Pontifícia Universidade Católica
do Rio de Janeiro

Hugo de Souza Oliveira

A RBF approach to the control of PDEs using Dynamic Programming equations

Tese de Doutorado

Thesis presented to the Programa de Pós–graduação em Matemática, do Departamento de Matemática da PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Matemática.

Advisor : Prof. Sinesio Pesco
Co-advisor: Prof. Alessandro Alla

Rio de Janeiro
September 2022

Pontifícia Universidade Católica do Rio de Janeiro

Hugo de Souza Oliveira

A RBF approach to the control of PDEs using Dynamic Programming equations

Thesis presented to the Programa de Pós–graduação em Matemática da PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Matemática. Approved by the Examination Committee:

Rio de Janeiro, September the 27th, 2022

Hugo de Souza Oliveira

Hugo de Souza Oliveira obtained his undergraduate degree in Economic Sciences from Universidade Federal do Rio de Janeiro (UFRJ) and a M.Sc degree in Applied Mathematics from Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio).

Bibliographic data

Oliveira, Hugo de Souza

A RBF approach to the control of PDEs using Dynamic Programming equations / Hugo de Souza Oliveira; advisor: Sinesio Pesco; co-advisor: Alessandro Alla. – 2022.

99 f: il. color. ; 30 cm

Tese (doutorado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Matemática, 2022.

Inclui bibliografia

1. Matemática – Teses. 2. Funções de Base Radial. 3. Programação Dinâmica. 4. Equações Diferenciais Parciais. 5. Problemas de Controle Ótimo. 6. Métodos Numéricos para EDPs. I. Pesco, Sinesio. II. Alla, Alessandro. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Matemática. IV. Título.

CDD: 510

Acknowledgments

I would like to acknowledge my mom for being the person by my side in any moment. There are no words to describe my gratitude here.

To Professor Alessandro to be my advisor and accepting to work with me in a very difficult moment, for the patience and academic advises. Thank you for everything.

To Professors Carlos, Mauricio, Sinesio, Ricardo and Yuri for accepting to be part of the committee.

To Professor Carlos and Professor Edgard for the support provided when I was about to drop everything. To Professor Marcos for the advices.

To Creuza, Carlos, Katia and Mariana for all the help and talks. Life at the Department is much easier with all of you.

To CAPES, for the aid granted, without which this work could not have been realized.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001

Abstract

Oliveira, Hugo de Souza; Pesco, Sinesio (Advisor); Alla, Alessandro (Co-Advisor). **A RBF approach to the control of PDEs using Dynamic Programming equations**. Rio de Janeiro, 2022. 99p. Tese de Doutorado – Departamento de Matemática, Pontifícia Universidade Católica do Rio de Janeiro.

Semi-Lagrangian schemes for the approximation of the dynamic programming principle are based on a time discretization projected on a state-space grid. The use of a structured grid makes this approach not feasible for high-dimensional problems due to the curse of dimensionality. In this thesis, we present a new approach for infinite horizon optimal control problems where the value function is computed using Radial Basis Functions (RBF) by the Shepard's moving least squares approximation method on scattered grids. We propose a new method to generate a scattered mesh driven by the dynamics and an optimal routine to select the shape parameter in the RBF. This mesh will help to localize the problem and approximate the dynamic programming principle in high dimension. Error estimates for the value function are also provided. Numerical tests for high dimensional problems will show the effectiveness of the proposed method. In addition to the optimal control of classical PDEs, we show how the method can also be applied to the control of nonlocal equations. We also provide an example analyzing the numerical convergence of a nonlocal controlled equation towards the continuous model.

Keywords

Radial Basis Functions; Dynamic Programming; Partial Differential Equations; Optimal Control Problems; Numerical Methods for PDEs.

Resumo

Oliveira, Hugo de Souza; Pesco, Sinesio; Alla, Alessandro. **Um método baseado em RBF para o controle de EDPs usando equações de Programação Dinâmica**. Rio de Janeiro, 2022. 99p. Tese de Doutorado – Departamento de Matemática, Pontifícia Universidade Católica do Rio de Janeiro.

Esquemas semi-Lagrangeanos usados para a aproximação do princípio da programação dinâmica são baseados em uma discretização temporal reconstruída no espaço de estado. O uso de uma malha estruturada torna essa abordagem inviável para problemas de alta dimensão devido à maldição da dimensionalidade. Nesta tese, apresentamos uma nova abordagem para problemas de controle ótimo de horizonte infinito onde a função valor é calculada usando Funções de Base Radial (RBFs) pelo método de aproximação de mínimos quadrados móveis de Shepard em malhas irregulares. Propomos um novo método para gerar uma malha irregular guiada pela dinâmica e uma rotina de otimizada para selecionar o parâmetro responsável pelo formato nas RBFs. Esta malha ajudará a localizar o problema e aproximar o princípio da programação dinâmica em alta dimensão. As estimativas de erro para a função valor também são fornecidas. Testes numéricos para problemas de alta dimensão mostrarão a eficácia do método proposto. Além do controle ótimo de EDPs clássicas mostramos como o método também pode ser aplicado ao controle de equações não-locais. Também fornecemos um exemplo analisando a convergência numérica de uma equação não-local controlada para o modelo contínuo.

Palavras-chave

Funções de Base Radial; Programação Dinâmica; Equações Diferenciais Parciais; Problemas de Controle Ótimo; Métodos Numéricos para EDPs.

Table of contents

List of figures

List of tables

List of Symbols

DP	Dynamic Programming
DPP	Dynamic Programming Principle
GHJB	Generalized Hamilton-Jacobi-Bellman
HJ	Hamilton-Jacobi
HJB	Hamilton-Jacobi-Bellman
MLS	Moving Least Squares
MPC	Model Predictive Control
NN	Neural Network
PMP	Pontryagin Maximum Principle
POD	Proper Orthogonal Decomposition
RBF	Radial Basis Function
TSA	Tree Structure Algorithm
WLS	Weigted Least Squares

It is my great hope someday, to see science and decision makers rediscover what the ancients have always known. Namely that our highest currency is respect.

Nassim Nicholas Taleb, *The Black Swan.*

1
Introduction

The importance of optimal control of dynamical systems is steadily increasing in applied mathematics, as industrial applications are becoming widespread. Fields like aerospace engineering, social science, economics and the design of optimal financial trading strategies rely in the solution of this type of control problem.

In this thesis we deal with infinite horizon control problems. The aim is to minimize a cost functional like

$$\inf_{u\in\mathcal{U}} \mathcal{J}_x(u) = \int_0^\infty g(y(s), u(s))e^{-\lambda s} ds, \tag{1-1}$$

subject to a controlled dynamical system

$$\begin{cases} \dot{y}(t) = f(y(t), u(t)), & t > 0, \\ y(0) = x \in \mathbb{R}^d. \end{cases} \tag{1-2}$$

As usual, the control $u : [0,\infty) \to U \subset \mathbb{R}^m$ is taken from the set of admissible controls $\mathcal{U}$, x is the initial condition, y is the trajectory, λ is the discount factor, g is the running cost and f is the dynamics.

The solution of the optimal control problems can be obtained by two different approaches. One is called open-loop: it assumes that the optimal control $u^*(t)$ is a function depending only on the initial condition and the time variable. The second is the closed-loop and in this case the optimal control is a function of the current state $y(t)$ and it is able to adapt itself according to changes in the trajectories by a feedback law: $u^*(t) = \mu^*(t, y(t))$. Next, we discuss different methods in each approach.

Open Loop. One of the most used approaches to deal with optimal control problems is the Pontryagin Maximum Principle (PMP [1]). It gives necessary conditions to find the optimal control and the optimal trajectory that are obtained solving a two-point boundary problem in the hamiltonian system given by state and costate equations. This method is commonly used in finite horizon problems, but extensions to infinite horizon cases have been formulated (see e.g. [2, 3, 4]).

The numerical solution of the two-point boundary problem is usually done by a shooting method (single or multiple shooting) [5] that depends on the initial guesses of the control and costate variables. The selection of those variables to start the shooting method can be realized using results of a direct method.

The direct discretization method is an alternative way to solve finite horizon optimal control problems. This method consists first in a suitable discretization of the problem along a discrete time horizon and then, solving the finite dimensional nonlinear programming problem considering constraints imposed by the discrete control set and the discrete trajectory [6]. The solution here is also open-loop. A detailed discussion about this topic is found in e.g. [7, 8] along with numerical algorithms.

Closed Loop. Moving from an open-loop to a closed-loop paradigm, the Model Predictive Control (MPC) approach is based on the repeated solution of an open-loop problem for a given initial condition. The basic idea is to break the time horizon in a finite number of intervals and solve the open-loop problem obtaining a sequence of optimal controls and optimal states. The first controls from the previously computed sequence are used in the problem dynamics and a new state is observed. This new state is compared to the reference state and the open-loop problem is solved again to obtain a new sequence of controls and states. The process is repeated until the end of the time horizon (see e.g.[9, 10]). It is important to note that MPC can be used to approximate solutions for infinite time horizon problems. Numerically, after a suitable discretization, the model can be solved using linear and nonlinear programming.

We continue our discussion on closed loop control, focusing on the Dynamic Programming Principle, which is the focus of this thesis.

The Dynamic Programming Principle (DPP) was designed by Richard Bellman in the 1950s [11]. We define the value function $v(x)$

$$v(x) := \inf_{u \in \mathcal{U}} \mathcal{J}_x(u) \tag{1-3}$$

which gives the optimal cost associated with any initial condition x. This function can be characterized as the solution of a first order stationary non-linear partial differential equation, the Hamilton-Jacobi-Bellman (HJB) equation:

$$\lambda v(x) + \max_{u \in U}\{-f(x,u) \cdot \nabla v(x) - g(x,u)\} = 0.$$

The value function is known to be Lipschitz continuous under mild regularity conditions of the running cost and the dynamics (see Chapter III of [12] for a general result about Hölder continuity). The development of the theory of viscosity solutions by Crandall and Lions in the 1980s allows to characterize the value function as the unique viscosity solution of the HJB equation (see e.g. [12] and [13] for a treatment of deterministic and stochastic cases, respectively).

The knowledge of the value function permits the construction of a feedback control and the respective controlled trajectory. The analytical solution and numerical approximations of HJB equation may be difficult to obtain due to the lack of regularity. Despite this obstacle, several numerical schemes have been proposed to approximate the value function, ranging from semi-Lagrangian [14, 15] to finite differences [16, 17] and finite volumes [18], for instance. Theoretically, these schemes can be implemented to solve problems in any dimension, but computationally limitations become quickly evident.

The dimension of a HJB equation related to a specific optimal control problem inherits the dimension of the dynamical system. The dimension can be greater than 10 or even of order $\mathcal{O}(10^3)$ if the dynamical system comes from a PDE discretization. Thus, numerical approximations become computationally challenging for such problems and it is a major bottleneck for the Dynamic Programming (DP) approach, known as the *curse of dimensionality* [19].

In terms of low dimensional problems the numerical solution of the HJB equation can be obtained using standard PDE methods as cited above. We refer to e.g. [14] and [12, Appendix A] for a discussion of the method for the static HJB equation together with error estimates in the context of semi-Lagrangian schemes

The approximation of the DPP in the fixed point form is

$$v^{\Delta t}(x) = \min_{u \in U}\{\Delta t g(x,u) + (1 - \lambda \Delta t) v^{\Delta t}(x + \Delta t f(x,u))\}.$$

Under adequate assumptions and considering $\Delta t < 1/\lambda$, this scheme is contractive and the solution can be obtained by iterations on state space (*Value Iteration* algorithm).

In the right-hand side of the above equation, the term $v^{\Delta t}(x + \Delta t f(x,u))$ is the approximate value function evaluated in a point that may not be part of the grid. This calculation is usually performed using linear interpolation.

A different solution method is to iterate on the policy space instead of the state space. Such algorithm is called *Policy Iteration.* A complete treatment of these algorithms is in e.g. [20] and they are also recalled in Chapter 2 of this thesis.

The value iteration algorithm is always convergent to any initial guess, but this convergence can be very slow depending on the dimension of the state space and refinements in the discretization (e.g. a reduction in Δt). The policy iteration algorithm has a faster convergence but depends on the choice of initial guesses. In [21] a coupling between value and policy iteration has been proposed in order to accurately approximate the value function in a reduced amount of time if compared to traditional methods. The value iteration is solved in a coarse grid and its solution is used as a smart initial guess of policy iteration algorithm in a finer mesh. This approach considerably improved performance and showed effectiveness in tests with dimension up to 5.

Most of the traditional methods used to approximate the value function are designed to work on structured grids and, due to the curse of dimensionality, are limited to low dimensional problems. Semi-Lagrangian schemes permit to deal with unstructured grids since it exists an approximation method able to perform the spatial reconstruction in this context. It is possible to use different techniques such as interpolation and approximation based on radial basis functions (RBF), for example. An RBF approximation approach based on least squares is Shepard's method. The Shepard approximation of a function for a given point is a convex combination of function values from points in nearby regions.

In [22] Shepard's method is exemplified in low dimensional examples (up to dimension 3) in structured grids. The RBFs are compactly supported Wendland's functions and are tuned under an empirically selected shape parameter. This quantity is used to modify the width of RBFs and has a direct influence on the quality of the approximate value function, as will be discussed in this thesis.

RBF interpolation or approximation can be coupled to semi-Lagrangian schemes to approximate the value function in unstructured grids. In [23], the authors suggest a method to realize surface (or curve) reconstruction using level set methods. The semi-Lagrangian scheme uses RBF interpolation and the shape parameter is selected following a predefined rule according to each example.

In [24] the author introduces a quantity reflecting the quality of the approximation of the HJB equation called the residual. It is defined as the absolute difference between the numerical approximation of the value function and the operator of the fixed point iteration. The residual is calculated at each simplex of the discretization and used as a proxy for which regions it should be improved.

In [25], the authors have worked using a semi-Lagrangian scheme with Shepard approximation as reconstruction tool. They propose an algorithm where the solution of the discrete HJB is done constructing an iteractively sequence of approximations. At each iteration of the algorithm and for each node of the discrete state space, the respective residual is computed and compared to a threshold. If the residual is above the threshold, then more basis are needed in the neighborhood of the respective node. The important point is that the residual is used as a refinement indicator and a clue about the quality of the approximation.

When the optimal control problem is related to a high dimensional dynamical system, traditional methods are no longer feasible to solve the HJB equation. In order to mitigate the curse of dimensionality, different strategies were proposed. One strategy would be to reduce the dimension of the dynamical system, aiming to use traditional PDE schemes to solve the low-dimensional HJB equation. A different strategy would be to use methods to directly solve the HJB equation in the high dimensional scenario.

One of such methods is a coupling between DP and model order reduction techniques (e.g. Proper Orthogonal Decomposition (POD), [26, 27]). In POD, the general idea is to reduce the dimension d of the dynamical system using a ℓ basis representation in space with $\ell \ll d$. Thus, a reference trajectory of the dynamical system is calculated and, at certain times, snapshots are selected and used to obtain the basis functions. Then, an ideal number ℓ of basis elements are selected in order to well represent the dynamical system according to its complexity. After the dimensional reduction, the HJB associated to the reduced dynamical system can be approximated using traditional low dimensional methods (e.g. semi-Lagrangian schemes) which allow to obtain the optimal control in feedback form. This turns up to be efficient if the reduction is up to dimension $\ell \leqslant 5$. However, the low dimensional representation may not be accurate when nonlinear or advection effects of the dynamical system are relevant. More information about this method and the coupling with HJB equations can be found in [28] and [29] with an analysis of error estimates.

A major disadvantage of many traditional approaches used in DP is the need to determine a numerical domain large enough to include different trajectories of the dynamical system. In general, the dimension of such domains can be large and refinements in the discretization can impose constraints to simulations due to memory allocations. In order to overcome this problem, the authors in [30] propose the solution of the time dependent HJB equation in a tree structure. Using the discrete version of the dynamics, a suitable initial condition and the discrete control set, a tree is formed collecting nodes obtained in the evolution of dynamics. After an adequate pruning of the tree (from the Lipschitz continuity of the value function) the value function can be approximated using a semi-Lagrangian scheme since the tree covers only regions of interest in the model. As all nodes are obtained by the system evolution, the forward step in the dynamics is always part of the tree and there is no necessity of interpolation in the scheme. This approach has shown effectivenes in the direct solution of problems with dimensions of order $\mathcal{O}(10^3)$. The coupling between the tree structure algorithm (TSA) and POD was presented in [31]. It allows the reduction of the dimension of the problem with the advantage that the calculation of the value function is more efficient.

In [32] the authors present a different strategy to reduce restrictions imposed by the curse of dimensionality. In order to address the space complexity, the use of Quasi-Monte Carlo grids is proposed in place of regular grids. Quasi-Monte Carlo points are formed by sequences where the elements present some correlation with each other. It generates a grid that is more uniformly distributed than a grid originated by traditional Monte Carlo, resulting in a domain that is densely populated. This strategy reduces the complexity of the problem, but introduces another difficulty: the necessity to deal with scattered data. To overcome this barrier and approximate the value function using a semi-Lagrangian scheme, the authors propose the use of Kriging regression as interpolation method to perform the spatial reconstruction. The Kriging regression is formed by a polynomial term and a white noise process (in the cited work the authors used a second order regression polynomial and a generalized exponential correlation function in the white noise), both selected accordingly to the problem at hand. The work deals with examples in dimension up to 6 and served as inspiration in the search for a computationally efficient way to attack problems in higher dimensions.

So far, the discussed strategies to mitigate the curse of dimensionality involve grids or scattered data meshes. Recently, new methods that do not depend on these structures have been shown to be a computationally viable alternative.

In e.g. [33] it has been proposed a method that relies on a pseudospectral collocation method to discretize semi-linear parabolic PDEs. This discretization provides a meaningful representation of the dynamics using a low number of degrees of freedom. The HJB equation associated with the control problem can be numerically solved provided that each coordinate in the problem dynamics is a sum of separable functions. The feasibility of the method is guaranteed and allows to control systems with dimension up to 14 using parallelization tools.

Another possible way to overcome the curse of dimensionality is given by the max-plus algebra theory (see e.g. [35, 36]). Here, the time dependent value function is described as a combination of coefficients and the max-plus basis. The number of coefficients increases exponentially with the number of time steps.

In [37] the use of adaptative sparse grids was proposed to approximate the HJB equation related to specific front propagation models. The number of grid nodes used to represent the functions depends on a threshold error and can be small if compared to a full grid. The semi-Lagrangian scheme is defined using sparse multilevel basis and a sparse tensor product construction. Here, the feasibility of the method is shown in dimension up to 8. The scheme is not monotone and convergence to the viscosity solution is not guaranteed.

Recently, the use of neural networks (NN) in the estimation of the value function is becoming more relevant. The works [38] and [39] are examples of recent advances in this area, which have the benefit of being completely mesh free and not relying on discretization. In the former work, the authors claim to be the first known result that presents the formal mathematical connection between an NN architeture with a specific evolutive first order Hamilton-Jacobi (HJ) equation. Such architeture exactly represents the viscosity solution of the HJ PDE if some conditions on the parameters are set. The work provides examples of HJ equations and their NN representation from given initial conditions and hamiltonians relating the number of neurons to the dimension of each problem, with examples of dimension up to 16 in the HJ equation and higher dimensions in the solution of inverse problems related to these equations. In [39] the authors present two architetures to solve an evolutive HJ PDE with viscosity solution given by the Lax-Oleinik formula. These structures are shown to exactly represent the viscosity solutions without errors and the effectivess is showed by examples in dimension up to 10 where the NN solution perfectly represent the solutions with zero smoothing effects when these are non differentiable functions.

1.1 Contributions

This thesis introduces an approach to mitigate the curse of dimensionality and, thus, enables the application of the dynamic programming equations to the control of high dimensional dinamical systems generated by PDEs. Our method [40] does not rely on a high dimensional regularly discretized state space. Instead, we take advantage of the problem dynamics and only populate certain regions using points from the discretized dynamics itself. The resulting unstructured grid is formed by high dimensional nodes. In order to directly apply a semi-Lagrangian scheme to aproximate the value function, the Shepard method is used as the reconstruction tool. We also present a way to automatize the selection of the shape parameter used to tune the RBFs. The selection is not realized based in ad hoc methods, but by comparisons of the residual quantity.

The method uses a discrete version of the dynamical system that can be obtained by an Euler or higher order schemes. After a suitable choice of parameters (a fixed time step, initial conditions in state space, discretized control space and a sufficient large final time) the discrete trajectories are stored until the final time. These trajectories are taylored to populate specific regions of the domain and will be the unstructured grid used to apply the dynamic programming framework and locally approximate the value function. This thesis uses the coupling between semi-Lagrangian scheme and the Shepard's method to obtain the approximation of the value function in a high dimensional scattered data domain.

The adequate choice of RBFs is crucial in the construction of the Shepard's method and in other least squares or interpolation methods. This choice has a direct effect in the efficiency and in the quality of the results. The radial nature of these functions demands the use of distances between data points. These distances are stored in possibly large matrices depending on the size of the scattered data set. The choice of compactly supported RBFs is important to obtain efficiency because this radial function maps distance matrices in sparse matrices (Chapter 3).

The choice of the RBF is usually followed by the selection of a parameter that multiplies the distance variable r. It is called the shape parameter $\sigma > 0$ and its value crucially affects the quality of the approximations since scales the support of compactly supported RBFs. In general, it is selected by trial and error or by ad-hoc methods (cross-validation or maximum likelihood estimation [42]).

Our work uses the residual of the HJB equation as an indicator of the quality of the approximation. From a set of different parameter values (a discrete bounded interval) we approximate the value function and calculate the residual when using each parameter. The chosen parameter is the one that minimizes the residual.

We also provide error estimates to the approximation of the value function. Our error estimates are based on well-known results from semi-Lagrangian schemes adapted to the scattered data context, the construction of our localized grid and its parameters, together with the structure of the shape parameter, as explained in Chapter 4.

We present results showing the suitability of the method for the control of a different type of PDEs: nonlocal equations. This type of equation differs from classical PDEs since points in the domain can interact with any other point in the space. Due to this property, such models are getting a lot of attention in applied sciences.

The control of nonlocal and fractional operator equations is usually performed with the open-loop approach. Basic models were recently object of study, such as the nonlocal Poisson equation [57, 67, 68] and the fractional heat equation [71]. The control of nonlocal models with nonlocality different from the fractional case was studied in [69, 70]. In this thesis we present a novelty that is the use of the DP approach to control nonlinear parabolic PDEs with fractional operator. We also analyse the convergence of the controlled solution towards the continuous model.

Summarizing, the contributions are:

– A versatile way to obtain a grid based on the dynamics of the problem populating only specific parts of the domain. This helps to mitigate the curse of dimensionality since there is no need to fix a high dimensional structured grid;

– The approximation of the value function in a high dimensional scenario obtained by a semi-Lagrangian scheme using the Shepard's method;

– The automatic selection of the shape parameter used in RBFs;

– Numerical error estimates for the proposed method;

– Control of PDEs using our algorithm and considering different types of initial conditions;

– Applications to the control of nonlocal PDEs.

1.2 Organization of the Thesis

This thesis is divided into 5 Chapters

– **Chapter 2** reviews some aspects of semi-Lagrangian schemes related to the infinite horizon optimal control problem and the minimum time problem. The value iteration and the policy iteration methods are discussed together with a finite difference implementation to solve a 2D minimum time problem. We conclude this chapter with numerical experiments.

– **Chapter 3** recalls the main properties of radial functions and some points about the construction of Wendland's compactly supported RBFs [43]. This presentation is followed by a discussion about scattered data interpolation and approximation methods, emphasizing the Shepard's method. Here, we present the concepts of fill distance and separation distance, relevant in considerations about the notion of good spreading of points on a scattered data framework. We conclude with numerical tests about scattered data RBF interpolation and Shepard approximation.

– **Chapter 4** is the core of this thesis, since we introduce our algorithm. Here we develop and analyse the semi-Lagrangian scheme using the Shepard's method. Using the discretized dynamics, a choice of discrete control space, time steps, final time and initial conditions we present how to construct the dynamics-driven grid. The choice of parameters to generate the grid not only affects the grid but also the error estimates.

  The selection of the shape parameter is given together with the algorithm that summarizes the whole process (create the mesh, select the parameter, obtain the value function). Then, we demonstrate the method using low and high dimensional examples arising from PDEs.

– **Chapter 5** introduces the notion of nonlocal PDEs and the model used to test our approach: the fractional heat equation. Then, we present the finite element discretization of the model and analyse three test cases. The first test compares the solutions obtained by open-loop approach and the solution obtained using our RBF dynamic programming method. In this case we also study the convergence towards the continuous problem. The second test is devoted to the control of a linear equation in a subset of the problem domain. The third test consists in the control of a equation with a nonlinear term. In the first and third tests we show the robustness of the method in controlling the models with the presence of a disturbance term.

– **Chapter 6** provides a summary with conclusions and future directions.

1.3 Original Material

Chapter 4 is based on the paper "HJB-RBF based approach for the control of PDEs" [40] , submitted to the Journal of Scientific Computing.

Chapter 5 is based on the paper "Control of fractional diffusion problems via dynamic programming equations" [41], submitted to the Journal of Peridynamics and Nonlocal Modeling.

2
Infinite horizon control problem

In this chapter we present optimal control problems that will be studied throughout this thesis. We first discuss the infinite horizon control problem, the related Dynamic Programming Principle (DPP) and the associated first order PDE: the Hamilton-Jacobi-Bellman (HJB) equation. We present the minimum time problem and its rescaling by means of Kruzkhov transformation in order to write it as an infinite horizon problem. Then, we discuss semi-Lagrangian schemes (the policy and value iteration algorithms) and the finite differences method. We conclude with numerical experiments.

2.1
Dynamic Programming Principle and the Hamilton-Jacobi-Bellman equation

In this section we provide the main results for the infinite horizon optimal control problem. A complete treatment can be found in e.g. [12].

Let us consider an autonomous dynamical system described by

$$\begin{cases} \dot{y}(t) = f(y(t), u(t)), & t > 0, \\ y(0) = x \in \mathbb{R}^d. \end{cases} \tag{2-1}$$

We denote by $y : [0, \infty) \to \mathbb{R}^d$ the state variable, $u : [0, \infty) \to \mathbb{R}^m$ the control variable and by $f : \mathbb{R}^d \times \mathbb{R}^m \to \mathbb{R}^d$ the dynamics that drives the system. Let $u \in \mathcal{U} := \{u : [0, \infty) \to U, \text{ measurable}\}$ with $\mathcal{U}$ the set of admissible controls and $U \subset \mathbb{R}^m$ a compact set. In order to ensure the existence and uniqueness of a solution to the Cauchy problem (2-1), the function f needs to fullfill some assumptions: f be a continuous function in both variables, Lipschitz continuous in the first variable with constant $L_f > 0$ and measurable. Under those assumptions, the Carathéodory Theorem (see e.g. [12]) guarantee existence and uniqueness of solution to the initial value problem. Specifically, for any choice of $u \in \mathcal{U}$ there is a unique trajectory $y(\cdot)$ satisfying (2-1).

In order to select the optimal trajectory, let us introduce the cost functional $\mathcal{J} : \mathcal{U} \to \mathbb{R}$ in infinite horizon context

$$\mathcal{J}_x(u) := \int_0^\infty g(y(s), u(s)) e^{-\lambda s} ds \tag{2-2}$$

where the running cost $g : \mathbb{R}^d \times \mathbb{R}^m \to \mathbb{R}$ is assumed to be bounded and Lipschitz continuous with respect to the first variable. The constant $\lambda > 0$ is the discount factor and the term $e^{-\lambda s}$ guarantees the convergence of the integral whenever g is bounded.

Let us introduce the value function. This is defined for any initial condition x as

$$v(x) := \inf_{u\in\mathcal{U}} \mathcal{J}_x(u). \tag{2-3}$$

The value function has an important characterizaton given by the Dynamic Programming Principle (DPP)

$$v(x) = \inf_{u\in\mathcal{U}} \left\{ \int_0^\tau g(y_x(s), u(s))e^{-\lambda s} ds + e^{-\lambda\tau}\, v(y_x(\tau))\right\} \quad \forall x \in \mathbb{R}^d, \tau > 0, \tag{2-4}$$

where $y_x(\tau)$ is the trajectory starting at point x evaluated at time τ. Having introduced the DPP, one can also characterize the value function by a nonlinear first order partial differential equation, the HJB equation (see e.g. [12]):

$$\lambda v(x) + \max_{u\in U}\{-f(x,u)\cdot\nabla v(x) - g(x,u)\} = 0, \qquad x \in \mathbb{R}^d, \tag{2-5}$$

with $\nabla v(x)$ being the gradient of v. Equation (2-5) is known to possess nonsmooth solutions in many cases. Thus, we consider the solution of this equation in the viscosity sense (see e.g. [12, 20]). The value function is known to be the unique viscosity solution of HJB equation (2-5). In general, only Lipschitz continuity is guaranteed if f and g are Lipschitz.

