Kungliga Tekniska Högskolan
Department of Mathematics, division of Numerical Analysis

Delft University of Technology
Faculty of Electrical Engineering, Mathematics & Computer Science

Master Thesis Computer Simulation for Science and Engineering

COMPARING TWO APPROACHES OF MODELLING FISH HARVESTING STRATEGIES USING OPTIMAL CONTROL

Author:
Niels Floris Leonardus in 't Veld

Supervisors:
Mattias Sandberg
Cornelis Vuik

Examiner:
Olof Runborg

August 17, 2022

Sammanfattning

Optimal kontroll är ett paradigm för att lösa optimeringsproblem som omfattar dynamiska system som ska kontrolleras. Den kan lösa problem med skörd av fisk där vi vill optimera skörd av fisk genom att betrakta fisket som en kontrollfunktion som verkar på tillståndet i det dynamiska systemet, som representerar tillväxten av fiskarter i miljön. Andra modelleringsaspekter av optimal styrning är att definiera slutkostnader och löpande kostnader, t.ex. maximering av vinsten. Vi håller terminalvillkoret jämförbart för ett antal olika arter. Det baseras på den ursprungliga populationen.
Genom att använda Hamiltonianen för optimal styrning och Pontryagins maximiprincip kan vi beräkna de optimala tillståndsbanorna som motsvarar lämpliga optimala styrningar. Hamiltonianen är beroende av tillståndsekvationen och driftskostnaderna. Vi presenterar två metoder för att modellera driftskostnaderna. Ett tillvägagångssätt som inte är direkt överförbart till problemet med skörd av fisk, men som leder till en slät Hamiltonian, vilket förenklar härledning och beräkning avsevärt. Den andra metoden, som är likvärdig med vinstmaximering, leder till en icke slät Hamiltonian. Detta leder till hopp-diskontinuerliga derivator som behövs för beräkningen. Vi föreslår att man reglerar Hamiltonianens derivator med hjälp av lämpliga släta funktioner, så att det är likvärdigt med att reglera Hamiltonianen direkt. Vi ger detaljer för genomförandet av båda tillvägagångssätten upp till system med n konkurrerande arter. Därefter går vi in i detalj på algoritmer och den implementerade programmeringsstrukturen. Slutligen visar vi genom numeriska experiment, för en och två arter, sambandet mellan den optimala kontrollen och slutkostnaderna. Men mer intressant är att de släta hamiltoniska modellerna är otillräckliga, vilket ger upphov till att reglerade hamiltoniska modeller är att föredra. Intressant nog resulterar det senare tillvägagångssättet i en stabil lösning, där kontrollen fungerar som en stabilisator.

Abstract

Optimal control is a paradigm for solving optimization problems involving dynamical systems, which are to be controlled. It is able to solve fish harvesting problems, in which we want to optimize harvesting out-take by considering fishing as a control function that acts on the state of the dynamical system, which represents the growth of fish species in the environment. Other modelling aspects of optimal control are defining terminal costs and running costs, e.g. maximizing profit. We keep the terminal condition comparable for a different number of species. It is based on the initial population.
By using the optimal control Hamiltonian and Pontryagin's Maximum Principle we can calculate the optimal state trajectories corresponding to suitable optimal controls. The Hamiltonian is dependent on the state equation and the running costs. We present two approaches of modelling the running costs. An approach that is not directly translatable to the fish harvesting problem, but it leads to a smooth Hamiltonian, which greatly simplifies derivation and computation. The other, which is equivalent to maximizing profit, leads to a non-smooth Hamiltonian. This leads to jump-discontinuous derivatives needed for computation. We propose to regularize the derivatives of the Hamiltonian using suitable smooth functions, such that it is equivalent to regularizing the Hamiltonian directly. We give details for implementing both approaches up to systems of n competing species. After which we go into detail on algorithms and programming structure implemented. Finally, in modest numerical experiments, for one and two species, we show the relation between the optimal control and the terminal costs. But more interestingly, that the smooth Hamiltonian models are inadequate and regularized Hamiltonian models are the preferred choice. Intriguingly, the latter approach results in steady state solution, where the control acts as a stabilizer.

Acknowledgements

First and foremost, I would like to use this opportunity to thank my supervisor Mattias Sandberg for his guidance and the freedom to shape my thesis into what it has become. He extended me the opportunity to use this project to develop a detailed program for solving optimal fish harvesting problems. This greatly improved both my interest in computational science and object oriented programming. Next, I want to thank my family and friends in supporting me throughout this tumultuous year. I would like to express my appreciation to two persons in particular. Firstly, Rianne 't Jong, who helped both with her linguistic feedback and by motivating me throughout the year. Additionally, special thanks to Storm van de Linde for his listening ear, support with programming issues and structuring.
To whomever it may concern, I hope you find this report enjoyable and informative. For the interested programmer my code has been added to Github.

Stockholm, May 28, 2022

Niels Floris Leonardus in 't Veld

Contents

1 Introduction

Fishing is a practice practice dating back at least 40.000 years all the way to hunter-gatherers, who, in contrast to what the name suggests, were mostly fishermen and foragers [1]. Fishing is a relatively stable non-dangerous method of gathering food, which made it the preferred practice next to foraging. Nowadays, most of us (luckily) do not have to gather our own food anymore and can simply rely on supermarkets. However, people all across the world still depend greatly on fish harvesting for their livelihood. The world population is increasing rapidly, which increases the need for food in order to sustain itself, making it necessary to fish more (intensively). This, however, puts more stress on the environment and its ecosystems. The number of fisheries that are unsustainable increases, which increases the risk of ecosystem overfishing [2]. The necessity arises to fish in a sustainable way.
Mathematics can help in modelling fishing strategies in order to get a better idea what optimal fishing strategies look like and how to make them sustainable. Optimal control theory is able to deal with such questions in a flexible way. It combines different areas of mathematics, e.g. differential equations, optimization and dynamical systems.
Through the use of Pontryagin's maximum principle, the Lagrange multiplier method and the Hamilton-Jacobi-Bellman equation we will explain the relation between the state equation, costate equation and optimal control Hamiltonian. Details for single species, two species and n-species models are set out, analyzed and computed. A central role in computing optimal fishing strategies is reserved for the Hamiltonian, independent of the control α,

$$H(x, \lambda) := \min_{\alpha \in \mathcal{A}} \{L(x, \alpha) + \lambda \cdot f(x, \alpha)\},$$

which is a combination of the running cost, L, an (auxiliary) costate λ and state equation $x' = f(x, \alpha)$, where x is the state and α is the control. In this thesis we set out two approaches of formulating the running cost. In the first approach the running cost is not directly translatable to the fishing problem but the resulting Hamiltonian is smooth (concave) in both x and λ. The second approach is intuitively related to the fishing problem, however, the resulting Hamiltonian is non-smooth. In order to solve such problems we regularize the Hamiltonian with smooth Hamiltonians, H^δ, that satisfy

$$H^\delta \to H \quad \text{as} \quad \delta \to 0.$$

We iteratively solve the problem by decreasing δ and use the solution obtained in the previous iteration as a first guess for the current iteration. Technically, for computations we only need to use the partial derivatives of the Hamiltonian, which for the non-smooth Hamiltonian are jump discontinuous. By regularizing the heaviside step function appropriately we make sure that it is equivalent to regularizing the Hamiltonian as a whole.
Finally, for both approaches the advantages and disadvantages followed from mathematical analysis, numerical implementations and simulations are set out. Based on these observations we give preliminary conclusions on modelling sustainable fish harvesting strategies.

2 | Mathematical Background

We find optimal fishing strategies by combining various mathematical areas. In order to explain the road ahead we need to digest some relevant theory related to differential equations, optimization, optimal control and numerical analysis. In this chapter we give a concise overview of all relevant topics related to differential equations and optimization. Chapter 3 and 4 are reserved for optimal control and numerical analysis, respectively. The reader who has sufficient knowledge about mathematics in general is advised to skip to Chapter 5.

2.1 Preliminary Definitions

For the sake of brevity we will list the relevant definitions, starting with elementary definitions and moving on to more complicated mathematical structures.

- A **set** A is a collection of elements, such that $a \in A$ (a is an element of A). Examples are $\mathbb{N} = \{1, 2, \cdots\}$ and $\mathbb{R} = \{\cdots, -\pi, -2, -1, 0, 1, 2, \pi, \cdots\}$.
- A **function** f is a mapping from a set A into B, denoted $f : A \to B$ such that for $a \in A : f(a) = b \in B$.
- A function f is **continuous** in a point d if its limit exists, i.e.

  $$\lim_{x \to d} f(x) = f(d).$$

  And it is continuous on a set D if it is continuous in every $d \in D$.
- A function $f : A \to \mathbb{R}$ **lipshitz continuous** if there exists a constant $L > 0$, s.t.

  $$|f(x) - f(y)| \leq L|x - y| \quad \forall x, y \in A.$$

  The constant L is called the Lipschitz constant.
- A function f is called **convex** if

  $$\forall \lambda \in [0, 1], \forall x, y \in D : f(\lambda x + (1 - \lambda)y) \geq \lambda f(x) + (1 - \lambda) f(y).$$

  A function is called **concave** if $-f$ is convex.
- A function $f : A \to \mathbb{R}$ is called **linear** in a variable x if $\forall x, y \in A : f(x + y) = f(x) + f(y)$ and $\forall c \in \mathbb{R} : f(cx) = cf(x)$. If a function is linear it is both convex and concave.
- A function $f : \mathbb{R} \to \mathbb{R}$ is **differentiable** if the limit

  $$f'(x) := \lim_{h \to 0} \frac{f(x + h) - f(x)}{h},$$

  exists and is finite. The notations $\frac{df}{dx}$ an $f_x(x)$ are also possible.
- A function f is called **smooth** on a domain D if it is differentiable everywhere (hence continuous).

- The **partial derivative**, w.r.t. the i-th variable, of a function $f : \mathbb{R}^n \to \mathbb{R}$ is defined via

$$\frac{\partial f(x)}{\partial x_i} := \lim_{h \to 0} \frac{f(x_1, \ldots, x_i, x_i + h, x_{i+1}, \ldots, x_n) - f(x)}{h},$$

where $x = (x_1, \ldots, x_n)$, assuming it exists and is finite. The notations $\partial_{x_i} f(x)$ and $f_{x_i}(x)$ are also possible. We can summarize all partial derivatives into a vector, called the **gradient** of f

$$\nabla f = \begin{pmatrix} \partial_{x_1} f(x) \\ \vdots \\ \partial_{x_n} f(x) \end{pmatrix}.$$

- The **Jacobian**, $\mathcal{J} f$, of a multi-variable function f is the matrix of its partial derivatives, i.e.

$$\mathcal{J} f(x) = \left(\frac{\partial^2 f(x)}{\partial x_i \partial x_j} \right)_{ij}.$$

- **Minima** and **maxima** of a function f are called **extrema**. A point x is called a **global** minimum if $f(x) < f(y), \forall y \neq x$. It is called a **local** minimum if $f(x) \leq f(y)$ for every y in the neighbourhood of x. Local and global maxima are defined similarly.

- A point c is called a **stationary point** of a function f if the derivative at that point vanishes, i.e. if $f'(c) = 0$. Extrema are examples.

- We highlight the mathematical structure of a **space**, which is a set with some added structure. Usually in the form of a distance, metric or a norm. We only restrict ourselves to examples, forgoing all the tedious details. For instance, adding the notion of distance to the set of real numbers, $\mathbb{R}$, turns to a space of numbers, where one can say 1 is closer to 0 than 3. Similarly, one can define a function space by taking a certain set of functions and adding some structure.

- A **vector space** $\mathbb{R}^n$, where the distance between two elements $x, y \in \mathbb{R}^n$ is defined by a norm, e.g. the **maximum norm**: $\|x - y\|_{\max} = \sup_n |x_i - y_i|$.

- The **space of continuous functions** $\mathcal{C}$ between two fixed sets A and B, is defined as follows

$$\mathcal{C} := \{f : A \to B |\ f \text{ is continous}\},$$

where one would need to define the distance (or metric) between two function-elements. E.g. the **infinity norm**: $f, g \in \mathcal{C} : \|f - g\|_{\max} = \sup_x |f(x) - g(x)|$.

- The **space of differentiable functions** on a set D is defined as

$$\mathcal{C}^1(D) := \{f | f : D \to \mathbb{R}\, f \text{ differentiable}\}.$$

The **space of** n **times differentiable functions**, $\mathcal{C}^n$, is defined similarly.

- A **functional**, $J : \mathcal{F} \to \mathbb{R}$ is a function that maps the function u (from a function space $\mathcal{F}$) to a scalar, i.e. $J(u) \in \mathbb{R}\, \forall u \in \mathcal{F}$.
- A **(functional) derivative** in u of a functional $J : \mathcal{F} \to \mathbb{R}$ in a direction $g \in \mathcal{F}$ is defined via

$$\frac{\mathrm{d}J(u+\epsilon g)}{\mathrm{d}\epsilon}\Big|_{\epsilon=0} := \lim_{\epsilon \to 0} \frac{F(u+\epsilon g) - F(u)}{\epsilon}.$$

  This is known as the Gateaux derivative. If the limit exists for all directions $g \in \mathcal{F}$ we say J is (Gateaux) differentiable in u. Furthermore, if all the gateaux derivatives (in u) vanish we will use the notation $\partial_u J(u) = 0$. Integrals are functionals, we treat functionals only in the context of integration, hence restricting ourselves to functional of the form

$$J(u(x)) = \int_{x_1}^{x_2} L(u, x)\, \mathrm{d}x.$$

  Note that we can add a constant arbitrarily without changing the definition.

2.2 Differential Equations

Now we can properly define a Differential Equation (DE). DE's are equations involving unknown functions and their derivatives. They are in some sense similar to algebraic equations, but instead of the solution being a number (or a set of numbers) the solution is a function (or a set of functions). They are classified by their highest order derivative. Systems of coupled DE's also exist. They are coupled in the sense that a variable of one DE can be part of another DE.

Equilibrium Solutions

In general, solving a DE is more complicated than solving algebraic equations. Sometimes solutions do not even exist and only in special cases is the solution analytic - one can write down a closed form expression. Usually, one is more interested in finding equilibrium solutions. An equilibrium solution is a solution for which the derivatives are equal to zero. In such solutions the rate of change is zero and the solution is constant.

Relevance of DE's

One may wonder why DE's are important. Generally, functions and derivatives represent physical quantities and their rate of change. With a DE one can define relationships between them. They are ubiquitous concepts in various scientific fields, including but not limited to engineering, physics, finance and biology. For example, there are differential equations describing the orbitals of planets, growth of populations and fluid flows. In some sense, the world is governed by DE's. But, perhaps it is more precise to say that DE's are an elaborate toolbox to describe (or model) the (infinitely) complex nature of the world around us.

2.2.1 Ordinary Differential Equations

Different types of DE's exist. The simplest ones are called Ordinary Differential Equations (ODE's). An ODE contains an unknown function of only one variable, say x, its derivative(s) and possibly additional given functions (of x). To get a feeling for ODE's we present an example.

Example 1: Newton's second law of motion, $F = ma$,[1] can be written as a second-order ODE. Namely,

$$m\frac{d^2x}{dt^2} = F(x(t)),$$

where $x(t)$ is the position of a particle for every point in time, $F(\cdot)$ is a known function and m is the mass of the particle in question. One example is Hooke's law

$$m\frac{d^2x}{dt^2} = -kx(t),$$

which describes the movement of an undamped spring. It has the following set of analytic solutions

$$x(t) = c_1\cos(\sqrt{\tfrac{k}{m}}t) + c_2\cos(\sqrt{\tfrac{k}{m}}t),$$

where $(c_1, c_2) \in \mathbb{R}^2$. Another more complicated (non-linear) example is

$$m\frac{d^2x}{dt^2} = -kx(t)(1 - ax(t)),$$

sparing the details of the (hideous) analytic solutions.[2]

W.l.o.g.[3], we restrict ourselves to ODE's of the first order of the following form

$$\frac{dx}{dt} = g(x(t), t),$$

where g is a known (multivariable) function of the variable $x(t)$, which is itself an unknown function t. Let $\mathcal{F}$ denote the space of possible solution-functions. By adding an initial condition, the value of x at time t_0, the ODE is called an Initial Value Problem (IVP), as follows

$$\begin{cases} \dfrac{dx}{dt} = g(x(t), t) \\ x(t_0) = x_0 \end{cases},$$

where $x_0 \in \mathbb{R}$. The solution is an unique element of $\mathcal{F}$ (if it exists). If such a solution exists and is unique then the IVP is called well-posed. If the function g is continuous (in every point of (x, t)) and Lipschitz continuous in x then the IVP will be well-posed.

[1] Force is mass × acceleration.

[2] The interested reader is invited to check the solution via their favourite ODE solver. My personal favourite is WolframAlpha: m x"(t) = -kx(t)(1-ax(t)).

[3] Without loss of generality

2.2.2 Partial Differential Equations

Another type of DE is a Partial Differential Equation (PDE). A PDE contains functions of multiple variables and their partial derivatives. They are more difficult than ODE's in terms of interpretation, solutions and solvability. Very rarely does a solution to a PDE have an explicit expression. And numerically solving PDE's is far more computationally extensive than solving ODE's, yet not impossible. They are classified via their highest order partial derivative. The set of solutions for a PDE is infinite (if it exists). Similar to an IVP a PDE can be turned into a Boundary Value Problem (BVP) if boundary conditions are specified. A BVP is well-posed if the solution exists and is unique. Even if the solution is unique it is still highly dependent on the type of boundary constraints in question. We will not go into detail, since they are mostly beyond the scope of this thesis. To get a general idea we provide an example.

Example 2: The Heat Equation is a first-order time second-order space PDE, it describes the flow of heat in a medium. In its most general form it is defined as

$$\frac{\partial u}{\partial t} = \triangle u,$$

where $\triangle = \frac{\partial^2}{\partial x_1^2} + \cdots + \frac{\partial^2}{\partial x_n^2}$ is a differential operator called the Laplace operator. In one dimension this translates into

$$u_t = u_{xx},$$

where $u := u(x,t)$ is the heat at position x at time point t, it can describe the heat evolution in a one-dimensional rod. And in two dimensions

$$u_t = u_{xx} + u_{yy},$$

where $u := u(x,y,t)$ is the heat at position (x,y) at time point t, it can describe the heat evolution in a two-dimensional plate.

2.2.3 Non-linear Differential Equations

Lastly, we will reflect on the difference between a linear and non-linear DE. A non-linear DE is a differential equation that is not linear in the unknown function or its derivatives. Example 1 clearly show the distinction between a linear and non linear DE. Eventhough the difference looks subtle, solving a non-linear DE is far more complicated. This fact holds both in terms of analytic solutions as well as for solving the DE numerically.

2.3 Optimization

Another area which is necessary to touch upon is mathematical optimization. We will mostly restrict ourselves to the basics and essential topics in relation to optimal control. In general, optimization aims at minimizing (or maximizing) a decision function over a set of (decision) variables (or points) under a set of constraints. Some additional notes: we use the term points instead of variables, since this is more appropriate in our context. For readability, we omit the

ambiguity between minimizing or maximizing.[4] If there are no constraints in the Optimization Problem (OP) then we are dealing with unconstrained OP. In its most general form it is

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x)$$

For f we would like to find a global minimizer x^*, which is called the optimal solution. By finding the stationary points of f we hope to find the minima of f, but those may be local minima. If f is a convex function then it is a lot easier, since, for a convex function every local minimum is a global minimum. In general, it is more difficult to know if a local minimum is global.

2.3.1 Constrained Optimization

If constraints are involved, we are dealing with constrained optimization. They are more difficult to solve, since feasibility plays a role. We say a point x is feasible if no constraint is violated. If one (or more) constraints are violated the point is deemed infeasible. In usual terminology, let n be the dimension of x and m be the number of constraints. We can write a constrained OP in the following form

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & f(x, y) \\ \text{subject to} \quad & y = g(x) \end{aligned} \tag{2.1}$$

where $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is called the decision function, $y \in \mathbb{R}^m$, and $g(x) : \mathbb{R}^n \to \mathbb{R}^m$ the constraint function. The latter consists of m known functions g_i conveniently summarized into a function $g(x) = (g_1(x), \ldots, g_m(x))$. To be completely precise we also assume $f, g \in \mathcal{C}^1$. For (2.1) it is even more difficult to find a global minimum with feasibility playing a role.

