta t} \|\mathbf{y}_x(\tau, \mathbf{u}^*(\tau)) - \mathbf{x}_k\| d\tau \\
&\quad + L_2 \int_0^{\Delta t} \|\mathbf{y}_x(\tau, \mathbf{u}^*(\tau)) - \mathbf{x}_{k+1}\| d\tau.
\end{aligned}$$

Considering that $\|\mathbf{f}\| \leq M, \forall \mathbf{x}, \mathbf{u}$, it follows that

$$\|\mathbf{y}_x(\tau, \mathbf{u}^*) - \mathbf{x}_k\| \leq M\tau,$$

and we arrive at the bound

$$\begin{aligned}\|\mathbf{y}_x(\Delta t, \mathbf{u}^*) - \mathbf{x}_{k+1}\| &\leq (L_1 + L_2) \int_0^{\Delta t} \|\mathbf{y}_x(\tau, \mathbf{u}^*) - \mathbf{x}_{k+1}\| d\tau \\ &\quad + \frac{(L_1 + L_2)M\Delta t^2}{8}, \\ \|\mathbf{y}_x(\Delta t, \mathbf{u}^*) - \mathbf{x}_{k+1}\| &\leq \bar{L} \int_0^{\Delta t} \|\mathbf{y}_x(\tau, \mathbf{u}^*) - \mathbf{x}_{k+1}\| d\tau + \bar{M}\Delta t^2,\end{aligned}$$

which, from the integral form of the Gronwall-Bellman inequality leads to

$$\|\mathbf{y}_x(\Delta t, \mathbf{u}^*) - \mathbf{x}_{k+1}\| \leq \bar{M}\Delta t^2 e^{\bar{L}\Delta t}.$$

Considering that

$$\begin{aligned}g(\mathbf{x}_k, \hat{\mathbf{u}}_k) - \int_0^{\Delta t} \ell(\mathbf{y}_x(t, \mathbf{u}^*), \mathbf{u}^*) dt &= \int_0^{\frac{\Delta t}{2}} \left(\ell(\mathbf{x}_k, \hat{\mathbf{u}}) - \ell(\mathbf{x}_k, \mathbf{u}^*) + \ell(\mathbf{x}_k, \mathbf{u}^*) \right. \\ &\quad \left. - \ell(\mathbf{y}_x(t, \mathbf{u}^*), \mathbf{u}^*) \right) dt \\ &\quad + \int_{\frac{\Delta t}{2}}^{\Delta t} \left(\ell(\mathbf{x}_{k+1}, \hat{\mathbf{u}}) - \ell(\mathbf{y}_x(\Delta t, \mathbf{u}^*), \mathbf{u}^*) \right) dt \\ &\quad + \int_{\frac{\Delta t}{2}}^{\Delta t} \left(\ell(\mathbf{y}_x(\Delta t, \mathbf{u}^*), \mathbf{u}^*) - \ell(\mathbf{y}_x(t, \mathbf{u}^*), \mathbf{u}^*) \right) dt,\end{aligned}$$

and, considering that ℓ is convex in \mathbf{u} , from Jensen's inequality, it follows that

$$\begin{aligned}- \int_0^{\frac{\Delta t}{2}} \ell(\mathbf{x}_k, \mathbf{u}^*) dt &\leq -\frac{\Delta t}{2} \ell(\mathbf{x}_k, \hat{\mathbf{u}}_k) \\ - \int_{\frac{\Delta t}{2}}^{\Delta t} \ell(\mathbf{y}_x(\Delta t, \mathbf{u}^*), \mathbf{u}^*) dt &\leq -\frac{\Delta t}{2} \ell(\mathbf{y}_x(\Delta t, \mathbf{u}^*), \hat{\mathbf{u}}_k),\end{aligned}$$

so that

$$\begin{aligned}
g(\mathbf{x}_k, \hat{\mathbf{u}}_k) - \int_0^{\Delta t} \ell(\mathbf{y}_x(t, \mathbf{u}^*), \mathbf{u}^*) dt &\leq \\
&\int_0^{\frac{\Delta t}{2}} (\ell(\mathbf{x}_k, \mathbf{u}^*) - \ell(\mathbf{y}_x(t, \mathbf{u}^*), \mathbf{u}^*)) dt \\
&+ \int_{\frac{\Delta t}{2}}^{\Delta t} (\ell(\mathbf{x}_{k+1}, \hat{\mathbf{u}}_k) - \ell(\mathbf{y}_x(\Delta t, \mathbf{u}^*), \hat{\mathbf{u}}_k)) dt \\
&+ \int_{\frac{\Delta t}{2}}^{\Delta t} (\ell(\mathbf{y}_x(\Delta t, \mathbf{u}^*), \mathbf{u}^*) - \ell(\mathbf{y}_x(t, \mathbf{u}^*), \mathbf{u}^*)) dt.
\end{aligned}$$

Considering that ℓ has Lipschitz constant K with respect to \mathbf{x} , it follows that

$$\begin{aligned}
g(\mathbf{x}_k, \hat{\mathbf{u}}_k) - \int_0^{\Delta t} \ell(\mathbf{y}_x(t, \mathbf{u}^*), \mathbf{u}^*) dt &\leq \int_0^{\frac{\Delta t}{2}} KM\tau d\tau \\
&+ \int_{\frac{\Delta t}{2}}^{\Delta t} (K\bar{M}\Delta t^2 e^{\bar{L}\Delta t} + KM\tau) d\tau, \\
g(\mathbf{x}_k, \hat{\mathbf{u}}_k) - \int_0^{\Delta t} \ell(\mathbf{y}_x(t, \mathbf{u}^*), \mathbf{u}^*) dt &\leq \frac{KM\Delta t^2}{2} + \frac{K\bar{M}\Delta t^3}{2} e^{\bar{L}\Delta t}.
\end{aligned}$$