The solution of HJB equation gives rise to the optimal feedback control $u^*(x)$:

$$u^*(x) = \arg\max_{u\in U}\{-f(x,u)\cdot\nabla v(x) - g(x,u)\}, \qquad x \in \mathbb{R}^d. \tag{2-6}$$

On the next section, we focus on minimum time problem and its connection with infinite horizon optimal control problems.

2.2 Minimum time problem

The objective of a minimum time problem is to steer the dynamics given in (2-1) from its initial state $x \in \mathbb{R}^d$ to a target set $\mathcal{T} \subset \mathbb{R}^d$ in the shortest possible time. We assume $\mathcal{T}$ closed. The time of arrival is defined as

$$t(x,u) := \begin{cases} \inf\limits_s\{s \in \mathbb{R}_+ : y_x(s,u) \in \mathcal{T}\} & \text{if } y_x(s,u) \in \mathcal{T} \text{ for some } s, \\ +\infty & \text{otherwise,} \end{cases} \tag{2-7}$$

where we write $y_x(s,u)$ to emphasize the dependence of the trajectory of the initial condition x, time s and control u. The value function here is the minimum time to reach the target and it is called the minimum time function, defined as

$$T(x) := \inf_{u\in\mathcal{U}} t(x,u). \tag{2-8}$$

Since not all initial conditions can be steered to $\mathcal{T}$ in finite time, this function is not defined everywhere. Thus, in order to define the DPP we first need to define the domain of the problem.

The set of points that allows the dynamics to reach the target is called the reachable set $\mathcal{R}$, i.e.

$$\mathcal{R} := \Big\{x \in \mathbb{R}^d : T(x) < +\infty\Big\}. \tag{2-9}$$

From equation (2-8) it is possible to obtain the Dynamic Programming Principle. For all $x \in \mathcal{R}$ and $\tau \in (0, T(x))$:

$$T(x) = \inf_{u\in\mathcal{U}}\{\tau + T(y_x(\tau,u))\}. \tag{2-10}$$

Assuming regularity of function $T(x)$, from equation (2-10) we obtain the corresponding HJB equation (see [20]):

$$\max_{u\in U}\{-f(x,u)\cdot\nabla T(x)\} = 1. \tag{2-11}$$

The minimum time problem can also be written as an infinite horizon problem. In order to rewrite the problem, we first need to rescale $T(x)$ by the so called Kruzkhov transformation. For $\mu > 0$,

$$v(x) := \begin{cases} \frac{1}{\mu} & \text{if } T(x) = +\infty, \\ \frac{1}{\mu} - e^{-\mu T(x)} & \text{else.} \end{cases} \tag{2-12}$$

The value function $v(x)$ is associated with the cost functional $\mathcal{J}_x(u) = \int_0^{t(x,u)} e^{-\mu s} ds$. Thus, we are in the scenario of an infinite horizon control problem with constant running cost $g(x,u) = 1$. The function $v(x)$ is the unique viscosity solution of

$$\begin{cases} \mu v(x) + \max_{u\in U}\{-f(x,u)\cdot\nabla v(x)\} - 1 = 0, & x \in \mathbb{R}^d \setminus \mathcal{T}, \\ v(x) = 0 & x \in \mathcal{T}. \end{cases} \tag{2-13}$$

The Kruzkhov transform can be inverted and it is possible to recover $T(x)$ directly from $v(x)$, since

$$T(x) = -\frac{1}{\mu}\ln\left(\frac{1}{\mu} - v(x)\right).$$

2.3 Numerical methods for the HJB equation

The HJB equation can be difficult to solve using analytical tools, despite the existence of the theory of viscosity solutions. An alternative strategy to solve the HJB equation is to approximate these solutions using numerical schemes such as, e.g. semi-Lagrangian or finite differences. These schemes are effective in solving low-dimensional problems such as those considered in this chapter. In what follows we discuss some of these methods with particular focus on semi-Lagrangian schemes. We consider the problem domain Ω discretized in equally distributed points, forming the numerical grid G.

2.3.1 Semi-Lagrangian schemes: Value Iteration and Policy Iteration

We will present the discrete version of DPP. First, let us consider a temporal discretization with fixed size Δt such that $t_k = k\Delta t$ with $k \in \mathbb{N}$. Then, the dynamics (2-1) is discretized with an explicit Euler scheme

$$\begin{cases} y_k &= y_{k-1} + \Delta t f(y_k, u_k), \\ y_0 &= x, \end{cases} \tag{2-14}$$

where u_k is a fixed control value in $[t_k, t_{k+1})$ and $u^{\Delta t}(s) = u_k$ for $s \in [t_k, t_{k+1})$. The control space U is discretized in m regularly distributed points. Let $y^{\Delta t}$ represent the dynamics from equation (2-14). Considering these information, the discretization of the cost functional (2-2) can be obtained by the rectangular method as

$$\mathcal{J}_x^{\Delta t}(y^{\Delta t}, u^{\Delta t}) := \Delta t \sum_{k=0}^{\infty} g(y_k, u_k) e^{-\lambda t_k}, \tag{2-15}$$

here $y^{\Delta t}$ is used to emphasize the dependence on the discrete dynamics. The associated value function is defined as

$$v^{\Delta t}(x) := \inf_{u^{\Delta t}} \mathcal{J}_x^{\Delta t}(y^{\Delta t}, u^{\Delta t}). \tag{2-16}$$

Similarly to the continuous case, equation (2-16) can also be split in two terms, resulting in the Discrete Dynamical Programming Principle (DDPP)

$$v^{\Delta t}(x) = \inf_{u^{\Delta t}} \left\{ \Delta t \sum_{k=0}^{p-1} g(y_k, u_k) e^{-\lambda t_k} + e^{-\lambda t_p} v^{\Delta t}(y_p) \right\}, \tag{2-17}$$

for $p \in \mathbb{N}$. Equation (2-17) defines semi-Lagrangian schemes for all $p \in \mathbb{N}$. In this thesis we will consider $p = 1$, obtaining

$$v^{\Delta t}(x) = \min_{u \in U} \{ \Delta t g(x, u) + (1 - \lambda \Delta t) v^{\Delta t}(x + \Delta t f(x, u)) \}, \tag{2-18}$$

since $u^{\Delta t}(s) = u \in U$ for $s \in [t_0, t_1)$.

Value Iteration. To fully discretize equation (2-18) we project it in the regular spatial grid G using linear interpolation for spatial reconstruction. We denote by V the vector whose entries are $V_j = v(x_j)$, with $x_j \in G$. Thus, the fully discretized scheme reads as

$$V_j = \min_{u \in U} \{\Delta t g(x_j, u) + (1 - \lambda \Delta t) I[V](x_j + \Delta t f(x_j, u))\}, \tag{2-19}$$

with $I[\cdot]$ being the interpolation operator. The right-hand side of equation (2-19) is a contraction if the interpolation operator I is non-expansive and $\Delta t < 1/\lambda$. Let us denote by V^n the vector evaluated at iteration n . If the scheme (2-19) is applied in a iterative manner, it converges to a fixed point for any inital condition V^0. What was described is the *Value Iteration* algorithm which is summarized in Algorithm 1.

Algorithm 1: Value Iteration for the Infinite Horizon control problem

```
1: INPUT: initial guess $V^0$, grid $G$, $\Delta t$, tolerance $tol$
2: while $||V^{n+1} - V^n|| \geq tol$ do
3:     for all $x_i \in G$ do
4:         $V_i^{n+1} = \min_{u \in U} \{\Delta t g(x_i, u) + (1 - \lambda \Delta t) I[V^n](x_i + \Delta t f(x_i, u))\}$
5:     end for
6:     $n = n + 1$
7: end while
```

We remark that, in pratical terms, the minimization phase in Algorithm 1 and in future algorithms described in this thesis are realized by comparison. The control set U is discretized in equally distributed points and for each $x_i \in G$ and $u_j \in U$ the respective value at iteration n is calculated and stored in entry R_{ij} of a matrix R. The value V_i^{n+1} is selected as the minimum value in row i from matrix R.

As mentioned in Section 2.2, the minimum time problem can be rewritten as an infinite horizon control problem. In this way, one can use the value iteration algorithm to approximate the value function.

Let us consider a time discretization with fixed temporal step size Δt, and let the number of steps to reach the target set $\mathcal{T}$ be defined by $N(x, u^{\Delta t})$. The discrete version of the minimum time (2-8) is now the minimum number of steps until the trajectory starting at x reaches the target:

$$N(x) := \min_{u^{\Delta t}} N(x, u^{\Delta t}). \tag{2-20}$$

From equation (2-20) it is possible to obtain the Discrete Dynamic

Programming Principle used in the minimum time case:

$$N(x) = \inf_{u^{\Delta t}} \{p + N(y_p)\} \tag{2-21}$$

at time p such that $0 \le p \le N(x)$. Considering $p = 1$ in (2-21) we obtain the equation

$$N(x) = \min_{u \in U} \{1 + N(x + \Delta t f(x,u))\}. \tag{2-22}$$

As $T(x) = \Delta t N(x)$, the Kruzkhov transform (2-12) can be applied to $T(x)$ with $\mu = 1$ obtaining $v^{\Delta t}(x) = 1 - e^{-\Delta t N(x)}$. Multiplying both sides of equation (2-22) by $-\Delta t$ and exponentiating, we obtain

$$e^{-\Delta t N(x)} = e^{-\Delta t} e^{-\Delta t \min\limits_{u \in U}\{N(x+\Delta t f(x,u))\}}.$$

Using the fact that $-e^{-\Delta t N(x)} = v^{\Delta t}(x) - 1$ and organizing the terms results in

$$v^{\Delta t}(x) = e^{-\Delta t} \min_{u \in U} \{v^{\Delta t}(x + \Delta t f(x,u))\} + 1 - e^{-\Delta t}, \tag{2-23}$$

if $x \in \mathbb{R}^d \setminus \mathcal{T}$ and $v^{\Delta t}(x) = 0$ if $x \in \mathcal{T}$. The spatial discretization of equation (2-23) gives us

$$\begin{cases} V_i^{n+1} = e^{-\Delta t} \min\limits_{u \in U} \{I[V^n](x_i + \Delta t f(x_i,u))\} + 1 - e^{-\Delta t}, & x_i \in \mathbb{R}^d \setminus \mathcal{T}, \\ V_i^{n+1} = 0 & x_i \in \mathcal{T}. \end{cases} \tag{2-24}$$

We refer to [20] for more details on these topics.

The discretization (2-24) leads to an approximation scheme similar to Algorithm 1 as shown in Algorithm 2. The differences are that in minimum time case the boundary condition needs to be forced to 0 at each point $x_i \in \mathcal{T}$ at each iteration.

Algorithm 2: Value Iteration for the Minimum Time control problem

```
1: INPUT: initial guess $V^0$, grid G, $\Delta t$, tolerance $tol$
2: while $||V^{n+1} - V^n|| \ge tol$ do
3:   for all $x_i \in G$ do
4:     $V_i^{n+1} = \min_{u \in U}\{1 - e^{-\Delta t} + e^{-\Delta t} I[V^n](x_i + \Delta t f(x_i,u))\}$
5:   end for
6:   $n = n + 1$
7: end while
```

Policy Iteration. Another method in the context of semi-Lagrangian schemes discussed here is the approximation in policy space, or Policy Iteration algorithm (see e.g. [19, 44]). Instead of computing a minimum value of V_i^n at each iteration n, this method works as follows: First, an initial guess of the control policy u_i^0 is given for equation (2-19) together with an initial guess V_i^0. Since the vector V^1 is evaluated substituting these guesses in the right hand side of (2-19), there is no search for a minimum and the equation becomes a linear system. This step is called the *policy evaluation step*. In order to find the policy to feed the next iteration, a search for the optimal value u_i^1 is done in each point of the grid G. This phase is called the *policy improvement step*. Now, the policy evaluation step restarts using V^1 and u^1 in place of V^0 and u^0. This process continues until a desired tolerance is achieved.

We point here that the acceleration of this method can be improved by the ideal choice of initial guesses (see [21]) together with the proposal of a suitable coupling between value iteration and policy iteration methods (the Accelerated Policy Iteration algorithm), as pointed out in the introduction.

Algorithm 3: Policy Iteration for the Infinite Horizon control problem

```
1: INPUT: initial guess $V^0$ and $u^0$, grid G, $\Delta t$, tolerance $tol$
2: while $||V^{n+1} - V^n|| \geq tol$ do
3:     Policy evaluation step:
4:     for all $x_i \in G$ do
5:         $V_i^{n+1} = \min_{u \in U}\{\Delta t g(x_i, u_i^n) + (1 - \Delta t\lambda)I[V^n](x_i + \Delta t f(x_i, u_i^n))\}$
6:     end for
7:     Policy improvement step:
8:     for all $x_i \in G$ do
9:         $u_i^{n+1} = \operatorname{argmin}_u\{\Delta t g(x_i, u) + (1 - \lambda\Delta t)I[V_i^n](x_i + \Delta t f(x_i, u))\}$
10:    end for
11:    $n = n + 1$
12: end while
```

2.3.2 Finite Differences

Finite differences schemes are traditionally used to approximate differential equations. In the field of optimal control problems and HJB equations, those schemes are commonly used in the numerical approximation of stochastic optimal control problems and mean field games (see e.g. [45]).

The theoretical guarantee of convergence of the finite differences approximation to the viscosity solution is presented in [16, 17]. Although finite differences approximation of viscosity solutions is well known, a restriction on the use of these schemes is due to the computational infeasibility of dealing with high-dimensional problems, a restriction that other schemes (e.g. semi-

Lagrangian, finite volumes, finite differences and others) also suffer. To ease the notations, the discussion here is restricted to a two-dimensional framework along the lines of [16].

The type of HJ equations considered in [16] is the evolutive first order like (2-25) where $H \in C(\mathbb{R}^2)$ is the Hamiltonian function. In [17] it is presented a more general problem, with the possibility of working with second order equations. We focus in the method proposed in [16] due to its similarity with conservative schemes and the straighforward implementation

$$\begin{cases} v_t + H(\nabla v) = 0, & \text{in } \mathbb{R}^2 \times (0, \infty), \\ v(x,0) = v_0(x) & \text{in } \mathbb{R}^2 \times \{t = 0\}. \end{cases} \tag{2-25}$$

We consider a regular grid discretized with space step Δx_i in dimension $i \in \{1,2\}$ and a time horizon discretized with time step Δt. Also consider a Lipschitz continuous function $\mathcal{H}$ called the *numerical Hamiltonian* of the scheme. This function is related to the numerical flux in conservation laws (a discussion can be found in e.g. [46]). Thus, equation (2-25) can be approximated by a scheme in differenced form (2-26).

Definition 2.1 *The scheme is said to be in differenced form if it has the form*

$$v_j^{n+1} = v_j^n - \Delta t \mathcal{H}\Big(D_{1,j-p}[V^n], \ldots, D_{1,j+q}[V^n], D_{2,j-p}[V^n], \ldots, D_{2,j+q}[V^n]\Big) \tag{2-26}$$

and here we denote by $D_{i,j}[V] = \frac{v_{j+e_i} - v_j}{\Delta x_i}$ with $i \in \{1,2\}$, that is, the forward approximation of spatial derivative v_{x_i} at the point x_j. The subscript j is a multi-index representing the coordinates of the point in the two-dimensional grid, i.e. $j \in \mathbb{Z}^2$ where $j = \{j_1, j_2\}$ and $x_j = \{j_1 \Delta x_1, j_2 \Delta x_2\}$. The multi-indices e_i with $i \in \{1,2\}$ represent the canonical vectors $e_1 = (1,0)$ and $e_2 = (0,1)$. Here, p and q represent general multi-indices with positive components.

A scheme in differenced form is said consistent if

$$\mathcal{H}(a, \ldots, a, b, \ldots, b) = H(a,b) \quad \text{for } a, b \in \mathbb{R},$$

and it is said to be monotone on $[-R, R]$ if $\mathcal{H}$ is a nondecreasing function in each of its arguments and $|D_{i,j}[V]| \leq R$.

A result from Crandall-Lions [16] asserts that if $H : \mathbb{R}^2 \to \mathbb{R}$ is continuous, the initial condition v_0 is Lipschitz continuous with constant L, the scheme (2-26) is monotone in $[-(L+1), L+1]$ and consistent for a locally Lipschitz continuous numerical Hamiltonian $\mathcal{H}$, then the approximation converges to the viscosity solution for $\Delta t \to 0$.

The previous result provides conditions under which the convergence of the numerical solution to the viscosity solution happens in an evolutive problem like (2-25). The Barles-Sougadinis theorem [17] states that any monotone, stable and consistent scheme converges to the viscosity solution provided that exists a comparison principle for the limiting equation, but it does not give any estimate.

As discussed, in the infinite horizon optimal control problem the objective is to find a solution to an equation of the following type

$$\lambda v(x) + \max_{u \in U}\{-f(x,u)\cdot \nabla v(x) - g(x,u)\} = 0, \qquad x \in \mathbb{R}^d.$$

We consider that such equation can be rewritten as

$$\lambda v(x) + H(\nabla v(x)) = 0. \tag{2-27}$$

Now, we follow [20] and consider a evolutive equation that converges to the regime (2-27)

$$v_t(x) + \lambda v(x) + H(\nabla v(x)) = 0. \tag{2-28}$$

The choice of a suitable discretization of equation (2-28) leads, for example, to the differenced form

$$\begin{aligned}\frac{v_j^{n+1} - v_j^n}{\Delta t} + \lambda v_j^n + \mathcal{H}\Big(D_{1,j-p}[V^n], \dots, D_{1,j+q}[V^n], D_{2,j-p}[V^n], \dots \\ \dots D_{2,j+q}[V^n]\Big) = 0\end{aligned} \tag{2-29}$$

The spatial derivatives can be built in a forward or backward manner like in upwind schemes (2-30) or in a centered manner (2-31).

$$D_{i,j}[V] = \frac{v_{j+e_i} - v_j}{\Delta x_i}. \tag{2-30}$$

$$\frac{D_{i,j}[V] + D_{i,j-e_1}[V]}{2} = \frac{v_{j+e_i} - v_{j-e_i}}{2\Delta x_i}. \tag{2-31}$$

If the scheme is built in an upwind way a special care should be considered due to changes in the signal of the velocity coefficient. The scheme (2-29) can be verified to be consistent and monotone under a suitable choice of the numerical hamiltonian and parameters. Using the time index as an iteration index, the scheme becomes

$$\begin{aligned}v_j^{n+1} = (1-\Delta t\lambda)v_j^n - \Delta t\mathcal{H}\Big(D_{1,j-p}[V^n], \dots, D_{1,j+q}[V^n], D_{2,j-p}[V^n], \dots \\ \dots, D_{2,j+q}[V^n]\Big),\end{aligned} \tag{2-32}$$

and the iteration converges to the viscosity solution. The implementation can be summarized as

Algorithm 4: Finite Differences for the Infinite Horizon control problem

```
1: INPUT: initial guesses $V^0$, grid G regularly discretized, $\Delta t$, tolerance $tol$
2: while $||V^{n+1} - V^n|| \geq tol$ do
3:     for all $x_j \in G$ do
4:         $V_j^{n+1} = (1 - \Delta t\lambda)V_j^n - \Delta t\mathcal{H}\Big(D_{1,j-p}[V^n], \dots, D_{1,j+q}[V^n], D_{2,j-p}[V^n], \dots, D_{2,j+q}[V^n]\Big)$
5:     end for
6:     $n = n + 1$
7: end while
```

2.3.3 Numerical Examples

In order to exemplify the methods discussed in this section we present tests performed on a two dimensional problem. Let us consider a minimum time problem in $\Omega = [-1, 1]^2$ with the following dynamics and control space

$$f(x, u) = \begin{pmatrix} \cos(u) \\ \sin(u) \end{pmatrix}, \quad U = [0, 2\pi]. \tag{2-33}$$

The cost functional to be minimized is $\mathcal{J}_x(u) = \int_0^{t(x,u)} e^{-\lambda s} ds$ with $t(x, u)$ being the time of arrival to the target $\mathcal{T} = (0, 0)$ for each $x \in [-1, 1]^2$ (see equation (2-7)).

The analytical solution $V^*(x)$ for this problem is the Kruzkhov transformation of the distance to the target $\mathcal{T}$ (see equation (2-12)):

$$V^*(x) := \begin{cases} 1, & \text{if } v^*(x) = \infty, \\ 1 - \exp(-v^*(x)), & \text{else} \end{cases}$$

with $v^*(x) = ||x||_2$ for each $x \in [-1, 1]^2$, see Figure 2.1.

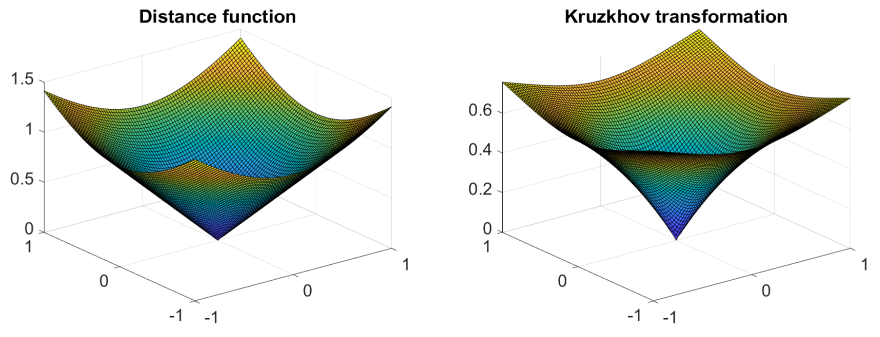

Figure 2.1: Distance function and its Kruzkhov transformation.

The relative errors between the analytical solution and the numerical approximations V are obtained by

$$\mathcal{E}(V) = \frac{||V - V^*||_\infty}{||V^*||_\infty} \tag{2-34}$$

Example 1: Value Iteration and Policy Iteration

We present numerical results to compare the use of Value Iteration and Policy Iteration in solving the two dimensional minimum time problem. The control space is discretized in 64 points, $\lambda = 1$ and $\Delta t = 0.8\Delta x$. The tolerance is 10^{-6}.

Numerical results of simulations are presented in Table 2.1 and Table 2.2. We note that the resulting relative errors and, as consequence, the rate of convergence obtained by both methods are the same. The difference between both methods lies in computational time with Policy Iteration having a faster convergence than Value Iteration. Plots of those solutions are in Figure 2.2.

Δx	Points	Error	CPU time (s)	Rate
0.1	21^2	0.0232	0.33	
0.05	41^2	0.0165	0.70	0.49
0.025	81^2	0.0108	2.05	0.6
0.0125	161^2	0.0068	23.90	0.67
0.00625	321^2	0.0041	291.90	0.7

Table 2.1: Results of Value Iteration.

Δx	Points	Error	CPU time (s)	Rate
0.1	21^2	0.0232	0.20	
0.05	41^2	0.0165	0.37	0.49
0.025	81^2	0.0108	1.13	0.6
0.0125	161^2	0.0068	12.80	0.67
0.00625	321^2	0.0041	156.20	0.7

Table 2.2: Results of Policy Iteration.

Example 2: Finite Differences

Now, we approximate the solution of the minimal time problem using finite differences. The HJB equation related to the problem written following (2-13) is

$$\begin{cases} v(x) + \max\limits_{u \in U} \Big\{ -\cos(u)\frac{\partial v}{\partial x_1} - \sin(u)\frac{\partial v}{\partial x_2} \Big\} - 1 = 0, & x \in \Omega \setminus \mathcal{T}, \\ v(x) = 0 & x \in \mathcal{T}. \end{cases} \tag{2-35}$$

Here we can follow the 2D scheme presented in [16] analog to a Lax-Friedrichs scheme and adapt it to a time-marching version like (2-29). The scheme becomes

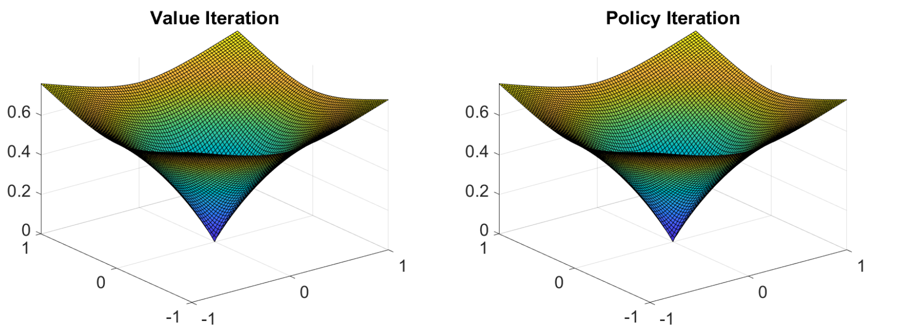

Figure 2.2: Value Iteration and Policy Iteration solutions to minimum time problem with grid formed by 81^2 points. $\Delta x = 0.025$ and $\Delta t = 0.02$.

$$\begin{cases} v_j^{n+1} &= (1-\Delta t)v_j^n - \Delta t \mathcal{H}\Big(D_{1,j}[V^n], D_{1,j-e_1}[V^n], D_{2,j}[V^n], D_{2,j-e_2}[V^n]\Big) - \Delta t, \\ v_j^{n+1} &= 0, \quad x \in \mathcal{T}, \end{cases} \tag{2-36}$$

where

$$\mathcal{H}\Big(D_{1,j}[V^n], D_{1,j-e_1}[V^n], D_{2,j}[V^n], D_{2,j-e_2}[V^n]\Big) = H\left(\frac{v_{j+e_1}-v_{j-e_1}}{2\Delta x_1}, \frac{v_{j+e_2}-v_{j-e_2}}{2\Delta x_2}\right) \\ - \theta\frac{\Delta x_1}{\Delta t}\left(\frac{v_{j+e_1}+v_{j-e_1}-2v_j}{\Delta x_1}\right) - \theta\frac{\Delta x_2}{\Delta t}\left(\frac{v_{j+e_2}+v_{j-e_2}-2v_j}{\Delta x_2}\right) \tag{2-37}$$

The parameter θ is chosen such that $0 < \theta < 1/4$ and then $\frac{\Delta t}{\Delta x_i}$ is selected in order to respect $\theta - \frac{\Delta t}{\Delta x_i}|H_i'(\alpha,\beta)|/2 \geq 0$ with H_i' being the derivative of the Hamiltonian with respect to the i-th argument.

Using Algorithm 4 with Ω discretized in regularly distributed points like in Example 1, U discretized with 32 points, a tolerance of 10^{-6} and a fixed relation $\Delta t = 10^{-3}\Delta x$ we obtain the Kruzkov transform of distance function as in Figure 2.3 and summarize the results in Table 2.3.

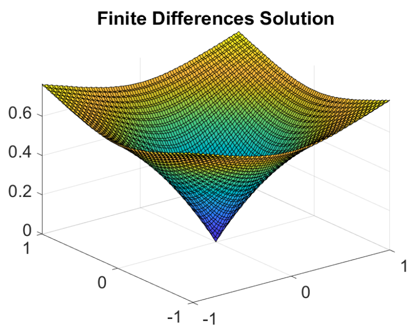

Figure 2.3: Finite Differences solution to minimum time problem with grid formed by 81^2 points. $\Delta x = 0.025$ and $\Delta t = 2.5 \times 10^{-5}$.

Δx	Points	Error	CPU time (s)	Rate
0.1	21^2	0.0363	50	1.3
0.05	41^2	0.0212	$2.7e+03$	0.77
0.025	81^2	0.0134	$6.6e+04$	0.66

Table 2.3: Data from Finite Differences tests.

From Table 2.3 we see the decay of the relative error (2-34) with the increase of the grid size, assuring that the method converges to the exact solution. It is clear that the computational time increases very fast, restricting our test to only 81^2 grid nodes.