2.3.2 Lagrange Multiplier Method

One way of tackling a constrained optimization problem is by applying the Lagrange multiplier theorem:

Theorem 1 (Lagrange Multiplier Theorem)**.** *Consider a constrained optimization problem with decision function $f : \mathbb{R}^n \to \mathbb{R}$ and constraint function $g(x) : \mathbb{R}^n \to \mathbb{R}^m$, both in $\mathcal{C}^1$. Let x^* be an optimal solution to*

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & c = g(x) \end{aligned} \tag{2.2}$$

Then there exists $\lambda \in \mathbb{R}^m$ such that

$$\nabla f(x^*) = \sum_{i=1}^{m} \lambda_i \nabla g_i(x^*).$$

Using this theorem a constrained OP can be transformed into an unconstrained OP by using the so-called Lagrangian function. The Lagrangian formulation of (2.1) is

$$\mathcal{L}(x, y, \lambda) = f(x, y) + \lambda^\top (y - g(x)),$$

[4] Note that maximizing a function f is the same as minimizing $-f$.

where $\lambda \in \mathbb{R}^m$ are known as Lagrangian Multipliers. Every element λ_i corresponds to a constraint function g_i. In a way, the Lagrange multipliers act as coefficients of the gradient for the optimal point and contains relevant information of the optimal point.

Necessary Condition

As previously stated, being a stationary point is a necessary condition for a minimum. Hence, we need the derivatives of the Lagrangian to vanish, i.e.

$$\nabla\mathcal{L}(x,y,\lambda) := \begin{pmatrix} \partial_x\mathcal{L}(x,y,\lambda) \\ \partial_y\mathcal{L}(x,y,\lambda) \\ \partial_\lambda\mathcal{L}(x,y,\lambda) \end{pmatrix} = 0 \iff \begin{cases} 0 = y - g(x) \\ 0 = \nabla_y f(x,y) + \lambda \\ 0 = \nabla_x f(x,y) - \sum_{i=1}^{m} \lambda_i \nabla g_i(x^*) \end{cases},$$

where $\nabla_x f(x,y)$ and $\nabla_y f(x,y)$ are the gradient w.r.t. x and y, respectively. Note that the first condition corresponds to the original constraint, the third condition is the Lagrange multiplier theorem and the second translates into

$$\lambda_i = -\frac{\partial f(x,y)}{\partial y_i}, \quad i = 1, \ldots, n.$$

Hence for (local) minima there necessarily needs to be a relation between the Lagrange multipliers and the gradient of f w.r.t. y.
The idea of Lagrangian multipliers is implemented in optimal control theory through the use of costates, which will be elaborated upon in the next chapter.

3 | Optimal Control

By first covering optimal control in general we lay out the framework in which we will define the problem of finding optimal fishing strategies. To this extent we need to cover relevant topics like (co)states, control functions, dynamic programming, the Hamilton Jacobi Bellman Equations and Pontryagin's Maximum Principle. Chapter 6 is reserved for the application of optimal control in the case of fish harvesting.

3.1 Control Theory

Control theory is the area of mathematics that deals with dynamical systems which can be controlled, in the sense that the evolution is influenced by an external force (or agent). A dynamical system is a system in which a function describes the time dependence of a point in a so-called state space. We will consider dynamical forms as a system of ODE's, as follows

$$\begin{cases} x_t' = f(x_t, \alpha_t), \quad 0 < t < T \\ x_0 = x^0 \end{cases}, \tag{3.1}$$

where $x_t := x(t)$ is a n-dimensional function, dependent on t, known as the **state** (of the system), x^0 is the initial condition and α_t is another function representing the **control**. To be very technical, $x : [0, T] \to \mathbb{R}^n$ and $\alpha : [0, T] \to A$, where T is the end-time and A is called the control set, hence

$$f : \mathbb{R}^n \times A \to \mathbb{R}^n.$$

For each initial point there are multiple trajectories describing the evolution of the system, depending on the choice of the control. One can wonder which set of final states can be reached depending on the control in question, or which control would be possible and which would be optimal for a certain objective. Formally we define a set of (all possible) admissible controls, via

$$\mathcal{A} = \{\alpha_t : [0, T] \to A \mid \quad \alpha \text{ is measurable}\},$$

where a measurable function is a function between two underlying spaces that preserves the structure of the spaces.[1] Now we are ready to formulate an optimal control problem (CP) in a general form, as follows

$$\begin{aligned} J[\alpha_t^*] := \underset{\alpha \in \mathcal{A}}{\text{minimize}} \quad & \int_0^T L(x_s, \alpha_s)\, \mathrm{d}s + g(x_T) \\ \text{subject to} \quad & x_t' = f(x_t, \alpha_t), \\ & x_0 = x^0, \end{aligned} \tag{3.2}$$

where α_t^* is called an **optimal control**, $L(x_t, \alpha_t) : \mathbb{R}^n \times A \to \mathbb{R}$ is known as the **running cost** and it models the cost from 0 to T, $g(x_T) : \mathbb{R}^n \to \mathbb{R}$ is called the

[1] A function f is measurable if the pre-image of any measurable set is measurable. Technically one needs this condition, since a non-measurable function is really weird, creating various problems in subsequent derivations.

terminal cost and is related to the terminal condition of the state trajectory. Both L and g are problem specific. The entire problem is subject to an IVP, known as the **state equation**, where f is called the flux. An optimal control solution α_t^* is dependent on T, L, f, $g(x_T)$ and x_0.

3.2 Lagrange Multiplier Functions

Analogous to the Lagrangian Multiplier Method for a constrained OP we can rewrite the problem in the form of a Lagrangian. To this extent define

$$\mathcal{L}(x, \lambda, \alpha) := g(x_T) + \int_0^T L(x_s, \alpha_s) + \lambda_s \cdot (f(x_s, \alpha_s) - x_s') \, \mathrm{d}s,$$

where $\lambda_t : [0, T] \to \mathbb{R}^n$ is the Lagrange multiplier function. In optimal control this is called the **costate**. The notation $x \cdot y$ denotes the inner product between two vectors x and y.[2] It holds that

$$\alpha_t^* = \arg\min_{\alpha \in \mathcal{A}} \mathcal{L}(x, \lambda, \alpha). \tag{3.3}$$

Hence, we would like to satisfy

$$\begin{cases} \partial_x \mathcal{L}(x, \lambda, \alpha) = 0 \\ \partial_\lambda \mathcal{L}(x, \lambda, \alpha) = 0 \\ \partial_\alpha \mathcal{L}(x, \lambda, \alpha) = 0 \end{cases}.$$

The derivation of the following conclusions can be found in Appendix A.1. The first equation leads to the following system

$$\begin{cases} -\lambda_t' = \partial_x \left(L(x_t, \alpha_t) + \lambda_t \cdot f(x_t, \alpha_t) \right) \\ \lambda_T = \nabla g(x_T) \end{cases}, \tag{3.4}$$

which is known as the **costate equation**. And satisfying the second equation is identical to satisfying the state equation, i.e.

$$x_t' = f(x_t, \alpha_t), \quad 0 < t < T.$$

The last equation leads to

$$\partial_\alpha \left(L(x_t, \alpha_t) + \lambda_t \cdot f(x_t, \alpha_t) \right) = 0. \tag{3.5}$$

Optimal Control Hamiltonian

The following might seem slightly arbitrary, but we will motivate it in a later section. Notice that $f(x, \alpha) = \partial_\lambda \left(L(x_t, \alpha_t) + \lambda_t \cdot f(x_t, \alpha_t) \right)$. This suggest defining a function via

$$h(x, \lambda, \alpha) := L(x_t, \alpha_t) + \lambda_t \cdot f(x_t, \alpha_t), \tag{3.6}$$

which we will call the **Hamiltonian function**. Noting that (3.5) indicates

$$\alpha_t^* = \arg\min_{\alpha \in \mathcal{A}} h(x, \lambda, \alpha).$$

[2] The inner product $x \cdot y$ is another way of saying $x^\top y$, but the former notation leads to less confusion in subsequent sections.

This motivates the idea of defining the **optimal control Hamiltonian**, via

$$H(x, \lambda) := \min_{\alpha \in \mathcal{A}} h(x, \lambda, \alpha). \tag{3.7}$$

3.3 Dynamic Programming

There is another way of solving CP (3.2), which makes use of what is known as dynamic programming. In this degree, define the value function

$$v(x, t) := \inf_{\alpha:[t,T]\to A} \left\{ g(x_T) + \int_t^T L(x_s, \alpha_s)\, \mathrm{d}s \,\middle|\, x_t = x \right\}. \tag{3.8}$$

The value function represents the optimal value of CP (3.2) but for a shorter time period $[t, T]$. Notice that v is independent on the control, α_t, and that

$$J[\alpha_t^*] = v(x^0, 0) \quad \wedge \quad v(x, T) = g(x_T), \quad \forall x.$$

If we look at the problem backwards, from end, $t = T$, to beginning, $t = 0$ and we manage to solve the similar problem of finding the optimal solution for $t \in [\tau, T]$, where τ is very close to T, then this help us in solving the original problem. This new problem is embedded in the original problem. Solving a problem in this manner is usually called backward induction and it motivates dynamic programming.

3.3.1 Hamilton-Jacobi-Bellman Equation

Using definition (3.8) one can derive a PDE to which v must adhere to. The entire derivation is in Appendix A.2. We state that $v(x, t)$ must satisfy,

$$\begin{cases} \partial_t v(x,t) + \min_{\alpha \in \mathcal{A}} \{ L(x_t, \alpha_t) + \partial_x v(x,t) \cdot f(x_t, \alpha_t) \} = 0 \\ v(x, T) = g(x_T) \end{cases}.$$

This is known as the Hamilton-Jacobi-Bellman Equation (HJB). This PDE is the defining relation of the dynamic programming approach. If we insert (3.6) into the HJB we get

$$\begin{cases} \partial_t v(x,t) + \min_{\alpha \in \mathcal{A}} \{ h(x, \partial_x v(x,t), \alpha) \} = 0 \\ v(x, T) = g(x_T) \end{cases}. \tag{3.9}$$

3.4 Hamiltonian System

By setting $\lambda_t := \partial_x v(x, t)$ and using (3.7) we get

$$\begin{cases} \partial_t v(x,t) + H(x, \lambda)) = 0 \\ v(x, T) = g(x_T) \end{cases}.$$

The HJB equation and the optimal control Hamiltonian can be summarized into the following theorem from [3]. It relates the optimal control problem (and its dynamical system) to a Hamiltonian system, i.e. showing that the characteristics of the HJB equation is a Hamiltonian system.

Theorem 2 (Hamiltonian System)**.** *Assume the (value function) $v \in \mathcal{C}^2$ and $H \in \mathcal{C}^1$, and*

$$x_t' = \partial_\lambda H(x, \lambda),$$

where $\lambda_t := \partial_x v(x,t)$. Then x_t and λ_t satisfy the Hamiltonian System

$$\begin{cases} x_t' = \partial_\lambda H(x,\lambda) \\ -\lambda_t' = \partial_x H(x,\lambda) \end{cases}. \tag{3.10}$$

Note that using this theorem the optimal control problem turns into a system of coupled ODE's, which is significantly easier to solve than the HJB equation, which is a PDE.
There are two remarkable properties that hold for every Hamiltonian. Firstly, the change over time is constant, i.e. $\partial_t H(x,\lambda) = 0$. Thus, H is constant over time. And that the flow of the Hamiltonian system is a symplectic mapping. In other words, the flow in the phase-space is volume-preserving, i.e. the area of a phase space is unchanged after the symplectic mapping is applied. We will not go into detail on this. But the fact itself leads to a more appropriate numerical scheme. For details on symplectic mappings we refer to [4].

3.4.1 Pontryagin's Maximum Principle

If we can find the optimal control Hamiltonian, then we can find a family of optimal control trajectories that is a local minimum. This is a central result in optimal control, called Pontryagin's Maximum Principle (PMP). It was first formulated by Pontryagin [5], who thought of optimal control in terms of maximization problems. We will formulate it for minimization problems.

Theorem 3 (Pontryagin's Minimum Principle)**.** *The optimal state x_t^*, control α_t^* (and costate λ_t^*) must minimize the Hamiltonian function, i.e.*

$$H(x^*, \lambda^*) := h(x^*, \lambda^*, \alpha^*) \leq h(x^*, \lambda^*, \alpha), \quad \forall \alpha \in \mathcal{A}.$$

Additionally, λ_t^* must satisfy the costate equation, i.e.

$$\begin{cases} -\lambda_t' = \partial_x h(x^*, \lambda, \alpha^*) \\ \lambda_T = \nabla g(x_T). \end{cases}.$$

These conditions are necessary conditions for an optimal control.

Note that PMP is not a sufficient condition for optimality. Using PMP alone, we cannot conclude a control is globally optimal. We state without proving that the a solution to HJB equation does offer sufficient conditions. But this involves solving a PDE, which in general is very difficult. On the other hand, any control that does not satisfy PMP cannot be optimal. As such it is useful to find candidate optimal controls.

4 | Numerical Analysis

Now that we understand optimal control from a theoretical perspective we need to transform it using numerical procedures in order to numerically compute the optimal control. But first we explain how to simulate the non-controlled state equations, which also helps us understand the numerical scheme for optimal control computations.

4.1 Simulating Differential Equations

Consider a generic ODE of the form $\{x'(t) = f(x,t), x(0) = x_0\}$. W.l.o.g. assume x is one-dimensional. We discretize the time-dimension into N time points (with step-size $\Delta t = \frac{T}{N}$) as follows

$$t_n = \frac{n}{N}T, \quad n = 0, \dots, N. \tag{4.1}$$

Next we define $x_i := x(t_i), \forall i$ and $\bar{x} = \big(x_0, \dots, x_N\big)$.

4.1.1 Forward Euler

This discretization leads us to approximate the derivative $x'(t_i)$ via

$$x_i' = \frac{x_{i+1} - x_i}{\Delta t},$$

with approximation error $\mathcal{O}(\Delta t)$. This is known as the forward-difference approximation. Hence, the forward difference numerical scheme for solving an ODE would be

$$x_{i+1} = x_i + \Delta t f(t_i, x_i), \quad \forall i,$$

which is known as the **forward Euler method**. It is an explicit method, i.e. the solution at x_{i+1} depends on x_i and t_i only. It can be solved iteratively by using the solution of the previous time step in the next iteration. However, it can be numerically unstable, meaning that the numerical solution can blow up for equations where the exact solution does not.

4.1.2 Backward Euler

By adding a slight modification to the forward Euler method, via

$$x_{i+1} = x_i + \Delta t f(t_{i+1}, x_{i+1}), \tag{4.2}$$

the **backward Euler method** is created. This method is implicit, the solution of x_{i+1} depends on x_{i+1} and t_{i+1} as well. An advantage of using backward Euler is numerical stability - it will not blow up. Conversely, we have to solve the numerical scheme as a whole, making the implementation more costly.

4.2 Numerical Optimal Control

We wish to approximate the solutions to CP (3.2) numerically. The functional $J[\alpha^*]$ can be approximated (numerically) via the value function $v(x,t)$ of Equation (3.8), i.e.

$$\bar{v}(\bar{x}, \Delta t) = \min_{\bar{\alpha} \in A^N} \left\{ g(x_N) + \sum_{n=0}^{N-1} L(x_n, \alpha_n) \Delta t \right\}, \tag{4.3}$$

where $\alpha_i := \alpha(t_i)$, $\bar{\alpha} = \big(\alpha_0, \ldots, \alpha_N\big)$. Since the control is discrete we can 'simplify' the set of admissible controls to be $A^N \subset \mathbb{R}^N$, where A is the control set. Note the simplifying elegance of integration in the numerical world.

4.2.1 Symplectic Euler

The CP under consideration is equivalent to Equation (3.10) in Theorem 2, the dynamical system's Hamiltonian formulation. One of properties of a Hamiltonian is that every flow is symplectic. It would therefore make sense to use a method that has this property as well. Using forward difference, we can transform Equation (3.10) into

$$\begin{cases} x_{i+1} = x_i + \Delta t \; H_\lambda(x_i, \lambda_{i+1}), & x_0 = x^0 \\ \lambda_i = \lambda_{i+1} + \Delta t \; H_x(x_i, \lambda_{i+1}), & \lambda_N = \nabla g(x_N) \end{cases}. \tag{4.4}$$

Notice that the Hamiltonian is a function of x_i and λ_{i+1}, which makes the whole scheme implicit. This numerical scheme is called the **symplectic Euler** method.

Convergence

Symplectic Euler works well for approximating the solution provided the control is a smooth function. A rigorous proof is found in [6]. The main conclusion is the following.

Theorem 4. *If (x^*, α^*) is a smooth optimal solution of (3.2) and $(\bar{x}^*, \bar{\alpha}^*)$ is the approximated solution computed using (4.4), then*

$$\|v - \bar{v}\|_{\mathcal{C}} = \mathcal{O}(\Delta t),$$

where $\|\cdot\|_{\mathcal{C}}$ is the infinity-norm for continuous functions, i.e. for $f \in \mathcal{C}(A)$,

$$\|f\|_{\mathcal{C}} = \sup_{f: A \to \mathbb{R}} |f(x)|.$$

Thus if $\Delta t \to 0$, i.e. $N \to \infty$, the numerical solutions converge to the continuous optimal solution.

4.2.2 Regularization

However, non-smooth controls exist and are relevant, as we will see in subsequent chapters. If the the optimal control α_t^*, as a function of x_t and t is non-smooth, then the Hamiltonian will also be non-smooth and the above conclusion about convergence does not hold any more. Luckily [6] gives insight on how to proceed, which we summarize in a theorem.

Theorem 5. *Let H be the corresponding non-smooth Hamiltonian, and define a regularization, H^δ, such that in the limit they are equivalent, i.e.*

$$\|H - H^\delta\|_{\mathcal{C}} = \mathcal{O}(\delta).$$

Then the solution of the regularized problem, $\bar{v}^\delta(\bar{x}, \Delta t)$, also converges to the continuous solution, i.e.

$$\|v - \bar{v}^\delta\|_{\mathcal{C}} = \mathcal{O}(\Delta t + \delta + \frac{(\delta t)^2}{\delta}).$$

This result is central to this thesis and will be used extensively to calculate the harvesting strategies (controls) for the optimal control problems formulated in Chapter 6.

5 | Mathematical Biology

After all the relevant background is introduced it is time to develop the necessary mathematical framework to study the evolution of fish species over time. Firstly, we introduce a single species model, and describe the evolution of fish densities. Then we will move on to competing species models, starting with two species and then generalizing to n-species.

5.1 Modelling Species

Biological systems are, in general, incredibly complex systems. For ecosystems, e.g. ,they consisting of numerous different species interacting amongst each-other. Each component can be of utmost importance; from the wolf - being the apex predator it can transform barren wastelands into lush forests [7], to bees being the key pollinators to our own crops [8], to name a few examples. With the help of differential equations we try to model these natural phenomena as dynamical systems; up to a certain degree of correctness. Such complicated systems can have a substantial number of parameters. Usually, the more parameters the model takes into account, the more complicated it gets and the harder it will be to compute numerical solutions, and infer relevant conclusions.

5.2 Single Species Model

We will start by modelling a single species of fish. To this extent we use the Logistic Equation, which was first formulated by Verhulst, in 1838, in his paper titled a 'Note on the Law of Population Growth' [9]. To this day it is still a widely used model, as a basis for more complex models. It is able to describe the evolution of people [10], bacteria [11] and fish species [12].

5.2.1 Logistic Equations

The Logistic Equation is a non-linear ODE of the following form

$$x_t' = rx_t\left(1 - \frac{x_t}{k}\right), \tag{5.1}$$

where $x(t)$ describes the evolution of the size the population over time. It is subject to $r > 0$, the intrinsic growth rate, and $k > 0$, the carrying capacity of the environment. Both parameters depend on the species in question and k can be thought of as the maximal sustainable population. In dynamical systems theory x_t is known as the state trajectory - it evolves over time subject to the state equation (5.1). Firstly, we note that $x < 0$ makes no sense from a populations perspective.

Equilibrium Solutions

Firstly, note that the logistic equation describes a parabola. Hence, there are two equilibrium solutions, namely

$$x_t = 0 \wedge x_t = k \quad \because \quad 0 = x_t - \frac{x_t^2}{k}.$$

The first solution corresponds to no population and the second coincides exactly with the carrying capacity. The trivial solution is unstable since, $0 < x_t < k$ the solution will increase, by virtue of $x_t' > 0$. If $x_t > k$ the solution will decrease, by virtue of $x_t' < 0$. Hence the $x_t = k$ is stable. By characterizing the equilibrium solutions we get a better understanding of the possible trajectories of the dynamical system.

Analytic Solution

Furthermore, an analytical solution can be found,

$$x(t) = \frac{kx^0}{x^0 + (k - x^0)e^{-rt}}, \tag{5.2}$$

where x^0 is the initial population - the initial condition. In Figure 5.1 various solutions for different initial conditions are given. Here $r = 1$, $K = 150$. The orange lines indicate the stable equilibrium solutions.