If we consider that, \bar{v} is Lipschitz continuous, it follows that

$$\begin{aligned}
g(\mathbf{x}_k, \hat{\mathbf{u}}_k) - \int_0^{\Delta t} \ell(\mathbf{y}_x(t, \mathbf{u}^*), \mathbf{u}^*) dt &\leq C_4\Delta t^2 + C_5\Delta t^3 e^{\bar{L}\Delta t} \\
|\bar{v}(\mathbf{x}_{k+1}) - \bar{v}(\mathbf{y}_x(\Delta t, \mathbf{u}^*))| &\leq C_6\Delta t^2 e^{\bar{L}\Delta t}.
\end{aligned}$$

Considering also that the worst case upper bound in (A.4) happens when one of the value functions is equal to 1, while the other is equal to $1 - \varepsilon$, it follows that

$$\bar{v}_k(\mathbf{x}_k) - \bar{v}(\mathbf{x}(t)) \leq \frac{C_4\Delta t^2 + (C_5\Delta t^3 + C_6\Delta t^2) e^{\bar{L}\Delta t} + \|\bar{v} - \bar{v}_k\|_\infty}{1 + \varepsilon\bar{g}\Delta t}. \quad (\text{A.5})$$

By combining (A.3) and (A.5), we have that

$$\begin{aligned}
\|\bar{v} - \bar{v}_k\|_\infty &\leq \frac{C_7\Delta t^2 + (C_5\Delta t^3 + C_6\Delta t^2) e^{\bar{L}\Delta t}}{\varepsilon\bar{g}\Delta t} \\
&\approx \frac{C_7\Delta t^2 + (C_5\Delta t^3 + C_6\Delta t^2)(1 + \bar{L}\Delta t)}{\varepsilon\bar{g}\Delta t},
\end{aligned}$$

in which the last approximation comes from considering that, for θ close to zero, $e^\theta \approx 1 + \theta$. Asymptotically, this bound is *dominated* by the linear term when $\Delta t \rightarrow 0$, since the higher order terms vanish faster. In that regard, we will write

$$\|\bar{v} - \bar{v}_k\|_\infty \leq C\Delta t. \quad (\text{A.6})$$

For the space discretization error, we analyze the errors of the value function on the grid points, by comparing (8) and (9). To differentiate them, we will denote the value on the grid points of (9) by \bar{V}_k . If we consider that the inf is attained by $\bar{\mathbf{u}}$ in (8), it follows that:

$$\bar{V}_k(\mathbf{x}_k) - \bar{v}_k(\mathbf{x}_k) \leq \frac{\mathcal{I}_{\bar{V}_{k+1}}[\mathbf{x}_{k+1}] - \bar{v}_{k+1}(\mathbf{x}_{k+1})}{\Omega_{\bar{V}}(\mathbf{x}_k, \bar{\mathbf{u}})\Omega_{\bar{v}}(\mathbf{x}_k, \bar{\mathbf{u}})},$$

with $\Omega_{\bar{V}}$ from (A.1) and

$$\Omega_{\bar{v}}(\mathbf{x}_k, \bar{\mathbf{u}}) = 1 + (1 - \bar{v}_{k+1}(\mathbf{x}_{k+1}))g(\mathbf{x}_k, \bar{\mathbf{u}}).$$

By considering that \bar{V} is Lipschitz, $g(\mathbf{x}_k, \bar{\mathbf{u}}) > \bar{g}\Delta t$, and that the worst case upper bound happens when one of the value functions is equal to 1, while the other is equal to $1 - \varepsilon$ leads to:

$$\bar{V}_k(\mathbf{x}_k) - \bar{v}_k(\mathbf{x}_k) \leq \frac{C_8\Delta x + \|\bar{V}_k - \bar{v}_k\|_\infty}{1 + \varepsilon\bar{g}\Delta t}.$$

with C_8 being a constant and Δx being the largest distance between any point and a grid point.

Since a similar bound can be found for $\bar{v}_k(\mathbf{x}_k) - \bar{V}_k(\mathbf{x}_k)$, then:

$$\begin{aligned} \|\bar{V}_k - \bar{v}_k\|_\infty &\leq \frac{C_8\Delta x + \|\bar{V}_k - \bar{v}_k\|_\infty}{1 + \varepsilon\bar{g}\Delta t} \\ \|\bar{V}_k - \bar{v}_k\|_\infty &\leq C_9 \frac{\Delta x}{\Delta t}. \end{aligned} \quad (\text{A.7})$$

with C_9 a constant. Combining (A.6) and (A.7), we have that:

$$\|\bar{v} - \bar{V}\|_\infty \leq C_{10}\Delta t + C_9 \frac{\Delta x}{\Delta t},$$

with C_{10} a constant. Similarly to the original Kruzkov transformation, the best coupling between Δt and Δx , in this case, is given by $\Delta x = \Delta t^2$,

indicating that **our grid resolution should be finer than our time discretization resolution.**

Since we have shown that our scheme is **monotone, a contraction mapping, and consistent**, from the Barles-Souganidis Theorem, we have proven that it is **convergent**. \square