3
Scattered Data Approximation using RBFs

This chapter recalls interpolation and approximation methods with scattered data using radial basis functions (RBFs). First we define what RBFs are and present basic properties of radial functions with compact support, specifically the Wendland's functions. Then, we provide examples of such functions in one and two dimensional scenarios. Finally we present a discussion of interpolation using RBFs and also approximations based in least squares methods. We give special focus to the Shepard approximation. In conclusion, we present numerical experiments of RBF interpolation and Shepard's method.

We refer to e.g. [49, 43, 22] and the references therein for a complete description of the topic.

3.1
Radial Basis Functions

The scattered data approximation problem consists in finding an approximant for a function $f : \Omega \subset \mathbb{R}^d \to \mathbb{R}$ at a point $x \in \Omega$ with unstructured grids. Thus, we consider a set of pairwise distinct data sites $X = \{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^d$ and a set of data values $Y = \{y_1, y_2, \ldots, y_n\} \subset \mathbb{R}$ with $y_i = f(x_i)$, $1 \leq i \leq n$. The objective is to find a function $\tilde{f}(x)$ that approximates the value $f(x)$ for $x \in \Omega$ and it is built as a linear combination of basis functions $\varphi_i : \Omega \subset \mathbb{R}^d \to \mathbb{R}$,

$$\tilde{f}(x) = \sum_{i=1}^{n} c_i \varphi_i(x). \tag{3-1}$$

Here, the basis functions φ_i are radial basis functions as described below.

Definition 3.1 (Positive definite function) *A continuous function $\phi : X \subset \mathbb{R}^d \to \mathbb{R}$ is positive definite if and only if it is even and*

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j \phi(x_i - x_j) \geq 0 \tag{3-2}$$

for any pairwise different points $\{x_1, x_2, \ldots, x_n\} \in \mathbb{R}^d$ and coefficients $c = [c_1, c_2, \ldots, c_n]^T \in \mathbb{R}^n$. The function ϕ is strictly positive definite on $\mathbb{R}^d$ if the quadractic form (3-2) *is zero only for $c = 0$.*

Definition 3.2 (Radial Basis Function - RBF) *A RBF $\varphi : \mathbb{R}^d \to \mathbb{R}$ is a radially invariant function $\varphi(x) = \varphi(\|x\|_2)$.*

Let $x_i, x_j \in X$ such that $\|x_i\| = \|x_j\|$. Then, by the definition above $\varphi(x_i) = \varphi(x_j)$. Also, note that if $\|x_i - x_j\| = r$ we have $\varphi(\|x_i - x_j\|) = \varphi(r)$. Let us fix x_i as the center of a RBF, we consider the distance from any point x_j to the center and denote as $\varphi_i(x_j) \coloneqq \varphi(\|x_j - x_i\|)$ the RBF centered on x_i.

Let us consider a set of pairwise distinct data sites $X \subset \Omega$ and $\varphi_i^\sigma(x)$ being a RBF centered at x_i with shape parameter σ evaluated at $x \in \Omega$. The radial nature of RBFs can be used to tune their spread by means of the parameter $\sigma > 0$. This parameter has the effect of rescaling $\|x\|_2$ (by the multiplication $\sigma\|x\|_2$) resulting in changes in the width and the shape of the function.

One example of RBF modified by the shape parameter is the gaussian $\varphi^\sigma(x) = e^{-\sigma\|x\|_2^2}$. This function is similar to the normal probability density function without the normalization factor. Here, the shape parameter σ works in similarly to the inverse of variance in probability theory, i.e, it is responsible for the dispersion of the probability density function in an inverse manner: the greater the term σ the more concentrated is the density function around the origin, as illustrated in Figure 3.1.

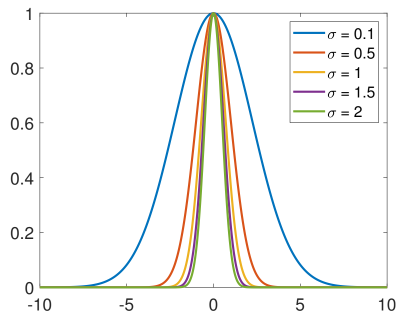

Figure 3.1: Example of Gaussian RBF $\varphi^\sigma(x) = e^{-\sigma\|x\|_2^2}$ with $\sigma = \{0.1, 0.5, 1, 1.5, 2\}$ and $x \in [-10, 10]$.

In this thesis we only work with compactly supported positive definite RBFs such as the Wendland's functions.

Wendland's functions are compactly supported RBFs constructed by an integral operator $\mathcal{I}$ applied to the truncated power function

$$\varphi_\ell(r) = (1 - r)_+^\ell, \quad r = \|x\|_2, \tag{3-3}$$

which is radial on $\mathbb{R}^d$ and strictly positive for $\ell > \lfloor \frac{d}{2} \rfloor + 1$. These functions are compactly supported in $[0, 1]$. Here, $(1 - r)_+^\ell \coloneqq \max\{0, (1 - r)^\ell\}$ and $\lfloor \cdot \rfloor$ the floor function.

Definition 3.3 *Let φ be such that $x \to x\varphi(x) \in L^1[0,\infty)$. The operator $\mathcal{I}$ is defined*

$$(\mathcal{I}\varphi)(r) := \int_r^\infty x\varphi(x)dx, \qquad r \geq 0.$$

If φ has compact support then $\mathcal{I}\varphi$ has compact support, as explained in e.g. [49]. Wendland's functions can be written as $\varphi_{d,k} = \mathcal{I}^k\varphi_{\lfloor \frac{d}{2} \rfloor+k+1}$, with $\varphi_{\lfloor \frac{d}{2} \rfloor+k+1}$ being the function (3-3). [49, Theorem 11.3] states that functions $\varphi_{d,k}$ are strictly positive definite and radial in $\mathbb{R}^d$ with an unique univariate polynomial representation having the minimal possible degree. Moreover, $\varphi_{d,k} \in C^{2k}(\mathbb{R})$.

In Figure 3.2 we see how one dimensional Wendland's functions behaves with different values of k for a fixed shape parameter $\sigma = 1$.

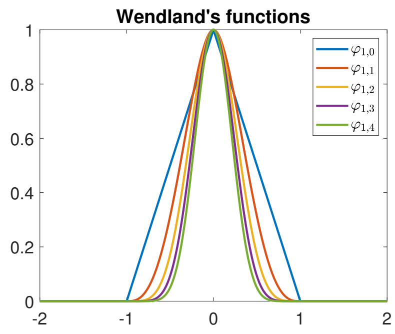

Figure 3.2: 1D Wendland's functions for $k \in \{0, 1, 2, 3, 4\}$ and $\sigma = 1$.

Let us show how the parameter σ affects a function φ^σ using a two dimensional example with $k = 2$ (i.e. $\varphi^\sigma \in C^4(\mathbb{R})$). Figure 3.3 shows the Wendland RBF

$$\varphi^\sigma(r) = \max\{0, (1 - \sigma r)^6(35\sigma^2r^2 + 18\sigma r + 3)\}, \;\; r := \|x\|_2, \tag{3-4}$$

with $\sigma = 0.8$ on the left panel and $\sigma = 2$ on the right panel. We can see how the parameter influences the *shape* of the basis functions and makes them flat (left) or spiky (right). The RBF is scaled so that $\varphi^\sigma(0) = 1$.

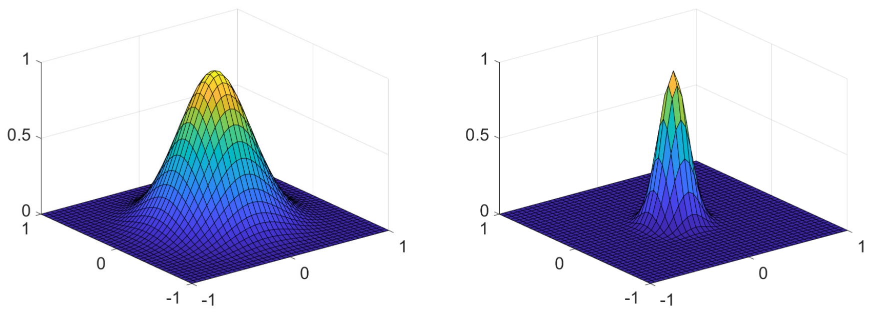

Figure 3.3: Wendland function (3-4) with $\sigma = 0.8$ (left) and $\sigma = 2$ (right).

RBFs are the basic tool used for scattered data approximation methods we will focus on. Next paragraphs explain some of these methods and how these functions are used in each context.

3.1.1
Interpolation using RBFs

We consider a continuous function $f : \Omega \subset \mathbb{R}^d \to \mathbb{R}$. The objective of the interpolation problem is to find a close fit to given data $X = \{x_1, x_2, \dots, x_n\}$ by a function $\tilde{f}$ such that $\tilde{f}(x_i) = f(x_i)$,$\forall x_i \in X$ and $\tilde{f}$ is written as a linear combination as in equation (3-1).

To perform the construction of $\tilde{f}$ we need to obtain the coefficient vector $c = [c_1, c_2, \dots, c_n]^T$ solving the system $Ac = f(x)$ where $f(x) = [f(x_1), \dots, f(x_n)]^T$ and

$$A = \begin{bmatrix} \varphi(\|x_1 - x_1\|) & \varphi(\|x_1 - x_2\|) & \dots & \varphi(\|x_1 - x_n\|) \\ \varphi(\|x_2 - x_1\|) & \varphi(\|x_2 - x_2\|) & \dots & \varphi(\|x_2 - x_n\|) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi(\|x_n - x_1\|) & \varphi(\|x_n - x_2\|) & \dots & \varphi(\|x_n - x_n\|) \end{bmatrix} \in \mathbb{R}^{n \times n}.$$

The choice of a strictly positive definite RBF guarantees the solvability of this system, since a positive definite RBF implies A to be a positive definite matrix and, by consequence, nonsingular.

This operation can be computationally expensive depending on the number of nodes in X and the number of times it needs to be performed (e.g. if it is performed at each iteration in an iterative scheme).

3.2
Weigthed least squares and Shepard approximation

Let us suppose that we want to approximate a function
$f : \Omega \to \mathbb{R}$ in the approximation space formed by M multivariate polynomials $\mathcal{P} = \text{span}\{p_1, p_2, \dots, p_M\}$ with $p_j \in \prod_s^d$, space of multivariate polynomials in d variables with total degree less or equal than s. Let us consider pairs (x_i, y_i), $i = 1, 2, \dots, n$ as for interpolation problem and $M < n$. The objective is to find the best possible approximation of f in $\mathcal{P}$. Differently from interpolation, we do not demand that $\tilde{f}(x_i) = f(x_i)$.

We want to find an approximant

$$\tilde{f}(x) = \sum_{j=1}^{M} c_j p_j(x), \qquad x \in \mathbb{R}^d,$$

where the coefficients $c = [c_1, c_2, \dots, c_M]^T$ are obtained solving the linear system $Gc = f_p$. Here, the matrix G is formed by $G_{i,j} = \langle p_i, p_j \rangle_w$ and vector $f_p = [\langle f, p_1\rangle_w, \langle f, p_2\rangle_w, \dots, \langle f, p_M\rangle_w]^T$. The system is originated by considering

$$\langle f - \tilde{f}, p_j \rangle_w = 0, \quad 1 \leq j \leq m \tag{3-5}$$

with the discrete inner product

$$\langle f, g \rangle_w := \sum_{i=1}^{n} f(x_i) g(x_i) w(x_i) \tag{3-6}$$

and given weights $w : \Omega \to [0, \infty)$. The approximation $\tilde{f}$ is the one that minimizes $\|\tilde{f} - f\|_{2,w}$ with $\|\cdot\|_{2,w}$ being the norm defined by the inner product (3-6). This formulation is called weighted least squares (WLS).

Based on the WLS, one can perform a similar construction considering only points x_j that are relatively close to x. In other words, the approximation problem can be localized. In order to fulfill this task, let us consider a different weight function to be used in the discrete weighted inner product, that decays increasing the distance between a specific point x and nodes x_j. This function is denoted by $w : \Omega \times \Omega \to \mathbb{R}^+$ and it is called moving weight function, where $w(x_j, x)$ decays when $\|x_j - x\|_2$ increases. An example of function that has the described properties and can be a moving weight function is a compactly supported RBF, since $w(x, x_i) = \varphi(\sigma \|x - x_i\|_2)$ (e.g. see Figure 3.3 and equation (3-4)).

Thus, the problem of approximate a function f at a point x is to find a function $\tilde{f}^x \in \operatorname{span}\{p_1, p_2, \dots, p_M\}$ such that $\|f - \tilde{f}^x\|_{2,w(x,\cdot)}$ is minimized. Here, the norm $\|\cdot\|_{2,w(x,\cdot)}$ comes from the discrete inner product

$$\langle f, g \rangle_{w(x,\cdot)} := \sum_{i=1}^{n} f(x_i) g(x_i) w(x, x_i), \tag{3-7}$$

with $w(x, x_i)$ a moving weight function. At any point $x \in \Omega$ we want to approximate the function f, we need to obtain coefficients c_j^x solving a system $G^x c^x = f^x$ with $G^x_{i,j} = \langle p_i, p_j \rangle_{w(x,\cdot)}$, $c^x = [c_1^x, \dots, c_M^x]^T$ and $f^x = [\langle f, p_1\rangle_{w(x,\cdot)}, \langle f, p_2\rangle_{w(x,\cdot)}, \dots, \langle f, p_M\rangle_{w(x,\cdot)}]^T$.

For each different point x we need to solve a distinct linear system to obtain specific coefficients c^x, that is the reason of the superscript in the notation. This method is known by Moving Least Squares (MLS) and from this formulation we can build the main tool used in this work: Shepard's method (see e.g. [49, Chapter 23] and the original paper [50]).

We start the presentation of Shepard's method considering $\mathcal{P} = \{p_1\}$ and $\tilde{f}^x(x) = c_1^x p_1(x)$. Solving the Moving Least Squares in this configuration and considering $p_1(x) = 1$ leads to

$$\langle p_1, p_1 \rangle_{w(x,\cdot)} \cdot c_1^x = \langle f, p_1 \rangle_{w(x,\cdot)}$$

that will result in

$$c_1^x = \sum_{i=1}^{n} f(x_i)\psi_i(x),$$

where $\psi_i(x) = \frac{w(x,x_i)}{\sum_{j=1}^{n} w(x,x_j)}$. The Shepard approximation of the function f in x is given by

$$Sf(x) = \sum_{i=1}^{n} f(x_i)\psi_i(x)$$

In this case, the RBF bases are used to form n weights

$$\psi_i^\sigma(x) := \frac{\varphi^\sigma(\|x - x_i\|_2)}{\sum_{j=1}^{n} \varphi^\sigma(\|x - x_j\|_2)}, \quad 1 \leq i \leq n, \tag{3-8}$$

and the Shepard approximant $S^\sigma[f](x)$ is formed as

$$S^\sigma[f](x) := \sum_{i=1}^{n} f(x_i)\psi_i^\sigma(x). \tag{3-9}$$

Observe that each $\psi_i(x)$ is compactly supported in $B(x_i, 1/\sigma) \subset \Omega$, non negative function, and the weights form a partition of unity, i.e., $\sum_{i=1}^{n} \psi_i^\sigma(x) = 1$ for all $x \in \Omega$ with

$$\Omega_{X,\sigma} := \bigcup_{x \in X} B(x, 1/\sigma) \subset \mathbb{R}^d. \tag{3-10}$$

This implies that $S^\sigma[f](x)$ is actually a convex combination of the function values $f(x_i)$. Moreover, the compact support of the weights leads to a computational advantage (due to sparsity of distance vectors) and localization. In particular, the Shepard weights are evaluated by constructing a distance vector $\Lambda \in \mathbb{R}^n$ with $\Lambda_i := \|x - x_i\|_2$ by computing only the entries Λ_i such that $\|x - x_i\| \leq 1/\sigma$. This operation can be implemented by a range search [74].

An additional advantage of the Shepard's method is that the construction of the approximant (3-9) can be directly obtained from the function values and the evaluation of the weights, without solving any linear system.

As RBF-based methods work with unstructured meshes, to obtain error estimates in this context it is common to consider the *fill distance* and the *separation distance*

$$h := h_{X,\Omega} := \sup_{x \in \Omega} \min_{x_i \in X} \|x - x_i\|, \qquad q = q_X := \min_{x_i \neq x_j \in X} \|x_i - x_j\|.$$

The fill distance h replaces the mesh size and it is the radius of the largest ball in Ω which does not contain any point from the set $X = \{x_1, x_2, \ldots, x_n\}$, and it gives a quantification of the well spread of the approximation nodes in the domain. On the other hand, the separation distance q quantifies the minimal separation between different approximation points. We remark that, for any sequence of points, $\frac{1}{2}q \leq h$, but the inverse inequality $h \leq Cq$, $C > 0$, does not hold unless the points are asymptotically uniformly distributed.

Two important concepts about convergence of interpolation or approximation methods are *non-stationarity* and *stationarity* of these methods. We say that a method is non-stationary if, with a denser set X (resulting in a possible decrease in h), we mantain the shape parameter σ fixed. A method is said to be stationary if, with a denser set X, the shape parameter is scaled as $\sigma = \theta/h$ for some $\theta > 0$, i.e. the denser the set, the smaller the support of the RBFs used on the method.

The point here is to note that interpolation methods using RBF are convergent in a non-stationary scenario (Theorem 15.3, [49]) while the Shepard approximation method is convergent in a stationary one, i.e. scaling the support of RBFs according to h. A more detailed discussion of stationarity and non-stationarity can be found in e.g. [49].

Example. Shepard approximation and RBF interpolation

We present an example of RBF interpolation and Shepard approximation using the squared distance function with respect to the origin in $\mathbb{R}^2$ i.e. $||x||_2$. We restrict our case to $\Omega = [-1, 1]^2$ and analyse the quality of Shepard approximation and RBF interpolation with different meshes. We first fix a set X of 200 points randomly selected in Ω according to a uniform probability distribution function, these are the evaluation points. Then we generate regular and random scattered meshes of size 21^2,41^2,81^2 and 161^2. The scattered meshes were obtained first by selecting a set of 40000 points according to a uniform distribution function and then selecting the desired number of points to each mesh size by the *kmeans* algorithm [73]. In every mesh we calculate the Shepard approximation and the RBF interpolation of distance function to evaluation points in X and compare the results with the exact solution computing the relative error

$$\mathcal{E}(\Lambda) = \frac{||\Lambda - \Lambda^*||_\infty}{||\Lambda^*||_\infty} \tag{3-11}$$

where Λ^* is a vector with all exact distances and Λ is a vector with Shepard approximations or RBF interpolation values. In the case of randomly scattered meshes we perform a sequence of 10 tests and give results in average (average relative errors and average fill distances). For Shepard approximation in regular and scattered meshes we fix $\sigma = 0.75/h$ and use an Wendland's RBF

$\varphi_{2,2}(r)$ (the exact formula is given in equation (3-4)). As RBF interpolation is convergent in a *non-stationary* scenario, we fix $\sigma = 0.75$ for all meshes.

In Figure 3.4 we can see the difference between the structured and unstructured grids, both with the same evaluation points.

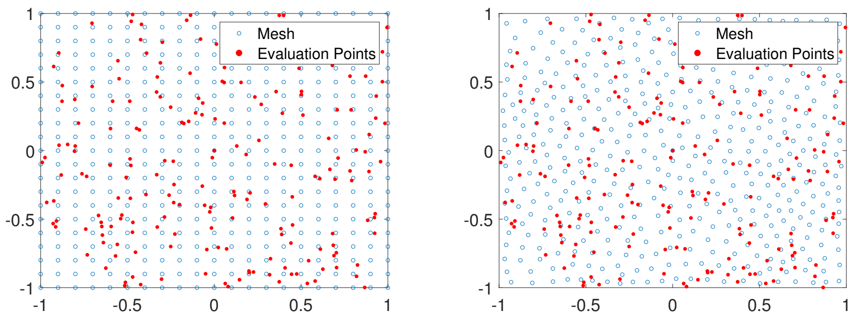

Figure 3.4: Regular mesh with evaluation points and scattered mesh with evaluation points. 41^2 points used to generate meshes and 200 evaluation points.

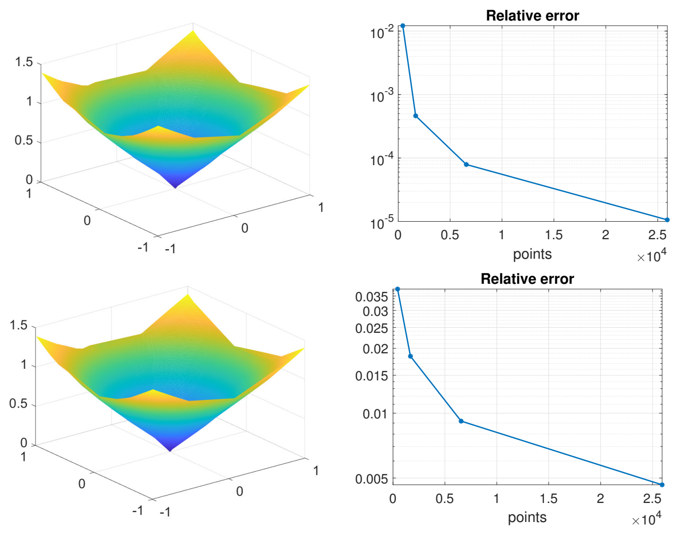

Figure 3.5: Distance functions and errors obtained in the regular grid case. Upper left: distance function calculated in evaluation points using Shepard approximation. Upper right: Relative errors in Shepard approximation case, logarithmic scale. Bottom left: distance function calculated in evaluation points using RBF interpolation. Bottom right: Relative errors in interpolation case, logarithmic scale.

h	Points	Interpolation error	Rate	CPU time (s)	Shepard error	Rate	CPU time (s)
0.0692	21^2	1.2e−2		0.02	3.7e−2		0.03
0.0340	41^2	4.6e−4	4.9	0.17	1.8e−2	1	0.01
0.0175	81^2	7.8e−5	2.4	4.14	9.2e−3	0.9	0.05
0.0084	161^2	1.0e−5	3	163.01	4.7e−3	1	0.22

Table 3.1: Data relative to interpolation using RBF and Shepard approximation on a regular grid.

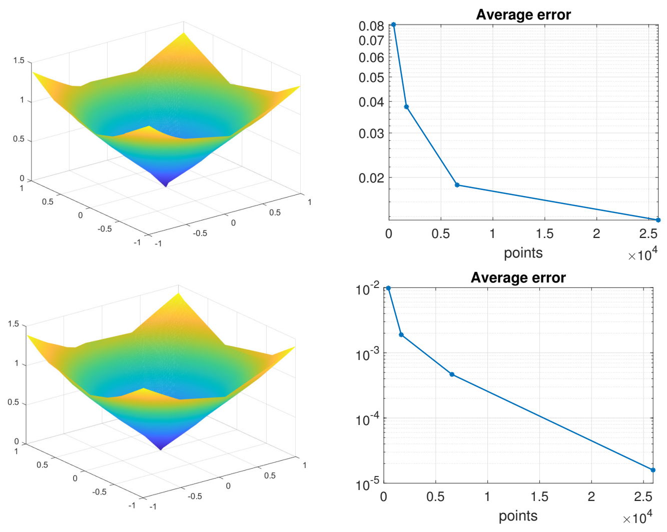

Figure 3.6: Distance functions and errors obtained in the scattered mesh case. Upper left: distance function calculated in evaluation points using Shepard approximation. Upper right: Average relative errors in Shepard approximation case, logarithmic scale. Bottom left: distance function calculated in evaluation points using RBF interpolation. Bottom right: Average relative errors in interpolation case, logarithmic scale.

h	Points	Interpolation error	Rate	CPU time (s)	Shepard error	Rate	CPU time (s)
0.0807	21^2	9.8e−3		0.02	8.0e−2		0.01
0.0340	41^2	1.9e−3	2.6	0.23	4.0e−2	1.2	0.02
0.0175	81^2	4.6e−4	2.4	5.97	1.9e−2	1.2	0.05
0.0084	161^2	1.5e−5	9.8	139.90	1.3e−2	0.9	0.23

Table 3.2: Data relative to interpolation using RBF and Shepard approximation on scattered grids.

In Figure 3.5 and Table 3.1 we see the results related to the Shepard approximation and RBF interpolation in a regular mesh whereas in Figure 3.6 and in Table 3.2 we show the results in the case of scattered meshes. In both cases the error in the Shepard approximation decays almost linearly and this is expected since Shepard's method has approximation of order $\mathcal{O}(h)$ (see e.g. [49]). The RBF interpolation error decays very fast and we can perceive a higher rate of convergence. This behavior is also noted in [49, Chapter 12]. It is important to stress that we are working with interpolation in a non-stationary scenario, which is convergent, but inefficient, since it deals with dense interpolation matrices. This point becomes clear in a comparison between the increase in CPU times (measured in seconds) of RBF interpolation and Shepard approximation with the increase in grid size. For instance, in the case of 161^2 points the CPU time of RBF interpolation is $740\times$ bigger than the time of Shepard approximation in the case of a regular grid and $608\times$ bigger in the case of a scattered grid.

From the results, we can also verify that both interpolation and approximation methods have lower relative errors in structured grids than in unstructured grids.

4
HJB-RBF based approach to the control of PDEs

In this chapter we present our new approach to mitigate the curse of dimensionality coupling the semi-Lagrangian scheme and the Shepard approximation method. The method is based on a scattered data mesh across a region of interest. We also develop a method to automatize the selection of the RBF shape parameter which minimizes the residual. We derive error estimates of the overall approximation method and, finally, summarize everything in one algorithm. In order to test the effectiveness of the numerical scheme we perform tests for low and high dimensional dynamical systems.

4.1
Semi-Lagrangian scheme with Shepard for HJB equation

This section discusses the semi-Lagrangian scheme with the Shepard approximation in place of linear interpolation. Then, we present (i) the localized mesh, (ii) the method used to select the shape parameter, (iii) the error estimates and (iv) the algorithm.

Let us consider a temporal step size $\Delta t > 0$ and build a grid in time such that $t_k = k\Delta t$ with $k \in \mathbb{N}$. We will discuss in the following how to define a spatial discretization, and for now we just denote it as $X = \{x_1, x_2, \dots, x_n\} \subset \Omega$. Furthermore, the set U is also discretized by setting the control $u(t) = u_k \in U$ for $t \in [t_k, t_{k+1})$ piecewise constant. To introduce the approximation of the value function, we represent the Shepard approximant as an operator

$$S^\sigma : (L^\infty, \|\cdot\|_\infty) \to (\mathcal{W}, \|\cdot\|_\infty), \tag{4-1}$$

where $\mathcal{W} = \text{span}\{\psi_1^\sigma, \psi_2^\sigma, \cdots \psi_n^\sigma\}$ as in (3-8). *We remark that the Shepard approximation uses as interpolation nodes the same points X defined above.*

We aim at the reconstruction of the vector $\{V_j\}_{j=1}^n \in \mathbb{R}^n$ where V_j is the approximate value for $v(x_j)$ for each $x_j \in X$. The full discretization of equation (2-5) is obtained starting from a classical approach (see e.g. [20]), as discussed in Section 2.3 , but replacing the local linear interpolation operator on a structured grid (equation (2-19)) with the Shepard approximation operator following [22]. This discretization reads

$$V_j = [W_\sigma(V)]_j := \min_{u\in U}\left\{\Delta t\, g(x_j,u)+(1-\Delta t\lambda)S^\sigma[V](x_j+\Delta t\, f(x_j,u))\right\}. \quad (4\text{-}2)$$

In (4-2) the set U is discretized in a finite number $M \in \mathbb{N}$ of points $U := \{u_1,\dots,u_M\}$, and the minimum is computed by comparison. The full approximation scheme of the value function is the Value Iteration (VI) method, as discussed in Chapter 2, it is obtained by iteration of (4-2), i.e.,

$$V^{k+1} = W_\sigma(V^k), \quad k = 0,1,\dots. \quad (4\text{-}3)$$

In this context, the Shepard operator offers a striking benefit in comparison with RBF interpolation. Indeed, S^σ in (4-1) has unit norm, and this implies that the right hand side of (4-2) is a contraction if $\Delta t \in (0, 1/\lambda]$ (see Lemma 4.1 below). Therefore, the convergence of the value iteration scheme is guaranteed.