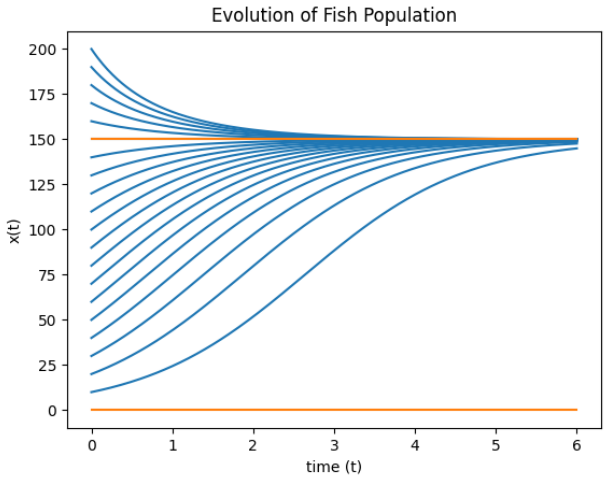

Figure 5.1: A plot showing the evolution of the logistic equation ($r = 1, K = 150$) for various starting conditions, $x^0 = \{0, 10, 20, \ldots, 200\}$. The orange lines indicate the stable equilibrium solutions.

5.3 Competing Species Model

Moving on two interacting species. For which we will make use of what is known as the Lotka-Volterra equations. Lotka first published these equations

in 1920 [13] and a few years later Volterra independently published them in another paper [14]. These equations are applied to predator-prey interactions. The system of ODE's is of the form

$$\begin{cases} x'(t) = \alpha x - \beta xy \\ y'(t) = \delta xy - \gamma y \end{cases},$$

where $x(t), y(t)$ correspond to the prey and predator population, respectively. The parameters $\alpha, \beta, \delta, \gamma > 0$ are parameters describing the predator-prey interactions.

5.3.1 Competing Lotka–Volterra Equations

The predator-prey model does not take into account the carrying capacity and growth rate, like in the logistic equation. In our analysis we will use a generalization of the predator-prey model, known as the Competing Lotka-Volterra equations, which assumes that both species 'compete' for the same resources. The model takes into account the carrying capacity and growth rate of each species directly. It is a coupled system of non-linear ODE's, as follows

$$\begin{cases} x_t' = r_1 x_t \left(1 - \dfrac{x_t + m_{12} y_t}{k_1}\right) \\ y_t' = r_2 y_t \left(1 - \dfrac{m_{21} x_t + y_t}{k_2}\right) \end{cases}, \tag{5.3}$$

where x_t, y_t correspond to two different species each with their own growth rate r_i and carrying capacity k_i. The parameters m_{ij} represents the effect species i has on species j. The terms in the numerator impact the rate of change in the population. Firstly, the size of the population of a species impacts its rate change, similar to the single species model. Secondly, the population of 'one' species is also impacted by the size of the 'other' population. If 'one' population is large it requires a lot of resources, decreasing the amount of resources available for sustaining the 'other' population.

Equilibrium Solutions

Being a system of ODE's the mathematical analysis is also more involved. It is possible to reformulate the (5.3). To this extent define the state of the dynamical system via $\boldsymbol{x}_t = (x_t, y_t)$ and then we end up with

$$\boldsymbol{x}_t' = \boldsymbol{r}\boldsymbol{x}_t^\top \left(\mathbf{1} - K^{-1} M \boldsymbol{x}\right), \tag{5.4}$$

where the parameters are summarized in vectors $\boldsymbol{r} = (r_1, r_2)$, $\boldsymbol{k} = (k_1, k_2)$ ($K = \mathrm{diag}(\boldsymbol{k})$) and a matrix $M = (m_{ij})_{ij}$, with all-ones on the diagonal. This is known as the interaction matrix. Here $\boldsymbol{r}\boldsymbol{x}^\top = \mathrm{diag}(r_1 x_t, r_2 y_t)$, this is called an outer-product. For convenience we will define $f = (f_1, f_2)$ s.t. $\boldsymbol{x}' = f(x, y)$. By analysing $\boldsymbol{x}_t' = 0$ we distinguish three cases:

- firstly, $\boldsymbol{x}_t = (0, 0)$ is the trivial solution of no populations,
- secondly, if $\boldsymbol{x}_t = (0, k_2)$, the model reduces to a single species model (r_2, k_2) for y and by symmetry $\boldsymbol{x}_t = (k_1, 0)$; a single species model (r_1, k_1) for x,

- lastly, and most interestingly, for $\boldsymbol{x}_t \neq 0$, we get

$$\mathbf{1} = K^{-1}M\boldsymbol{x} \iff \boldsymbol{x} = M^{-1}\boldsymbol{k} \iff \boldsymbol{x} = \frac{1}{1 - m_{12}m_{21}} \begin{pmatrix} K_1 - m_{12}K_2 \\ K_2 - m_{21}K_1 \end{pmatrix},$$

  the equilibrium solution being a point in the (x, y)-plane.

Before we analyze the latter further, we conclude that for the two species model, there are situation where it stabilizes to a one species model. One species dies out and the other will move to its stable equilibrium solution. This phenomenon is called **competitive exclusion**. If two species both survive, then it is called **competitive coexistence**. Additionally, notice that the equilibrium solutions do not depend on $\boldsymbol{r}$. The time it takes to reach such an equilibrium solution will however by dependent on $\boldsymbol{r}$.

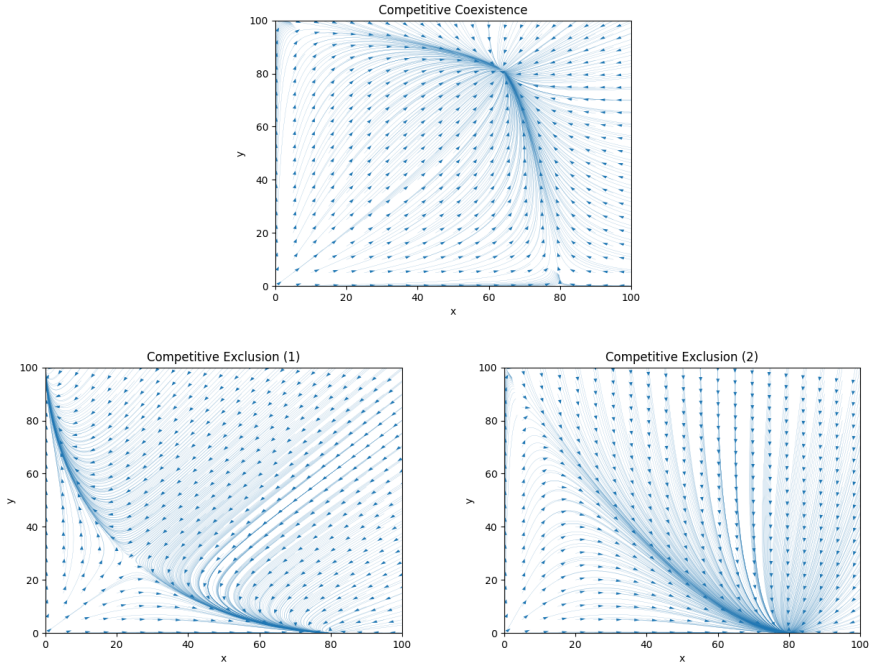

Figure 5.2: Phase space plots for different situations of two competing species. The top plot indicates competitive coexistence, i.e. a stable equilibrium solution at $\boldsymbol{x} = (64, 60)$. The Bottom line shows two instances of competitive exclusion. In the first there is an unstable equilibrium in $\boldsymbol{x} = (24, 28)$, which depending on the initial condition moves to the stable equilibrium solutions $(k_1, 0) = (80, 0)$ or $(0, k_2) = (0, 100)$. In the second a stable equilibrium solution $\boldsymbol{x} = (80, 0)$. The lines indicate solution-trajectories and the arrow indicates the direction which the solution propagates.

The third case is more difficult. We will be less rigorous and use Figure 5.2 to describe the different cases. In Figure 5.2 three different phase plots, for different transition matrices, are presented. Here

$$r = (1, 1), k = (80, 100) \quad \text{and} \quad M_1 = \begin{pmatrix} 1 & 0.2 \\ 0.3 & 1 \end{pmatrix}, M_2 = \begin{pmatrix} 1 & 2 \\ 3 & 1 \end{pmatrix}, M_3 = \begin{pmatrix} 1 & 0.2 \\ 3 & 1 \end{pmatrix},$$

where M_i corresponds to Figure i, counting from from top to bottom and left to right. The lines indicate solution-trajectories and the arrow indicates the

direction which the solution propagates. The phase plot changes drastically depending on the parameters. Let $\boldsymbol{x}^*$ be the solution to the linear system, i.e. $M\boldsymbol{x} = \boldsymbol{k}$. If $\boldsymbol{x}^* > 0$ then it describes an equilibrium point in the phase plane of the two species. Depending on M two things occur: the point is stable or unstable. If the point is stable competitive coexistence occurs, which is visible in the top plot of Figure 5.2. If the point is unstable competitive exclusion occurs, where the surviving species is dependent on the initial condition, visible in the right plot of Figure 5.2. If $x^* \vee y^* < 0$ then the solution does not lie in the first quadrant ($\boldsymbol{x} > 0$) and hence competitive exclusion occurs, like in the right plot Figure 5.2.

Stability Of Equilibrium Solutions

Mathematically one can distinguish if a point is stable or unstable by investigating the eigenvalues of the Jacobian of $f = (f_1, f_2)$ at $\boldsymbol{x}^*$, i.e. solving for λ:

$$\mathcal{J}f(\boldsymbol{x}^*) - \lambda I = \begin{pmatrix} r_1(1 - \frac{1}{k_1}(2x^* + m_{12}y^*) & -\frac{r_1}{k_1}x^* \\ -\frac{r_2}{k_2}y^* & r_2(1 - \frac{1}{k_2}(m_{21}y^* + 2x^*) \end{pmatrix} - \lambda I = 0.$$

We will not go into detail on this, since it is of minor importance. Systems where competitive exclusion occurs are less relevant, since we can (safely) assume that in nature those systems would already be in stable equilibrium. None the less, for comparing models for devising optimal fishing strategies, they can give additional insight on the behaviour of the controls.

Analytic Solutions

An analytic solution for two species model if all the (6) parameters are considered independent is yet unknown [15]. Notably, if one or more parameters are connected, then there exists a analytic solution, but this is beyond the scope of this thesis. Having said that we do not need an analytic solution, considering we can compute one numerically using the methods described in Section 4.1.

5.3.2 n-Species Competitive Model

The dynamics presented in (5.3) can easily be extended for allowing more than two species. Consider n species, $x_i,\ i = 1, \ldots, n$; each with their own growth rate r_i and carrying capacity K_i. This leads to the following system of coupled (non-linear) ODE's

$$\begin{cases} x_i'(t) = r_i x_i \left(1 - \dfrac{1}{K_i} \displaystyle\sum_{j=1}^{n} m_{ij} x_j\right) \end{cases} \quad , \quad i = 1, \ldots, n, \tag{5.5}$$

where m_{ij} represents the interaction between species i and j, $m_{ii} = 1 \forall i$. System (5.5) is identical to (5.4), where $\boldsymbol{r} = (r_1, \ldots, r_n)$, $\boldsymbol{k} = (k_1, \ldots, k_n)$, $K = \text{diag}(\boldsymbol{k})$ and $M = (m_{ij})_{ij}$, with all-ones on the diagonal.

Equilibrium Solutions

With two species it is already complicated to write out all possible equilibrium solutions. As explained the stable solution of one species is also an equilibrium solution for the two species model. For n species, we have n different equilibrium solutions in which a single species survives. For two or more species competitive existence and competitive exclusion plays a role. And every subset $k < n$ of species could be in competitive existence or exclusion. An n-species model has 2^n subsets. But there is a structured way to think about all the possibilities.

Consider a k species model, where $k < n$. If only one species dies out and the rest survives there is competitive existence between those $k - 1$ species. Hence, the case where all species survive is interesting in its own right, since it is one of the equilibrium solutions for a model with more than k species. In other words, we can use analysis of interaction of $0 < k < n$ species in evaluating all equilibrium states for the n species model. By following the same line of reasoning as in the two species models we can distinguish three types of stationary states:

- Firstly, trivial solution, $\boldsymbol{x} = 0$.
- Secondly, 'k-subset' solutions, let $I \subset \{1, \ldots, n\}$ s.t. $|I| = k$, be the indices of the k species that survive, where $0 < k < n$. This describes a k species model with parameters (r', K', M') corresponding to the fish that coexists after the rest has gone extinct, which will have the equilibrium solution $\boldsymbol{y} \in \mathbb{R}^k$ s.t.
$$M'\boldsymbol{y} = \boldsymbol{k}'.$$
- Thirdly, the 'principal' solution which corresponds to all the fish surviving. The equilibrium solution is a point $\boldsymbol{x}^* \in \mathbb{R}^n$ s.t.
$$M\boldsymbol{x}^* = \boldsymbol{k}.$$

6 | Modelling Harvesting Strategies

The previous chapter is devoted to analyzing non-controlled differential equations, which we will use to define controllable differential equations. Together with running and terminal costs we can construct an optimal control problem. We use the results of Chapter 3, PMP and the Hamiltonian Systems formulation, to derive the derivatives of the optimal control Hamiltonian necessary to implement the models. The derivation is in the sense of numerical approximation, we will therefore be less rigorous. The results are reserved for Chapter 8.

6.1 Modelling strategies

To construct an optimal control problem for the modelling of fish harvesting strategies we need to define the following:

- a state equation $\{x_t' = \tilde{f}(x_t, \alpha_t),\, x_0 = x^0\}$; related to the dynamical system in question,
- the running cost $L(x_t, \alpha_t)$ and terminal cost $g(x_T)$,
- and the time horizon, $[0, T]$.

The running cost and terminal costs are partly subjective. Consider a single species model, we need to define $\tilde{f}$, s.t.

$$x_t' = \tilde{f}(x_t, \alpha_t), \quad 0 < t < T,$$

where $\alpha : [0, T] \to \mathbb{R}$ is the control function that represents fish harvesting. Clearly, for fishing we want $\alpha(t) \geq 0, \quad \forall t$, since α_t denotes the fish harvesting intensity. But beyond that choice of α is free.

Constant Control

We start with the simplest version, fixed control $a \in \mathbb{R}$. To this extent let

$$x_t' = f(x_t) - a,$$

where $f(x_t) = rx_t(1 - \frac{x_t}{k})$, corresponding to the Logistic Equation. From an analytical perspective the logistic equation, which is a parabola, is shifted by $-a$. We can find the equilibrium solutions of $x' = 0$ via

$$f(x_t) = a \quad \therefore \quad x^2 - kx - \frac{ak}{r} = 0 \quad \therefore \quad x_\pm = \tfrac{1}{2}\left(k \pm \sqrt{k(k - 4\frac{a}{r})}\right),$$

where x_- is an unstable solution and x_+ is a stable solution of the controlled ODE. Hence, the optimal control is exactly the height of the parabola, i.e.

$$a^* = \frac{kr}{4}.$$

The state equation has one equilibrium solution $x^* = \frac{k}{2}$, which is unstable. If the fish population drops below x^* then the population will die out. This is clearly undesirable, motivating the subsequent state equations in which the size of the population at every timestep is taken into account.

6.2 Single Species

For a single species model we will apply the state equation:

$$x_t' = f(x_t) - \alpha_t x_t,$$

where f is the logistic equation, i.e. $f(x_t) = rx_t(1 - \frac{x_t}{k})$. Since f is concave in x, $\tilde{f}$ is concave as well.

Equilibrium Solution

Note that the equilibrium solution of the above state equation, i.e. the α_t that makes $x_t' = 0$ is precisely

$$\alpha_t = r(1 - \frac{x_t}{k}).$$

Then the control and state are in balance and the rate of change is zero. It would be ideal if we could find a control that balances the fish out take and increase of fish population, generating a steady state solution for the x state.

Terminal Costs

First we will define the terminal costs. From a sustainability perspective, it would make sense to have terminal costs that are related to the initial population. For one species consider,

$$g(x_T) = C(x_T - x^0)^2$$

where $C > 0$ is the weight of the terminal cost. This would penalize a control depending on the final endpoint of the trajectory. A higher C would force the solution to end up closer to x_0. Note that the terminal condition penalizes more fish as much as less fish. Since this condition is natural (and only used in the ending condition of the costate equation) we keep it fixed and only differentiate in the running costs, $L(x, \alpha)$.

6.2.1 Smooth Hamiltonian

The Hamiltonian function (3.6) depends on the state equation, $\tilde{f}(x_t, \alpha_t)$, and the running costs, $L(x_t, \alpha_t)$. If the running costs is a smooth function, then the resulting optimal control Hamiltonian is smooth as well. And we can directly apply Theorem 2 with PMP to find an optimal control. To this extent we will first consider

$$L(x_t, \alpha_t) = \tfrac{1}{2}\alpha_t^2 - c\alpha_t x_t,$$

where c is the reward for fishing an amount $\alpha_t \cdot x_t$, at time t. We assume c to be the (constant) price of fish on the market. Although it is not completely clear what the running costs means. One can think of the first term as representing

the cost of fishing with an intensity α_t. The second term relates to maximizing profit. An optimal control strategy would in this case be

$$h(x,\lambda,\alpha) = L(x_t,\alpha_t) + \lambda_t \tilde{f}(x_t,\alpha_t) = \frac{1}{2}\alpha_t^2 - \alpha_t(c+\lambda_t)x_t + \lambda_t f(x_t)$$
$$\frac{\partial h}{\partial \alpha} = \alpha_t - (c+\lambda_t)x_t = 0 \quad \therefore \quad \alpha_t^*(x,\lambda) = (c+\lambda_t)x_t.$$

For readability we set $\nu := \alpha_t^*$. Hence, the optimal control Hamiltonian is

$$H(x,\lambda) = -\tfrac{1}{2}\nu^2 + \lambda_t f(x_t),$$

which is a concave function of x_t and λ_t. Written as such to clarify the use of the fact, for a function η

$$\frac{\partial}{\partial x}\tfrac{1}{2}(\eta(x,y))^2 = \eta\frac{\partial \eta}{\partial x},$$

which will be used extensively throughout this chapter. This leads to the following (smooth) derivatives

$$\begin{cases} H_x(x,\lambda) = -\nu\nu_x + \lambda_t f'(x_t) = -(c+\lambda_t)^2 x_t + \lambda_t f'(x_t) \\ H_\lambda(x,\lambda) = -\nu\nu_x + f(x_t) = -(c+\lambda_t)x_t^2 + f(x_t) \end{cases}.$$

The optimal fishing strategy can be found through the PMP using the symplectic Euler scheme (4.4), calculating the optimal state x_i^* (and costate λ_i^*), corresponding to the optimal control α_i^*. Afterwards we compute the (numerical) optimal control via

$$\alpha_i^* = \nu(x_i^*,\lambda_i^*), \quad i = 0,\dots,N.$$

6.2.2 Non-smooth Hamiltonian

Conversely, if we consider

$$L(x_t,\alpha_t) = -c\alpha_t x_t,$$

it is completely clear what the running costs mean, namely, maximizing profit. But the optimal control Hamiltonian,

$$H(x,\lambda) = \min_{\alpha:[0,T]\to A}\{-\alpha_t(c+\lambda_t)x_t + \lambda_t f(x_t)\},$$

is linear in both λ and α and concave in x.[1] Thus, the optimal control α_t^* depends on the minimum and maximum of control set A, depending if $(c+\lambda_t)x_t$ is negative or positive. Noting that $\alpha_t \geq 0$ we can safely assume $[0,\alpha_{\max}]$, otherwise $\alpha_t^* = \infty$, which is undesirable.[2] Thus

$$H(x,\lambda) = -\alpha_{\max}((c+\lambda_t)x_t)^+ + \lambda_t f(x_t),$$

[1] A function that is linear is both convex and concave and the minimum of a concave function is again concave.