Lemma 4.1 *The Shepard operator $S^\sigma : (L^\infty, \|\cdot\|_\infty) \to (\mathcal{W}, \|\cdot\|_\infty)$ has norm equal 1.*

Proof. By the assumption of the basis functions $\psi_i^\sigma > 0$ for $i = 1,\dots,n$ and by the fact that for each $x \in \Omega$, $S^\sigma v(x)$ is a convex combination of the values $v(x_1), v(x_2),\dots, v(x_n)$ we have

$$|S^\sigma v(x)| \le \sum_{i=1}^n |S^\sigma v(x_i)\psi_i^\sigma(x)| \le \max_{i=1,\dots,n}|v(x_i)|\sum_{i=1}^n |\psi_i^\sigma(x)| \le \max_{i=1,\dots,n}|v(x_i)| \le \|v\|_\infty$$

■

As soon as we obtain an approximation of the value function, we can compute the numerical feedback control as

$$u_n^*(x) = \arg\min_{u\in U}\{\Delta t g(x,u) + (1-\lambda\Delta t)S^\sigma[V](x+\Delta t f(x,u))\}, \quad (4\text{-}4)$$

with $x = y(t_n)$. Thus, we are able to perform a reconstruction of an optimal trajectory y^* and optimal control u^*.

Under the assumption that the fill distance h decays to zero and that the shape parameter scales as $\sigma = \theta/h$, the approximation scheme (4-2) converges [22]. More precisely, under suitable assumptions on f and g Theorem 3 in [22] guarantees that $\|v - V\|_\infty \le (C/\theta)h$, where C depends on the dynamics but not on the discretization.

4.1.1 The Scattered Mesh

Despite these convincing theoretical guarantees, the requirement that $h = h_{X,\Omega}$ decays to zero is too restrictive in our setting, since filling the entire computational domain Ω may be out of reach for high dimensional problems.

Moreover, as already mentioned in Section 3.2, Shepard's method performs approximations in high dimensions and unstructured grids, while in [22] the authors focused on a given configuration for the shape parameter and an equidistant grid. In the next paragraphs, we will explain how to select the shape parameter and to generate unstructured meshes to solve high dimensional problems. This approach can also be used for minimum time problem as we will see in Section 4.2.

Mesh. Different possibilities are available for the discretization of the spatial domain. A standard choice would be to use an equi-distributed grid, which covers the entire space and usually provides accurate results for interpolation problems. Unfortunately, for higher dimensional problems it is impossible to think to work on equi-distributed grids, as their size grows exponentially. This is a particular limitation in our case, since our goal is to control PDEs, whose discretization leads to high-dimensional problems e.g. $d \gtrsim \mathcal{O}(10^3)$. On the other hand, a random set of points is computationally efficient to generate and to use, but additional care should be taken since the distribution of points is usually irregular (some regions can be more densely populated than others) and the fill distance may decrease only very slowly when increasing the number of points.

In general terms, there is a tradeoff between keeping the grid at a reasonable size and the need to cover the relevant part of the computational domain. In particular, it is well known that the fill distance h for any sequence of points $\{X_n\}_{n\in\mathbb{N}}$ can decrease at most as $h \leq c_\Omega n^{-1/d}$ in $\mathbb{R}^d$ for a suitable constant $c_\Omega > 0$ depending only on the geometry of the domain. Observe that uniform points have precisely this asymptotic decay of the fill distance. Thus, an exponentially growing number of points is required to obtain a well spread covering of Ω as d increases.

The key point to overcome these limitations is to observe that the evolution of the system provides itself an indication of the regions of interest within the domain. Following this idea, we propose a discretization method driven by the dynamics of the control problem (2-1). Observe that a similar idea has been used in [51] to compute the value function along the trajectories of open-loop control problems. Also in [30] the grid has been generated by points of the dynamics leading to the solution of the HJB equation on a tree structure. We stress once again that this kind of approach is an effective way to address the curse of dimensionality, since only the parts of the space that are visited by some system trajectories are considered, without the need of filling the entire space. This is possible at the price of having a local approximation, but this locality is taken in full account in the following estimates, so that one can balance between the efficiency of the method and the coverage of the domain.

To define our dynamics-driven grid we fix a time step $\overline{\Delta t} > 0$, a maximum number $\overline{K} \in \mathbb{N}$ of discrete times and, for $\overline{L}, \overline{M} > 0$, some initial conditions of interest and a discretization of the control space, i.e.,

$$\overline{X} := \{\bar{x}_1, \bar{x}_2, \dots, \bar{x}_{\overline{L}}\} \subset \Omega, \quad \overline{U} := \{\bar{u}_1, \bar{u}_2, \dots, \bar{u}_{\overline{M}}\} \subset U.$$

Observe that all these parameters do not need to coincide with the ones used in the solution of the value iteration (4-2), but they are rather used to construct the grid. In general we use $\overline{\Delta t} > \Delta t$ and $\overline{M} < M$, i.e., the discretization used to construct the mesh is coarser than the one used to solve the control problem.

Given an initial condition $\bar{x}_i \in \overline{X}$ and a control $\bar{u}_j \in \overline{U}$, we solve numerically for $i = 1, \dots, \bar{L}$ and $j = 1, \dots, \bar{M}$ equation (2-1) to obtain trajectories

$$\begin{aligned} x_{i,j}^{k+1} &= x_{i,j}^k + \overline{\Delta t}\, f(x_{i,j}^k, \bar{u}_j), \quad k = 1, \dots, \bar{K} - 1, \\ x_{i,j}^1 &= \bar{x}_i, \end{aligned} \tag{4-5}$$

such that $x_{i,j}^k$ is an approximation of the state variable with initial condition $\bar{x}_i$, constant control $u(t) = \bar{u}_j$, for all time $t = k\overline{\Delta t}$. For each pair $(\bar{x}_i, \bar{u}_j)$ we obtain the set $X(\bar{x}_i, \bar{u}_j) := \{x_{i,j}^1, \dots, x_{i,j}^{\bar{K}}\}$ containing the discrete trajectory, and our mesh is defined as

$$X := X(\overline{X}, \overline{U}, \overline{\Delta t}, \overline{K}) := \bigcup_{i=1}^{\bar{L}} \bigcup_{j=1}^{\bar{M}} X(\bar{x}_i, \bar{u}_j). \tag{4-6}$$

In this view, the values of $\overline{X}, \overline{U}, \overline{\Delta t}, \overline{K}$ should be chosen such that X contains points that are suitably close to the points of interest for the solution of the control problem. In the following proposition we provide a quantitative version of this idea, which will be the base of our error estimate in Theorem 4.5.

Proposition 4.2 *Let $X := X(\overline{X}, \overline{U}, \overline{\Delta t}, \overline{K})$ be the dynamics-dependent mesh of* (4-6)*, and assume that f is uniformly bounded i.e., there exists $M_f > 0$ such that*

$$\sup_{x \in \Omega, u \in U} \|f(x,u)\| \leq M_f.$$

Then, for each $x \in X$, $\Delta t > 0$ and $u \in U$,

$$dist(x + \Delta t f(x,u), X) \leq M_f \Delta t, \tag{4-7}$$

where the distance between a point x to a set X is defined as $dist(x, X) := \inf\{dist(x, y)|\ y \in X\}$. Assume furthermore that f is uniformly Lipschitz continuous in both variables, i.e., there exist $L_x, L_u > 0$ such that

$$\|f(x,u) - f(x',u)\| \le L_x\|x - x'\| \ \ \forall x, x' \in \Omega, \ \ u \in U,$$
$$\|f(x,u) - f(x,u')\| \le L_u\|u - u'\| \ \ \forall x \in \Omega, \ \ u, u' \in U.$$

Then, if $x := x^k(x_0, u, \Delta t) \in \Omega$ is a point on a discrete trajectory with initial point $x_0 \in \Omega$, control $u \in U$, timestep $\Delta t > 0$, and time instant t_k, with $k \in \mathbb{N}$, $k \le \bar{K}$,

$$dist(x, X) \le \left(|\Delta t - \overline{\Delta t}|\bar{K} M_f + \min_{\bar{x} \in \bar{X}} \|\bar{x} - x_0\| + \bar{K}\overline{\Delta t} L_u \min_{\bar{u} \in \bar{U}} \|\bar{u} - u\| \right) e^{\bar{K}\overline{\Delta t} L_x}. \tag{4-8}$$

Proof. If $x \in X$ we simply have

$$\begin{aligned} \mathrm{dist}(x + \Delta t f(x,u), X) &= \min_{x' \in X} \|x + \Delta t f(x,u) - x'\| \le \|x + \Delta t f(x,u) - x\| \\ &= \Delta t \|f(x,u)\| \le M_f \Delta t, \end{aligned}$$

which gives the bound (4-7) using only the boundedness of f.

To prove (4-8) we need to work explicitly with the initial points and the control values. By assumption we have

$$x = x^k(x_0, u, \Delta t) = x_0 + \Delta t \sum_{p=0}^{k-1} f(x^p(x_0, u, \Delta t), u),$$

and since $x' \in X$, using the definition (4-6) we can choose it as

$$x' = x^k_{\ell,m} = \bar{x}_\ell + \overline{\Delta t} \sum_{p=0}^{k-1} f(x^p_{\ell,m}, \bar{u}_m),$$

for some $\ell \in \{1, \dots, \bar{L}\}$, $m \in \{1, \dots, \bar{M}\}$, and with $x^0_{\ell,m} = x_0$.

It follows that

$$\begin{aligned} x^k(x_0, u, \Delta t) - x^k_{\ell,m} = &\ x_0 - \bar{x}_\ell + \\ &+ \Delta t \sum_{p=0}^{k-1} f(x^p(x_0, u, \Delta t), u) - \overline{\Delta t} \sum_{p=0}^{k-1} f(x^p_{\ell m}, \bar{u}_m), \end{aligned}$$

and thus adding and subtracting $\overline{\Delta t}\sum_{p=0}^{k-1} f(x^p(x_0,u,\Delta t),u)$ we obtain

$$\begin{aligned}
\|x^k(x_0,u,\Delta t) - x^k_{\ell,m}\| \leq \; & \|x_0 - \bar{x}_\ell\| + |\Delta t - \overline{\Delta t}| \sum_{p=0}^{k-1} \|f(x^p(x_0,u,\Delta t),u)\| + \\
& + \overline{\Delta t} \sum_{p=0}^{k-1} \|f(x^p(x_0,u,\Delta t),u) - f(x^p_{\ell m},\bar{u}_m)\| \\
\leq \; & \|x_0 - \bar{x}_\ell\| + |\Delta t - \overline{\Delta t}| k M_f + \\
& + \overline{\Delta t} \sum_{p=0}^{k-1} \|f(x^p(x_0,u,\Delta t),u) - f(x^p_{\ell m},\bar{u}_m)\|.
\end{aligned}$$

Now adding and subtracting $f(x^p(x_0,u,\Delta t),\bar{u}_m)$ in the sum, and using the Lipschitz continuity of f, we get

$$\begin{aligned}
\|x^k(x_0,u,\Delta t) - x^k_{\ell,m}\| \leq \; & \|x_0 - \bar{x}_\ell\| + |\Delta t - \overline{\Delta t}| k M_f + \\
& + \overline{\Delta t} \sum_{p=0}^{k-1} \Big(L_u \|u - \bar{u}_m\| + L_x \|x^p(x_0,u,\Delta t) - x^p_{\ell,m}\| \Big) \\
= \; & \|x_0 - \bar{x}_\ell\| + |\Delta t - \overline{\Delta t}| k M_f + k\overline{\Delta t} L_u \|u - \bar{u}_m\| + \\
& + \overline{\Delta t} L_x \sum_{p=0}^{k-1} \|x^p(x_0,u,\Delta t) - x^p_{\ell,m}\|.
\end{aligned}$$

Applying the discrete Grönwall lemma to this inequality gives

$$\|x^k(x_0,u,\Delta t) - x^k_{\ell,m}\| \leq \Big(\|x_0 - \bar{x}_\ell\| + |\Delta t - \overline{\Delta t}| k M_f + k\overline{\Delta t} L_u \|u - \bar{u}_m\| \Big) e^{k\overline{\Delta t} L_x},$$

and since ℓ and m are free, we can choose them as $\bar{u}_m := \underset{\bar{u}\in\bar{U}}{\arg\min} \|u - \bar{u}\|$ and $x_\ell := \underset{\bar{x}\in\bar{X}}{\arg\min} \|x_0 - \bar{x}\|$. Finally, bounding k by $\bar{K}$ gives (4-8). ■

4.1.2 Selection of the shape parameter

The quality of Shepard approximation strongly depends on the choice of the shape parameter σ. As mentioned in the introduction, several techniques exist to tune the shape parameter in the RBF literature, such as cross validation and maximum likelihood estimation (see e.g. Chapter 14 in [52]), but they are designed to optimize the value of σ in a fixed approximation setting. In our case, on the other hand, we need to construct an approximant at each iteration k within the value iteration (4-2). This makes the existing methods computationally expensive and difficult to adapt to the target of minimizing the error in the iterative method.

For these reasons, we propose here a new method to select the shape parameter based on the minimization of a problem-specific indicator, namely the residual $R(\sigma) := R(V_\sigma)$. We will denote by V_σ the discrete value function related to the choice σ in the Shepard approximation. Assuming that the value iteration with parameter σ has been stopped at iteration k_{final} giving the solution $V_\sigma := V^{k_{\text{final}}}$, we define the residual as

$$R(\sigma) := \|V_\sigma - W_\sigma(V_\sigma)\|_\infty, \tag{4-9}$$

and we choose the shape parameter that minimizes this quantity with respect to σ.

To get a suitable scale for the values of σ, we parametrize it in terms of the grid X, similarly to what it is done in [22], we set $\sigma := \theta/h_{\Omega,X}$ for a given $\theta > 0$. Since $h_{\Omega,X}$ is difficult to compute or even to estimate in high dimensional problems, we resort to setting $\sigma = \theta/q_X$ and we optimize the value of $\theta > 0$. Observe that the separation distance q_X is an easily computable quantity that depends only on X, and it is thus actually feasible to use this parametrization even in high dimensions.

Choosing an admissible set of parameters $\mathcal{P} := [\theta_{\min}, \theta_{\max}] \subset \mathbb{R}^+$, the parameter is thus chosen by solving the optimization problem

$$\bar{\theta} := \underset{\theta\in\mathcal{P}}{\arg\min}\, R(\theta/q_X) = \underset{\theta\in\mathcal{P}}{\arg\min}\, \|V_{\theta/q_X} - W_{\theta/q_X}(V_{\theta/q_X})\|_\infty. \tag{4-10}$$

Remark. This problem can be solved by using a comparison method or e.g. an inexact gradient method. The former means to discretize the set $\mathcal{P}$ as $\{\theta_1, \dots, \theta_{N_p}\} \subset \mathcal{P}$ and to compute all the value functions for all θ_i, $i = 1, \dots, N_p$. The latter considers a projected gradient method where the parameter space $\mathcal{P}$ is continuous and the derivative is approximated by

$$R_\theta = \frac{R(\theta + \varepsilon) - R(\theta)}{\varepsilon},$$

for some fixed $\varepsilon > 0$. In the numerical tests, we will compare both minimization strategies in the low dimensional case, while we will concentrate on the comparison method in high dimensional one.

4.1.3 Error Estimates

We adapt the classical theory that is used to prove rates of convergence for the value iteration when linear interpolation is performed (see [20, Section 8.4.1]).

The idea is to estimate the time and space discretizations separately. Since the time discretization is independent of the interpolation scheme used in the space discretization, we just recall the following result from [20, Section 8.4.1]. The original result holds for the supremum norm over Ω, we formulate it here for a general compact subset $\tilde{\Omega} \subset \Omega$ in order to deal with the space discretization later. Let $\|f\|_{\infty,\tilde{\Omega}} := \sup_{x\in\tilde{\Omega}} |f(x)|$.

Theorem 4.3 *Let v be the exact value function, and $v^{\Delta t}$ be the solution of the value iteration* (4-2) *without space discretization, i.e.,*

$$v^{\Delta t}(x) = \min_{u\in U} \left\{ \Delta t\, g(x,u) + e^{-\lambda\Delta t} v^{\Delta t}(x + \Delta t\, f(x,u)) \right\}. \tag{4-11}$$

If v is Lipschitz continuous, for each compact subset $\tilde{\Omega} \subset \Omega$ there exists a constant $C := C(\tilde{\Omega}) > 0$ such that

$$\left\| v - v^{\Delta t} \right\|_{\infty,\tilde{\Omega}} \leq C\Delta t^{1/2}. \tag{4-12}$$

Assume additionally that the following hold:

1. *f is uniformly bounded, U is convex, $f(x,u)$ is linear in u.*

2. *$g(\cdot,u)$ is Lipschitz continuous and $g(x,\cdot)$ is convex.*

3. *There exists an optimal control $u^\star \in \mathcal{U}$.*

Then there exists $C' := C'(\tilde{\Omega}) > 0$ such that

$$\left\| v - v^{\Delta t} \right\|_{\infty,\tilde{\Omega}} \leq C'\Delta t. \tag{4-13}$$

The proof of estimate (4-12) can be found in Chapter VI of [12]. Here, we focus in the proof of (4-13) since it is used in the construction of our error estimate.

Proof. In order to prove (4-13) using the assumptions above, let us consider the explicit minimum in (4-11) and in (2-4) with optimal controls $\hat{u}$ and $u^*(s)$

$$\begin{aligned} v^{\Delta t}(x) &= \Delta t\, g(x,\hat{u}) + e^{-\lambda\Delta t} v^{\Delta t}(x + \Delta t\, f(x,\hat{u})) \\ v(x) &= \int_0^{\Delta t} g(y_x(s), u^*(s)) e^{-\lambda s} ds + e^{-\lambda\Delta t}\, v(y_x(\Delta t)). \end{aligned}$$

To ease the notations, we omit the dependence of the trajectory on the control $y_x(\Delta t) = y_x(\Delta t, u)$.

For the first part of the proof, we consider $u(s) = \hat{u}$ and verify the following inequality:

$$\begin{aligned} v(x) - v^{\Delta t}(x) \leq \int_0^{\Delta t} g(y_x(s), \hat{u}) e^{-\lambda s} ds + e^{-\lambda \Delta t}\, v(y_x(\Delta t)) \\ - \Delta t\, g(x, \hat{u}) - e^{-\lambda \Delta t} v^{\Delta t}(x + \Delta t\, f(x, \hat{u})). \end{aligned} \tag{4-14}$$

Then, using approximation arguments on $g(y_x(s), \hat{u})e^{-\lambda s}$ and integrating we obtain

$$\int_0^{\Delta t} g(y_x(s), \hat{u}) e^{-\lambda s} ds - \Delta t g(x, \hat{u}) = \mathcal{O}(\Delta t^2). \tag{4-15}$$

On the other hand, working with the remaining terms of (4-14)

$$\begin{aligned} v(y_x(\Delta t)) - v^{\Delta t}(x + \Delta t\, f(x, \hat{u})) &= v(y_x(\Delta t)) - v(x + \Delta t\, f(x, \hat{u})) + \\ &+ v(x + \Delta t\, f(x, \hat{u})) - v^{\Delta t}(x + \Delta t\, f(x, \hat{u})) \\ &\leq L_v \left\| y_x(\Delta t) - x - \Delta t\, f(x, \hat{u}) \right\| + \left\| v - v^{\Delta t} \right\|_{\infty, \tilde{\Omega}} \\ &\leq \mathcal{O}(\Delta t^2) + \left\| v - v^{\Delta t} \right\|_{\infty, \tilde{\Omega}}, \end{aligned} \tag{4-16}$$

where we have used the Lipschitz continuity of the value function and an approximation argument in $y_x(\Delta t) - x - \Delta t\, f(x, \hat{u})$. We then plug inequalities (4-15) and (4-16) into (4-14) and we end up with the unilateral estimate

$$v(x) - v^{\Delta t}(x) \leq \mathcal{O}(\Delta t^2) + e^{-\lambda \Delta t} \left\| v - v^{\Delta t} \right\|_{\infty, \tilde{\Omega}}. \tag{4-17}$$

To the estimate of $v^{\Delta t}(x) - v(x)$ we consider the control $\bar{u}$, which is suboptimal in $v^{\Delta t}(x)$. Thus,

$$\begin{aligned} v^{\Delta t}(x) - v(x) \leq \Delta t\, g(x, \bar{u}) + e^{-\lambda \Delta t} v^{\Delta t}(x + \Delta t\, f(x, \bar{u})) \\ - \int_0^{\Delta t} g(y_x(s), u^*(s)) e^{-\lambda s} ds - e^{-\lambda \Delta t}\, v(y_x(\Delta t)). \end{aligned} \tag{4-18}$$

The control $\bar{u}$ is chosen in order to $\bar{u} = \frac{1}{\Delta t} \int_0^{\Delta t} u^*(s) ds$. Using *assumption 1* that $f(x, u)$ is linear in u and explicitly considering $f(x, u) = f_1(x) + f_2(x)u$, with $f_1 : \mathbb{R}^d \to \mathbb{R}^d$ and f_2 a $d \times m$ matrix, we can write

$$\begin{aligned}
y_x(\Delta t) &= x + \int_0^{\Delta t} f(y_x(s), u^*(s))ds \\
&= x + \int_0^{\Delta t} f_1(y_x(s))ds + \int_0^{\Delta t} f_2(y_x(s))u^*(s)ds \\
&= x + \int_0^{\Delta t} (f_1(x) + \mathcal{O}(\Delta t))ds + \int_0^{\Delta t} (f_2(x) + \mathcal{O}(\Delta t))u^*(s)ds \\
&= x + \Delta t(f_1(x) + f_2(x)\bar{u}) + \mathcal{O}(\Delta t^2) \\
&= x + \Delta t f(x, \bar{u}) + \mathcal{O}(\Delta t^2). \qquad (4\text{-}19)
\end{aligned}$$

We can develop the term $\int_0^{\Delta t} g(y_x(s), u^*(s))e^{-\lambda s}ds$ as

$$\begin{aligned}
\int_0^{\Delta t} g(y_x(s), u^*(s))e^{-\lambda s}ds &= \int_0^{\Delta t} (g(x, u^*(s)) + \mathcal{O}(\Delta t))(1 + \mathcal{O}(\Delta t))ds \\
&= \int_0^{\Delta t} g(x, u^*(s))ds + \mathcal{O}(\Delta t^2) \\
&\geq \Delta t g(x, \bar{u}) + \mathcal{O}(\Delta t^2), \qquad (4\text{-}20)
\end{aligned}$$

where we have used the fact that $g(x, \cdot)$ is convex combined with Jensen inequality in the last part. Using results (4-19), (4-20) and working in (4-18) similarly as done in the proof of the first unilateral estimate (4-17), we have

$$\begin{aligned}
v^{\Delta t}(x) - v(x) &\leq \Delta t\, g(x, \bar{u}) + e^{-\lambda \Delta t} v^{\Delta t}(x + \Delta t\, f(x, \bar{u})) \\
&\quad - \int_0^{\Delta t} g(y_x(s), u^*(s))e^{-\lambda s}ds - e^{-\lambda \Delta t}\, v(y_x(\Delta t)) \\
&\leq \Delta t\, g(x, \bar{u}) - \Delta t\, g(x, \bar{u}) + \mathcal{O}(\Delta t^2) \\
&\quad + e^{-\lambda \Delta t} v^{\Delta t}(x + \Delta t\, f(x, \bar{u})) - e^{-\lambda \Delta t}\, v(y_x(\Delta t))
\end{aligned}$$

resulting in

$$v^{\Delta t}(x) - v(x) \leq \mathcal{O}(\Delta t^2) + e^{-\lambda \Delta t} \left\| v - v^{\Delta t} \right\|_{\infty, \tilde{\Omega}}. \qquad (4\text{-}21)$$

Using (4-17) and (4-21)

$$|v(x) - v^{\Delta t}(x)| \leq \mathcal{O}(\Delta t^2) + e^{-\lambda \Delta t} \left\| v - v^{\Delta t} \right\|_{\infty, \tilde{\Omega}} \qquad (4\text{-}22)$$

Passing to the norm $\|.\|_\infty$ and adjusting terms give us the final estimate

$$\begin{aligned}
(1 - e^{-\lambda \Delta t}) \left\| v - v^{\Delta t} \right\|_{\infty, \tilde{\Omega}} &\leq \bar{C} \Delta t^2 \\
\left\| v - v^{\Delta t} \right\|_{\infty, \tilde{\Omega}} &\leq C' \Delta t. \qquad (4\text{-}23)
\end{aligned}$$

■

It remains now to quantify the error that is committed by introducing a space discretization, i.e., the error associated to the interpolation scheme in (4-2). To do this, we first require a bound on the error of Shepard interpolation, and we report in the next proposition a result obtained in [22]. In this case, the key idea is to scale the shape parameter of the RBF basis according to the fill distance of the mesh. Following Proposition 4.2, we can control the fill distance $h_{X,\tilde{\Omega}}$ of the mesh X within the set $\tilde{\Omega}$ obtained as the collection of the different trajectories of the discrete dynamics. In other words, we set

$$\tilde{\Omega} := \tilde{\Omega}(\tilde{X}, \tilde{U}, \tilde{T}) := \left\{ x := x^k(x_0, u, \Delta t) : x_0 \in \tilde{X}, u \in \tilde{U}, \Delta t \in \tilde{T}, k \leq \bar{K} \right\}. \tag{4-24}$$

For this set, equation (4-8) gives

$$\begin{aligned} h_{X,\tilde{\Omega}} &:= \sup_{x \in \tilde{\Omega}} \operatorname{dist}(x, X) \\ &\leq \left(\sup_{\Delta t \in \tilde{T}} |\Delta t - \overline{\Delta t}| \bar{K} M_f + \sup_{x_0 \in \tilde{X}} \min_{\bar{x} \in \bar{X}} \|\bar{x} - x_0\| + \bar{K}\overline{\Delta t} L_u \sup_{u \in \tilde{U}} \min_{\bar{u} \in \bar{U}} \|\bar{u} - u\| \right) e^{\bar{K}\overline{\Delta t} L_x} \end{aligned} \tag{4-25}$$

We now recall from [22] the error estimate for the Shepard approximation.

Proposition 4.4 *Let $L_v > 0$ be the Lipschitz constant of $v : \Omega \to \mathbb{R}$. Let S^σ be the Shepard approximation of v on X obtained using a kernel with support contained in $B(0,1)$, and let $\sigma := C/h_{X,\tilde{\Omega}}$ for a positive constant $C > 0$. Then*

$$\|v - S^\sigma[v]\|_{\infty,\tilde{\Omega}} \leq C L_v h_{X,\tilde{\Omega}}. \tag{4-26}$$

We can finally prove our main result.

Theorem 4.5 *Let $\tilde{\Omega}$ given in (4-24), and assume that $\tilde{U}$ contains the two controls that are optimal for $v^{\Delta t}$ and for V. Then, under the assumptions of Proposition 4.4,*

$$\|V - v^{\Delta t}\|_{\infty,\tilde{\Omega}} \leq \frac{C L_v}{\lambda} \frac{h_{X,\tilde{\Omega}}}{\Delta t} \tag{4-27}$$

$$\leq \frac{C L_v}{\lambda} \frac{e^{\bar{K}\overline{\Delta t} L_x}}{\Delta t} \left(\sup_{\Delta t \in \tilde{T}} |\Delta t - \overline{\Delta t}| \bar{K} M_f + \sup_{x_0 \in \tilde{X}} \min_{\bar{x} \in \bar{X}} \|\bar{x} - x_0\| \quad + \right.$$
$$\left. + \quad \bar{K}\overline{\Delta t} L_u \sup_{u \in \tilde{U}} \min_{\bar{u} \in \bar{U}} \|\bar{u} - u\| \right)$$

Proof. We consider the control $\hat{u}$ that is optimal for $v^{\Delta t}$, and we define $\hat{z} := x_j + \Delta t f(x_j, \hat{u})$ for $1 \leq j \leq n$. This implies in particular that $\hat{u}$ is sub-optimal for V, and thus by the definition (4-2) for each $1 \leq j \leq n$,

$$V_j \leq \Delta t\, g(x_j, \hat{u}) + (1 - \Delta t \lambda) S^{\sigma}[V](\hat{z}).$$

Combining this inequality with the definition (4-11) of $v^{\Delta t}(x_j)$, we have

$$\begin{aligned} V_j - v^{\Delta t}(x_j) &\leq \Delta t\, g(x_j, \hat{u}) + (1 - \Delta t \lambda) S^{\sigma}[V](\hat{z}) - \Delta t\, g(x_j, \hat{u}) - \\ &\qquad - (1 - \Delta t \lambda) v^{\Delta t}(\hat{z}) \\ &= (1 - \Delta t \lambda) \left(S^{\sigma}[V](\hat{z}) - v^{\Delta t}(\hat{z}) \right). \end{aligned}$$

Now, we add and subtract the interpolation $S^{\sigma}[v^{\Delta t}]$ of $v^{\Delta t}$ evaluated at $\hat{z}$, use the bound (4-26) and the fact that S^{σ} is a contraction to obtain

$$\begin{aligned} V_j - v^{\Delta t}(x_j) &\leq (1 - \Delta t \lambda) \left(S^{\sigma}[V](\hat{z}) - S^{\sigma}[v^{\Delta t}](\hat{z}) + S^{\sigma}[v^{\Delta t}](\hat{z}) - v^{\Delta t}(\hat{z}) \right) \\ &\leq (1 - \Delta t \lambda) \left(\|V - v^{\Delta t}\|_{\infty, \tilde{\Omega}} + C L_v h_{X, \tilde{\Omega}} \right). \end{aligned}$$

We can now repeat the same reasoning with the control $\tilde{u}$ that is optimal for V, and in this way we can obtain a bound on the opposite quantity $v^{\Delta t}(x_j) - V_j$. These two inequalities, when combined, give the desired bound:

$$\|V - v^{\Delta t}\|_{\infty, \tilde{\Omega}} \leq \frac{C L_v}{\lambda} \frac{h_{X, \tilde{\Omega}}}{\Delta t}.$$

■

By the triangular inequality from (4-13) and (4-27), one can obtain

$$\|v - V\|_{\infty, \tilde{\Omega}} \leq C' \Delta t + \frac{C L_v}{\lambda} \frac{h_{X, \tilde{\Omega}}}{\Delta t}.$$

4.1.4 Algorithm

The generation of the localized mesh, the value iteration algorithm with Shepard approximation and the selection of the shape parameter can be summarized in the following algorithm.

Algorithm 5: Value Iteration with shape parameter selection

```
1: INPUT: $\Omega, \Delta t, U, \mathcal{P}$ parameter range, tolerance, RBF and system dynamics $f$,
   flag
2: initialization;
3: Generate Mesh
4: if flag == Comparison then
5:     for $\theta \in \mathcal{P}$ do
6:         Compute $V_\theta$;
7:         $R(\theta) = ||V_\theta - W(V_\theta)||_\infty$
8:     end for
9:     $\bar{\theta} = \underset{\theta \in P}{\arg\min}\ R(\theta)$;
10: else
11:     $R_\theta = 1, \theta = \theta_0$, tol, $\varepsilon$
12:     while $\|R_\theta\| > tol$ do
13:         Compute $V_\theta$ and $V_{\theta+\varepsilon}$
14:         Evaluate $R(V_\theta)$ and $R(V_{\theta+\varepsilon})$
15:         $R_\theta = \dfrac{R(V_{\theta+\varepsilon}) - R(V_\theta)}{\varepsilon}$
16:         $\theta = \theta - R_\theta$
17:         $\theta = \max(\min(\theta_{\max}, \theta), \theta_{\min})$        (projection into $\mathcal{P}$)
18:     end while
19:     $\bar{\theta} = \theta, V_{\bar{\theta}} = V_\theta$
20: end if
21: OUTPUT: $\{\bar{\theta}, V_{\bar{\theta}}\}$
```

4.2 Numerical experiments

In this section we present three numerical tests to illustrate the effectiveness of the proposed algorithm. The first test is a two dimensional minimum time problem with well-known analytical solution. In this test we analyse results using regular and scattered grids. The second and the third test deal with a advection equation and a nonlinear heat equation, respectively. We discretize in space both PDEs using finite differences. The dimension of the semi-discrete problem will be 10201 for the advection equation and 961 for the parabolic problem. Note that the high dimensions of the PDEs examples make the problems impossible to solve by classical methods. For each test we provide examples of feedback reconstruction for different initial conditions that may not belong to the grid. In the parabolic case we present the effectiveness of the feedback control under disturbances of the system.

In every experiment, we define an admissible interval $\mathcal{P}$ to solve the minimization problem by comparison in Algorithm 5 as follows: we start with a large interval $\mathcal{P}_1$ and we coarsely discretize it. Then, we run Algorithm 5 and obtain $\bar{\theta}_1$. Later, we choose a set $\mathcal{P}_2 \subset \mathcal{P}_1$ such that $\bar{\theta}_1 \in \mathcal{P}_2$. Using $\mathcal{P}_2$ and a finer refinement, the Algorithm provides $\bar{\theta}_2$. We iterate this procedure, and finally we set $\mathcal{P} = \mathcal{P}_n$.

For the purpose of numerical computations we consider only finite horizons $t \in [0, T]$ with a given $T > 0$ large enough to simulate the infinite horizon problem. We also assume that the dynamics evolve for each initial value and control parameter within a compact set $\Omega \subset \mathbb{R}^d$.

In our tests, we use the Wendland RBF defined in (3-4).

Test 1: Eikonal equation in 2D

The first test is the same problem studied in Subsection 2.3.3. We consider a two dimensional minimum time problem in $\Omega = [-1, 1]^2$ with the following dynamics and control space

$$f(x, u) = \begin{pmatrix} \cos(u) \\ \sin(u) \end{pmatrix}, \quad U = [0, 2\pi]. \tag{4-28}$$

The cost functional to be minimized is $\mathcal{J}_x(y, u) = \int_0^{t(x,u)} e^{-\lambda s} ds$ with

$$t(x, u) := \begin{cases} \inf_s\{s \in \mathbb{R}_+ : y_x(s, u) \in \mathcal{T}\} & \text{if } y_x(s, u) \in \mathcal{T} \text{ for some } s \\ +\infty & \text{otherwise,} \end{cases} \tag{4-29}$$

being the arrival time to the target $\mathcal{T} = (0, 0)$ for each $x \in [-1, 1]^2$.

The analytical solution $V^*(x)$ for this problem is the Kruzkov transform of the distance to the target $\mathcal{T}$ (see e.g. [12]):

$$V^*(x) := \begin{cases} 1 & \text{if } v^*(x) = \infty, \\ 1 - \exp(-v^*(x)) & \text{else} \end{cases}$$

with $v^*(x) = ||x||_2$ for each $x \in [-1, 1]^2$.

We tested Algorithm 5 for two different cases. The first one considers an unstructured grid generated by random points, whereas the second case studies an unstructured grid generated by the problem dynamics. Due to the randomness of our grid we compute an average error. In the first case we will average over 10 tests whereas on the second one over 5. We do not discuss in detail the case with a regular grid since it has been extensively analyzed in [22]. However, in Example 4.2, we show optimal trajectories using also the regular grid with linear interpolation and the Shepard's approximant. The *relative error* reads

$$\mathcal{E}(V_\theta) = \frac{||V_\theta - V^*||_\infty}{||V^*||_\infty} \tag{4-30}$$

where V_θ is the discrete value function obtained with the shape parameter $\sigma = \frac{\theta}{h}$ and V^* is the exact solution. We will denote by θ^* the parameter selected in order to minimize the *relative error* from (4-30):

$$\theta^* := \underset{\theta \in \mathcal{P}}{\arg\min}\ \mathcal{E}(V_\theta)$$

Clearly, $\mathcal{E}(V_{\theta^*})$ will be a lower bound for $\mathcal{E}(V_\theta)$. In the following three cases we set $\lambda = 1$ and U is discretized with 16 equidistant controls.

Example 1. Random Grid

The first test with the Eikonal equation is performed using an unstructured grid generated by random points. In order to obtain a grid which densely covers our numerical domain, a set of 40000 randomly distributed points in $[-1,1]^2$ is clustered using the *k-means* algorithm, where k is the number of desired points in the grid. Examples of this type of grids are shown in Figure 4.1.

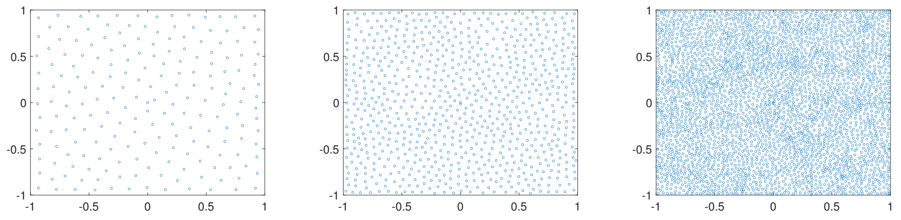

Figure 4.1: Example 1. Random generated grids. Left: 200 points and fill distance 0.1618. Center: 800 points and fill distance 0.0846. Right: 3200 points and fill distance 0.0461.

We found $\mathcal{P} = [1,3]$ a suitable parameter space discretized with 0.1 as step size. In the HJB equation we set $\Delta t = h$. In the left panel of Figure 4.2 we see an example of residual $R(\theta)$ when the unstructured grid is formed by 3200 nodes. The residual is minimized between 1.5 and 2. The average value after 10 tests is $\bar{\theta} = 1.76$ as can be seen in the fourth column of Table 4.1. In the middle panel of Figure 4.2 we see a plot of $\mathcal{E}(V_\theta)$ for different number of points in the grid; the values θ^* are in the fifth column of Table 4.1. The right panel of Figure 4.2 shows the behavior of $\mathcal{E}(V_{\bar{\theta}})$ and $\mathcal{E}(V_{\theta^*})$ decreasing the fill distance h. The error decays as h does.

Table 4.1 shows the quality of our results. The first column presents the fill distance h and the relative number of points is shown in the second column. The third column presents the CPU time (in seconds). The fourth column presents the values $\bar{\theta}$, outputs of Algorithm 5 using the comparison

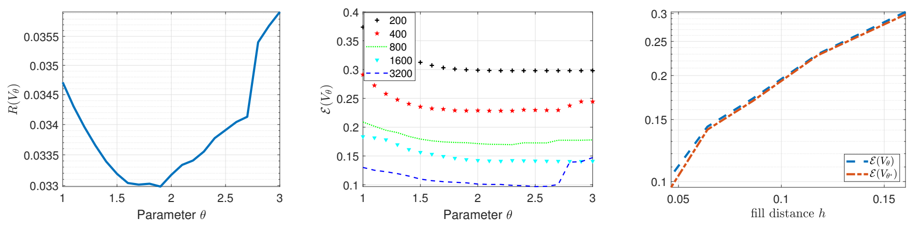

Figure 4.2: Example 1. Left: Average residual for 3200 points. Middle: $\mathcal{E}(V_\theta)$. Right: $\mathcal{E}(V_{\bar{\theta}})$ and $\mathcal{E}(V_{\theta^*})$ variation with h.

method in the minimization procedure. The fifth column presents the values of θ^*. The sixth and seventh columns present the values of $\mathcal{E}(V_{\bar{\theta}})$ and $\mathcal{E}(V_{\theta^*})$.

We see the average decay of the fill distance h when increasing the number of nodes. Accordingly, the average CPU time increases. The parameters $\bar{\theta}$ and θ^* assume values close to each other, with $\theta^* > \bar{\theta}$. The errors $\mathcal{E}(V_{\bar{\theta}})$ and $\mathcal{E}(V_{\theta^*})$ reduce according to h. We can also observe that the rate of convergence with respect to the choice of $\bar{\theta}$ and θ^* is similar.

h	Points	CPU time (s)	$\bar{\theta}$	θ^*	$\mathcal{E}(V_{\bar{\theta}})$	rate	$\mathcal{E}(V_{\theta^*})$	rate
0.1603	200	9.8	1.91	2.16	0.303		0.2981	
0.1177	400	14.6	1.86	2.06	0.230	0.89	0.2284	0.86
0.0861	800	31.8	1.92	2.21	0.172	0.92	0.1697	0.95
0.0641	1600	115.0	2.04	2.42	0.1432	0.62	0.1407	0.63
0.0464	3200	504.0	1.76	2.06	0.1037	0.99	0.0969	1.1

Table 4.1: Example 1. Numerical Results with random unstructured grid.

Figure 4.3 presents in the left panel the exact solution evaluated on the scattered grid with 3200 points and $h = 0.0464$. The middle picture is the solution obtained by value iteration algorithm with Shepard approximation and the last picture is the absolute error between the two solutions. The error has an erratic behavior always below 10^{-1}.

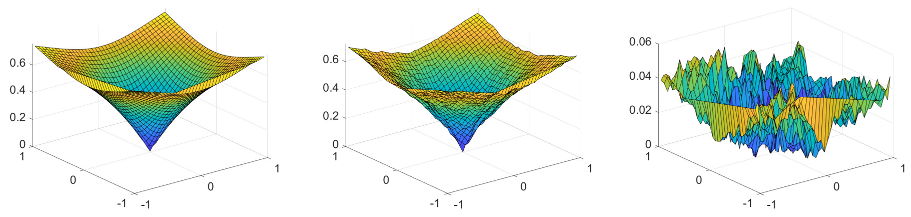

Figure 4.3: Example 4.2. Value Functions generated in a Random Unstructured Grid formed by 3200 points. Left: Exact solution. Center: Solution obtained by VI and Shepard approximation. Right: Absolute error of exact solution and value function obtained by Shepard approximation Value Iteration.

Example 2. Grid driven by the dynamics

In this example we test our novel grid proposed in Subsection 4.1.1. We set $\Delta t = h$ in the HJB equation. To generate the trajectories which will be our grid points, we set $(\overline{\Delta t}, \bar{L}, \bar{M}) = \{(0.1, 4, 16), (0.05, 8, 16), (0.025, 16, 16)\}$ in (4-6). Figure 4.4 shows some examples of meshes generated in this case, with randomly selected initial conditions. These meshes follow the pattern of problem dynamics.

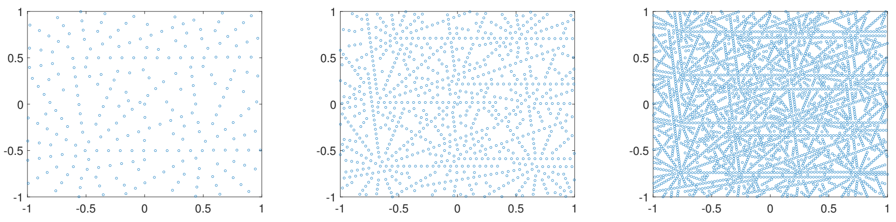

Figure 4.4: Example 2. Meshes generated by the dynamics. Left: 246 points and fill distance 0.1436. Middle: 909 points and fill distance 0.0882. Right: 3457 points and fill distance 0.0439.

We set, again, $\mathcal{P} = [1, 3]$. The behavior of $R(\theta)$ is shown in the left panel of Figure 4.5 when the grid has 3469 points on average. The minimum value is achieved for $\bar{\theta} = 1.7$. The middle panel of Figure 4.5 shows a plot of $\mathcal{E}(V_\theta)$ for a different number of points. The right picture shows the error behavior in $\mathcal{E}(V_{\bar{\theta}})$ and $\mathcal{E}(V_{\theta^*})$ with the reduction of h. It decreases according to Theorem 4.5. We stress that all of these quantities are computed averaging 5 simulations.

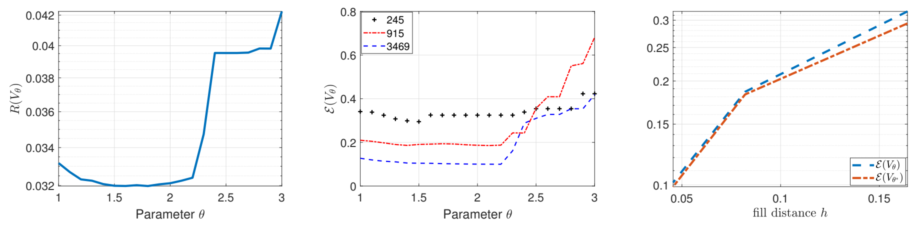

Figure 4.5: Example 2. Left: Average residual to case with 3469 points. Middle: $\mathcal{E}(V_\theta)$. Right: $\mathcal{E}(V_{\bar{\theta}})$ and $\mathcal{E}(V_{\theta^*})$ variation with h.

Table 4.2 summarizes the results of Example 2 as discussed in Example 1. All the considerations are very similar to the previous example but now we have a different grid which will help us to deal with higher dimensional problems as it will be presented in the next sections. We stress that the values of the error indicator in the sixth and eighth column of Table 4.2 are very close to each other. Once again, this is very interesting since we are using an a-posteriori criteria for the computation of the shape parameter. We can also observe that the rate of convergence with respect to the choice of $\bar{\theta}$ and θ^* is similar.

h	Points	CPU time (s)	θ	θ^*	$\mathcal{E}(V_{\bar{\theta}})$	rate	$\mathcal{E}(V_{\theta^*})$	rate
0.1642	245	8.5	1.58	1.82	0.3182		0.2949	
0.0820	915	55.6	1.66	1.76	0.1861	1.29	0.1855	1.49
0.0455	3469	654.0	1.7	1.82	0.1016	0.97	0.0997	0.94

Table 4.2: Example 4.2. Numerical results with a grid driven by the dynamics.

Finally, Table 4.3 presents the results of Algorithm 5 using a gradient descent method with $\varepsilon = 10^{-6}$ in step 11 of Algorithm 5. If we compare the results with Table 4.2 we can see that this method is computationally slower and the accuracy has the same order of the comparison method. Thus, we will only show the performance of our method where the minimization is computed by comparison. We do not provide results with smaller h in Table 4.3 because it is clear that it will be slower than the comparison method.

h	Points	CPU time	$\bar{\theta}$	$\mathcal{E}(V_{\bar{\theta}})$
0.1364	248	9.7	1.62	0.2780
0.0865	924	152.0	1.72	0.1859

Table 4.3: Example 4.2. Results using gradient method.

Example 3. On the feedback reconstruction.
The approximated value functions computed by means of Algorithm 5 with different grids allow us to obtain the optimal trajectories and optimal controls for any initial conditions. In Figure 4.6, we present an example of optimal controls and trajectories computed for $x = (0.7, -0.7)$ using value functions obtained in Example 1 and Example 2. For completeness, we also show the results considering the value function obtained by the traditional value iteration algorithm using linear interpolation and Shepard method on a regular grid as done in [22].

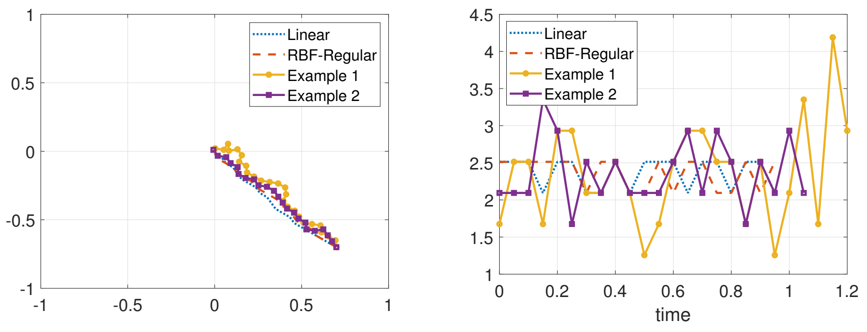

Figure 4.6: Example 3. Optimal trajectories (left) and optimal controls (right) for $x = (0.7, -0.7)$.

All trajectories reach the target $\mathcal{T}$ with different costs. The value of the cost functional, for a given initial condition, is equal when dealing with a structured grid with linear interpolation and Shepard approximation (named RBF-Regular in the table). The value of cost functional with value function from Example 2 is always smaller or equal than the value of the cost functional from Example 1, for both initial conditions.

In Table 4.4 we provide the evaluation of the cost functional for different initial conditions and different methods. It is interesting to see that linear interpolation and Shepard approximation coincide on a equidistributed grid and that the use of Shepard with a grid driven by the dynamics lead to lower cost functional values with respect to the random mesh.

x	Linear	*RBF-Regular*	*Example 1 (RG)*	*Example 2 (GDD)*
$(-0.7,-0.7)$	0.6664	0.6664	0.7315	0.7006
$(0.7,0.7)$	0.6664	0.6664	0.7847	0.6839
$(-0.7,0.7)$	0.6664	0.6664	0.7458	0.6839
$(0.7,-0.7)$	0.6664	0.6664	0.7458	0.7006

Table 4.4: Example 3. Evaluation of the cost functional for different methods and initial conditions x. Example 1 is the random grid (RG) and Example 2 is the grid driven by the dynamics (GDD).

Test 2: Bilinear advection Equation

The second test deals with the control of a two-dimensional advection equation with constant velocity $c \in \mathbb{R}$

$$\begin{cases} y_t(\xi,t) + c\nabla_\xi y(\xi,t) = u(t)y(\xi,t) & (\xi,t) \in D \times [0,T] \\ y(\xi,t) = 0 & \xi \in \partial D \times [0,T] \\ y(\xi,0) = x(\xi) & \xi \in D, \end{cases} \tag{4-31}$$

and x the initial condition. Equation (4-31) can be written as equation (2-1) using finite differences in space (see e.g. [53]) which leads to a system of ODEs:

$$\begin{cases} \dot{y}(t) = Ay(t) + u(t)y(t) & t \in (0,\infty) \\ y(0) = x & x \in \Omega, \end{cases} \tag{4-32}$$

where $A \in \mathbb{R}^{d\times d}$ is the discretization of the gradient term, $y(t) \in \mathbb{R}^d$ and control $u(t) \in U$. Our goal is to steer the solution to 0, minimizing the following cost functional:

$$\mathcal{J}_x(y,u) \equiv \int_0^\infty (\|y(s)\|_2^2 + \gamma|u(s)|^2)e^{-\lambda s}ds \tag{4-33}$$

where $y(t)$ solves (4-32). The parameters in (4-31) are $D = [0,5]^2$, $c = 1$ and $T = 2.5$ is chosen large enough to simulate the infinite horizon. We also set $\gamma = 10^{-5}$ in (4-33).

In this test we select initial conditions from a class of parametrized functions:

$$\mathcal{C} := \left\{ k \sin(\pi\xi_1)\sin(\pi\xi_2)\chi_{[0,1]^2};\ k \in (0,1] \right\}. \tag{4-34}$$

To generate the grid driven by the dynamics we chose $k = \{0.5, 1\}$ in (4-34), 11 equidistributed controls in $U = [-2, 0]$ and $\bar{\Delta}t = 0.1$. Thus, in (4-6), we set: $\bar{\Delta}t = 0.1; \bar{M} = 11; \bar{L} = 2$. Equation (4-32) is then solved for each initial condition and each control with D discretized with $d = 10201$ points, and final time $T = 2.5$. After computing the grid generated by the dynamics, we run Algorithm 5 with $\mathcal{P} = [0.4, 0.7]$ discretized with step size 0.05, fixing $\Delta t = 0.05$ to discretize (4-32) by an implicit Euler method and 21 equidistributed controls. The residual $R(\theta)$ reaches its minimum with $\bar{\theta} = 0.65$ as shown in the bottom-left panel of Figure 4.7. The CPU time to run our algorithm is 583.6 seconds.

To obtain the feedback control and optimal trajectories we have further discretized the set U with 81 points. Thus, we have studied the control problem for different initial conditions selected from the set (4-34) using the value function already stored. In Figure 4.7, we can compare in the top panels the uncontrolled solution, i.e., the transport of initial condition in D considering $u(t) = 0$, with controlled solution for $y(\xi, 0) = 0.75 \sin(\pi\xi_1)\sin(\pi\xi_2)\chi_{[0,1]^2}$. The respective optimal control is shown in the bottom-right panel of Figure 4.7. Note that this initial condition does not belong to the grid.

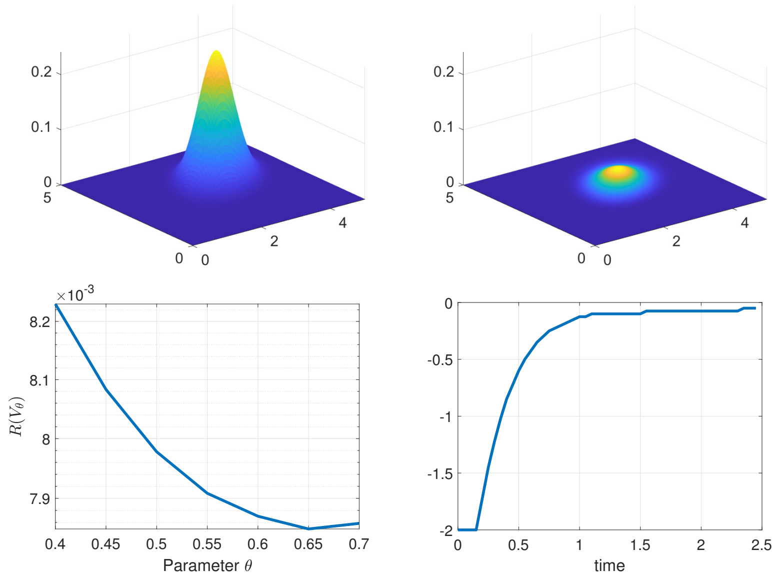

Figure 4.7: Test 2. Initial condition $y(\xi, 0) = 0.75 sin(\pi\xi_1) sin(\pi\xi_2)\chi_{[0,1]^2}$. Top: uncontrolled solution (left) and controlled solution (right). Bottom: residual (left) and optimal control (right).

We have also studied other initial conditions, with $k = \{0.5, 1\}$ in (4-34). The behavior of the solution is similar to Figure 4.7. We show a plot of the cost functionals in Figure 4.8, and we see that the controlled solution has always a lower cost functional than the uncontrolled solutions. We are able to reach the desired configuration with the three initial conditions considered.

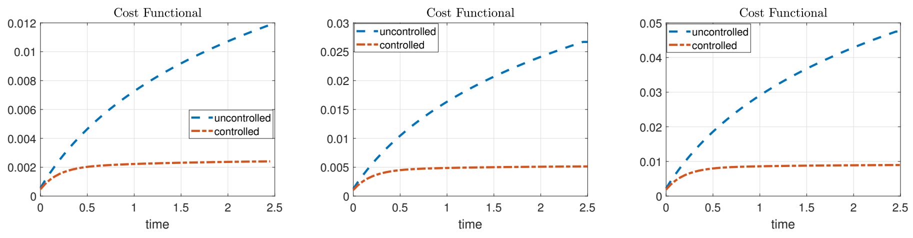

Figure 4.8: Test 2. Cost functional. Left: $y(\xi,0) = 0.5 sin(\pi\xi_1)sin(\pi\xi_2)\chi_{[0,1]^2}$. Center: $y(\xi,0) = 0.75 sin(\pi\xi_1)sin(\pi\xi_2)\chi_{[0,1]^2}$. Right: $y(\xi,0) = sin(\pi\xi_1)sin(\pi\xi_2)\chi_{[0,1]^2}$.

Test 3: Nonlinear Heat Equation

This test deals with the control of a two-dimensional parabolic equation with polynomial nonlinearities:

$$\begin{cases} y_t(\xi,t) = \alpha\Delta y(\xi,t) + \beta(y^2(\xi,t) - y^3(\xi,t)) + u(t)x(\xi) & (\xi,t) \in D \times (0,\infty) \\ \partial_n y(\xi,t) = 0 & \xi \in \partial D \times (0,\infty) \\ y(\xi,0) = x(\xi) & \xi \in D \end{cases} \tag{4-35}$$

with $D = [0,1]^2, \alpha = \frac{1}{100}, \beta = 6$ and x the initial condition. Finite differences in space for (4-35) leads to

$$\begin{cases} \dot{y}(t) = \alpha A y(t) + Bu(t) + \beta\mathbf{F}(y(t)) & t \in (0,\infty) \\ y(0) = x & x \in \Omega \end{cases} \tag{4-36}$$

where $A \in \mathbb{R}^{d\times d}$ is the discretization of the laplacian, $B \in \mathbb{R}^d$ with $B_i = x(\xi_i)$ for $i = 1,\dots,d$ and ξ_i a node of the discretization of D. Here the nonlinear term is $\mathbf{F}(y(t)) = y(t)^2 - y(t)^3$.