[2] Note we could also define a minimal fishing effort $\alpha_{\min} > 0$. However this would only complicate the problem further and will not yield more satisfying results than the $\alpha_{\min} = 0$ case.

where $R(\nu) := [\nu]^+ = \max\{0, \nu\}$, which is known as the ramp function. The ramp function is continuous, but not smooth, since the derivative at $\nu = 0$ is undefined. Thus the resulting optimal control Hamiltonian is non-smooth. By excluding the case $(c + \lambda_t)x_t = 0$ we could write

$$\begin{cases} H_x = -\alpha_{\max} \mathcal{H}[(c + \lambda_t)x_t](c + \lambda_t) + \lambda_t f'(x_t) \\ H_\lambda = -\alpha_{\max} \mathcal{H}[(c + \lambda_t)x_t]x_t + f(x_t) \end{cases}, \tag{6.1}$$

where

$$\mathcal{H}[\nu] = I\{\nu > 0\} = \begin{cases} 0 & \text{if } \nu > 0 \\ 1 & \text{else} \end{cases},$$

is the heaviside step function (and $I\{\cdot\}$ is the indicator function). The terms $c + \lambda_t$ and x_t are due to the chain rule. The heaviside step function is discontinuous at $x = 0$. Hence, both derivatives are discontinuous for $x_t = 0$, when the population died out, and $\lambda_t = -c$. Even though the first case would never be part of an optimal control strategy the second case is far from ideal.

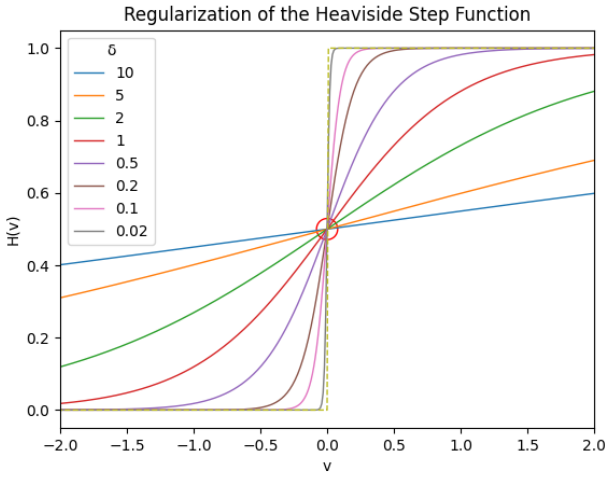

Figure 6.1: A plot showing the heaviside step function (dashed line) and the regularization for different δ. The key issue with a regularization is that there will always be values approximately $\frac{1}{2}$. This is clearly visible in the red circle.

6.2.3 Regularization

As we will see in the next chapter implementing the above formulation directly will have problematic results. We therefore apply a regularization procedure by defining a smooth Hamiltonian that in the limit is equal to the above defined non-smooth Hamiltonian. Noting that the ramp function is the only non-smooth part we could regularize this with an appropriate function, i.e. $S^\delta(\nu)$ such that

$$\|R(\nu) - S^\delta(\nu)\|_{\mathcal{C}} = \mathcal{O}(\delta).$$

This would lead to a regularized Hamiltonian, $H^\delta(x, \lambda) := -\alpha_{\max} S^\delta(\nu) + \lambda_t f(x_t)$, which we then need to differentiate in two directions. Since we only

need the derivatives in the numerical implementation we regularize the discontinuous derivatives separately. The discontinuity of both derivatives is due to the heaviside step function, which we regularize via

$$s^\delta(\nu) = \frac{1}{1+\exp(-\frac{2}{\delta}\nu)}.$$

Then

$$\|\mathcal{H}(\nu) - s^\delta(\nu)\|_\mathcal{C} = \mathcal{O}(\delta).$$

In Figure 6.1 various steps of the regularization and the heaviside step function (dashed line) are given. No matter how close to zero δ will be, there will always be values approximately $\frac{1}{2}$. This fact is pointed out by the red circle. By setting $\nu := (c+\lambda_t)x_t$ the regularization (of the derivatives) looks like

$$\begin{cases} H_x^\delta(x,\lambda) = -\alpha_{\max}\, s^\delta(\nu)\nu_\lambda + \lambda_t f'(x_t) \\ H_\lambda^\delta(x,\lambda) = -\alpha_{\max}\, s^\delta(\nu)\nu_x + f(x_t) \end{cases}. \tag{6.2}$$

A word of Note

We need to be careful approaching the regularization this way, since it is not trivial that regularizing the derivatives separately is equivalent to regularizing the Hamiltonian directly. However, note that the ramp function can also be defined as

$$R(\nu) = \int_{-\infty}^{\nu} \mathcal{H}(\eta)\,\mathrm{d}\eta,$$

we can see that

$$\frac{\partial}{\partial x}R(\nu(x,\lambda)) = \frac{\partial \nu}{\partial x}\frac{dR(\nu)}{d\nu} = \nu_x\mathcal{H}(\nu) \quad \wedge \quad \frac{\partial}{\partial \lambda}R(\nu(x,\lambda)) = \nu_\lambda\mathcal{H}(\nu).$$

Conversely, by defining $s^\delta(\nu)$ as the regularization above, and setting

$$S^\delta(\nu) := \int_{-\infty}^{\nu} s^\delta(\eta)\,\mathrm{d}\eta,$$

we can make sure that the regularized Hamiltonian and its derivatives align.

6.3 Two Competing Species

The two competing species model will be based on (5.3), which will be used in the form

$$\begin{cases} x_t' = \tilde{f}_1(x_t, y_t) \\ y_t' = \tilde{f}_2(x_t, y_t) \end{cases}.$$

For every two species model we will use the terminal condition

$$g(x_T, y_T) = C_1(x_T - x^0)^2 + C_2(y_T - y^0)^2,$$

where C_i is a parameter for how important species i is relative to the other species. The derivation is analogous to Section 6.2, we will therefore be more brief.

6.3.1 One-dimensional Control

We start by controlling the state equation(s) using a single control function, i.e.

$$\begin{cases} x_t' = f_1(x_t, y_t) - q_1\alpha_t x_t, & x_0 = x^0 \\ y_t' = f_2(x_t, y_t) - q_2\alpha_t y_t, & y_0 = y^0 \end{cases},$$

where $q_i > 0$ is the catchability of species i. The catchability is necessary in this case, since it would be restrictive to assume that both fish are caught at equal rate. This one-dimensional control mimics the situation where two fish species are harvested by a single fishing strategy - fished with the same method. For convenience define $f = (f_1, f_2)$. Although, being ad hoc we will use the term single control.

Smooth Hamiltonian

In line with Section 6.2.1 we define

$$L(x, y, \alpha) = \tfrac{1}{2}\alpha_t^2 - \alpha_t(c_1 q_1 x_t + c_2 q_2 y_t),$$

where c_i is the cost of species i. Using (3.6) we get

$$\begin{aligned} h(x, y, a) &= \tfrac{1}{2}\alpha_t^2 - \alpha_t(q_1(c_1 + \lambda_1)x_t + q_2(c_2 + \lambda_2)y_t) + \lambda_t \cdot f \\ \therefore \quad \alpha^* &= q_1(c_1 + \lambda_1)x_t + q_2(c_2 + \lambda_2)y_t, \end{aligned}$$

where $\lambda_i := (\lambda_i)_t$ are the costates and $\lambda_t := (\lambda_1, \lambda_2)$. By setting

$$\nu(x, y, \lambda) := q_1(c_1 + \lambda_1)x_t + q_2(c_2 + \lambda_2)y_t,$$

and filling (3.7) we get

$$H(x, y, \lambda) = -\tfrac{1}{2}\nu^2 + \lambda_t \cdot f.$$

with derivatives

$$\begin{cases} H_x = -\nu\nu_x + \lambda_t \cdot \nabla_x f \\ H_y = -\nu\nu_y + \lambda_t \cdot \nabla_y f \\ H_\lambda = -\nu\nu_{\lambda_1} + f_1 \\ H_\mu = -\nu\nu_{\lambda_2} + f_2 \end{cases}. \tag{6.3}$$

Regularized Hamiltonian

In line with Section 6.2.2 and 6.2.3 we define

$$L(x, y, \alpha) = -\alpha_t(c_1 q_1 x_t + c_2 q_2 y_t).$$

Set $\nu := (c_1 + \lambda_1)q_1 x_t + (c_2 + \lambda_2)q_2 y_t$ then

$$H(x, y, \lambda) = -\alpha_{\max}(\nu)^+ + \lambda \cdot f,$$

and

$$\begin{cases} H_x(x, y, \lambda) = -\alpha_{\max}\, s^\delta(\nu)\nu_x + \lambda_t \cdot \nabla_x f. \\ H_y(x, y, \lambda) = -\alpha_{\max}\, s^\delta(\nu)\nu_y + \lambda_t \cdot \nabla_y f \\ H_{\lambda_1}(x, y, \lambda) = -\alpha_{\max}\, s^\delta(\nu)\nu_{\lambda_1} + f_1 \\ H_{\lambda_2}(x, y, \lambda) = -\alpha_{\max}\, s^\delta(\nu)\nu_{\lambda_2} + f_2 \end{cases}.$$

By comparing all the previous partial derivatives we clearly see a pattern emerging.

6.3.2 Two-Dimensional Control

On the other hand, we could use a two-dimensional control functions for controlling the evolution of each species separately, i.e.

$$\begin{cases} x_t' = f_1(x_t, y_t) - \alpha_1 x_t, & x(0) = x_0 \\ y_t' = f_2(x_t, y_t) - \alpha_2 y_t, & y(0) = y_0 \end{cases},$$

where $\alpha_i := \alpha_i(t)$ and $\alpha_t := (\alpha_1, \alpha_2)$. This would model the situation where you would fish both species differently. We will use the term double control for short.

Smooth Hamiltonian

Defining

$$L(x, y, \alpha) = \tfrac{1}{2}\alpha_t \cdot \alpha_t - \alpha_t \cdot \begin{pmatrix} c_1 x_t \\ c_2 y_t \end{pmatrix},$$

then $h(x, y, \lambda, \alpha) = \tfrac{1}{2}\alpha_t \cdot \alpha_t - \alpha_t \cdot \begin{pmatrix} (c_1 + \lambda_1) x_t \\ (c_2 + \lambda_2) y_t \end{pmatrix} + \lambda_t \cdot f$ and thus

$$\frac{\partial h(x, y, \alpha)}{\partial \alpha} = \alpha)_t - \begin{pmatrix} (c_1 + \lambda_1) x_t \\ (c_2 + \lambda_2) y_t \end{pmatrix} = 0 \quad \therefore \quad \alpha_t^* = \begin{pmatrix} (c_1 + \lambda_1) x_t \\ (c_2 + \lambda_t) y_t \end{pmatrix}.$$

As expected the controls of species 1 and 2 are independent. Note that single control is the sum of the double control functions, but weighted by their catchability. By setting $\nu := (\nu_1, \nu_2) = \alpha^*$ we get $H(x, y, \lambda) = -\frac{1}{2}\nu \cdot \nu + \lambda_t \cdot f$. and

$$\begin{cases} H_x(x, y, \lambda) = -\nu_1(\nu_1)_x + \lambda_t \cdot \nabla_x f \\ H_y(x, y, \lambda) = -\nu_2(\nu_2)_y + \lambda \cdot \nabla_y f \\ H_\lambda(x, y, \lambda) = -\nu_1(\nu_1)_\lambda + f_1 \\ H_\mu(x, y, \lambda) = -\nu_2(\nu_2)_\mu + f_2 \end{cases}.$$

Regularized Hamiltonian

Define

$$L(x, y, \alpha) = -\alpha_t \cdot \begin{pmatrix} c_1 x_t \\ c_2 x_t \end{pmatrix},$$

then $h(x, y, \lambda, \alpha)$ is linear in α_i and hence

$$H(x, y, \lambda) = -\alpha_{\max} \cdot \begin{pmatrix} (\nu_1)^+ \\ (\nu_2)^+ \end{pmatrix} + \lambda \cdot f$$

where $\alpha_{\max} = (\alpha_{\max,1}, \alpha_{\max,2})$ is the maximal fishing effort for species i. By regularizing both controls separately, using s^{δ_1} and s^{δ_2} we get the following derivatives

$$\begin{cases} H_x = -\alpha_{\max,1} s^{\delta_1}(\nu_1)(\nu_1)_x + \lambda_t \cdot \nabla_x f \\ H_y = -\alpha_{\max,2} s^{\delta_2}(\nu_2)(\nu_2)_y + \lambda_t \cdot \nabla_y f \\ H_{\lambda_1} = -\alpha_{\max,1} s^{\delta_1}(\nu_1)(\nu_1)_{\lambda_1} + f_1 \\ H_{\lambda_2} = -\alpha_{\max,2} s^{\delta_2}(\nu_2)(\nu_2)_{\lambda_2} + f_2 \end{cases}.$$

6.4 n-Competing Species

We will use the system

$$x_t' = f(x_t) - \alpha_t * x_t,$$

where $f = (f_1, \dots, f_n)$ and f_i corresponds with the i^{th} equation of (5.5) and $*$ denotes element-wise multiplication, i.e.

$$a * x = \begin{pmatrix} a_1 x_1 \\ \vdots \\ a_n x_n \end{pmatrix}.$$

Smooth Hamiltonian

In the next chapter we will see that the smooth Hamiltonian is inadequate in some cases. But for the sake of completeness, we derive the derivatives needed to do the calculations. Consider

$$L(x, \alpha) = \frac{1}{2}\alpha_t \cdot \alpha_t - \alpha_t \cdot (\boldsymbol{c} * x_t),$$

where $\boldsymbol{c} = (c_1, \dots, c_n)$. This leads to $\alpha^* = (\boldsymbol{c} + \lambda_t) * x_t$ and

$$\nu := (\nu^1, \dots, \nu^n) \quad \text{s.t.} \quad \nu^i := (c_i + \lambda_i) x_i,$$

and the n derivatives,

$$\begin{cases} H_{x_i} = -\nu^i \nu^i_{x_i} + \lambda \cdot \nabla_{x_i} f \\ H_{\lambda_i} = -\nu^i \nu^i_{\lambda_i} + f_i \end{cases}, \quad i = 1, \dots, n.$$

Regularized Hamiltonian

Lastly, consider

$$L(x, \alpha) = -\alpha \cdot (c * x),$$

then the Hamiltonian is linear in α_i, hence

$$H(x, \lambda) = -\sum_{i=1}^{n} \alpha_{\max,i} \left((c_i + \lambda_i) x_i\right)^+ + \lambda \cdot f.$$

Define $\nu^i := (c_i + \lambda_i) x_i$. Applying the regularizations $s^{\delta_1}, \dots, s^{\delta_n}$ we get

$$\begin{cases} H_{x_i} = -\alpha_{\max,i} s^{\delta_i}(\nu^i) \nu^i_{x_i} + \lambda \cdot \nabla_{x_i} f \\ H_{\lambda_i} = -\alpha_{\max,i} s^{\delta_i}(\nu^i) \nu^i_{\lambda_i} + f_i \end{cases}, \quad i = 1, \dots, n.$$

7 | Experimental Setup

After all the relevant topics (and functions) have been derived it is time to construct a program for computing numerical solutions to the models from Chapter 6. In the next chapter we set out (interesting) results. But first we explain all the particulars of the discretization, methods and algorithms used to generate those results. This is followed by a concise overview of the Object Oriented (OOP) structure implemented.

7.1 Experimental Setup

For the numerical experiments we used `Python (3.9)` (in Pycharm 2021.3.2 Professional Edition) to write the code *structure*. The code was run on Mac OS X (x86_64 version 10.15.7) with a 2,5 GHz Dual-Core Intel Core i5 processor and a total RAM of 8 GB (1600 MHz DDR3).

7.2 Implementation

In Chapter 4 numerical optimal control has already been explained in a more general sense. We build onto this knowledge, start a bit trivially and moving onto more complex algorithms. We will try to distinguish between mathematical functions and functions in programming, by referring to the latter as methods. The algorithms and methods will be formulated for n-species models, in general.

7.2.1 Numerical Implementation

We use the same discretization procedure as formulated in Equation 4.1. For reference, the method that generates a discrete time interval $[T_0, T_{\text{end}}]$ of N points is denoted `generateTimeline(T_0,T_end,N)`.

Scaling

In order to make it easier to understand and relate all the different results we scaled all the state-trajectories corresponding to species, i.e.

$$\tilde{x}_i = \frac{x_i}{k_i}, \quad i = 1, \dots, n.$$

This makes sure that every fish trajectory will be between 0 and 1.

Solving Non-Linear Equations

Throughout this chapter we will make extensive use of the method[1] `fsolve(func,guess,...)`, which returns the roots a system of non-linear equations. It basically solves

$$F(Y) = 0,$$

[1] It is part of the `Scipy v1.80` module, where the `guess` is called `x_0` but this would be unnecessarily confusing.

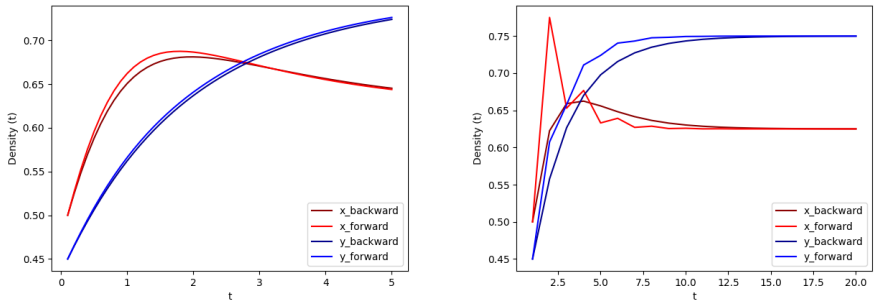

Figure 7.1: Two instances of the forward and backward Euler solutions, for Equation 5.3, where $r = [2, 1]$, $k = [1, 1]$, $x_0 = [0.5, 0.45]$ and $m_{12} = 0.5$, $m_{21} = 0.4$. In the left figure $(N, \Delta t) = (50, 0.1)$ is used; it is clear that Forward Euler 'overshoots' Backward Euler. In the right figure $(N, \Delta t) = (20, 1)$, here the instability of Forward Euler is clearly visible.

for a function $F : \mathbb{R}^n \to \mathbb{R}^m$ and a vector $Y \in \mathbb{R}^n$. The inputs of `fsolve` are a method `func` and an initial guess, `guess`. A proper initial guess is necessary for convergence. For completeness, we state that `fsolve` uses the Powell's dog leg method [16], which is an iterative optimization algorithm for the solution of non-linear least squares problems. We will not go into detail on this, since it is beyond the scope of this thesis.

Simulating ODEs

Moving on to `ForwardEuler` and `BackwardEuler`. The forward Euler method is relatively straight forward. Consider the system of ODE's, i.e. $x_t' = f(x)$ s.t. $f : \mathbb{R}^n \to \mathbb{R}^n$, then

Algorithm 1 Generate Forward Euler Solution

function ForwardEuler($\boldsymbol{x_0}, N, \Delta t$)
  $\boldsymbol{x}^0 \leftarrow \boldsymbol{x_0}$
  for $i = 1, \ldots, N$ **do**
    $\boldsymbol{x}^{i+1} \leftarrow \boldsymbol{x}^i + \Delta t \boldsymbol{f}(\boldsymbol{x}^i)$
  return $(\boldsymbol{x}^0, \ldots, \boldsymbol{x}^N)$

Here $\boldsymbol{x}^i := \boldsymbol{x}(t_i) = (x_1(t_i), \ldots, x_N(t_i))$. Backward Euler is a bit more involved, since we need to solve the implementation as a whole, for which we use the method `fsolve(f,guess)`.

Algorithm 2 Generate Backward Euler Solution

function BackwardEuler($\boldsymbol{Y}, \boldsymbol{x}_0, N, \Delta t$)
  $\boldsymbol{x_1}, \ldots, \boldsymbol{x_n} \leftarrow \text{split}(\boldsymbol{Y}, n)$
  $\boldsymbol{F}^0 \leftarrow -\boldsymbol{x}^0 + \boldsymbol{x}_0 + \Delta t \boldsymbol{f}(\boldsymbol{x}^0)$
  for $k = 1, \ldots, N$ **do**
    $\boldsymbol{F}^k \leftarrow -\boldsymbol{x}^k + \boldsymbol{x}^{k-1} + \Delta t \boldsymbol{f}(\boldsymbol{x}^k)$
  return $(\boldsymbol{F}^0, \ldots, \boldsymbol{F}^N)$
guess $\leftarrow$ ForwardEuler($\boldsymbol{x}_0, N, \Delta t$)
solution $\leftarrow$ fsolve(BackwardEuler, guess)

Note that the initial guess is the solution of the forward Euler method, which is close to the actual solution, another initial guess would also be possible. Figure 7.1 shows two instances of the forward and backward Euler solutions, for a two species state equation.

7.2.2 Numerical Optimal Control

Moving onto the optimal control part we will use similar algorithms to calculate optimal control strategies. As previously indicated we can distinguish two types of models, with a smooth Hamiltonian and with a non-smooth Hamiltonian. In the latter a regularization is used.