We want to minimize again (4-33) as in the previous test and consider the class of initial conditions (4-34). Here, we build an unstructured mesh using as initial condition $k = \{0.5, 1\}$ in (4-34). In (4-6), we set $\bar{\Delta t} = 0.1, \bar{M} = 41, \bar{L} = 2$. The state space $D = [0,1]^2$ was discretized in 31^2 points which is the dimension of the discretized problem (4-36). The control space $U = [-2,0]$ is discretized with 41 points. The time domain is $T = [0,5]$ which was discretized with 51 points. We set $\gamma = 10^{-4}$ in (4-33) and run Algorithm 5 using $\mathcal{P} = [2, 2.4]$ with step size of 0.05 and $\Delta t = 0.075$. The parameter that minimizes $R(\theta)$ is $\bar{\theta} = 2.25$ as shown in Figure 4.9. The time needed to approximate the value function is approximately 20 minutes.

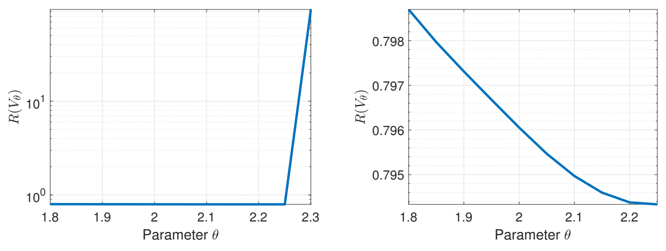

Figure 4.9: Test 3. Residual.

We present the controlled solutions for different initial conditions taken from $\mathcal{C}$ in (4-34). First, we consider the initial condition $y(\xi, 0) = 0.75\sin(\pi\xi_1)\sin(\pi\xi_2)$ and the results are shown in Figure 4.10. As one can see in the left panel the solution reach the equilibrium $y(t) = 1$, whereas the controlled solution goes to 0 as desired. The optimal control is then shown in the bottom panel of Figure 4.10. Note that the initial condition does not belong to the grid where we computed the value function.

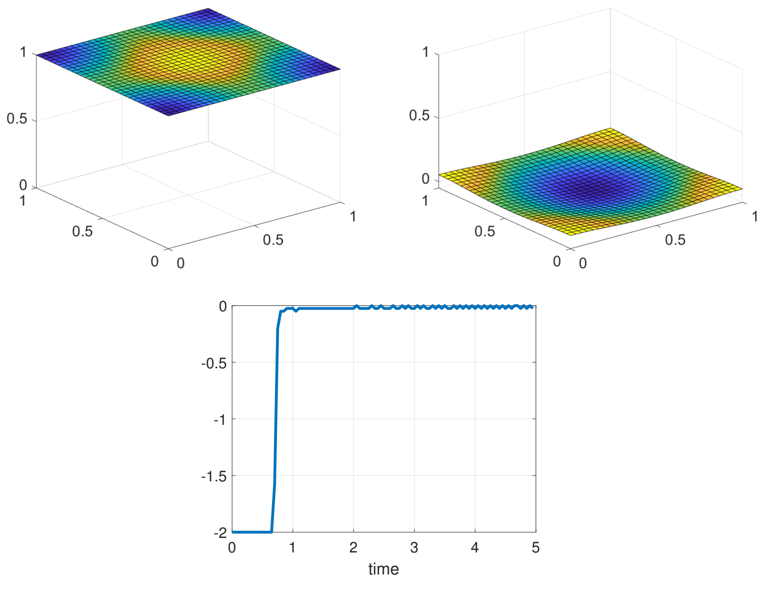

Figure 4.10: Test 3. Initial condition $y(\xi, 0) = 0.75\sin(\pi\xi_1)\sin(\pi\xi_2)$. Left: uncontrolled solution at time $t = 5$. Right: controlled solution at time $t = 5$. Bottom: optimal control.

Figure 4.11 presents the evaluation of the cost functional for the initial conditions considered. As expected, the cost of controlled solutions is always smaller than costs of uncontrolled solutions.

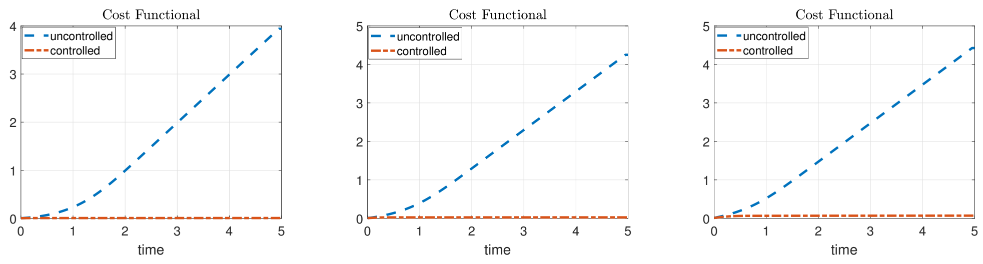

Figure 4.11: Test 3. Cost functional. Left: $y(\xi, 0) = 0.5sin(\pi\xi_1)sin(\pi\xi_2)$. Center: $0.75y(\xi, 0) = sin(\pi\xi_1)sin(\pi\xi_2)$. Right: $y(\xi, 0) = sin(\pi\xi_1)sin(\pi\xi_2)$

Simulation with noise. We now check the robustness of the method adding a noise term at each time instance. We keep using the same value function stored before and we computed the optimal trajectory for a initial condition with a small perturbation. Here, we consider $y(\xi, 0) = 0.75\sin(\pi\xi_1)\sin(\pi\xi_2) + \mathcal{N}(0, 6.25 \times 10^{-4})$ for each $(\xi_1, \xi_2) \in D$, where $\mathcal{N}(0, 6.25 \times 10^{-4})$ is a normally distributed random variable with zero mean and standard deviation 0.025. With these parameters we have a probability of $95, 45\%$ of selecting a number in the range $[-0.05, 0.05]$ at each iteration. At each time iteration a new independent perturbation $\mathcal{N}(0, 6.25 \times 10^{-4})$ has been added to the trajectory. Left picture of Figure 4.12 presents the uncontrolled trajectory and the solution converges (somehow) to $y(t) = 1$ with a perturbation. The right panel shows the controlled solution and how it is close to $y(t) = 0$, also with a perturbation. The left panel in the bottom line of Figure 4.12 presents the optimal control. A comparison of the cost functional for the perturbed problem is show in the bottom right panel of Figure 4.12.

Finally, to further show the effectiveness of our method we consider a non-smooth initial condition that does not belong to (4-34):

$$y(\xi, 0) = \max\{-(2|\xi_1 - 0.5| + 1)(2|\xi_2 - 0.5| + 1) + 2, 0\}$$

as shown in the left panel of Figure 4.13. Here we consider a time interval $T = [0, 8]$ discretized in 161 points.

Figure 4.13 shows the uncontrolled trajectory, which converges to $y(t) = 1$ and the controlled trajectory, which converges to the unstable equilibrium $y(t) = 0$ as desired. Figure 4.14 also presents the optimal control and the evaluation of the cost functional.

4.3 Policy Iteration

In Chapter 2 we have discussed semi-Lagrangian schemes to approximate the value function. In this chapter, the discussion has focused on the use of the

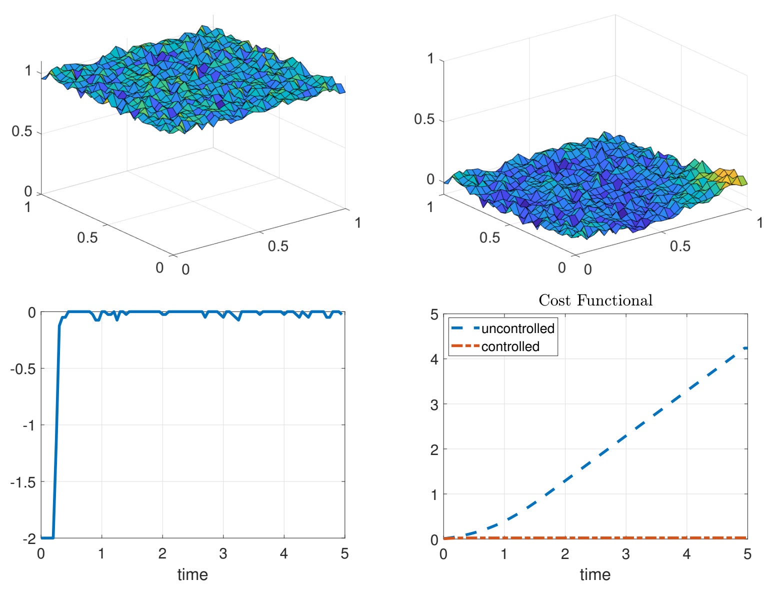

Figure 4.12: Test 3. Initial condition $y(\xi, 0) = 0.75sin(\pi\xi_1)sin(\pi\xi_2) + \mathcal{N}(0, 6.25 \times 10^{-4})$. Upper left: uncontrolled solution. Upper right: controlled solution. Lower left: optimal control. Lower right: Cost functional.

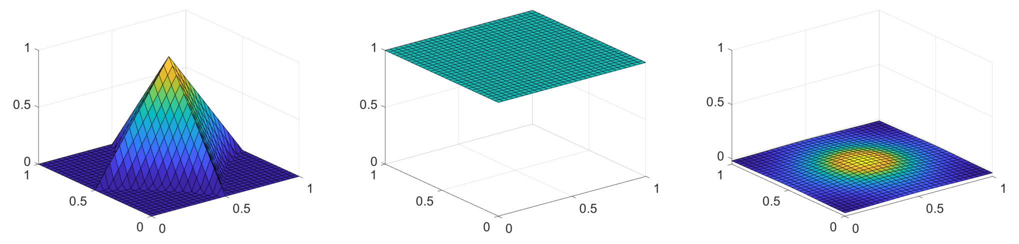

Figure 4.13: Test 3. Left: Initial condition $y(\xi, 0) = \max\{-(2|\xi_1 - 0.5| + 1)(2|\xi_2 - 0.5| + 1) + 2, 0\}$. Middle: uncontrolled solution. Right: controlled solution.

value iteration algorithm coupled with Shepard approximation as fundamental framework. A natural path would be to adapt the same method with a policy iteration scheme.

We have discussed in Chapter 2 the reduced computational time of policy iteration over value iteration due to the lack of a minimization phase and under the suitable choice of the initial guess. There, the discussion considered a regular grid. The use of policy iteration in Algorithm 5, despite being intuitively straightforward, is restricted due to computational time.

We work with the Van der Pol oscillator as a model problem to discuss the restricions on the use of policy iteration. Let $\Omega = [-2, 2]^2$, $U = [-1, 1]$, $\lambda = 1$ and dynamics given by

$$f(x, y, u) = \begin{pmatrix} y \\ (1 - x^2)y - x + u \end{pmatrix}.$$

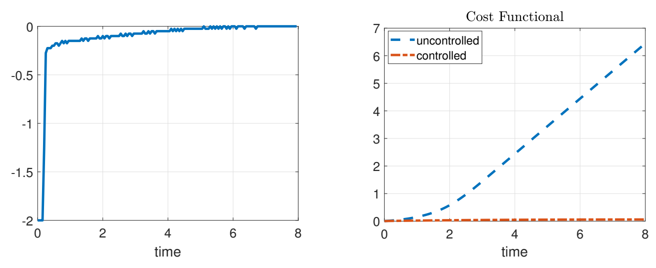

Figure 4.14: Test 3. Left: optimal control and (right) running Cost with initial condition $y(\xi, 0) = \max\{-(2|\xi_1 - 0.5| + 1)(2|\xi_2 - 0.5| + 1) + 2, 0\}$.

The running cost is given by

$$g(x, y, u) = x^2 + y^2$$

and the objective is to minimize the cost functional

$$\int_0^\infty g(x(s), y(s), u(s))e^{-\lambda s}ds. \tag{4-37}$$

We obtain the reference value function by value iteration in a regular grid of 321^2 points, U discretized in 32 equidistributed points, $\Delta t = 0.3\Delta x$ and the boundary value $v(x) = 3.5$. This configuration is the same studied in [21]. This reference solution is presented in Figure 4.15.

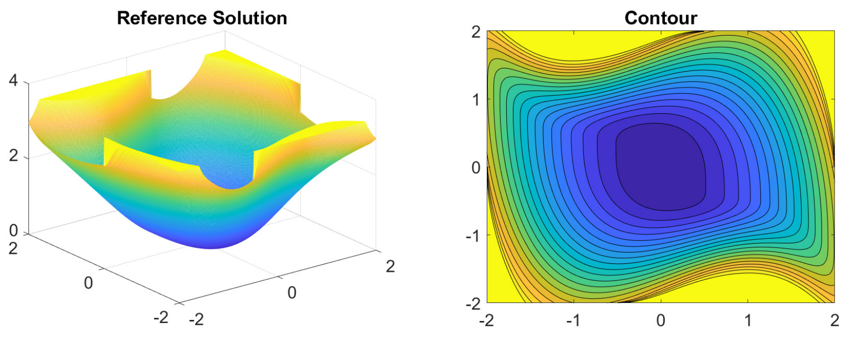

Figure 4.15: Left: Reference value function for Van der Pol oscillator. Right: Contour plot of the reference value function.

We run Algorithm 5 using both value iteration and policy iteration. The domain Ω is populated with random points generated by an uniform distribution such that Ω is densely populated. We use numerical domains of sizes $\{200, 400, 800, 1600\}$ points, $\Delta t = 0.85h$, U discretized in 17 points and a parameter space $\mathcal{P} = [0.1, 2]$. Due to the random nature of the meshes, for each mesh we calculate 10 tests and obtain average values. The average relative errors are calculated using the reference solution as the function that the value and policy iterations aim to achieve. The initial guess V^0 to start the policy iteration is a value function obtained by value iteration in a regular grid formed by 81^2 nodes.

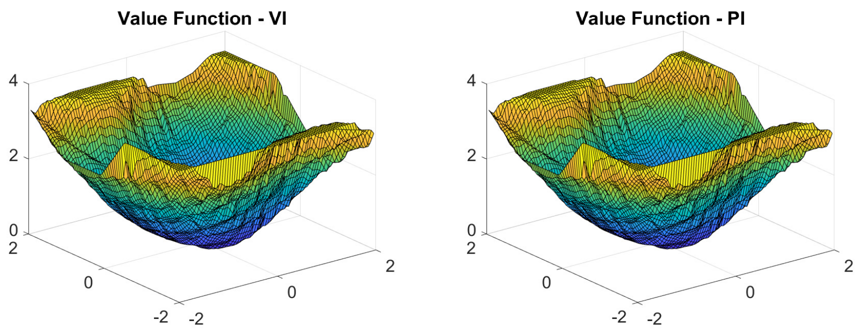

Figure 4.16: Left: value function obtained by value iteration in a mesh formed by 1600 scattered points. Right: value function obtained by policy iteration in a mesh formed by 1600 scattered points.

As we can see in Figure 4.16, the solutions obtained using value iteration and policy iteration in Algorithm 5 are close to the shape of the reference solution and the accuracy of the approximation increases as the decay in the average relative errors suggests. The focus of this example is a comparison between computational time of Algorithm 5 using each value iteration and policy iteration.

h	Points	Error-VI	Error-PI	CPU time - VI (s)	CPU time - PI (s)
0.3295	200	0.3228	0.3228	6.0	5.7
0.2359	400	0.2948	0.2948	9.9	13.0
0.1715	800	0.2776	0.2776	26.3	47.9
0.1252	1600	0.2493	0.2493	139.9	243.5

Table 4.5: Average computational times of value iteration and policy iteration implementations of Algorithm 5

The implementation of Algorithm 5 deals with storage of distance matrices and simultaneous search for minimum among them. Each of these matrices stores distances from all $x_i \in X$ to points with the form $x_j + \Delta t f(x_j, u_k)$, where $x_j \in X$ and $u_k \in U$ is fixed. The entries on those matrices are mapped by the RBF φ, resulting in a matrix A associated to the fixed control u_k: $A_{i,j}(u_k) = \varphi(\|x_i - x_j - \Delta t f(x_j, u_k)\|)$.

The size of these matrices $A(u_k)$ depends on the number of nodes in the unstructured grid X and the quantity of matrices depends on the number of discrete controls, since each matrix is associated to a specific u_k.

When using value iteration, at the n-th iteration the minimum value V^n is obtained comparing entries of different vectors. Each of these vectors is calculated using a pre-stored matrix associated to a discrete control, as described above. When working with policy iteration we have two phases:

policy evaluation and policy improvement steps. In the policy evaluation step each entrie of V^{n+1} is obtained by

$$V_i^{n+1} = \Delta t g(x_i, u_i^n) + (1 - \lambda \Delta t) S[V^n](x_i + \Delta t f(x_i, u_i^n)) \tag{4-38}$$

where each node x_i is associated to a specific control u_i^n evaluated in policy improvement step at previous iteration n. In order to calculate the Shepard approximation $S[V^n](x_i + \Delta t f(x_i, u_i^n))$ in equation (4-38), we need the distances between $x_i + \Delta t f(x_i, u_i^n)$ and all other nodes x_j in X. These values are stored as columns in each matrix $A(u_i^n)$. Thus, to each node, the method demands the search for specific entries between different matrices, which is computationally expensive with the increase of the mesh size or in the number of controls. This explains the CPU time differences when comparing value and policy iteration in Table 4.5.

5
The HJB-RBF approach to control nonlocal PDEs

In this chapter we apply our approach to control nonlocal models focusing on the fractional Laplacian operator. We will introduce our nonlocal model together with its discretization. Then, we describe the control problem we studied and test our approach on three distinct examples. The first test compares the solution of our DP approach with the open-loop method. In this case we also study the numerical convergence towards the continuous problem. In the second test, we control the dynamical system in a specific region of the domain and in the third we study a nonlinear model. We also test the robustness of the method under disturbance of the problem to show the effectiveness of the feedback control.

5.1
Problem setting

Nonlocal, integral models are valid alternatives to classical PDEs to describe systems where small scale effects or interactions affect the global behavior. In particular, nonlocal models are characterized by integral operators that embed length scales in their definitions, allowing to capture long-range space interactions. Furthermore, the integral nature of such operators reduces the regularity requirements on the solutions that are now allowed to feature discontinuous or singular behavior.

In its simplest form, the action of a nonlocal (spatial) operator $\mathcal{L}$, on a scalar function $y : \mathbb{R}^n \to \mathbb{R}$, is defined as

$$\mathcal{L}y(\xi) = \int_{B_\delta(\xi)} I(\xi, z, y)\, dz,$$

where $B_\delta(\xi)$ defines a nonlocal neighborhood of size δ surrounding a point $\xi \in \mathbb{R}^n$, n being the spatial dimension and δ the so-called horizon or *interaction radius*. The latter defines the extent of the nonlocal interactions and embeds the nonlocal operator with a characteristic length scale. The integrand function I is application dependent and plays the role of a constitutive law. Its definition is not straightforward and represents one of the most investigated problems in nonlocal research [56, 57, 58, 59, 60].

Fractional differential operators are almost as old as their integer counterpart [61]; however, their usability has increased during the last decades thanks to progress in computational capabilities and to a better understanding of their descriptive power. The simplest form of a fractional operator is the fractional Laplacian; in its integral form, its action on a scalar function y is defined as (see e.g. [61])

$$(-\Delta)^{\varsigma} y(\xi) = C_{n,\varsigma} \text{ p.v.} \int_{\mathbb{R}^n} \frac{y(\xi) - y(z)}{|\xi - z|^{n+2\varsigma}} dz, \tag{5-1}$$

where $\varsigma \in (0,1)$ is the *fractional order*, $C_{n,\varsigma}$ a normalization constant defined as $C_{n,\varsigma} := \frac{2^{2\varsigma}\varsigma\Gamma(\varsigma+n/2)}{\pi^{n/2}\Gamma(1-\varsigma)}$ and p.v. indicates the principal value. It follows from (5-1) that in fractional modeling the state of a system at a point depends on the value of the state at any other point in the space; in other words, fractional models are nonlocal. Specifically, fractional operators are special instances of more general nonlocal operators [62, 63, 64, 55] of the following form [54]:

$$-\mathcal{L}y(\xi) = \int_{B_\delta(\xi)} (y(\xi)\gamma(\xi,z) - y(z)\gamma(z,\xi))dz. \tag{5-2}$$

Here, interactions are limited to a norm-induced ball $B_\delta(\xi)$ of radius δ. The kernel $\gamma(\xi, z)$ is a modeling choice and determines regularity properties of the solution. Note that for $\delta = \infty$ and for the fractional-type kernel $\gamma(\xi,z) = |\xi - z|^{-n-2\varsigma}$ the nonlocal operator in (5-2) is equivalent to the fractional Laplacian in (5-1). Also, it has been shown in [63] that for that choice of γ solutions corresponding to the nonlocal operator (5-2) converge to the ones corresponding to the fractional operator (5-1) as $\delta \to \infty$ (see [62] for more convergence results and for a detailed classification of these operators and relationships between them).

Given the bounded, open domain $D \subset \mathbb{R}^n$ and a given constant $\delta > 0$, we define the *interaction domain* corresponding to D as

$$D_{\mathcal{I}_\delta} := \{z \in \mathbb{R}^n \text{ such that } z \in B_\delta(\xi) \text{ for some } \xi \in D\}.$$

If the interaction radius is $\delta = \infty$ we have the interaction domain $D_{\mathcal{I}_\delta} = \mathbb{R}^n \backslash \Omega$.

While in classical PDEs models the constraints of the problem (Dirichlet, Neumann or Robin conditions) are applied on the boundary ∂D, in nonlocal models analogous conditions are imposed in the interaction domain $D_{\mathcal{I}_\delta}$. They are called *volume constraints.*

5.2 The control problem and its discretization

For the fractional Laplacian operator defined in (5-1) we let D denote a bounded Lipschitz domain and we define the integral fractional Laplacian on a bounded domain to be the restriction of the full-space operator to functions satisfying a volume constraint on $D_{\mathcal{I}_\delta} = \mathbb{R}^n \setminus D$. Here, for simplicity, we only consider the homogeneous case, i.e. $y = 0$ in $D_{\mathcal{I}_\delta}$.

We focus on the control of a general parabolic problem with time dependent, reaction-diffusion equation. The objective is to find $y : \mathbb{R}^n \times (0,\infty) \to \mathbb{R}$ satisfying

$$\begin{cases} \partial_t y(\xi,t) &= -\alpha(-\Delta)^\varsigma y(\xi,t) + F(y(\xi,t)) + & (\xi,t) \in D \times (0,\infty), \\ & \qquad +b(\xi,t) + u(t)q(\xi) & \\ y(\xi,t) &= 0 & (\xi,t) \in D_{\mathcal{I}_\delta} \times (0,\infty), \\ y(\xi,0) &= x(\xi) & \xi \in D, \end{cases} \tag{5-3}$$

where the control term is $u(t)$, with $u : [0,\infty) \to U \subset \mathbb{R}$ a compact set and $q : D \to \mathbb{R}$. We also consider a non-linear function $F : \mathbb{R} \to \mathbb{R}$, a forcing term $b : D \times [0,\infty) \to \mathbb{R}$ and a constant $\alpha > 0$.
The general formulation of the cost functional that we intend to minimize is

$$\mathcal{J}_x^\infty(y,u) := \int_0^\infty (\|y(\cdot,s) - \bar{y}\|^2_{L^2(D)} + \gamma\|u(s)q(\cdot)\|^2_{L^2(D)})e^{-\lambda s}ds,$$

where x is the initial condition, $\bar{y}$ is the desired state, $\gamma > 0$ and $\lambda > 0$ is the discount factor. Then, the optimal control problem is formulated as

$$\min_{u\in U} \mathcal{J}_x^\infty(y,u) \text{ s.t. } y(u) \text{ satisfying (5-3)},$$

and we write $y = y(u)$ to emphasize the dependence of the solution on the control u.

We obtain the variational formulation of problem (5-3) by multiplying the equation with a test function w with $w|_{\mathbb{R}^n\setminus D} = 0$, $w \in \mathcal{Y}$. In order to define $\mathcal{Y}$, let us present the definition of the fractional Sobolev space H^ς via Fourier Transform $\mathcal{F}$ (see e.g. [65]) by

$$H^\varsigma(\mathbb{R}^n) := \left\{ y \in L^2(\mathbb{R}^n) : \int_{\mathbb{R}^n} (1 + |\xi|^{2\varsigma})|\mathcal{F}y(\xi)|^2 d\xi < \infty \right\}.$$

Considering the Dirichlet volume constraint $y = 0$, we define the space

$$\mathcal{Y} := \{y \in H^\varsigma(\mathbb{R}^n) : y = 0, \quad \forall \xi \in \mathbb{R}^n \setminus D\}$$

equipped with the norm

$$\|y\|_{\mathcal{Y}}^2 = \|y\|_{H^{\varsigma}(\mathbb{R}^n)}^2 = \|y\|_{L^2(D)}^2 + \int_{\mathbb{R}^n}\int_{\mathbb{R}^n} \frac{(y(\xi)-y(z))^2}{|\xi - z|^{n+2\varsigma}} d\xi dz.$$

Hence, the variational formulation of the parabolic problem (5-3) is given by

$$\begin{gathered}\text{Find } y \in \mathbb{V} \text{ such that } \forall w \in \mathcal{Y} \\ \langle \partial_t y, w\rangle = -\alpha a(y,w) + \langle F(y), w\rangle + \langle b, w\rangle + u\langle q, w\rangle, \end{gathered} \tag{5-4}$$

with the space $\mathbb{V}$ defined as

$$\mathbb{V} := \{y \in L^2((0,\infty);\mathcal{Y}) : \partial_t y \in L^2((0,\infty);\mathcal{Y}')\},$$

and $\mathcal{Y}'$ being the dual space of $\mathcal{Y}$. The term $a(y,w)$ is obtained using the fractional laplacian (5-1) resulting in

$$a(y,w) := \frac{C_{n,\varsigma}}{2}\int_{\mathbb{R}^n}\int_{\mathbb{R}^n} (y(\xi)-y(z))(w(\xi)-w(z))|\xi - z|^{-n-2\varsigma} dz d\xi.$$

Having introduced the variational formulation of equation (5-3), we now describe its discretization process.

Let $\mathcal{G} = \{\mathcal{K}\}$ be a conforming partition of $\overline{D}$ into simplices $\mathcal{K}$ with size $h_{\mathcal{K}} = \operatorname{diam}(\mathcal{K})$, and set $h_{\mathcal{G}} = \max_{\mathcal{K}\in\mathcal{G}} h_{\mathcal{K}}$. Given $\mathcal{G}$, we define the finite element space of continuous piecewise polynomials of degree one as

$$\mathcal{Y}(\mathcal{G}) = \left\{ w_{\mathcal{G}} \in C^0(\overline{\Omega}) : w_{\mathcal{G}|\mathcal{K}} \in \mathbb{P}_1(\mathcal{K}), \forall \mathcal{K} \in \mathcal{G},\ w_{\mathcal{G}} = 0 \text{ on } \partial D \right\}, \tag{5-5}$$

where $\mathbb{P}_1(\mathcal{K})$ is the space of linear functions on $\mathcal{K}$. We point that $\mathcal{Y}(\mathcal{G}) \subset \mathcal{Y}$.

Consider $\{\phi_i\}_{i=1}^d \subset \mathcal{Y}(\mathcal{G})$ be the nodal hat functions that span the space. We search for a function $y_{\mathcal{G}} \in \mathcal{Y}(\mathcal{G})$ which can be described by the ansatz $y_{\mathcal{G}}(\xi,t) = \sum_{i=1}^d y_i(t)\phi_i(\xi)$, i.e. for every fixed time t the function $y_{\mathcal{G}}$ is a continuous linear piecewise function with time dependent nodal values $y_i(t)$. Thus, using this ansatz in (5-4) we obtain the terms

$$a(y_{\mathcal{G}}, \phi_j) = \sum_{i=1}^d y_i a(\phi_i,\phi_j), \qquad (\dot{y_{\mathcal{G}}}, \phi_j) = \sum_{i=1}^d \dot{y}_i(\phi_i,\phi_j),$$

where $\dot{y} = \frac{dy}{dt}$. Let us denote by Ω the discretization of D considering d nodes. Using these spatial discretized elements and treating the other terms accordingly (note that the initial condition is written as the ansatz $x(\xi) = \sum_{i=1}^d x_i\phi_i(\xi)$), we have the semi-discrete version of equation (5-3) given by

$$\begin{cases} M\dot{Y}(t) = -\alpha AY(t) + \mathbf{F}(Y(t)) + B(t) + Qu(t), & t > 0, \\ Y(0) = x, & x \in \Omega. \end{cases} \tag{5-6}$$

Here, the matrices $A, M \in \mathbb{R}^{d\times d}$ are constructed with entries

$$A_{ij} = a(\phi_i, \phi_j), \qquad\qquad M_{ij} = (\phi_i, \phi_j).$$

The other elements are d-dimensional vectors with entries given by $Y_i(t) = y_i(t)$, $\dot{Y}_i(t) = \dot{y}_i(t)$, $\mathbf{F}_i(Y) = \langle F(y_{\mathcal{G}}), \phi_i\rangle$, $B_i(t) = \langle b, \phi_i\rangle$ and $Q_i(t) = \langle q, \phi_i\rangle$.

Relabelling $\dot{y}(t) = \dot{Y}(t)$, $y(t) = Y(t)$, multiplying the right-hand side of (5-6) by the inverse of matrix M, and the initial condition $y(0) = x$ being an element of $\mathcal{Y}(\mathcal{G})$, leads us to the system

$$\begin{cases} \dot{y}(t) = M^{-1}(-\alpha Ay(t) + \mathbf{F}(y(t)) + B(t) + Qu(t)), & t > 0, \\ y(0) = x, & x \in \Omega. \end{cases} \tag{5-7}$$

Equation (5-7) gives us a dynamical system in the format of equation (2-1) with $f(y(t), u(t)) = M^{-1}(-\alpha Ay(t) + \mathbf{F}(y(t)) + B(t) + Qu(t))$. It is similar to what we have introduced in Chapter 2 and used in Chapter 4. Thus, we are able to apply the DP approach.

We remark that in numerical approximations, the matrix M^{-1} is never computed.

5.3 Numerical Tests

In order to illustrate the DP approach to control fractional diffusion problems we present three tests. The first is a parabolic problem where the analytical solution is known. In this case, we show a comparison between the solutions obtained by open-loop method and by the DP approach. The same equation is also used to show the effectiveness of the feedback control when dealing with disturbances of the system. We also study the numerical convergence towards the continuous problem.

The second and third test are performed using exclusively the DP approach. The second test is a modification of the previous problem where we control the equation on a target subset $\mathcal{T} \subset D$. The third test is a fractional heat equation with a nonlinear term and a diffusive coefficient α. Our tests are one-dimensional ($n = 1$) and consider $\varsigma = 0.75$ in Section 5.3.

In our tests we use Wendland RBFs constructed according to the dimension d of each discrete problem. We remember that the problem dimension is the number of nodes in the spatial discretization. The functions are of class

$\mathcal{C}^4(\mathbb{R}^d)$ and are obtained using the general formula proposed in [66]

$$\varphi^\sigma(r) = \max\{0, (1-\sigma r)^{\ell+2}((\ell^2+4\ell+3)\sigma^2 r^2 + (3\ell+6)\sigma r + 3)\}, \quad r := \|x\|_2,$$

where $\ell = \lfloor \frac{d}{2} \rfloor + 3$.

5.3.1 Test 1: Linear example with exact solution

Let us present an artificial example with optimal solution that will be used for comparison with the results obtained using the DP approach. We refer to e.g. [71] and references therein for optimal control of parabolic fractional equations.

Let $\tilde{y}$ and $\tilde{b}$ be the exact solution and corresponding right-hand side of the fractional Poisson problem

$$\begin{cases} (-\Delta)^{\varsigma} y(\xi) &= \tilde{b}(\xi) \quad \xi \in D, \\ y(\xi) &= 0 \qquad \xi \in D_{\mathcal{I}_\delta}, \end{cases}$$

on D, with $\|\tilde{y}\|_{L^2(D)} = 1$, and let $\phi, \psi : (0,T) \to \mathbb{R}$ such that $\phi(0) = 1$, $\psi(T) = 0$. Let the space $U := [a,b]$ and take the control to be of the form $u(\xi,t) \in \mathcal{U}_{ad}$ where

$$\mathcal{U}_{ad} := \{u \in L^2(0,T;D) : a\tilde{y}(\xi) \le u(\xi,t) \le b\tilde{y}(\xi),\ (\xi,t) \in D \times (0,T)\}$$

and $u(\xi,t) = c(t)\tilde{y}(\xi)$ with $c : (0,T) \to U$. Consider the following functions that will be used in the construction of the cost functional and the PDE

$$\begin{aligned} y_d(\xi,t) &:= \phi(t)\tilde{y}(\xi) - \gamma\psi'(t)\tilde{y}(\xi) + \gamma\psi(t)\tilde{b}(\xi) + \lambda\gamma\psi(t)\tilde{y}(\xi), \\ u_d(\xi,t) &:= \operatorname{proj}_U(\psi(t))\tilde{y}(\xi), \\ b(\xi,t) &:= \phi'(t)\tilde{y}(\xi) + \phi(t)\tilde{b}(\xi) - u_d(\xi,t). \end{aligned}$$

We aim to minimize the cost functional

$$\begin{aligned} \mathcal{J}_x^T(y,u) &:= \frac{1}{2}\int_0^T (\|y(\cdot,s) - y_d(\cdot,s)\|^2_{L^2(D)} + \gamma\|u(\cdot,s)\|^2_{L^2(D)})e^{-\lambda s} ds \\ &= \frac{1}{2}\|y - y_d\|^2_{L^2_\nu(0,T;D)} + \frac{\gamma}{2}\|u\|^2_{L^2_\nu(0,T;D)}, \end{aligned}$$

where we have set $\nu(t) := e^{-\lambda t}$, subject to

$$\begin{cases} \partial_t y(\xi,t) + (-\Delta)^{\varsigma} y(\xi,t) &= b(\xi,t) + u(\xi,t) \quad (\xi,t) \in D \times (0,T), \\ y(\xi,t) &= 0 \qquad\qquad\qquad (\xi,t) \in D_{\mathcal{I}_\delta} \times (0,T), \\ y(\xi,0) &= x(\xi) \qquad\qquad\quad \xi \in D. \end{cases} \tag{5-8}$$

For fixed initial condition x, we can write the first order optimality conditions. Let $j(u) := \mathcal{J}_x^T(\mathcal{S}u, u)$, where $\mathcal{S}$ is the solution operator of the above PDE (5-8). The operator $\mathcal{S}$ is also called the fractional control-to-state operator

$$\mathcal{S} : L^2(0,T;D) \to \mathbb{V}$$

and $\mathcal{S}u = y(u)$, where $y(u)$ solves (5-8).

Then, the optimal control must satisfy the variational inequality (see Lemma 2.21 in [72])

$$\overline{u} = \operatorname{argmin} j(u) \Leftrightarrow (j'(\overline{u}), u - \overline{u}) \geq 0 \quad \forall u \in \mathcal{U}_{ad}.$$

The inequality can be written in the equivalent form

$$(\mathcal{S}^*(\mathcal{S}\overline{u} - y_d) + \gamma\overline{u}, u - \overline{u})_{L^2_\nu((0,T);D)} \geq 0,$$

where $\mathcal{S}^*$ is the adjoint solution operator. Hence, $\overline{p} := \mathcal{S}^*(\mathcal{S}\overline{u} - y_d)$ is the solution to

$$\begin{cases} -\partial_t\overline{p}(\xi,t) + \lambda\overline{p} + (-\Delta)^s\overline{p}(\xi,t) &= \overline{y}(\xi,t) - y_d(\xi,t) \quad (\xi,t) \in D \times (0,T), \\ \overline{p}(\xi,t) &= 0 \qquad\qquad\qquad\quad (\xi,t) \in D_{\mathcal{I}_\delta} \times (0,T), \\ \overline{p}(\xi,T) &= 0 \qquad\qquad\qquad\quad \text{for all } \xi \in D. \end{cases}$$

with $\overline{y} = S\overline{u}$.

Then, the following inequality

$$0 \leq (\overline{p} + \gamma\overline{u}, u - \overline{u})_{L^2_\nu(0,T;D)} = ((\overline{p}, \tilde{y})_{L^2(D)}\tilde{y} + \gamma\overline{u}, u - \overline{u})_{L^2_\nu(0,T;D)}$$

implies that $\overline{u} = \operatorname{proj}_U\left(-\frac{1}{\gamma}(\overline{p}, \tilde{y})_{L^2(D)}\right)\tilde{y}$.

If we set

$$y^*(\xi,t) := \phi(t)\tilde{y}(\xi), \quad p^*(\xi,t) := -\gamma\psi(t)\tilde{y}(\xi), \quad u^*(\xi,t) := u_d(\xi,t),$$

we obtain

$$\begin{aligned} y^*(\xi,0) &= \tilde{y}(\xi), \\ \partial_t y^*(\xi,t) + (-\Delta)^s y^*(\xi,t) &= \phi'(t)\tilde{y}(\xi) + \phi(t)\tilde{b}(\xi) \\ &= b(\xi,t) + u^*(\xi,t), \end{aligned}$$

where we have used that $\tilde{y}$ is the solution of Poisson equation. Hence, y^* is the state corresponding to the control u^* and the initial condition $x = \tilde{y}$.

Moreover,

$$\begin{aligned} p^*(\xi, T) &= 0, \\ -\partial_t p^*(\xi,t) + \lambda p^*(\xi,t) + (-\Delta)^s p^*(\xi,t) &= \gamma\psi'(t)\tilde{y}(\xi) - \lambda\gamma\psi(t)\tilde{y}(\xi) - \gamma\psi(t)\tilde{b}(\xi) \\ &= y^*(\xi,t) - y_d(\xi,t). \end{aligned}$$

and hence p^* solves the adjoint equation with right-hand side $y^* - y_d$.

Finally, $\operatorname{proj}_U \left(-\frac{1}{\gamma}(p^*, \tilde{y})_{L^2}\right)\tilde{y}(\xi) = \operatorname{proj}_U\left(\psi(t)\right)\tilde{y}(\xi) = u^*(\xi,t)$.

Therefore, for the initial condition $x = \tilde{y}$, the optimal control is u^*, and the optimal state is y^*.

Let us now link the problem to the infinite horizon framework. In order to do it, we make $T \to \infty$. In addition, let us choose $T_0 > 0$ and ψ such that $\psi(t) = 0$ for $t \geq T_0$. Moreover, choose U such that $0 \in U$. For $T > T_0$, the previous construction gives the solution of the optimal control problem on $(0,T)$. On the other hand, we have

$$\begin{aligned} \mathcal{J}_x^\infty(y^*, u^*) &= \mathcal{J}_x^T(y^*, u^*) + \frac{1}{2}\|y^* - y_d\|^2_{L^2_\nu(T,\infty;D)} + \frac{\gamma}{2}\|u^*\|^2_{L^2_\nu(T,\infty;D)} \\ &= \mathcal{J}_x^T(y^*, u^*) + \frac{\gamma^2}{2}\Big[\|\psi'\|^2_{L^2_\nu(T,\infty)} + \|\psi\|^2_{L^2_\nu(T,\infty)}\|\tilde{b}\|^2_{L^2(D)} - \lambda(\psi',\psi)_{L^2_\nu(T,\infty)} \\ &- 2(\psi',\psi)_{L^2_\nu(T,\infty)}(\tilde{y},\tilde{b})_{L^2(D)} + 2\frac{\lambda}{\gamma}\|\psi\|^2_{L^2_\nu(T,\infty)}(\tilde{y},\tilde{b})_{L^2(D)} + \lambda^2\|\psi\|_{L^2_\nu(T,\infty)}\Big] \\ &\quad + \frac{\gamma}{2}\|\operatorname{proj}_U(\psi)\|^2_{L^2_\nu(T,\infty)} \\ &= \mathcal{J}_x^T(y^*, u^*), \end{aligned}$$

where we have used that ψ and ψ' are zero on (T,∞). We also need to consider $\gamma > 0$. Therefore,

$$\min \mathcal{J}_x^\infty(y,u) \leq \min \mathcal{J}_x^T(y,u),$$

but the inverse inequality also holds, since $\mathcal{J}_x^T(y,u) \leq \mathcal{J}_x^\infty(y,u)$. Hence, the pair (y^*, u^*) is also optimal on $(0,\infty)$.

Let us set the following quantities

$$\begin{aligned} \gamma > 0, \qquad \lambda \geq 0, \qquad U = [a,b] \subset \mathbb{R}, \\ a \leq 0 \leq b, \qquad 0 < T_0 \leq T, \end{aligned}$$

where T is the final time used for the open loop approach. Moreover, let us choose the parameters of equation (5-8) as

$$\begin{aligned} D &= (-1,1), \\ \tilde{b}(\xi) &= 2^{2\varsigma}\Gamma(1+\varsigma)\frac{\Gamma(\varsigma+1/2)}{\Gamma(1/2)}\sqrt{\frac{\Gamma(2\varsigma+3/2)}{\Gamma(2\varsigma+1)\Gamma(1/2)}}, \\ \tilde{y}(\xi) &= \sqrt{\frac{\Gamma(2\varsigma+3/2)}{\Gamma(2\varsigma+1)\Gamma(1/2)}}\left(1-\xi^2\right)_+^{\varsigma}, \\ \phi(t) &= \cos(t), \\ \psi(t) &= (T_0-t)^2\chi_{\{t\leq T_0\}}, \\ u_d(\xi,t) &= \mathrm{proj}_U(\psi(t))\tilde{y}(\xi), \\ y_d(\xi,t) &= \phi(t)\tilde{y}(\xi) - \gamma\psi'(t)\tilde{y}(\xi) + \gamma\psi(t)\tilde{b}(\xi) + \lambda\gamma\psi(t)\tilde{y}(\xi), \\ b(\xi,t) &= \phi'(t)\tilde{y}(\xi) + \phi(t)\tilde{b}(\xi) - u_d(\xi,t). \end{aligned} \tag{5-9}$$

With $u(\cdot,t) = c(t)\tilde{y}(\cdot)$, we minimize

$$\mathcal{J}_x^\infty(y,u) = \frac{1}{2}\int_0^\infty (\|y(\cdot,s) - y_d(\cdot,s)\|^2_{L^2(D)} + \gamma\|u(\cdot,s)\|^2_{L^2(D)})e^{-\lambda s}ds \tag{5-10}$$

subject to

$$\begin{cases} \partial_t y(\xi,t) + (-\Delta)^\varsigma y(\xi,t) &= b(\xi,t) + u(\xi,t) & (\xi,t) \in D\times(0,\infty), \\ y(\xi,t) &= 0 & (\xi,t) \in D_{\mathcal{I}_\delta}\times(0,\infty), \\ y(\xi,0) &= x(\xi) & \xi \in D. \end{cases} \tag{5-11}$$

The solution for $x = \tilde{y}$ is given by

$$y^*(\xi,t) = \phi(t)\tilde{y}(\xi), \qquad u^*(\xi,t) = u_d(\xi,t). \tag{5-12}$$

Finally, to fit the cost functional into the setting of the DPP we make a change of variable

$$z(\xi,t) = y(\xi,t) - y_d(\xi,t).$$

Denoting the initial condition by $z_o = z(\xi,0)$, our problem will be able to minimize the cost functional

$$\mathcal{J}_{z_o}^\infty(z,u) := \frac{1}{2}\int_0^\infty (\|z(\cdot,s)\|^2_{L^2(D)} + \gamma\|u(\cdot,s)\|^2_{L^2(D)})e^{-\lambda s}ds \tag{5-13}$$

subject to

$$\begin{cases} \partial_t z(\xi,t) + (-\Delta)^{\varsigma} z(\xi,t) &= b(\xi,t) + u(\xi,t) - \\ & \quad -\partial_t y_d(\xi,t) - (-\Delta)^{\varsigma} y_d(\xi,t) & (\xi,t) \in D \times (0,\infty), \\ z(\xi,t) &= 0 & (\xi,t) \in D_{\mathcal{I}_\delta} \times (0,\infty), \\ z(\xi,0) &= x(\xi) - y_d(\xi,0) & \xi \in D. \end{cases} \tag{5-14}$$

Equation (5-14) is equivalent to (5-11). The great benefit is that the cost functional has only the norm of z. With this change of variables we can solve the problem using the HJB approach and then recover the solution y.

The finite element spatial discretization of equation (5-14) leads us to the system formed by ODEs

$$\begin{cases} M\dot{z}(t) &= -Az(t) + Qc(t) + B(t) - M\dot{y_d}(t) - Ay_d(t) & t \in (0,\infty) \\ z(0) &= x - y_d(0) & x, y_d \in \Omega, \end{cases} \tag{5-15}$$

with $A, M \in \mathbb{R}^{d\times d}$ and d-dimensional vectors $\dot{z}, z, Q$ and B as described in (5-6). Here, y_d and $\dot{y_d}$ are also d-dimensional vectors with construction analogue to the other terms. The vector x is equal to $\tilde{y}$ where $\tilde{y}_i = \tilde{y}(\xi_i)$, $i = 1, \dots, d$ and ξ_i is a node in the discretization of D.

Our goal is to minimize the cost functional

$$\mathcal{J}_x(z,u) = \frac{1}{2}\int_0^\infty (\|z(s)\|^2_{L^2(D)} + \gamma\|c(s)x\|^2_{L^2(D)})e^{-\lambda s} ds \tag{5-16}$$

where $z(t)$ solves (5-15).

In this example, we study the problem for $d = 63$ and $d = 127$. We compare the solution obtained by the DP approach (later denoted with y_{HJB}) with the exact solution y^* and with the solution obtained considering the exact optimal control plugged into the discrete system (5-15) (later denoted with y_*).

To approximate the value function and obtain the control in feedback form, using our Algorithm 5, we consider $\gamma = 0.01, \lambda = 0.5$ and $T_0 = 3$. To generate the scattered mesh the control space $U = [0,1]$ is discretized with 7 equidistributed controls and the time step used is $\overline{\Delta t} = 0.0125$. The problem (5-15) is solved up to $T = 4$ and the discrete trajectories are collected. The grid is formed by a total of 2248 nodes for both $d = 63$ and $d = 127$. The separation distances are 0.0248 and 0.0350, respectively.

We use the grid generated by the dynamics to run our algorithm using the parameter space $\mathcal{P} = [0.1, 0.3]$ discretized with step size 0.02. The residual is minimized with the parameter equal to 0.2. We, then, consider U discretized in 21 points and $\Delta t = 0.01$. We use these data and the grid X to approximate the value function. To obtain the feedback control and optimal trajectories we discretize the control set U in 1681 points and solve (5-15) up to the final time $T = 4$. We revert the change of variables used to obtain equation (5-14), then we can compare the solutions obtained via DP approach, denoted by $y_{HJB}(\xi, t)$, with the exact solution $y^*(\xi, t)$ (as described in equation (5-12)) and with the solution obtained using the optimal control $u^*(\xi, t)$ inserted on equation (5-15), denoted by $y_*(\xi, t)$.

Figure 5.1 presents the solutions for $d = 127$ and $\Delta t = 0.0125$. In the left panel we can see the uncontrolled solution of system (5-15), whereas in the middle panel we show the open-loop solution. In the right panel we show the controlled solution via the HJB equation. We note the difference between the shape of the uncontrolled solution to the other solutions. This difference is clear if we look at Figure 5.2 the plots of the absolute difference between the uncontrolled solution and the HJB solution and the uncontrolled solution to the open-loop solution. Also, the similarity between the shape of the solutions obtained by open-loop control and feedback control becomes evident when we look at their absolute difference in the right panel of Figure 5.2.

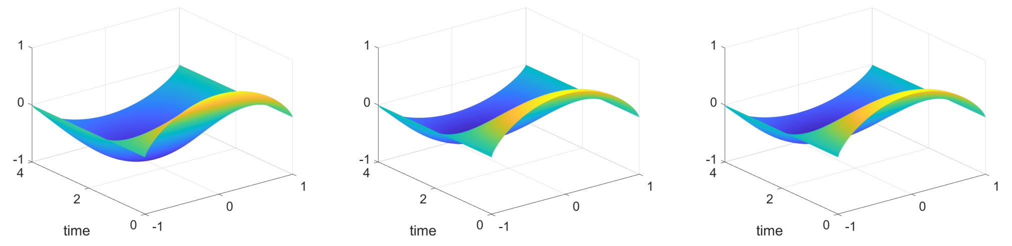

Figure 5.1: Test 1. Left: Uncontrolled solution. Middle: Open-loop solution. Right: HJB solution. Solutions in the case $\Delta t = 0.0125$ and $d = 127$.

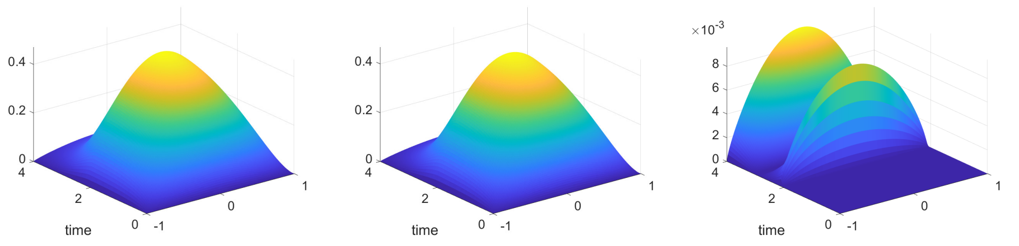

Figure 5.2: Test 1. Left: Absolute difference between the HJB solution and the uncontrolled solution. Center: Absolute difference between the open-loop solution and the uncontrolled solution. Right: Absolute difference between the HJB and open-loop solutions.

In the left panel of Figure 5.3, we compare the control obtained by the HJB approach with respect to the analytical control u^*. One can observe that the controls are very close up to T_0. Then, there is a slight difference which does not really affect the cost functional. Indeed, in the right panel of Figure 5.3 we show the evaluation of the cost functional for the uncontrolled solution, the analytical and the HJB driven approach. One can see that the cost functional of the two controlled solutions matches perfectly. This already highlights the quality of our approximation.

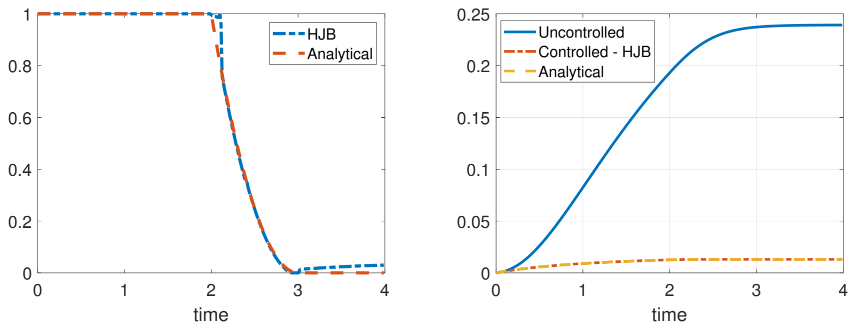

Figure 5.3: Test 1. Left: optimal control, Right: Cost functional (5-10) calculated for uncontrolled, analytical and controlled by HJB approach solutions.

To further validate our approach, we study the error of our approximation towards the control of the PDE. Thus, we have increased the dimension of the problem d (i.e. reduce the error in space of the discretized problem) and use different temporal step sizes Δt. Our convergence analysis is performed with the norm $\|\cdot\|_{L^2_\nu(0,T;D)}$, which we write as $\|\cdot\|_{L^2_\nu}$ to ease of notation. In Table 5.1 we show the values of $\|y_{HJB} - y^*\|_{L^2_\nu}$ in the second column, $\|y_{HJB} - y_*\|_{L^2_\nu}$ in the fourth column and $\|y^* - y_*\|_{L^2_\nu}$ in the sixth. Here we considered distinct values of Δt to solve the discrete problem. We note that these values decay with the reduction in Δt, as expected, since a more refined time discretization implies better approximations.

Δt	$\|y_{HJB} - y^*\|_{L^2_\nu}$	rate	$\|y_{HJB} - y_*\|_{L^2_\nu}$	rate	$\|y^* - y_*\|_{L^2_\nu}$	rate
0.05	0.0450		0.0351		0.0470	
0.025	0.0340	0.40	0.0225	0.64	0.0233	1
0.0125	0.0260	0.38	0.0197	0.2	0.0115	1

Table 5.1: Test 1. Data of simulation with dimension = 63. $h = 0.0248$. Number of nodes = 2248.

Table 5.2 presents results for $d = 127$. The decay in the norm of differences also happens with a reduction on time step. The comparison between values in Table 5.2 with values in Table 5.1 suggests that the refined spatial discretization implies a more accurate approximation, since the values in the second and fourth columns are smaller than their counterparty in Table 5.1. These values in Table 5.2 are almost the half of the respective values in Table 5.1. The entries in the sixth column of Table 5.1 are very similar to those in the same column of Table 5.2, indicating that the time discretization error has a greater influence than the spatial error.

Δt	$\|y_{HJB} - y^*\|_{L^2_\nu}$	rate	$\|y_{HJB} - y_*\|_{L^2_\nu}$	rate	$\|y^* - y_*\|_{L^2_\nu}$	rate
0.05	0.0423		0.0220		0.0473	
0.025	0.0281	0.59	0.0173	0.34	0.0236	1
0.0125	0.0149	0.9	0.0072	1.2	0.0117	1

Table 5.2: Test 1. Data of simulation with dimension = 127. $h = 0.0350$. Number of nodes = 2248.

One of the benefits of computing the control in closed form is the ability to react under disturbances of the system. In this example, we disturb the system at each time instance adding a noise vector with each component following an uniform distribution in the interval $[0, 0.02]$.

Simulation with noise Keeping the same value function obtained in the previous test (in the 127 dimensional case), we compute the feedback control with $\Delta t = 0.025$ and U discretized with 1681 nodes. We compare the solutions where we plug the open-loop control into the system and the solution using the HJB approach. The open-loop control is computed without taking into account the disturbance. We can see, in the left panel of Figure 5.4, the evaluation of the cost functional for both methods. It is clear that the feedback control leads to a lower cost function than the open-loop. For completeness, we show the solution in the middle and right panel of Figure 5.4. In the third panel of Figure 5.4 we can see that the solution controlled using the HJB approach under the presence of perturbation closely follows the shape of the HJB controlled solution without perturbation, as shown in Figure 5.1.

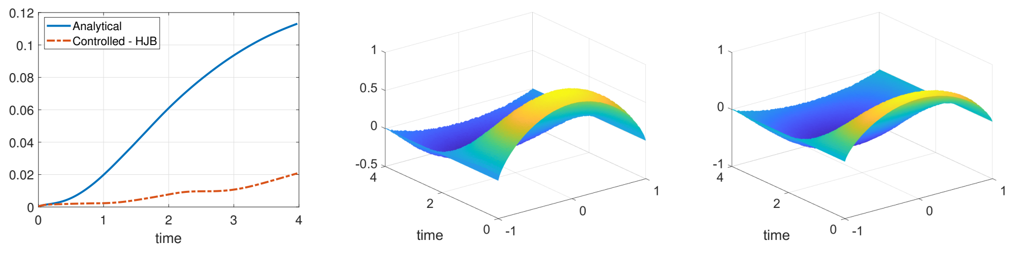

Figure 5.4: Test 1: Left: Cost functionals with noise term. Center: Open Loop solution with noise. Right: HJB approach solution with noise. $T = 4$, 1681 controls; $\Delta t = 0.025$

This test shows the effectiveness of our method under disturbances and further motivate this approach with respect to open-loop controls.

5.3.2
Test 2: Linear example with desired state restricted to a target

In this example we minimize the cost functional

$$\mathcal{J}_x^{\infty}(y,u) := \frac{1}{2}\int_0^{\infty} (\|y(\cdot,s)\|^2_{L^2(\mathcal{T})} + \gamma\|u(\cdot,s)\|^2_{L^2(D)})e^{-\lambda s}ds \tag{5-17}$$

that is similar to the cost functional (5-13), but the norm of y is calculated with the restriction to the target $\mathcal{T} = [-1/2, 1/2]$ and $\mathcal{T} \subset D$ with $D = (-1,1)$. It is minimized subject to the following dynamics

$$\begin{cases} \partial_t y(\xi,t) + (-\Delta)^{\varsigma} y(\xi,t) &= b(\xi,t) + u(\xi,t) \quad \text{for all } \xi \in D \times (0,T), \\ y(\xi,t) &= 0 \quad \text{for all } \xi \in D_{\mathcal{I}_\delta} \times (0,T), \\ y(\xi,0) &= x(\xi) \quad \text{for all } \xi \in D, \end{cases} \tag{5-18}$$

where

$$b(\xi,t) = (1-\cos(t))\chi_{[-1,-3/4]}(\xi) \tag{5-19}$$

and

$$u(\xi,t) = u_1(t)\chi_{[-3/4,-1/2]}(\xi) + u_2(t)\chi_{[1/2,3/4]}(\xi), \tag{5-20}$$

with $u_i(t) \in U = [-0.5, 0]$, $i \in \{1,2\}$, i.e., the control is formed by two independent terms, each of them acting on a specific region of the domain.