Smooth Hamiltonian

Firstly, for a smooth Hamiltonian the symplectic Euler method, described in Section 4.2.1. can be used 'directly'. For this we will use the following Algorithm, considering an n-species model, see Section 6.4. In order to derive convergence we first calculate the non-controlled solution, i.e. the solution if no fishing has been done on the whole time horizon. This leads to an initial guess for $\boldsymbol{x}$. To get an initial guess for the co-states $\boldsymbol{\lambda}$ we use the following algorithm, which is forward difference but backwards in time. It is based on the second equation in (3.10). Using this we can build a method generating non controlled

Algorithm 3 Generate a $\boldsymbol{\lambda}$

function GenerateLambda($\boldsymbol{x}, N, \Delta t$)
  $\boldsymbol{\lambda}^0 \leftarrow g(\boldsymbol{x}^N)$
  for $i = N-1, \dots, 1$ **do**
    $\boldsymbol{\lambda}^{i-1} \leftarrow \boldsymbol{\lambda}^i + \Delta t H_x(\boldsymbol{x}^i, \boldsymbol{\lambda}^i)$
  return $(\boldsymbol{\lambda}^0, \dots, \boldsymbol{\lambda}^N)$

solution, from x^0 N steps (of Δt) forward in time:

Algorithm 4 Generate a Non-Controlled Solution

function GenerateNonControlledSolution($x_0, N, \Delta t$)
  $\boldsymbol{x} \leftarrow$ BackwardEuler($\boldsymbol{x}_0, N, \Delta t$)
  $\boldsymbol{\lambda} \leftarrow$ GenerateLambda($\boldsymbol{x}, N, \Delta t$)
  guess$\leftarrow (\boldsymbol{x}^0, \boldsymbol{\lambda}^0, \dots, \boldsymbol{x}^N, \boldsymbol{\lambda}^N)$ **return** guess

Note that we could also have used the forward Euler approach in the above algorithms. But we have chosen for the backward Euler, since it is unconditionally stable and it does not 'overshoot' the solution. Using this procedure we can calculate an appropriate initial guess $(\boldsymbol{x}, \boldsymbol{\lambda})$ and solve the optimal control model via the algorithm:

Algorithm 5 Solving an n Species Model

function SYMPLECTICEULER($\boldsymbol{Y}, \boldsymbol{x}_0, N, \Delta t$)
  $\boldsymbol{x_1}, \boldsymbol{\lambda_1}, \ldots, \boldsymbol{x_n}, \boldsymbol{\lambda_n} \leftarrow \text{split}(\boldsymbol{Y}, 2n)$
  $N \leftarrow \text{length}(\boldsymbol{x_1})$
  $\boldsymbol{F}^0 \leftarrow -\boldsymbol{x}^0 + \boldsymbol{x}_0 + \Delta t H_\lambda(\boldsymbol{x}_0, \boldsymbol{\lambda}^0)$
  for $k = 1, \ldots, N$ **do**
    $\boldsymbol{F}^k \leftarrow -\boldsymbol{x}^k + \boldsymbol{x}^{k-1} + \Delta t H_\lambda(\boldsymbol{x}^{k-1}, \boldsymbol{\lambda}^k)$
    $\boldsymbol{G}^{k-1} \leftarrow -\boldsymbol{\lambda}^{k-1} + \boldsymbol{\lambda}^k + \Delta t H_x(\boldsymbol{x}^{k-1}, \boldsymbol{\lambda}^k)$
  $\boldsymbol{G}^N \leftarrow -\boldsymbol{\lambda}^N + \nabla g(\boldsymbol{x}^N)$
  return $(\boldsymbol{F}^0, \boldsymbol{G}^0, \ldots, \boldsymbol{F}^N, \boldsymbol{G}^N)$
guess $\leftarrow$ GenerateNonControlledSolution($\boldsymbol{x}_0, N, \Delta t$)
solution $\leftarrow$ fsolve(SymplecticEuler, guess)

Lastly, we point out that the optimal control is computed through

$$\boldsymbol{\alpha}_i = (c_i + \boldsymbol{\lambda}_i) * \boldsymbol{x}_i, \quad i = 1, \ldots, n.$$

In Experiment 7.1 we give an example of what the solution in terms of state, costate and control looks like. Technically speaking, the costate is an auxilliary function, and in the results this outcome will be omitted. But for completeness we will add it for the three subsequent experiments. Additionally, we will not draw any preliminary conclusions between the three subsequent experiments. This will be reserved for Chapter 8.

Experiment 7.1: Consider a single species of fish, with parameters

$$r = 1, x_0 = 0.50, K = 0.75,$$

with an optimal control time horizon $[0, 4]$, which we discretize with $N = 200$. For the running cost we set $c = 1$ and the weight of the terminal cost is $C = 4$. The results are visible in Figure 7.2. The left figure is the non-controlled solution, computed using Equation (5.2) and on the right the solution of the optimal control problem.

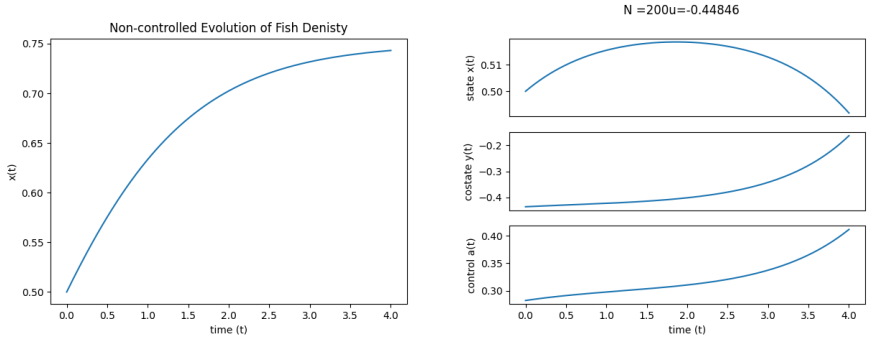

Figure 7.2: On the left a non-controlled solution computed via (5.2). On the right optimal control solution calculated via the smooth Hamiltonian of Section 6.2.1. Parameters: $r = 1, x_0 = 0.50, K = 0.75$ and $T = 4, c = 1, C = 4$.

Non-Smooth Hamiltonian

It is possible to write the Equations of (6.1) in programming language using boolean expressions, i.e.

$$((c+\lambda_i)x_i > 0) = \begin{cases} \text{True}, & \text{if} \quad (c+\lambda_i)x_i > 0, \\ \text{False}, & \text{else} \end{cases}, \; i = 0, \ldots, N,$$

where in computer bits True $\equiv$ 1 and False $\equiv$ 0, which is identical to the heaviside step function. This approach fails, because `fsolve` is not able to deal with these boolean expressions, since its derivative cannot be approximated. As is visible from the next example.

Experiment 7.2: Consider the same experiment as in Experiment 7.1, where we assume $\alpha_{max} = 1$. But with a non-smooth Hamiltonian programmed using boolean expressions. The computations reported 'RuntimeWarning: The iteration is not making good progress, as measured by the improvement from the last five Jacobian evaluations'. In other words the solution did not improve for 5 iterations, concluding that a root (x, λ) is not found. The results are visible in Figure 7.3. Clearly, $x(t)$ is not the correct evolution of the species, since the trajectory of $x(t)$ from $T = 0.9$ (the moment when there is no control applied) onward is very odd. This behaviour occurs around $\lambda = -1$, exactly where $c + \lambda = 0$.

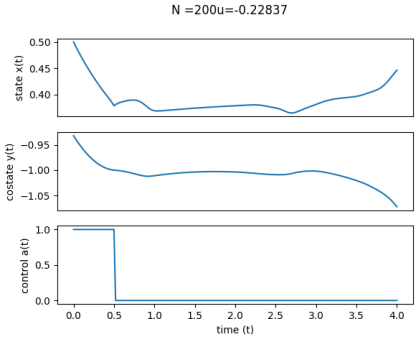

Figure 7.3: The non-optimal control solution calculated via the non-smooth Hamiltonian of Section 6.2.2, which did not converge. Parameters: $r = 1, x_0 = 0.50, K = 0.75$ and $T = 4, c = 1, C = 4$.

Regularized Hamiltonian

In case of the regularization we need to implement an algorithm that solves the problem for different δ's recursively. Consider an n-species problem, with n different regularization functions, $s^{\delta_1}, \ldots, s^{\delta_n}$, where δ_i is the regularization coefficient for species i. Then we can set a threshold, $\delta_{\min,i}$, for each regularization. For convenience we set all threshold to be the same. Beforehand we set a factor for dividing the δ's. To this extent set $\rho = 2$. Then we apply the algorithm:

Algorithm 6 Solving an n Species Model Recursively

guess $\leftarrow$ GenerateNonControlledSolution($\boldsymbol{x}_0, N, \Delta t$)
solution $\leftarrow$ fsolve(SymplecticEuler, guess)
$j \leftarrow 0$
while $\max_i \delta_i > \delta_{\min}$ **do**
  $\delta_{j \bmod(n)} \leftarrow \delta_{j \bmod(n)}/\rho$
  solution $\leftarrow$ fsolve(SymplecticEuler, solution)
  $j+ = 1$
return solution

The only difference in the `SymplecticEuler` method is that $H_x^\delta := (H_{x_1}^{\delta_1}, \ldots, H_{x_n}^{\delta_n})$ and $H_{\boldsymbol{\lambda}}^\delta$ are used instead of $H_{\boldsymbol{x}}$ and $H_{\boldsymbol{\lambda}}$, respectively. Finally, we would like to point out that the optimal control is computed through

$$\boldsymbol{\alpha}_i = \alpha_{\max,i}\, s_i^\delta(c_i + \boldsymbol{\lambda}_i * \boldsymbol{x}_i), \quad i = 1, \ldots, n.$$

Some remarks on the δ's

- The term $j \bmod(n)$ is division modulo n. Division where only the remainder is left, i.e. $j \bmod(n) = r$ where $j = qn + r : q \in \mathbb{N}$. Modulo division makes sure that at every iteration another parameter is shrunken. The algorithm can only terminate if for each species, the parameter for regularization is shrunk sufficiently, i.e. $\forall i : \delta_i < \delta_{\min}$.
- Note that if each regularization has a different threshold the algorithm would look a lot messier, since we should only shrink a δ that is higher than the threshold, and skip the other ones.
- Some argumentation of the procedure is insightful. It is possible to model every regularization with the same regularization parameter δ. This would speed up the procedure a lot.
- Conversely, a lot of iterations only changing a shrinking δ_i per iteration would lead to more stable results. Consider a case where there are 10 species, then we can either shrink one delta or shrink all 10 δ's. Every iteration would yield a slightly different results instead of one big change, which might lead the algorithm from converging to the wrong solution.

Continuing onto an experiment.

Experiment 7.3: Consider the same experiment as in Experiment 7.2, but with the regularized derivatives of Equation 6.2. We start with $\delta = 100$ and decrease it to $\delta_{\min} = 10^{-10}$.

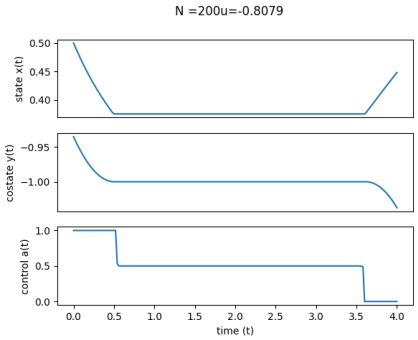

Figure 7.4: The optimal control solution calculated via the regularized Hamiltonian of Section 6.2.3 for which we employed Alogorithm 6. It corresponds to the the non controlled solution (left plot) of Figure 7.2. Parameters: $r = 1, x_0 = 0.50, K = 0.75$ and $T = 4, c = 1, C = 4$.

It is interesting to note that in Experiment 7.3 there is a period where the optimal control strategy is to fish at half the maximal rate. At first sight it may seem odd that such controls are even possible. But one needs to consider that the choice of optimal control functions is not restrictive, even more so if a non-smooth Hamiltonian is allowed. The choice of control at every discretization point is a number in $[0, \alpha_{\max}]$. One could think of a period where $\alpha = 0.5$ as a function which switches between 1 and 0 continuously. Hence, making $\alpha(t) = a$, possible at any point in time, $\forall a \in [0, \alpha_{\max}]$.

Long Time Horizon

Solving an optimal control problem on a long time horizon could lead to problems. Firstly, a long time horizon needs a large number of time points increasing the computation time. Secondly, the initial guess plays an important role in convergence. We initiate `fsolve` with a guess based on the non-controlled solution, but a non-controlled solution looks very different from a controlled solution. Therefore, it would be useful first solve a short time horizon problem, get an optimal solution, append it a little by a non controlled trajectory and use that as a next guess for the slightly longer time horizon. To this extent we will use the following algorithm.

Algorithm 7 Long Time Horizon Iteration

$N = N_0$
guess $\leftarrow$ GenerateNonControlledSolution($\boldsymbol{x}_0, N, \Delta t$)
solution $\leftarrow$ fsolve(SymplecticEuler, guess)
while $N \leq N_{\max}$ **do**
  appendage $\leftarrow$ GenerateNonControlledSolution($\boldsymbol{x}^N, dN, \Delta t$)
  guess $\leftarrow$ append(solution, appendage)
  $N+ = dN$
  solution $\leftarrow$ fsolve(SymplecticEuler, guess)
return solution

Here $\boldsymbol{x}^N$ is the last state of the trajectory, which is used as an initial condition from which the non controlled solution propagates N time steps. This algorithm is for the smooth-Hamiltonian models. The only difference using it for the regularized hamiltonians would be to add the recursive procedure of Algorithm 6 instead of `fsolve`.

7.3 Programming Implementation

A lot of time and effort is spent on the programming and implementation of the above specified numerical implementation. For reference we will shortly elaborate on the approach used. The full code can be found on Github.

7.3.1 OOP

In approaching a programming project of this size it is usually wise to think ahead how to structure the program. This alleviates a lot of the stress, error correcting and debugging later on. To approach it in a structured way, one can use Object Oriented Programming (OOP). OOP is a paradigm in which you write programs structured around classes of different objects. A class can be seen as a blue print for the object - which is an instance of a class. Classes can have subclasses which have similar structures. The 'parent' of a subclass is called the superclass. To keep it simple we will be talking about objects (of classes) mostly. An object can have different attributes and methods (or functions).

Example 3: Consider you want to built a computer game which involves training a (virtual) dog and teaching it tricks. Then one can think of the 'dog' as being a class, and 'golden retriever' (`Golden_Retriever`) is a subclass of `Dog`. All dogs have four legs (attribute) and they all bark (method), denoted `bark()`. Therefore we would define those in the class `Dog`. Additionally we will define an attribute `name`, which will be empty until we 'create' (initiate) a `Dog(name)`-object. Via what is called inheritance, the `Golden_Retriever`-class will also have a name, four legs and be able to bark. Since all golden retrievers are dogs, but not all dogs are golden retrievers, you can benefit from making this distinction. A golden retriever is able to learn tricks, but not all dogs can. So it would make sense to give the `Golden_Retriever` 'trick'-methods like, e.g. sit, fetch and spin. Since a dog needs to learn a trick to do it, we would have boolean values, like `can_sit`, `can_fetch` and `can_spin`, which will be `False`. All the above is class-structure; the blue-prints.
We can initiate an object of the class `Golden_Retriever(name)`, by giving it a name, say: Buttercup. After some training Buttercup learns 'sit', which changes the boolean `can_sit` to `True` and then the Buttercup-object can use the method `sit()`.

The concept of inheritance is driven by logic. It makes programming complex things substantially more convenient, a lot less error-prone and easier to debug. However, it takes some practice to learn how to program using OOP.

Program Class-structure

To get an idea of how we tackled the presented numerical setup an overview of the OOP-structure is given in Figure 7.5 and 7.6. The `self` variable in every method is related to the python code and is not important.[2] Two groups of classes are presented. The first `FishPool`, which represents the fish population, for one an two species. Attributes of these objects are, e.g. r, k and m_{ij}.

[2]It describes that the method, is a method of the object. Methods of classes are also possible.

Methods are, e.g., functions like the logistic equation or being able to plot ODE solutions.

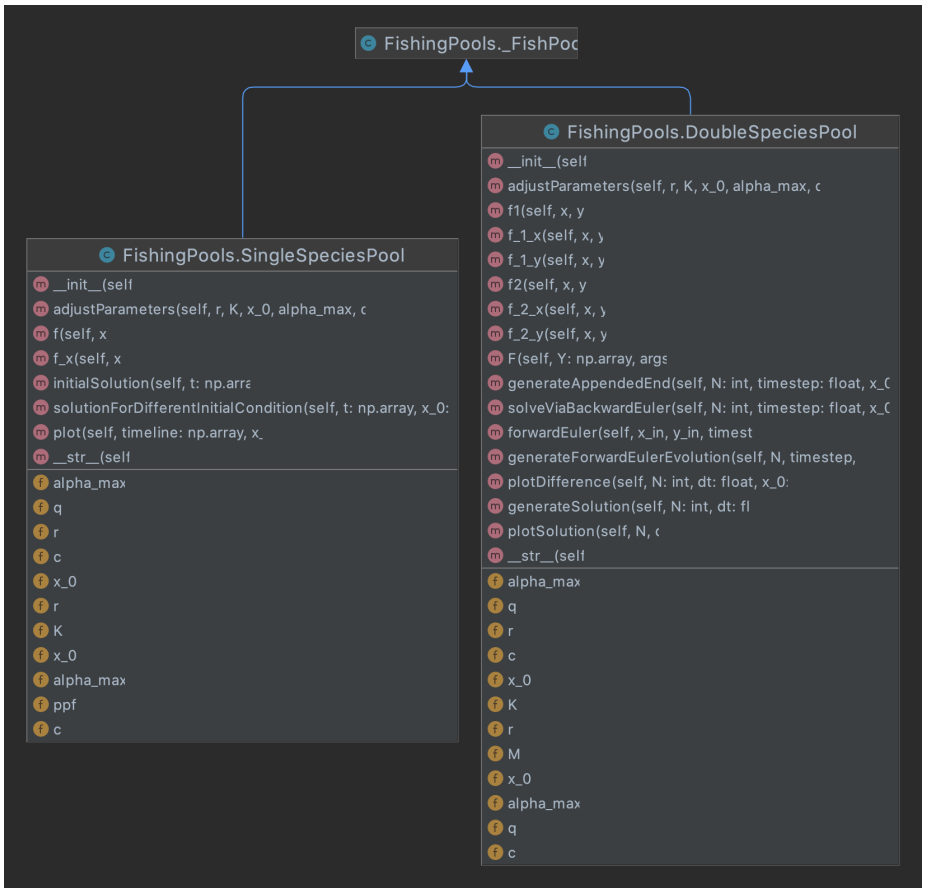

Figure 7.5: A schematic overview of the user-defined Classes for the one and two species models. Blue dots indicate Classes, red dots indicate methods and yellow dots attributes. The arrows relate to inheritance, where the Class the arrow points at is the parent of the subclass.

Secondly, we have an `OptimalControl` 'package', which has classes related to optimal control. An Optimal Control problem is a general class. By initializing it with a `FishPool` we turn it into a `FishingProblem`. All the main computation is done inside these classes and subclasses, like `solve` (from `FishingProblem`). The smooth hamiltonian-cases use this method to solve the Fishing Problem for certain initial conditions, e.g., x_0. Or in case of the regularized Hamiltonian it uses the method `solveRecursively()`, which makes use of the method `solve()` (from the superclass) to solve for different δ (until $\delta < \delta_{\min}$). Furthermore, the superclass defines methods like H_x, H_λ used in the Symplectic Euler scheme. This can be done by noting the similarity of Equation 6.1 and 6.2 for a single species, e.g., and then defining distinct methods for computing the optimal α (in the subclasses).

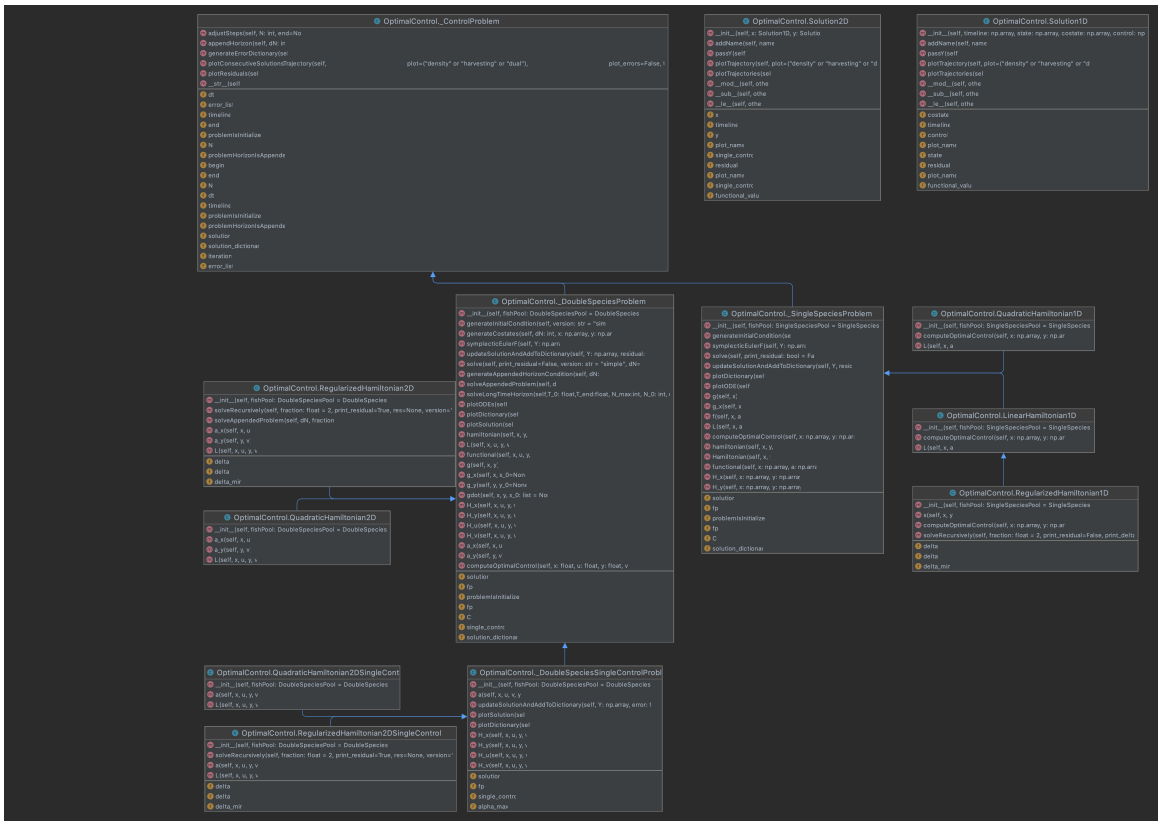

Figure 7.6: A schematic overview of two user-defined Classes related to the Optimal Control Problem, where the input is a Object (from the class) of `FishPools` (Figure 7.5). Blue dots indicate Classes, red dots indicate methods and yellow dots attributes. The arrows relate to inheritance, where the Class the arrow points at is the parent of the subclass.

Lastly, we point out two classes, which are not connected by inheritance, but by embedment. The `Solution1D` is initiated with $x^{1:N}, \lambda^{1:N}, \alpha^{1:N}$ and a timeline $0 = t_0, \ldots, t_N = T$. We elaborate on one method, `passY()`, which passes

$$Y = (x_1, \lambda_1, \ldots, x_N, \lambda_N).$$

The `Solution2D`-object is initiated with two `Solution1D`-objects, for species 1 and 2. And it also has a `passY()` method, denoted `Solution2D.passY()`, which uses the `Solution1D.passY()` methods of both to generate

$$Y = (Y_1, Y_2),$$

where Y_i is the output of `passY()` for species i.

8 Results

After a simple explanation on how the whole set-up is implemented, we set out several results for different experiments. Each experiment considers a particular model. The numbering of figures corresponds with the numbering of experiments. Due to time constraints no worthy results have been computed for the n-species models. Every plot is titled with the size of their discretization, N, and the functional value u, computed using (4.3). We will therefore omit those details from the text.

8.1 Comparison

Each model can have several solutions, which are not necessarily optimal. A solution consists of a triplet $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\alpha}})$ and a functional value $\bar{v}(\bar{\boldsymbol{x}}, \Delta t)$, where $\bar{\boldsymbol{x}} = (\boldsymbol{x}^0, \ldots, \boldsymbol{x}^N)$ with $\boldsymbol{x}^i \in \mathbb{R}^n$ the state (of n species) at time point t_i. We point out some additional remarks.

Comparison of Solutions

Firstly, comparison between solutions of the same model is relatively straightforward. A solution a is 'better', or more optimal, than another other solution b if the value of that functional is lower, i.e. if $\bar{u}_a \leq \bar{u}_b$. Secondly, a solution is more sustainable than another solution if the value of $g(x(T))$ is lower, i.e. $0 \leq g(x_a^N) \leq g(x_b^N)$.

Comparison of Models

Comparison between models is a lot more involved. If two models have the same functional (objective function) they can be compared, in the sense, that their solutions can be better (or worse).
Conversely, if the functional of two models are different in either the running cost or the terminal costs, then we cannot claim one solution is better based on the value of both functionals. However, we can still compare two solutions (of two different models) in the sense sustainability.
Usually, if two models have the same running costs but different terminal costs, then the solution with the heavier weighted terminal costs will be more sustainable then the other, yet (probably) worse than the other solution. For single species models this is trivial, but for two species it is not.

Example 4: Consider two different species, with initial population $(x^0, y^0) = (10, 50)$. For two solutions their end-time populations are

$$(x_T, y_T) = (13, 51) \quad \text{and} \quad (11, 53).$$

This makes the differences $(3, 1)$ and $(1, 3)$, but we cannot directly say which is more sustainable. It is like comparing apples and oranges. Maybe one species is a lot bigger than the other, a lot of those species would mean more biomass, but this is not (directly) a measure for sustainability. What is better for the planet is best left to the biologist, we only wish to address the complexness of comparison.

Lastly, if two models have different running costs then we can not compare the solutions since they are optimized for different situations. Alas, we cannot compare the solutions of a smooth Hamiltonian-model with its corresponding non-smooth (regularized) Hamiltonian-model, in the sense of optimality.

8.2 Single Species

We first start with the single species models, for details see Section 6.2 and its implementation Section 7.2.2.

8.2.1 Smooth Hamiltonian

In this first experiment we will vary only the terminal cost, to see how the optimal control is affected.

Experiment 8.1 (Varying Terminal Costs 1)**:** Consider 7.1, but with $C = 1$, $C = 4$ and $C = 10$ instead. The results are summarized in Figure 8.1, where the left, middle and right correspond to $C = 1, 4$ and 10, respectively. The functional values for all three models are very close and the trajectory x_t changes accordingly.

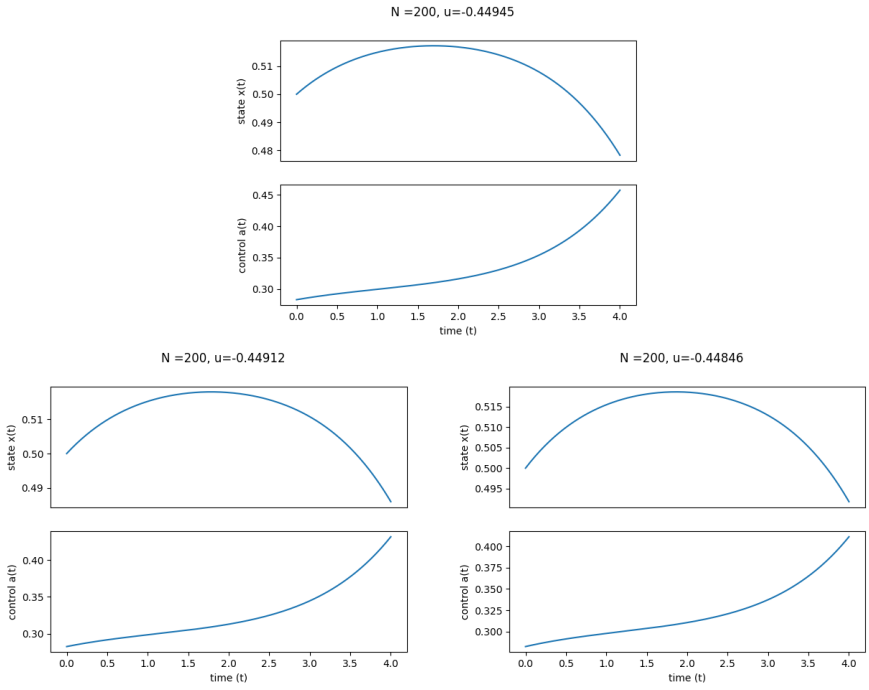

Figure 8.1: Optimal control solution calculated via the smooth Hamiltonian of Section 6.2.1 for three different weights (C) of the terminal costs. Parameters: $r = 1, x_0 = 0.50, K = 0.75$ and $T = 4, c = 1, C = 1, 4, 10$.

Lastly, we investigate the model for long time horizons:

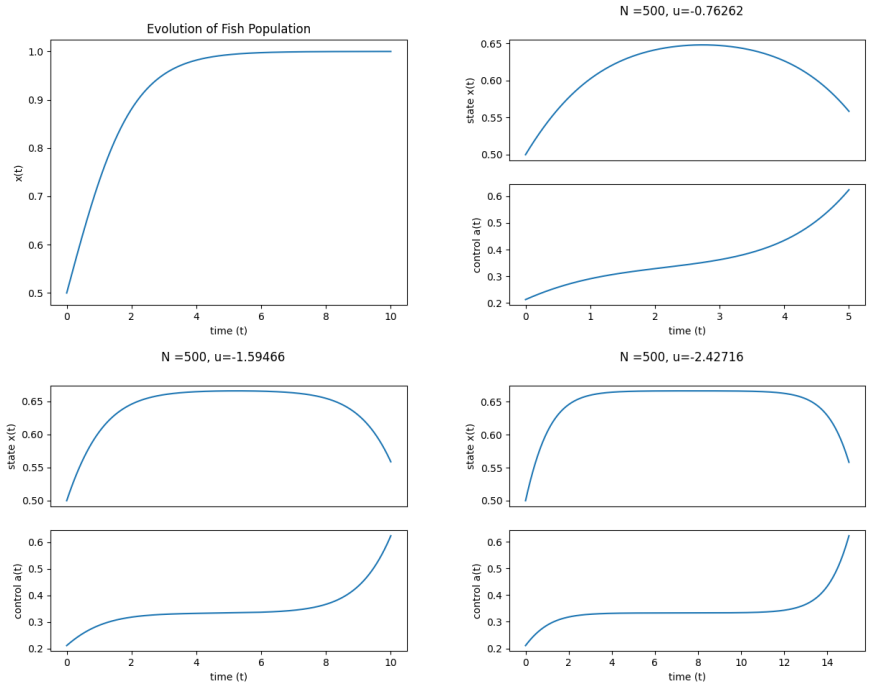

Figure 8.2: Optimal control solution calculated via the smooth Hamiltonian of Section 6.2.1 for three different time horizons. Parameters: $r = 1, x_0 = 0.50, K = 0.75$ and $T = 5, 10, 15\,, c = 1, C = 1$.

Experiment 8.2 (Long Time Horizon 1)**:** Consider a single species model with $r = 1, k = 1, x_0 = 0.5$. and optimal control parameters $T = 5, 10, 15, c = 1, C = 1$. The results are in Figure 8.2, where the upper-left is the non-controlled evolution. For long time horizon, the solution moves to a steady state for the fish population, stabilizing it until the end time comes close.

All in all the smooth control seems to work quite well: it can fish in such a way that it forces the optimal trajectory closer to the initial population.

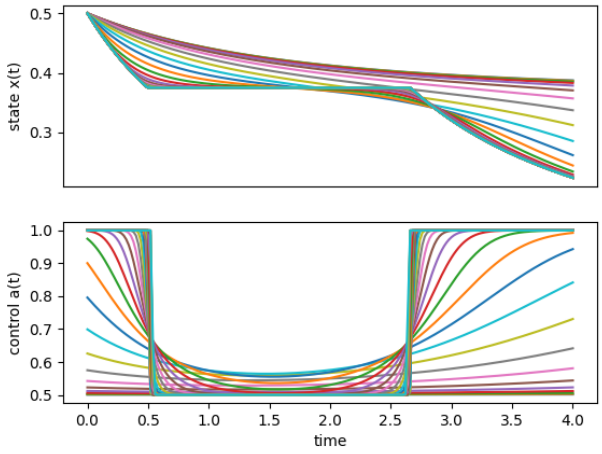

Figure 8.3: Optimal control solution calculated via the regularized Hamiltonian of Section 6.2.3, for different δ. The final solution is more clearly visible in Figure 8.4. Parameters: $r = 1, x_0 = 0.50, K = 0.75$ and $T = 4, c = 1, C = 4$.

8.2.2 Regularized Hamiltonian

For details on the regularized method see Section 6.2.3 and its implementation 7.2.2. We set $\delta = 100$ and $\delta_{\min} = 10^{-14}$. To get an idea how the solution changes for different δ we set up the following experiment:

Experiment 8.3 (Recursive Solutions)**:** Consider Experiment 7.3, but with $C = 1$. In Figure 8.3 various solutions for different regularizations are visible - each iteration the solution changes a little forcing the solution into a non-smooth form. The final form is more clearly visible in the top of Figure 8.4.

Next we varificate, that the regularized model is able to adjust to a different weight (C) of the terminal costs. This is done similar to Experiment 8.1.

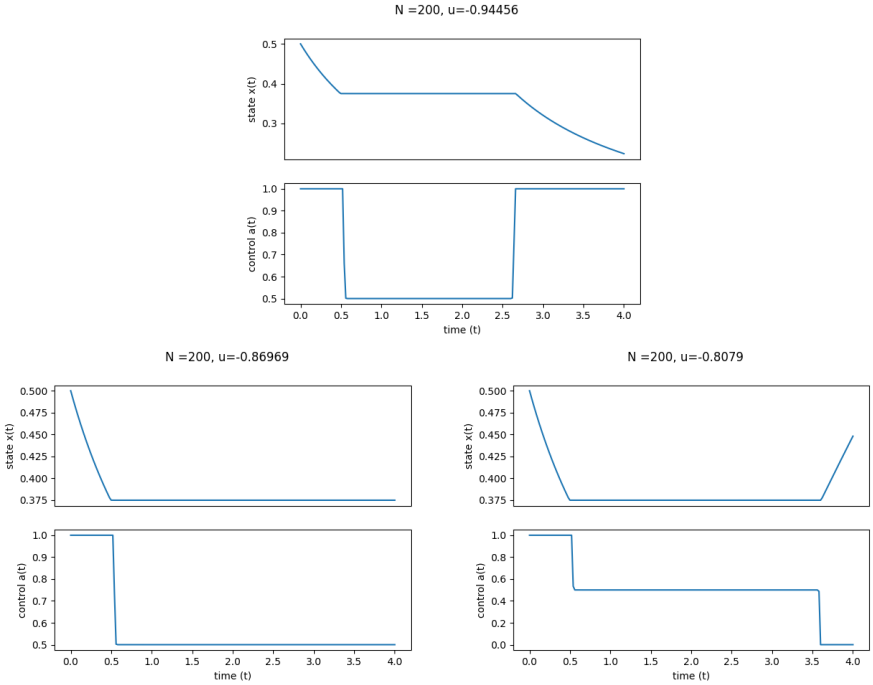

Figure 8.4: Optimal control solutions calculated via the regularized Hamiltonian of Section 6.2.3, for different C. Parameters: $r = 1, x_0 = 0.50, K = 0.75$ and $T = 4, c = 1, C = 1, 4, 10$.

Experiment 8.4 (Varying Terminal Costs 2)**:** Consider the same situation as in 7.3. The results can be seen in Figure 8.4, where the top, left, right corresponds to $C = 1, 4$ and 10, respectively. Again the functional values for all three models and the state trajectories change accordingly. Additionally, note that in all three models the optimal state trajectory is in equilibrium, for an extended period of time, as is the optimal control.

By comparing Experiment 8.1 and 8.4 we draw three conclusions.

- Firstly, the optimal control trajectory of regularized Hamiltonian changes more drastically, whereas the former increases more rapidly in the end if C gets smaller.

- Secondly, both models always produce terminal conditions which are lower than the initial population. It is rewarding to let the fish population decrease and stop fishing near the end-time to let the population recuperate. But for such a trajectory the end condition will always be less than the initial condition, i.e. $x_T < x^0$. Otherwise the optimal control could have kept fishing because the terminal cost does not distinguish between more or less than the initial population.

- Thirdly, the regularized Hamiltonian shows extended periods where both state is in an equilibrium solution, due to the optimal control being constant. Conversely, the smooth Hamiltonian does not show such a stabilizing behaviour between the state and the control in a short time horizon.

In the third experiment we will extend the time horizon to see how the optimal control adjusts to this change.

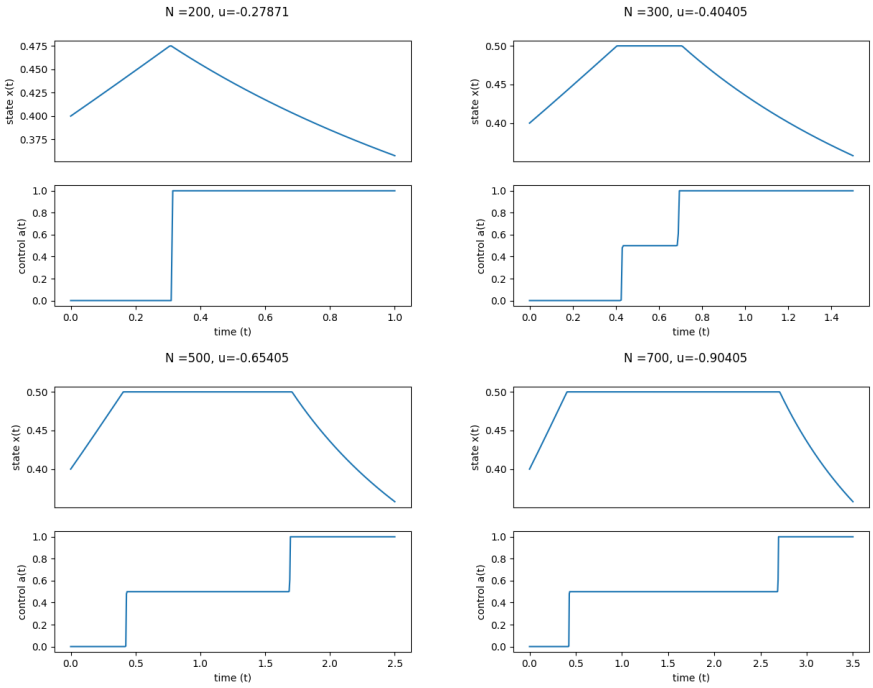

Figure 8.5: Optimal control solutions calculated via the regularized Hamiltonian of Section 6.2.3, for different (short) time horizons, $T = 1, 1.5, 2.5$ and 3.5. Parameters: $r = 1, x_0 = 0.50, K = 0.75$ and $c = 1, C = 4$.

Experiment 8.5 (Short Time Horizon)**:** Consider a single species model with $r = 1, k = 1, \alpha_{\max} = 1, x_0 = 0.4$ and optimal control parameters $c = 1, C = 10$, we vary

$$T = 1, 1.5, 2.5 \text{ and } 3.5.$$

The results are in Figure 8.5, from which it is clear that all the optimal controls aim for the same steady state. The longer the time horizon the longer the control keeps the state trajectory in steady state. Once the end time comes close the control forces the trajectory back to a lower condition. If we set

$C < 10$ then the state trajectory would have ended closer to the initial solution and the steady state would be maintained for a longer duration. Interestingly the optimal control stabilizes for short time periods.

Next we propose an experiment for long time horizons identical to the situation in Experiment 7.3.

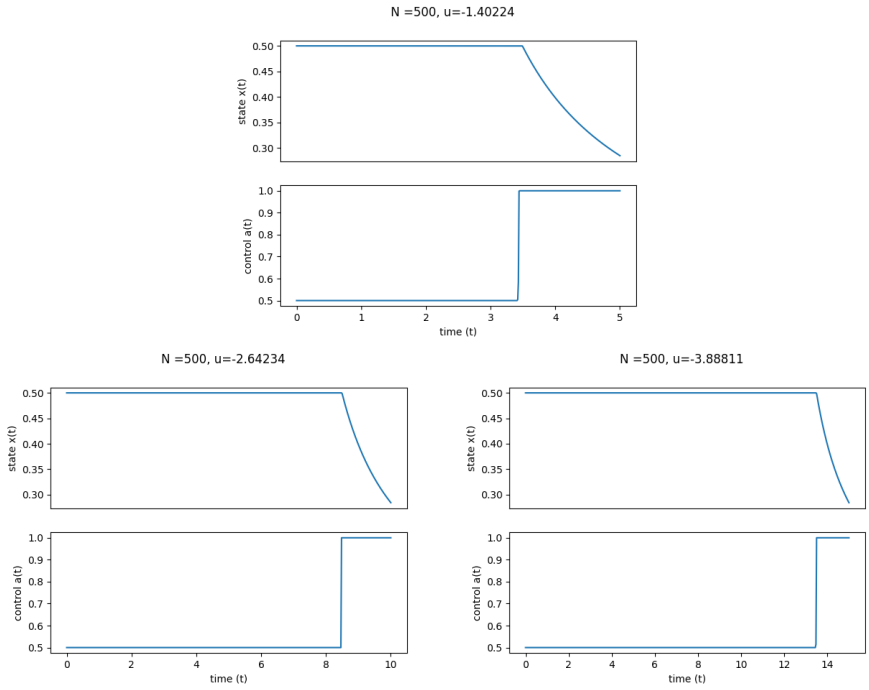

Figure 8.6: Optimal control solutions calculated via the regularized Hamiltonian of Section 6.2.3, for different (short) time horizons, $T = 5, 10$ and 15. Parameters: $r = 1, x_0 = 0.50, K = 0.75$ and $c = 1, C = 4$.