Here, the semi-discrete system in equation (5-18) is

$$\begin{cases} M\dot{y}(t) &= -Ay(t) + \Big(u_1(t)\chi_{[-3/4,-1/2]}(\xi) + u_2(t)\chi_{[1/2,3/4]}(\xi)\Big) \quad t \in (0,\infty) \\ & \qquad +B(t) \\ y(0) &= 0 \qquad 0 \in \Omega \end{cases} \tag{5-21}$$

where $\xi \in \mathbb{R}^d$ with ξ_i a node in the discretization of D, $i = 1,\dots,d$. The cost functional to be minimized is

$$\mathcal{J}_x(y,u) = \frac{1}{2}\int_0^{\infty} (\|y(s)\|^2_{L^2(\mathcal{T})} + \gamma\|u_1(s)\chi_{[-3/4,-1/2]} + u_2(s)\chi_{[1/2,3/4]}\|^2_{L^2(D)})e^{-\lambda s}ds \tag{5-22}$$

with $y(t)$ solution of (5-21) and $u(\xi,s)$ defined as (5-20).

We set the parameters of the equation as $\gamma = 10^{-6}$ and $\lambda = 0.5$. To build the scattered mesh we use $\overline{\Delta t} = 0.025$ and control pairs $(u_1,u_2) \in U \times U$, with U discretized in 5 equidistributed controls, resulting in a total of 25 control pairs. We solve equation (5-21) up to time $T = 6$ and collect a total of 6026 nodes to form our 63 dimensional grid. The separation distance is 0.0180.

If we consider the parameter $\theta = 0.01$, a time step $\Delta t = 0.01$ and use the same discrete control set of the mesh, we compute the value function using the unstructured grid. In order to execute the feedback reconstruction we used a $\Delta t = 0.025$ and a total of 1681 control points.

In Figure 5.5, we show the uncontrolled solution in the left panel and the controlled solution in the middle, both with final time $T = 10$. The difference between the solutions becomes clear in the evaluation of the cost functional in the right panel of Figure 5.1. The cost of the controlled solution is below the cost functional of the uncontrolled solution most of the time.

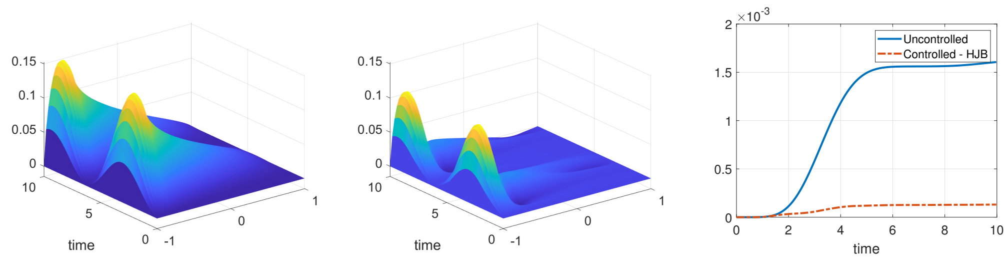

Figure 5.5: Test 2: Left: uncontrolled solution. Middle: controlled solution. Right: Cost Functionals.

In Figure 5.6, we present the two control variables u_1 and u_2 obtained with our approach in the left panel. The control u_1 acts in an interval that is very close to the region where the force term is positive. Then, it is clear that u_1 is more active than u_2 since it is closer to the interval where $b(\xi, t)$ acts. Finally, a further zoom of the uncontrolled and controlled solution in the target region in the middle and right panels, respectively.

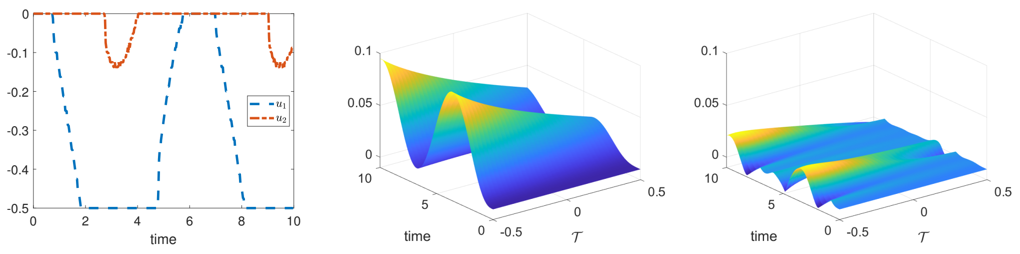

Figure 5.6: Test 2: Left: Controls u_1 and u_2. Center: uncontrolled solution in a plot restricted to target $\mathcal{T}$. Right: controlled solution in a plot restricted to target $\mathcal{T}$.

5.3.3 Test 3: A nonlinear example

In this test we consider the following nonlinear state equation

$$\left\{\begin{array}{rll} \partial_t y(\xi,t) + \alpha(-\Delta)^{\varsigma} y(\xi,t) + & = F(y(\xi,t)) + x(\xi)u(t) & \text{for all } \xi \in D \times (0,\infty), \\ y(\xi,t) & = 0 & \text{for all } \xi \in D_{\mathcal{I}_\delta} \times (0,\infty), \\ y(0,\xi) & = x(\xi) & \text{for all } \xi \in D, \end{array}\right. \tag{5-23}$$

where the nonlinear term is given by $F(y) = y^2(1-y)$ and the parameter $\alpha = 0.01$.

Here, the finite element spatial discretization results in the system

$$\begin{cases} M\dot{y}(t) &= -\alpha A y(t) + \mathbf{F}(y(t)) + Qu(t) \quad t \in (0,\infty) \\ y(0) &= x \qquad\qquad\qquad\qquad\qquad\quad x \in \Omega, \end{cases} \tag{5-24}$$

where the nonlinear term $\mathbf{F} : \mathbb{R}^d \to \mathbb{R}^d$ is $\mathbf{F}(y(t)) = y(t)^2 - y(t)^3$, $Q \in \mathbb{R}^d$ with $Q_i = x(\xi_i)$ for $i = 1, \dots, d$ and ξ_i a node in the discretization of D. The initial condition used in this case is the same used in Section 5.3.1, i.e. the solution of the Poisson problem. The cost functional we want to minimize is

$$\mathcal{J}_x(y,u) := \frac{1}{2}\int_0^\infty (\|y(s)\|^2_{L^2(D)} + \gamma \|u(t)x\|^2_{L^2(D)}) e^{-\lambda s} ds, \tag{5-25}$$

subjected to the semi-discrete system (5-24).

The parameters chosen in the cost are $\lambda = 0.5$ and $\gamma = 0.01$. To generate the scattered mesh we consider the control space $U = [-0.5, 0]$ discretized in 11 nodes, a temporal step $\overline{\Delta t} = 0.025$ and the same initial condition from the first test. We collect the trajectory points generated solving the system (5-24) up to final time $T = 6$ for $d = 63$. The scattered mesh has 2652 nodes and the separation distance is 0.12.

The space $\mathcal{P} = [0.08, 0.12]$ is discretized with step size 0.01. The residual is minimized when the parameter is equal to 0.09. We run the value iteration algorithm with $\Delta t = 0.01$ and U discretized in 21 nodes. We obtain the feedback control and solution considering 81 discrete control points and $\Delta t = 0.025$. In Figure 5.7, we show the uncontrolled solution on the left and the controlled solution on the right. We can visually see how the controlled solution reaches the desired configuration which is the zero equilibrium. In the

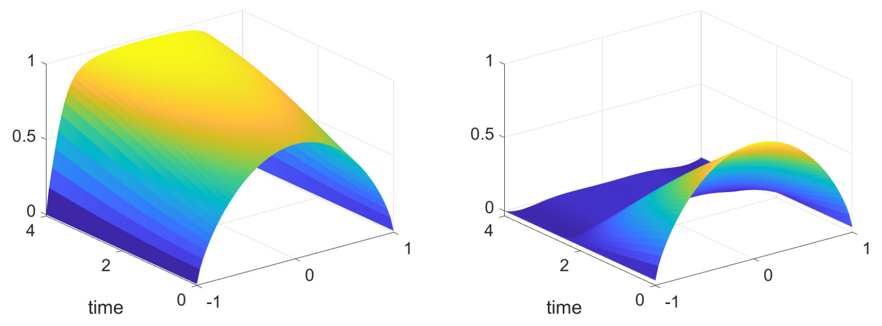

Figure 5.7: Test 3. Left: Uncontrolled solution. Right: Controlled solution.

left panel of Figure 5.8, we compare the evaluation of the cost functional for controlled and uncontrolled solution. As expected the controlled solution has a lower cost functional with respect to the the uncontrolled one. Finally, in the right panel of Figure 5.8 we show the control found. The control is active at the beginning then it is zero when the desired configuration is reached.

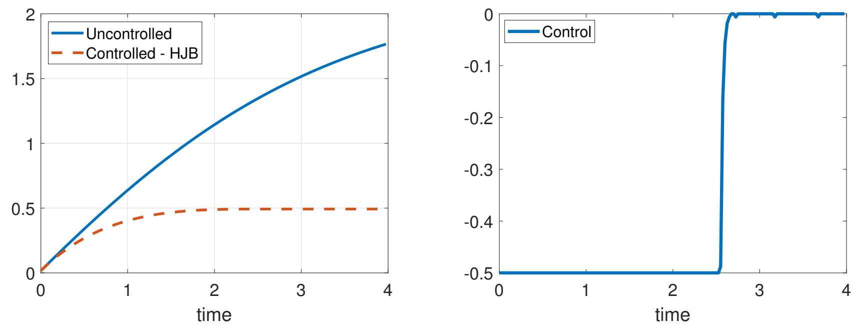

Figure 5.8: Test 3. Left: Cost functional. Right: Optimal control.

Using the same value function from previous test, now we compute the feedback control and trajectory with the presence of a noise term. Here we consider as perturbation a 63 dimensional vector with each component following a normal random variable with zero mean and standard deviation of 0.0025. With this configuration we have a probability of 95.45% of selecting a number in $[-0.005, 0.005]$. At each instance of time we add to the trajectory a new independent perturbation term.

Simulation with noise The feedback control and the trajectory are computed with $\Delta t = 0.025$ and U discretized in 81 nodes. In the left panel of Figure 5.9 we show the controlled solution under disturbances. We can see that, although the noise, we are able to reach our desired configuration. Then, in the right panel of Figure 5.9 we show the control found. The behaviour of the control is qualitatively similar to the picture in the right panel of Figure 5.8, but we can note how the feedback is able to react and adjust itself, see e.g. the pick around $t = 3.5$.

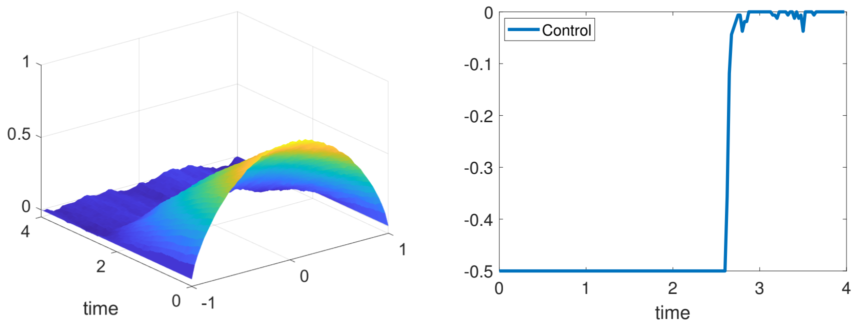

Figure 5.9: Test 3. Simulation with noise. Left: Controlled solution. Right: Optimal control.

6
Summary and Conclusions

In this thesis, we have proposed a new approach to approximate the solution of HJB equations for infinite horizon optimal control problems using Shepard approximation. The method mitigates the curse of dimensionality and provides a new approach to control high dimensional systems coming from PDEs. Our approach has some important advantages if compared to traditional methods that rely on the regular discretizations of time and state space. The first advantage is that there is no dependence of a regularly discretized state space since our method is based on a semi-Lagrangian scheme that uses the Shepard approximation as reconstruction tool.

Another advantage of our approach is the use of the problem dynamics itself to generate the scattered mesh. Since we have the parameters properly chosen (a time step, a discretization of the control set), the trajectories collected from the evolution of the discrete dynamical system will populate only certain regions of the domain. Thus, this strategy does not require a complete domain discretization and consequently reduces the computational memory requirements.

Furthermore, we have discussed RBFs and their use in the Shepard's method. We used Wendland functions since its compact support is important when working with large scale distance matrices as they become sparse. We also pointed out the importance of an adequate selection of the shape parameter to improve the quality of the Shepard approximation. This selection is performed comparing residual values [24] computed from different value functions, each of them obtained using a different shape parameter in the Shepard approximation. The selected shape parameter is the value associated to the minimal residual.

We have also adapted classical theoretical results presented in e.g. [20] to the use of Shepard method and to the generation of our grid.

The effectiveness of our method was tested for low and high dimensional problems. In fact, in our numerical tests, the desired state was always reached and the method was also stable under disturbances. Furthermore, we have also shown the possibility to control such systems for different types of initial conditions, specifically, coming from a class of initial conditions.

We briefly discussed nonlocal equations and their basic differences from traditional PDE models. The control of nonlocal equations is usually performed by the open-loop approach. The novelty we have presented in this thesis is the control of nonlocal models using our DP approach. We analyzed the convergence of the controlled solution towards the continuous model. Furthermore, we could also control nonlocal problems when the system is disturbed.

Finally, we would like to stress that the fundamental idea introduced in this thesis may be of interest also when applied to other local-approximation methods. Thus, the method is not necessarily bounded to the RBF-based Shepard approximation introduced in Section 3.2. In particular, the application of a semi-Lagrangian scheme and the construction of the scattered mesh that are discussed in this thesis, can all be applied together with any interpolation method that can work on high-dimensional and scattered meshes.

Future directions. The number of nodes that form the dynamics-driven mesh can be large depending on the system and the choice of the parameters used to run the discrete problem. The implementation of the algorithm demands that we save a distance matrix associated to each control used in the grid construction. The mesh size affects the dimension of the distance matrices and it can lead to a computational restriction due to the amount of memory required to store all the matrices. In order to overcome this problem a possible strategy would be to reduce the dimension of the dynamical system beforehand using Model Order Reduction methods (e.g. POD) in a similar way to [31] with the aim that the reduced system could be controlled using a lower number of variables and the computation of the value function could be more efficient.

Finally, we remark that the proposed method can be generalized also to other hyperbolic equations using semi-Lagrangian schemes.

Bibliography

[1] PONTRYAGIN, L. S.; BOLTYANSKII, V.; GAMKRELIDZE, R. ; MISHCHENKO, E. A.. **The Mathematical Theory of Optimal Processes**. Interscience, 1962.

[2] HALKIN, H.. **Necessary conditions for optimal control problems with infinite horizons**. Econometrica, 42(2):267–272, 1974.

[3] ASEEV, S. M.; KRYAZHIMSKIY, A. V.. **The Pontryagin Maximum Principle and transversality conditions for a class of optimal control problems with infinite time horizons**. SIAM Journal on Control and Optimization, 43(3):1094–1119, 2004.

[4] GRASS, D.; CAULKINS, J. P.; FEICHTINGER, G.; TRAGLER, G.; BEHRENS, D. A. ; OTHERS. **Optimal control of nonlinear processes**. Berlino: Springer, 2008.

[5] MAURER, H.. **Numerical solution of singular control problems using multiple shooting techniques**. Journal of optimization theory and applications, 18(2):235–257, 1976.

[6] GRÜNE, L.. **Numerical Methods for Nonlinear Optimal Control Problems**, p. 1–10. 01 2014.

[7] GERDTS, M.. **Optimal control of ODEs and DAEs**. Walter de Gruyter, 2011.

[8] BETTS, J. T.. **Practical methods for optimal control and estimation using nonlinear programming**. Society for Industrial and Applied Mathematics, 2010.

[9] GRÜNE, L.; PANNEK, J.. **Nonlinear model predictive control**. In: NONLINEAR MODEL PREDICTIVE CONTROL, p. 45–69. Springer, 2017.

[10] MAYNE, D. Q.; RAWLINGS, J. B.; RAO, C. V. ; SCOKAERT, P. O.. **Constrained model predictive control: Stability and optimality**. Automatica, 36(6):789–814, 2000.

[11] BELLMAN, R.. **Dynamic Programming**. Rand Corporation research study. Princeton University Press, 1957.

[12] BARDI, M.; CAPUZZO-DOLCETTA, I.. **Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations**. Springer, 1997.

[13] FLEMING, W. H.; SONER, H. M.. **Controlled Markov processes and viscosity solutions**. Springer Science & Business Media, 2006.

[14] FALCONE, M.. **A numerical approach to the infinite horizon problem of deterministic control theory**. Applied Mathematics and Optimization, 15(1):1–13, 1987.

[15] DOLCETTA, I. C.; ISHII, H.. **Approximate solutions of the Bellman equation of deterministic control theory**. Applied Mathematics and Optimization, 11(1):161–181, 1984.

[16] CRANDALL, M. G.; LIONS, P.-L.. **Two approximations of solutions of Hamilton-Jacobi equations**. Mathematics of computation, 43(167):1–19, 1984.

[17] BARLES, G.; SOUGANIDIS, P. E.. **Convergence of approximation schemes for fully nonlinear second order equations**. Asymptotic analysis, 4(3):271–283, 1991.

[18] KOSSIORIS, G.; MAKRIDAKIS, C. ; SOUGANIDIS, P. E.. **Finite volume schemes for Hamilton-Jacobi equations**. Numerische Mathematik, 83(3):427–442, 1999.

[19] BELLMAN, R. E.; DREYFUS, S. E.. **Applied Dynamic Programming**. Princeton University Press, 1962.

[20] FALCONE, M.; FERRETTI, R.. **Semi-Lagrangian approximation schemes for linear and Hamilton-Jacobi equations**. Society for Industrial and Applied Mathematics, 2013.

[21] ALLA, A.; FALCONE, M. ; KALISE, D.. **An efficient policy iteration algorithm for dynamic programming equations**. SIAM Journal on Scientific Computing, 37(1):A181–A200, 2015.

[22] JUNGE, O.; SCHREIBER, A.. **Dynamic programming using radial basis functions**. Discrete and Continuous Dynamical Systems, 35(9):4439–4453, 2015.

[23] CARLINI, E.; FERRETTI, R.. **A semi-Lagrangian scheme with radial basis approximation for surface reconstruction.** Computing and Visualization in Science, 18(2):103–112, 2017.

[24] GRÜNE, L.. **An adaptive grid scheme for the discrete Hamilton-Jacobi-Bellman equation.** Numerische Mathematik, 75(3):319–337, 1997.

[25] FERRETTI, G.; FERRETTI, R.; JUNGE, O. ; SCHREIBER, A.. **An adaptive multilevel radial basis function scheme for the HJB equation.** IFAC-PapersOnLine, 50(1):1643–1648, 2017.

[26] SIROVICH, L.. **Turbulence and the dynamics of coherent structures, parts i, ii and iii.** Quart. Appl. Math., p. 561–590, 1987.

[27] VOLKWEIN, S.. **Model reduction using Proper Orthogonal Decomposition.** Lecture Notes, University of Konstanz, 2013.

[28] KUNISCH, K.; VOLKWEIN, S. ; XIE, L.. **HJB-POD-based feedback design for the optimal control of evolution problems.** SIAM Journal on Applied Dynamical Systems, 3(4):701–722, 2004.

[29] ALLA, A.; FALCONE, M. ; VOLKWEIN, S.. **Error analysis for POD approximations of infinite horizon problems via the Dynamic Programming approach.** SIAM Journal on Control and Optimization, 55(5):3091–3115, 2017.

[30] ALLA, A.; FALCONE, M. ; SALUZZI, L.. **An efficient DP algorithm on a tree-structure for finite horizon optimal control problems.** SIAM Journal on Scientific Computing, 41(4):A2384–A2406, 2019.

[31] ALLA, A.; SALUZZI, L.. **A HJB-POD approach for the control of nonlinear PDEs on a tree structure.** Applied Numerical Mathematics, 155:192–207, 2020. Structural Dynamical Systems: Computational Aspects held in Monopoli (Italy) on June 12-15, 2018.

[32] CHILAN, C. M.; CONWAY, B. A.. **Optimal nonlinear control using Hamilton-Jacobi-Bellman viscosity solutions on unstructured grids.** Journal of Guidance, Control, and Dynamics, 43(1):30–38, 2020.

[33] KALISE, D.; KUNISCH, K.. **Polynomial approximation of high-dimensional Hamilton-Jacobi-Bellman equations and applications to feedback control of semilinear parabolic PDEs.** SIAM Journal on Scientific Computing, 40(2):A629–A652, 2018.

[35] MCENEANEY, W. M.. **A curse-of-dimensionality-free numerical method for solution of certain HJB PDEs**. SIAM journal on Control and Optimization, 46(4):1239–1276, 2007.

[36] MCENEANEY, W. M.. **Convergence rate for a curse-of-dimensionality-free method for Hamilton–Jacobi–Bellman PDEs represented as maxima of quadratic forms**. SIAM Journal on Control and Optimization, 48(4):2651–2685, 2009.

[37] BOKANOWSKI, O.; GARCKE, J.; GRIEBEL, M. ; KLOMPMAKER, I.. **An adaptive sparse grid semi-Lagrangian scheme for first order Hamilton-Jacobi Bellman equations**. Journal of Scientific Computing, 55(3):575–605, 2013.

[38] DARBON, J.; LANGLOIS, G. P. ; MENG, T.. **Overcoming the curse of dimensionality for some Hamilton-Jacobi partial differential equations via neural network architectures**. Research in the Mathematical Sciences, 7(3):1–50, 2020.

[39] DARBON, J.; MENG, T.. **On some neural network architectures that can represent viscosity solutions of certain high dimensional Hamilton-Jacobi partial differential equations**. Journal of Computational Physics, 425:109907, 2021.

[40] ALLA, A.; OLIVEIRA, H. ; SANTIN, G.. **HJB-RBF based approach for the control of PDEs**, 2021, Submitted to the Journal of Scientific Computing, https://arxiv.org/abs/2108.02987.

[41] ALLA, A.; D'ELIA, M.; GLUSA, C. ; OLIVEIRA, H.. **Control of fractional diffusion problems via dynamic programming equations**, 2022, Submitted to the Journal of Peridynamics and Nonlocal Modeling, Springer, https://arxiv.org/abs/2210.09827.

[42] FASSHAUER, G. E.; ZHANG, J. G.. **On choosing "optimal" shape parameters for RBF approximation**. Numerical Algorithms, 45(1):345–368, 2007.

[43] WENDLAND, H.. **Scattered Data Approximation**. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2004.

[44] HOWARD, R. A.. **Dynamic programming and Markov processes**. 1960.

[45] ACHDOU, Y.; BARLES, G.; ISHII, H. ; LITVINOV, G. L.. **Hamilton-Jacobi equations: approximations, numerical analysis and applications**. 2013.

[46] SETHIAN, J. A.. **Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science**. Cambridge university press, 1999.

[49] FASSHAUER, G. E.. **Meshfree Approximation Methods with Matlab**. WORLD SCIENTIFIC, 2007.

[50] SHEPARD, D.. **A two-dimensional interpolation function for irregularly-spaced data**. In: PROCEEDINGS OF THE 1968 23RD ACM NATIONAL CONFERENCE, p. 517–524, 1968.

[51] SCHMIDT, A.; HAASDONK, B.. **Data-driven surrogates of value functions and applications to feedback control for dynamical systems**. IFAC-PapersOnLine, 51(2):307–312, 2018.

[52] FASSHAUER, G.; MCCOURT, M.. **Kernel-based Approximation Methods using MATLAB**. WORLD SCIENTIFIC, 2015.

[53] LEVEQUE, R. J.. **Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems**. SIAM, 2007.

[54] D'ELIA, M.; DU, Q.; GUNZBURGER, M. ; LEHOUCQ, R.. **Nonlocal convection-diffusion problems on bounded domains and finite-range jump processes**. Computational Methods in Applied Mathematics, 29:71–103, 2017.

[55] PANG, G.; D'ELIA, M.; PARKS, M. ; KARNIADAKIS, G. E.. **nPINNs: nonlocal Physics-Informed Neural Networks for a parametrized nonlocal universal Laplacian operator. Algorithms and Applications**. to appear in Journal of Computational Physics, 2020.

[56] BURKOVSKA, O.; GLUSA, C. ; D'ELIA, M.. **An optimization-based approach to parameter learning for fractional type nonlocal models**. Computers and Mathematics with Applications, 2021. in print.

[57] D'ELIA, M.; GUNZBURGER, M.. **Identification of the diffusion parameter in nonlocal steady diffusion problems**. Applied Mathematics and Optimization, 73:227–249, 2016.

[58] XU, X.; D'ELIA, M. ; FOSTER, J.. **A machine-learning framework for peridynamic material models with physical constraints**. Computer Methods in Applied Mechanics and Engineering, 2021, accepted.

[59] YOU, H.; YU, Y.; SILLING, S. ; D'ELIA, M.. **Data-driven learning of nonlocal models: from high-fidelity simulations to constitutive laws**. Accepted in AAAI Spring Symposium: MLPS, 2021.

[60] YOU, H.; YU, Y.; TRASK, N.; GULIAN, M. ; D'ELIA, M.. **Data-driven learning of robust nonlocal physics from high-fidelity synthetic data**. Computer Methods in Applied Mechnics and Engineering, 374:113553, 2021.

[61] GORENFLO, R.; MAINARDI, F.. **Fractional calculus: integral and differential equations of fractional order**. Fractals and Fractional Calculus in Continuum Mechanics, p. 223–276, 1997.

[62] D'ELIA, M.; GULIAN, M.; OLSON, H. ; KARNIADAKIS, G.. **Towards a unified theory of fractional and nonlocal vector calculus**. Fractional Calculus and Applied Analysis, 24:1301–1355, 10 2021.

[63] D'ELIA, M.; GUNZBURGER, M.. **The fractional Laplacian operator on bounded domains as a special case of the nonlocal diffusion operator**. Computers and Mathematics with applications, 66:1245–1260, 2013.

[64] DU, Q.; GUNZBURGER, M.; LEHOUCQ, R. ; ZHOU, K.. **Analysis and approximation of nonlocal diffusion problems with volume constraints**. SIAM Review, 54:667–696, 2012.

[65] MCLEAN, W.; MCLEAN, W. C. H.. **Strongly elliptic systems and boundary integral equations**. Cambridge university press, 2000.

[66] FASSHAUER, G. E.. **On smoothing for multilevel approximation with radial basis functions**. Approximation theory IX, 2:55–62, 1998.

[67] ANTIL, H.; OTAROLA, E.. **A FEM for an optimal control problem of fractional powers of elliptic operators**. SIAM Journal on Control and Optimization, 53(6):3432–3456, 2015.

[68] D'ELIA, M.; GLUSA, C. ; OTÁROLA, E.. **A priori error estimates for the optimal control of the integral fractional laplacian**. SIAM Journal on Control and Optimization, 57(4):2775–2798, 2019.

[69] ALBI, G.; CHOI, Y.-P.; FORNASIER, M. ; KALISE, D.. **Mean field control hierarchy**. Applied Mathematics & Optimization, 76(1):93–135, 2017.

[70] DOLGOV, S.; KALISE, D. ; SALUZZI, L.. **Data-driven tensor train gradient cross approximation for Hamilton-Jacobi-Bellman equations**, 2022.

[71] GLUSA, C.; OTÁROLA, E.. **Error estimates for the optimal control of a parabolic fractional PDE**. SIAM Journal on Numerical Analysis, 59(2):1140–1165, 2021.

[72] TRÖLTZSCH, F.. **Optimal control of partial differential equations: theory, methods, and applications**. American Mathematical Soc., 2010.

[73] BRUNTON, S. L.; KUTZ, J. N.. **Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control**. Cambridge University Press, 2019.

[74] SEDGEWICK, R.; WAYNE, K.. **Algorithms**. Addison-Wesley Professional, 2011.