Experiment 8.6 (Long Time Horizon 2)**:** Consider Experiment 7.3 but with regularized Hamiltonians. The results are visible in Figure 8.6. The non-controlled solution and optimal control solutions using the smooth Hamiltonian is visible in Figure 8.2. Clearly the two steady state solutions do not align.

Note that in this last experiment $C = 1$. If $C > 1$ then the end-time population would be closer to the initial population. Thus from both Experiment 8.4 and 8.6 we can conclude that setting $C = 1$ is not restrictive and would not change the steady state solution. In other words, the optimal control is only dependent on the penalization of the terminal cost, once the end-time is close. This fact also holds for the smooth Hamiltonian case, but is less visible.

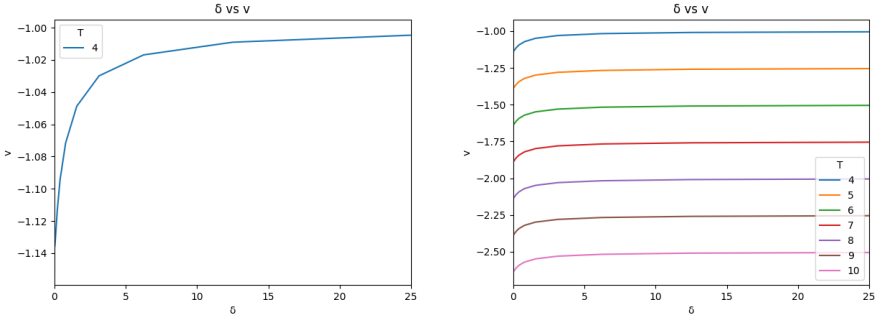

Figure 8.7: Plots showing the dependence of the functional value $\bar{v}$ on the regularization parameter δ. Values are based on Experiment 7.3, but for different time horizons $T = 4, \dots, 10$. Left is an enlarged plot of the case $T = 4$.

Finally, we will quantitatively compare how the δ parameter influences the functional value $\bar{v}$.

Experiment 8.7 (δ vs $\bar{v}$)**:** We consider the same situation as Experiment 7.1 and vary the time horizon, from $T = 4, \dots, 10$. The results are visible in Figure 8.7. On the left plot we zoomed in on the case $T = 4$. And on the right all T's are visible.

8.2.3 Conclusion

By comparing all previous experiments, we conclude that both approaches generate appropriate optimal controls. For long time horizon they both create steady state trajectories, because the control stabilizes the state trajectory. In the steady state the control is also constant. Note that the steady states of both approaches differs. But from a sustainability perspective they are both appropriate, since extending the steady state for larger time horizons would yield optimal controls as well.
Conversely, the results for short time horizons differ. The regularized Hamiltonian approach is able to stabilizes those systems quickly. But, the smooth Hamiltonian is not able to stabilize those systems, due to being less flexible, in the sense that the optimal control trajectory cannot change drastically (become non-smooth).

8.3 Two Competing Species

Moving onto the two species models we use the models explained in Section 6.3. In this situation competitive coexistence and competitive exclusion plays an important role, for details we refer to Section 5.2.1. We will therefore treat both results separately. We will discuss one-dimensional and two-dimensional control in parallel. To be able to compare both results we need to set $q = (1, 1)$, since then the functionals coincide.

8.3.1 Competitive Coexistence

To get an idea what both controls will look like for a system with species that coexist.

Smooth Hamiltonian

To get an idea what both controls will look like for the same dynamical system we use similar specifications for single control and double control with a smooth Hamiltonian using two different experiments.

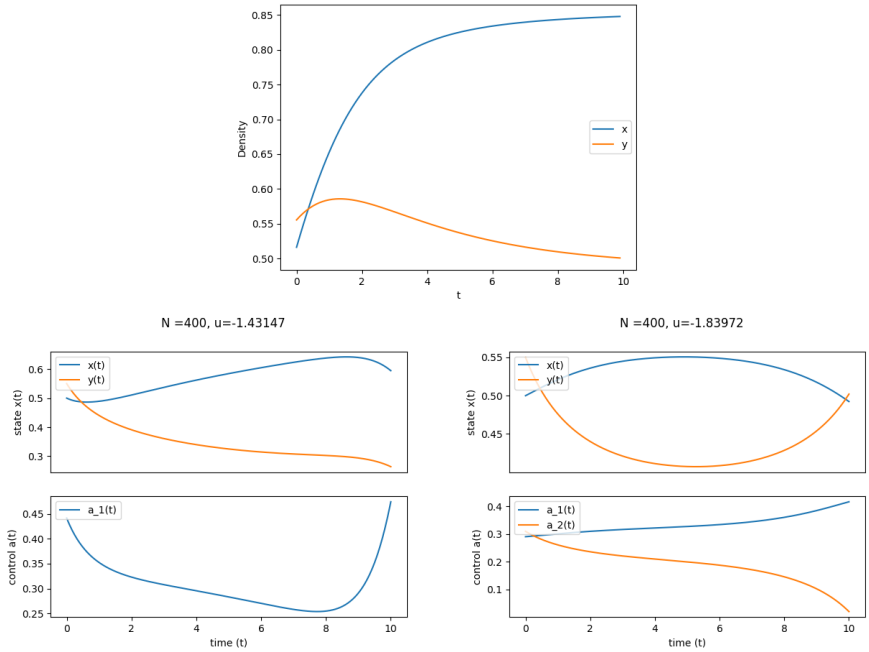

Figure 8.8: Optimal control solution calculated via the smooth Hamiltonian of Section 6.3 for the one-dimensional and two-dimensional control situations. On the top the related non-controlled solution is visible. Parameters: $r = (1,1), x_0 = (0.5, 0.55), K = (1, 0.75), m_{12} = 0.3, m_{21} = 0.6$ and $T = 10, c = (1,1), C = (10,10)$.

Experiment 8.8 (Competitive Coexistence 1)**:** Consider a two species model with parameters

$$r = (1,1), k = (1, 0.75), M = \begin{pmatrix} 1 & 0.3 \\ 0.6 & 1 \end{pmatrix}, x^0 = (0.5, 0.55).$$

The interaction between the species is not significant and both species coexist. Let the optimal control parameters be $T = 8, c = (1,1)$ and $C = (10,10)$, which coincides with T. The results are visible in Figure 8.8. The top, left, right figures are the non-controlled solution, single control and double control solutions, respectively. Both approaches seem to work to some extent. Obviously, is it more difficult for the single control to balance both end time populations with the initial population. A trade-off, favouring x above y, is visible. The double control is clearly more appropriate from a sustainability perspective.

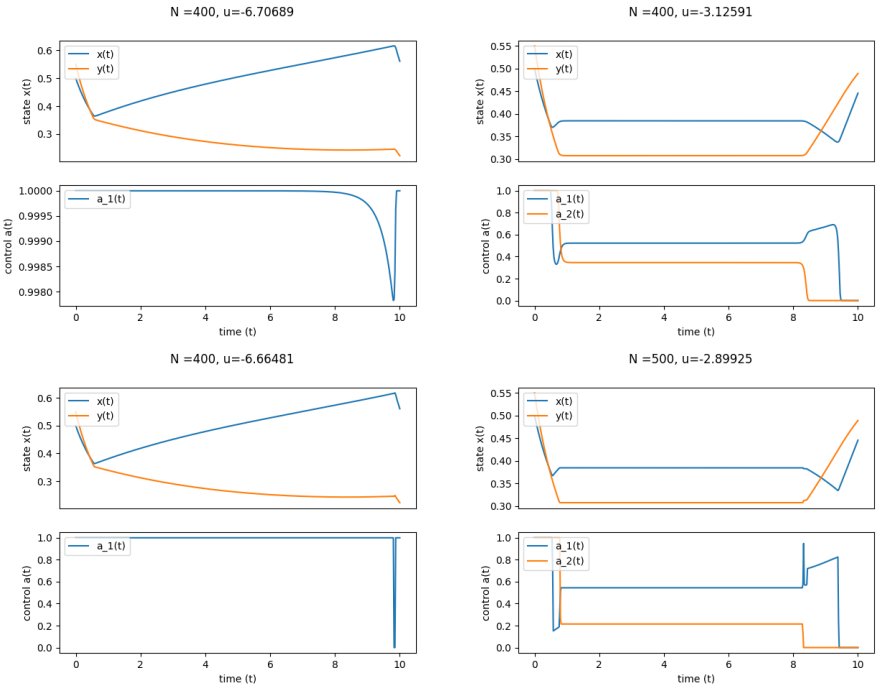

Figure 8.9: Optimal control solution calculated via the regularized Hamiltonian of Section 6.3 for the one-dimensional and two-dimensional control situations.The related non-controlled solution is visible in Figure 8.8. The bottom row corresponds a solution with a lower threshold. Parameters: $r = (1, 1), x_0 = (0.5, 0.55), K = (1, 0.75), m_{12} = 0.3, m_{21} = 0.6$ and $T = 10, c = (1, 1), C = (10, 10)$.

Regularized Hamiltonian

Next, we do the same experiment but with the regularized Hamiltonian approach:

Experiment 8.9 (Competitive Coexistence 2)**:** Consider Experiment 8.8 but with the non-smooth Hamiltonian. Let $\alpha_{\max} = 1$ and

$$\delta_{\min} = 10^{-4} \text{ and } 10^{-8},$$

for the single control model and $\alpha_{\max} = (1, 1)$,

$$\delta_{\min} = (10^{-4}, 10^{-4}) \text{ and } (10^{-8}, 10^{-8}),$$

for the double control model. The results are visible in Figure 8.9, where the bottom row is the smallest threshold. The left and right figures correspond to single and double controls, respectively. A smaller threshold forces the solutions to be more piecewise linear, whereas the larger threshold allows for more smooth transitions (considering the control trajectories). Interestingly in the double control case the control forces a steady state on both species, which clearly is desirable. Such a steady state could be extrapolated to form optimal controls for larger time horizons. All states appear similar, even for 'different' optimal controls. From the functional values we conclude that the resulting state trajectories are almost identical. Note that we used $N = 500$ in the last plot, since the algorithm did not converge with $N = 400$. The number of time points was insufficient for the large time horizon.

If we compare Experiment 8.8 with 8.9, we see that for the single control the resulting state trajectory is relatively similar, even thought the controls are not similar.[1]

8.3.2 Competitive Exclusion

Moving on to competitive exclusion, which is a more involved situation. One species is bound to die out. From a modelling perspective it is interesting to see how the model behaves on for these conditions. We will discuss one-dimensional and two-dimensional control in parallel. To compare we again set $q = (1, 1)$.

Smooth Hamiltonian

We do an experiment similar Experiment 8.8 from the previous section.

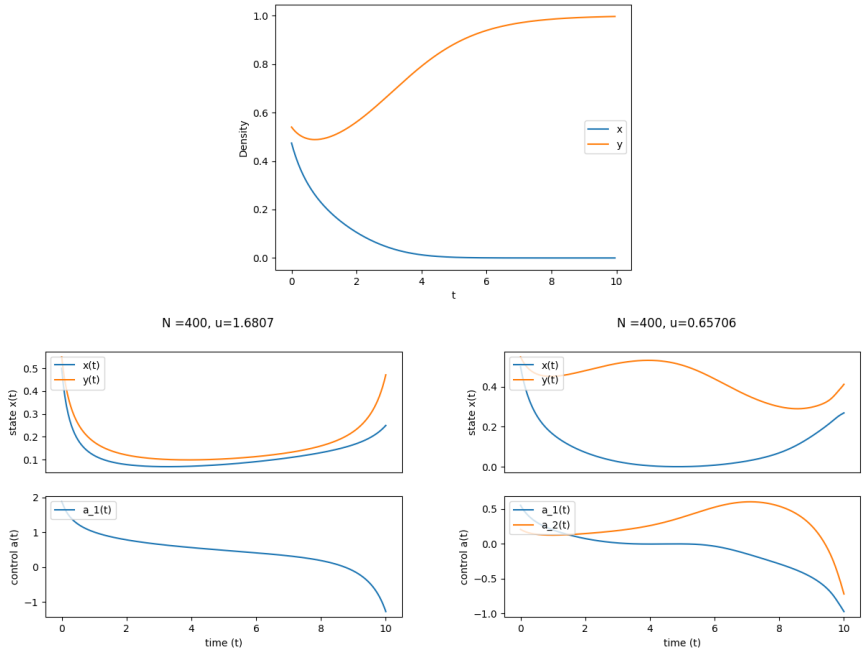

Figure 8.10: Optimal control solution calculated via the smooth Hamiltonian of Section 6.3 for the one-dimensional and two-dimensional control situations. On the top the related non-controlled solution is visible. Parameters: $r = (1, 1), x_0 = (0.5, 0.55), K = (1, 0.75), m_{12} = 3, m_{21} = 2$ and $T = 10, c = (1, 1), C = (10, 10)$.

Experiment 8.10 (Competitive Exclusion 1)**:** Consider Experiment 8.8 but with

$$r = (0.5, 1), M = \begin{pmatrix} 1 & 3 \\ 2 & 1 \end{pmatrix},$$

the rest of the parameters are identical.The results are visible in Figure 8.10. The top, left and right figures are the non-controlled solution, single control

[1] This comparison is justified since the resulting state trajectory is dependent on the controls only.

and double control solutions, respectively. The results are undesirable, since for both cases the optimal control is negative for a substantial part in the end. This clearly is not an acceptable fishing strategy.

One could try to produce appropriate controls based on the control in Experiment 8.10 by afterwards setting $\tilde{\alpha} = (\alpha)^+$, i.e. $\tilde{\alpha}_i = (\alpha_i)^+$ for $i = 1, 2$. However, this would also affect the state trajectory, hence, changing the whole solution. But we cannot afterwards be sure that we are in an optimal control any-more. This clearly is far from ideal.

Regularized Hamiltonian

Next, we do the same experiment but with the regularized Hamiltonian approach.

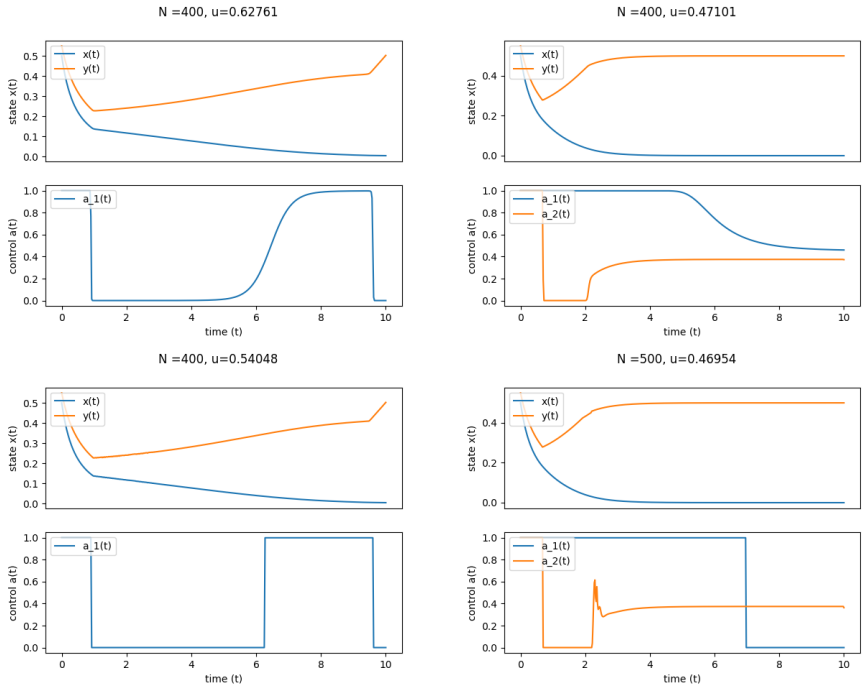

Figure 8.11: Optimal control solution calculated via the regularized Hamiltonian of Section 6.3 for the one-dimensional and two-dimensional control situations.The related non-controlled solution is visible in Figure 8.11. The bottom row corresponds a solution with a lower threshold. Parameters: $r = (1, 1), x_0 = (0.5, 0.55), K = (1, 0.75), m_{12} = 3, m_{21} = 2$ and $T = 10, c = (1, 1), C = (10, 10)$.

Experiment 8.11 (Competitive Exclusion 2)**:** Consider Experiment 8.10 but with a regularized Hamiltonian. Let $\alpha_{\max} = 1$ and

$$\delta_{\min} = 10^{-4} \text{ and } 10^{-8},$$

for the single control model and $\alpha_{\max} = (1, 1)$,

$$\delta_{\min} = (10^{-4}, 10^{-4}) \text{ and } (10^{-8}, 10^{-8}),$$

for the double control model. The results are visible in Figure 8.11, where the bottom row is the lowest threshold. The left and right figures correspond to single and double controls, respectively. Again, the threshold plays a role in how smooth the optimal control is and the resulting state trajectories are almost identical. Note that both the single control and double control are able to deal with competitive extinction, to some degree. One could suggest that the x-control (on the right) is inappropriate, but remembering it means fishing intensity, and is multiplied by $x_t$which is zero, this makes sense.

8.3.3 Conclusion

We conclude that in the two species models the smooth Hamiltonian is inadequate to find acceptable optimal controls. It makes it possible to boost the state trajectory via negative controls, i.e. increase the population. This happens at least in cases where competitive exclusion plays a role. Furthermore, the one-dimensional control is less flexible than the two-dimensional control, in the sense that it makes the dynamical system less controllable. The regularized Hamiltonian with two-dimensional control produces state trajectories that for most of the duration are in steady state, similar to the steady states in Experiment 8.5 and 8.6.

n Species

Using these results we could predict what the conclusions for the n-species models will be. However, we have to be careful projecting these conclusion drawn for 2 species, onto the n-species. For the single species the smooth Hamiltonian is adequate, but it has undesirable results in 2 species models. Having said that we can safely assume the smooth Hamiltonian approach would break down for $n \geq 2$. It is also likely that the single control models will be inadequate, consideration it only works to some extent in $n = 2$. It would be even harder to balance more than two species with just a single control. With more species competitive exclusion and competitive coexistence still place a role. Hence, in the n-dimensional control regularized Hamiltonian approach we anticipate similar behaviour for both the competitive exclusion and competitive coexistence. Thus, the model will (probably) behave in the same way for competitive coexistence, that is produce a state trajectory that is constant for each of the n-species. And the competitive exclusion model for which $k < n$ species survive will have k species in constant steady state, each controlled by their related control function. Additionally, the population of $n - k$ species dies will be equal to zero, even if the control functions are not identically zero, making the controls irrelevant.

8.4 Long Time Horizon

It is worthy to note that all the above results are computed by direct implementation of the algorithms. From the previous experiments it is clear that the non-controlled solutions and the controlled solutions are not similar, in the sense that the trajectories differ substantially. However, we use this non-controlled solution as a first guess in the direct implementation. But this may

not be ideal, and as described in detail in Section 7.2.2, it may lead to problems with convergence. It may happen that solving directly leads to worse optimal control. Hence, we devised Algorithm 7 to solve a problem with a long horizon. To check if the result differs we do an experiment where we apply the `LongTimeHorizon`$(T_0, N_0, N_{\max}, dN)$ to the regularized Hamiltonian approach with two-dimensional controls from Experiment 8.9 and 8.9.

Experiment 8.12 (Long Time Horizon vs Direct Implementation)**:** Consider Experiment 8.9 and 8.9 in a regularized Hamiltonian setting with two-dimensional controls. For the `LongTimeHorizon`-method we set

$$T_0 = 2.5\,, N_0 = 125\,, N_{\max} = 500\,, dN = 25.$$

Then $\Delta t = \frac{2.5}{125} = 0.02$, $T = N_{\max} * \Delta t = 10.0$ and the algorithm will take 15 steps of size dN. Additionally, we speeded up the `solveRecursively`-method, by shrinking δ_1 and δ_2 at each iteration (instead of seperately). The results are visible in Figure 8.12. By comparison of functional values we can see the left figure is actually quite close to the solution computed via direct implementation. The same cannot be said for the right figure. There the direct implementation clearly produced a control which is better, in the sense of optimality.

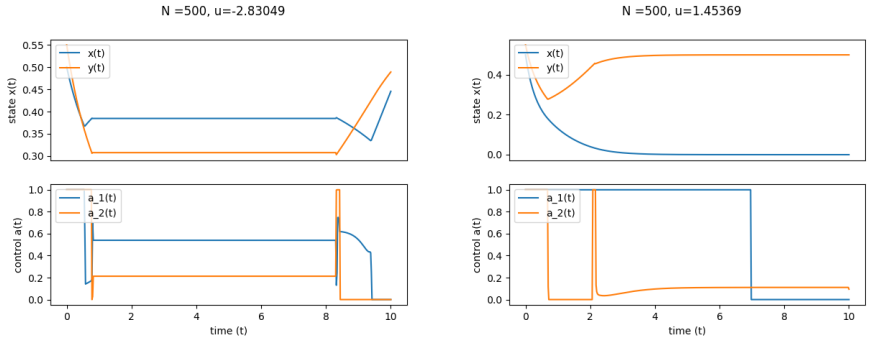

Figure 8.12: Two plots computed with `LongTimeHorizon`$(T_0, N_0, N_{\max}, dN)$-method. The left and right plot correspond to two-dimensional regularized Hamiltonian approach (with the lowest threshold) in Experiment 8.9 and 8.11, respectively. Parameters: $T_0 = 2.5\,, N_0 = 125\,, N_{\max} = 500\,, dN = 25$.

Technically, based on Experiment 8.12 it is very difficult to draw conclusions. As seen in previous examples optimal controls that are different still produce similar functional values, making it hard to justify that the method `LongTimeHorizon` does not work. Optimal control problems in general are difficult, but regularizing and calculating it numerically adds a lot of extra layers of complexity. Maybe the algorithm did not converge in this single case, or, maybe there are not enough time points to draw actual conclusions. Perhaps we should not have shrunken both parameter at the same time. But then the algorithm would take a lot longer to compute. To oppose this disconcerting note, we draw some relevant conclusions:

- For the competitive coexistence case, we know that we found the same optimal control, in a way, justifying that it is a suitable candidate for an optimal fish harvesting strategy.

- From a theoretical perspective both approaches should work, provided that the discretization is large enough. Clearly we can always double the number of time points and see if this improves our optimal control.
- For the competitive exclusion case, we see that the optimal control in both approaches lead to a steady state for the y species, and that the control calculated with the direct approach is more optimal. Therefore we can base a suitable candidate for a fishing strategy of species y as follows

$$\alpha_t = \begin{cases} 1 & \text{if} \quad 0 \leq t < 0.9 \\ 0 & \text{if} \quad 0.9 \leq t \leq 2.1 \\ 0.4 & \text{if} \quad 2.1 < t \leq 10 \end{cases}.$$

The hope is that we convey in some way the complexity of addressing practical problems in an optimal control framework and the extensiveness of optimal control solutions that can come out of it. The question which optimal control strategy is 'most' optimal is in some-ways irrelevant, instead, what matters is that we can actually find suitable strategies, which suit the same purpose.

8.5 Summary of Results

For reference we give a quick summary of the results drawing some additional conclusions. Starting with the single species models, the smooth Hamiltonian approach works quite well for the single species models, it is even able to stabilize the state trajectory for long time periods. Secondly, the regularized Hamiltonian approach is able to stabilize the resulting state trajectory in short and long time horizons. Lastly, both approaches suffer from penalizing trajectories ending above the initial population as much as below. This is not ideal and can be considered unsustainable.
When there are two species involved the smooth Hamiltonian approach starts to break down. It opts for negative fishing intensities, which is undesirable. The regularized Hamiltonian approach does not suffer from this disadvantage. Actually, we prematurely excluded the possibility by defining the control set to be

$$A = [0, \alpha_{\max}],$$

which leads to the non-smoothness in the first place.
Additionally, we conclude that the two-dimensional control is the preferred approach, since it is better able to control the resulting state trajectories. However, the one-dimensional control (with regularized Hamiltonian) remains an adequate model in case we assume that only one-dimensional controls are allowed.
Lastly, the long time horizon and direct implementation both produce the same results, in the case of competitive coexistence. Hence, we conclude that the optimal control computed via the direct approach, is a suitable candidate for being *the* optimal fishing strategy for a dynamical system where two fish species live in coexistence.

9 | Conclusion

In this thesis we have presented various ways of formulating an optimal control problem for generating (optimal) fish harvesting strategies, for a different number of species. After (mathematically) characterizing all the different models, they were analyzed and compared. From the comparisons we also draw (preliminary) conclusions on how to generate sustainable fishing strategies.

Mathematically, the models can be categorized into two groups. A group, which formulates a smooth (optimal control) Hamiltonian opposed to another group, which formulates a non-smooth Hamiltonian. The former is analytically tractable and results in a well-posed optimal control problem with a concave Hamiltonian, which is desirable, since it produces readily computable optimal control strategies. The latter is derived with the same ease, but results in a non-smooth Hamiltonian with jump-discontinuous partial derivatives. The derivatives are necessary for setting up the symplectic Euler scheme. Therefore, the use of a regularization for the discontinuous part of the partial derivatives has to be implemented. The derivatives are regularized separately, but in such a way, that it is equivalent to regularizing the Hamiltonian directly. This is necessary to obtain optimal control solutions of the regularized problem, which are also optimal control solutions for the non-regularized problem.

From a fish harvesting perspective, the non-smooth Hamiltonian related to a running cost is clearly defined. It concerns maximizing of profit. Conversely, the smooth Hamiltonian has a term which is not related to optimal harvesting. It can in some way be related to the cost of fishing, but this is partly an arbitrary choice. It is added to make the resulting Hamiltonian concave, and hence, more analytically tractable. However, it turned out that this additional term is (too) restrictive and in some cases produces optimal controls that are partly negative, which is undesirable from a fish harvesting perspective.
The non-smooth Hamiltonian is therefore the preferred choice, since it is able to stabilize the state trajectories for more than one species. This actually leads to constant steady states, which are independent of the end time. However, using the regularized version its is very hard to know if the optimal control, that is produced by numerical scheme is actually optimal. Fishing is a practical problem. Usually for more practically oriented optimal control problems it is satisfactory that a candidate control solution satisfies the Pontryagin's Maximum Principle, since this implies local optimality.

The mathematical analysis even gives insight in finding fishing strategies that can be considered sustainable. Firstly, in the non-smooth Hamiltonian models the optimal state & control solution will after some time settle on a steady state for both trajectories. This steady state is kept for an extended period of time. This suggests that one can stabilize the fish population by fishing the optimal amount over an extended period of time.
Secondly, by adjusting the weight of the terminal costs we can force the control solution's state trajectory to end closer to the initial population ; or further away. A disadvantage of the proposed terminal cost is that it penalizes a larger

end-time population, than the initial population, the same way as an end-time population which is lower, than the original population. Therefore, we propose further investigation into different terminal cost functions that penalize ending conditions differently, which could in turn lead to more sustainable optimal harvesting strategies.

In conclusion, addressing the problem of fish harvesting using optimal control theory is appropriate. The formulation of a non-smooth Hamiltonian creates a tractable problem. Specifically, in the case of fish harvesting when controls only affect a single species, the derivatives of the Hamiltonian are separable. Furthermore, optimal control theory is sufficiently flexible to adjust the optimal control to adhere to sustainability desires.

A | Appendix

A.1 Derivation of the Necessary Conditions for the Lagrange Multiplier Method

In this section we will formulate necessary conditions for Equation 3.3. We will (loosely) follow the derivation presented in [3]. For readability we will omit the t-dependence if functions, like x,λ or α are elements of another function, i.e.

$$f(x, \alpha) := f(x_t, \alpha_t).$$

Remember that a (local) minimizer is a stationary point. Thus an optimal solution would necessarily satisfy the following system (note that we changed the order)

$$\partial_\lambda \mathcal{L}(x, \lambda, \alpha) = 0, \tag{A.1}$$
$$\partial_x \mathcal{L}(x, \lambda, \alpha) = 0, \tag{A.2}$$
$$\partial_\alpha \mathcal{L}(x, \lambda, \alpha) = 0. \tag{A.3}$$

Depending on the dimension these conditions can be either in one or multiple dimensions. We will derive corollaries for each equation. Throughout this section we will use the following theorem.

Theorem 6 (Fundamental Lemma of Calculus of Variation)**.** *If a function f is continuous on an open interval (a,b) and satisfies the equality*

$$\int_a^b f(t)h(t)\,\mathrm{d}t = 0,$$

for all smooth functions h vanishing at the boundary of (a, b), then f is zero.[1]

Equation (A.1)

Since $\partial_\lambda \mathcal{L}(x, \lambda, \alpha) = 0$ we know

$$\frac{d}{d\epsilon}\mathcal{L}(x, \lambda + \epsilon v, \alpha)\Big|_{\epsilon=0} = 0, \quad \forall v(t).$$

Let $\tilde{\lambda}_t := \lambda_t + \epsilon v_t$ be a small change in λ, then

$$\begin{aligned} 0 &= \lim_{\epsilon \to 0} \frac{\mathcal{L}(x, \tilde{\lambda}, \alpha) - \mathcal{L}(x, \lambda, \alpha)}{\epsilon}, \\ &= \lim_{\epsilon \to 0} \frac{\int_0^T \epsilon v_s \cdot (f(x_s, \alpha_s) - x'_s)\,\mathrm{d}s}{\epsilon}, \\ &= \int_0^T v_s \cdot (f(x_s, \alpha_s) - x'_s)\,\mathrm{d}s \end{aligned}$$

Hence,

$$x'_t = f(x_t, \alpha_t), \quad \forall 0 < t < T.$$

The latter consequence is justified by Theorem (6).

[1] Technically, the functions h should be compactly supported on (a, b), but this is beyond the scope of this thesis, and usually vanishes suffices.

Equation (A.2)

We will follow the same line of reasoning as above. To this extent let $\tilde{x}_t := x_t + \epsilon u_t$ be a small change in x, then the derivative x' also changes a little, i.e. $\tilde{x}'_t = x'_t + \epsilon u'_t$. First note by Taylor expansion that

$$\begin{aligned} g(\tilde{x}_T) &= g(x_T + \epsilon u_T) = g(x_T) + \epsilon \nabla g(x_T) \cdot u_T + \mathcal{O}(\epsilon^2), \\ L(\tilde{x}, \lambda, \alpha) &= L(x, \lambda, \alpha) + \epsilon \left(\partial_x L(x, \lambda, \alpha) \cdot u_t\right) + \mathcal{O}(\epsilon^2), \\ f(\tilde{x}, \alpha) &= f(x, a) + \epsilon \mathcal{J}_x f(x, \alpha) u_s + \mathcal{O}(\epsilon^2), \end{aligned}$$

where $\mathcal{J}_x f(x, \alpha)$ is the Jacobian, the matrix of partial derivatives of f w.r.t. x, i.e. $\mathcal{J}_x f(x, \alpha) = (\partial_{x_i} f_j(x, \alpha))_{ij}$. Combining everything leads to

$$\mathcal{L}(\tilde{x}, \lambda, \alpha) - \mathcal{L}(x, \lambda, \alpha) = \epsilon \left(\nabla g(x_T) \cdot u_T + \int_0^T \partial_x L(x, \alpha) \cdot u_s + \lambda_s \cdot (\mathcal{J}_x f(x, \alpha) u_s - u'_s) \, \mathrm{d}t \right) + \mathcal{O}(\epsilon^2).$$

Hence, $\partial_x \mathcal{L}(x, \lambda, \alpha) = 0$ leads to

$$\begin{aligned} 0 &= \frac{d}{d\epsilon} \mathcal{L}(\tilde{x}, \lambda, \alpha) \Big|_{\epsilon=0} \\ &= \nabla g(x_T) \cdot u_T + \int_0^T \partial_x L(x, \alpha) \cdot u_s + \lambda_s \cdot (\mathcal{J}_x f(x, \alpha) u_s - u'_s) \, \mathrm{d}t \\ &= \nabla g(x_T) \cdot u_T + \int_0^T (\partial_x L(x, \alpha) \cdot u_s + \mathcal{J}_x f(x, \alpha) \lambda_s) \cdot u_s - \lambda_s \cdot u'_s \, \mathrm{d}t \\ &= \nabla g(x_T) \cdot u_T - \left[u_s \cdot \lambda_s \right]_0^T + \int_0^T (\partial_x L(x, \alpha) + \mathcal{J}_x f(x, \alpha) \lambda_s + \lambda'_s) \cdot u_s \, \mathrm{d}t \\ &= (\nabla g(x_T) - \lambda_T) \cdot u_T + \int_0^T (\partial_x L(x, \alpha) + \mathcal{J}_x f(x, \alpha) \lambda_s + \lambda'_s) \cdot u_s \, \mathrm{d}t \end{aligned}$$

In the third step integration by parts is used and in the fifth $u_0 = 0$, since $\tilde{x}_0 = x_0 = x^0$. Concluding (by Theorem 6)

$$0 = \partial_x L(x, \alpha) + \mathcal{J}_x f(x, \alpha) \lambda_t + \lambda'_t \quad \text{and} \quad 0 = \nabla g(x_T) - \lambda_T.$$

To see the latter, remember that this must hold for every u_T, thus eliminating the case that the terms would cancel each other out. Summarizing, (A.2) is equivalent to the following system

$$\begin{cases} -\lambda'_t = \partial_x \left(L(x, \alpha) + \lambda_t \cdot f(x, \alpha) \right) \\ \lambda_T = \nabla g(x_T) \end{cases}, \tag{A.4}$$

which is known as the costate equation.

Equation (A.3)

Let $\tilde{\alpha}_s := \alpha_s + \epsilon w_s$ be a small change in α, then

$$\begin{aligned} 0 &= \frac{d}{d\epsilon} \mathcal{L}(x, \lambda, \tilde{\alpha}) \Big|_{\epsilon=0} \\ &= \int_0^T \partial_\alpha L(x, \alpha) \cdot w_s + \lambda_s \cdot \mathcal{J}_\alpha f(x, \alpha) w_s \, \mathrm{d}t \\ &= \int_0^T \left(\partial_\alpha \left(L(x, \alpha) + \lambda_s \cdot f(x, \alpha) \right) \right) \cdot w_s \, \mathrm{d}t \end{aligned}$$

Then using Theorem 6

$$\partial_\alpha \left(L(x,\alpha) + \lambda_t \cdot f(x,\alpha) \right) = 0.$$

A.2 Derivation of the HJB-Equation

In this section we will derive the HJB equations from the value function defined in 3.8. Again we will follow [3]. To this extent

$$\begin{aligned} v(x,t) :&= \inf_{\alpha:[t,t+\tau]\to\mathbb{R}^n} \inf_{\alpha:[t+\tau,T]\to\mathbb{R}^n} \left\{ g(x_T) + \int_t^T L(x,\alpha)\,\mathrm{d}t \,\middle|\, x_t = x \right\} \\ &= \inf_{\alpha:[t,t+\tau]\to\mathbb{R}^n} \left\{ \inf_{\alpha:[t+\tau,T]\to\mathbb{R}^n} \left\{ g(x_T) + \int_{t+\tau}^T L(x,\alpha)\,\mathrm{d}t \right\} + \int_t^{t+\tau} L(x,\alpha)\,\mathrm{d}t \right\} \\ &= \inf_{\alpha:[t,t+\tau]\to\mathbb{R}^n} \left\{ v(x,t+\tau) + \int_t^{t+\tau} L(x,\alpha)\,\mathrm{d}t \right\} \end{aligned}$$

Hence, we can conclude by subtracting $v(x,t)$ on both sides:

$$0 = \inf_{\alpha:[t,t+\tau]\to\mathbb{R}^n} \left\{ v,t+\tau) - v(x,t) + \int_t^{t+\tau} L(x,\alpha)\,\mathrm{d}t \right\}.$$

Consider the value function, but slightly evolved in time, i.e. $\tilde{t} = t + \tau$, then, by Taylor expansion

$$v(x,t+\tau) = v(x,t) + \tau(\partial_t v(x,t) + f(x,\alpha)\cdot\partial_x v(x,t)) + \mathcal{O}(\tau^2),$$

where where $x_t' = f(x,a)$. Then

$$\begin{aligned} 0 &= \inf_{\alpha\in\mathcal{A}} \left\{ \tau(\partial_t v(x,t) + f(x,\alpha)\cdot\partial_x v(x,t)) + \int_t^{t+\tau} L(x,\alpha)\,\mathrm{d}t \Big|_{\tau=0} \right\} \\ 0 &= \inf_{\alpha\in\mathcal{A}} \left\{ \partial_t v(x,t) + f(x,\alpha)\cdot\partial_x v(x,t) + L(x,\alpha) \right\} \\ 0 &= \partial_t v(x,t) + \inf_{\alpha\in\mathcal{A}} \left\{ L(x,\alpha) + f(x,\alpha)\cdot\partial_x v(x,t)) \right\}. \end{aligned}$$

A word of note, in the above derivation we assume that the value function is differentiable. However, this is not true in the general case. But since we only touch upon the HJB equation, through the fact that for an optimal control the HBJ equation is a sufficient condition, it will be treated as a footnote importance our purpose.

Bibliography

[1] Carol R. Ember. Myths about hunter-gatherers. *Ethnology*, 17(4):439–448, 1978.

[2] Marta Coll, Simone Libralato, Sergi Tudela, Isabel Palomera, and Fabio Pranovi. Ecosystem overfishing in the ocean. *PloS one*, 3:e3881, 02 2008.

[3] Anders Szepessy Raul Tempone Georgios Zouraris Jesper Carlsson, Kyoung-Sook Moon. *Stochastic Differential Equations: Models and Numerics (Draft)*. March 31, 2021.

[4] C. Lubich E. Hairer and G. Wanner. *Geometric Numerical Integration*. Springer Series in Computational Mathematics.

[5] L. Bittner. L. s. pontryagin, v. g. boltyanskii, r. v. gamkrelidze, e. f. mishechenko, the mathematical theory of optimal processes. viii + 360 s. new york london 1962. john wiley and sons. preis 90. *ZAMM - Zeitschrift für Angewandte Mathematik und Mechanik*, 43:514–515, 1963.

[6] Mattias Sandberg and Anders Szepessy. Convergence rates of symplectic pontryagin approximations in optimal control theory. *ESAIM: Mathematical Modelling and Numerical Analysis*, 40(1):149–173, 2006.

[7] Daniel S. Licht, Joshua J. Millspaugh, Kyran E. Kunkel, Christopher O. Kochanny, and Rolf O. Peterson. Using Small Populations of Wolves for Ecosystem Restoration and Stewardship. *BioScience*, 60(2):147–153, 02 2010.

[8] Keng-Lou Hung, Jennifer Kingston, Matthias Albrecht, David Holway, and Joshua Kohn. The worldwide importance of honey bees as pollinators in natural habitats. *Proceedings of the Royal Society B: Biological Sciences*, 285:20172140, 01 2018.

[9] Maurice Vogels, Rita Zoeckler, Donald M. Stasiw, and Lawrence C. Cerny. P. f. verhulst's "notice sur la loi que la populations suit dans son accroissement" from correspondence mathematique et physique. ghent, vol. x, 1838. *Journal of Biological Physics*, 3:183–192, 1975.

[10] Cesare Marchetti, Perrin S. Meyer, and Jesse H. Ausubel. Human population dynamics revisited with the logistic model: How much can be modeled and predicted? *Technological Forecasting and Social Change*, 52(1):1–30, 1996.

[11] Hiroshi Fujikawa, Akemi Kai, and Satoshi Morozumi. A new logistic model for bacterial growth. *Shokuhin eiseigaku zasshi. Journal of the Food Hygienic Society of Japan*, 44:155–60, 07 2003.

[12] Michael G. Neubert. Marine reserves and optimal harvesting. *Ecology Letters*, 6(9):843–849, 2003.

[13] Alfred J. Lotka. Analytical note on certain rhythmic relations in organic systems. *Proceedings of the National Academy of Sciences*, 6(7):410–415, 1920.

[14] V Volterra. Variatzioni e fluttuazioni de1 numero d'individui in specie animali conviventi, mem. *R. Accad. Linei Ser*, 6, 1926.

[15] Mostafa A. Abdelkader. Exact solutions of lotka-volterra equations. *Mathematical Biosciences*, 20(3):293–297, 1974.

[16] Burton S. Garbow, Kenneth E. Hillstrom, and Jorge J. More. Documentation for minpack subroutine hybrj, Mar 1980.