



HAL
open science

Performance of a Markovian neural network versus dynamic programming on a fishing control problem

Mathieu Laurière, Gilles Pagès, Olivier Pironneau

► To cite this version:

Mathieu Laurière, Gilles Pagès, Olivier Pironneau. Performance of a Markovian neural network versus dynamic programming on a fishing control problem. *Probability, Uncertainty and Quantitative Risk*, In press, 10.48550/arXiv.2109.06856 . hal-03891220

HAL Id: hal-03891220

<https://cnrs.hal.science/hal-03891220>

Submitted on 9 Dec 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Performance of a Markovian neural network versus dynamic programming on a fishing control problem

Mathieu Laurière¹, Gilles Pagès², Olivier Pironneau³

Abstract

Fishing quotas are unpleasant but efficient to control the productivity of a fishing site. A popular model has a stochastic differential equation for the biomass on which a stochastic dynamic programming or a Hamilton-Jacobi-Bellman algorithm can be used to find the stochastic control – the fishing quota. We compare the solutions obtained by dynamic programming against those obtained with a neural network which preserves the Markov property of the solution. The method is extended to a similar multi species model to check its robustness in high dimension.

Keywords: Stochastic optimal control, partial differential equations, neural networks.

1. Introduction

Too much fishing can deplete the biomass to a disastrous level and leave fishermen out of work, or even, in some part of the world, starving. There are several ways to control fishing. One is to forbid fishing in regions and to alternate fishing and non-fishing zones [21]. Another is by imposing quotas. In [2] statistical learning was shown to be very efficient to calibrate the parameters of the fishing model of [9]. In [24] a stochastic control problem was derived from the model used in [9] and to speed up the computation of optimal quotas, a solution by statistical learning was proposed and compared to standard stochastic control solutions like the Hamilton-Jacobi-Bellman equations (HJB). However the solution provided by the neural network was not Markovian as it used the future states, given by the stochastic model, to optimise the present. In this article we propose to study the performance of a Markovian neural network, in the spirit of [12, 14, 10].

The unpopularity of severe quotas is modeled by a penalty in the cost of an optimisation problem to compute the best fishing strategy which preserves the fish biomass, *i.e.*, keeps it close to an ideal state X_d . Quotas on fishing should also be as stable in time as possible because fishermen need to know that the quota will not move too much from one day to the next. This is modeled by another penalty on the time variations of the quota.

Let X_t be the fish biomass at time t , E_t the fishing effort, interpreted as the number of boats at sea. In [9], qX_t , with the *catchability* constant q , is the maximum weight of fish that a boat can mechanically catch, meaning that the more fish there is the more fishermen will catch them, but proportionally to the capacity of his equipment.

¹*lauriere@princeton.edu*, ORFE dept. Princeton University, USA

²*gilles.pages@sorbonne-universite.fr*, LPMA, Sorbonne Université, Paris, France

³*olivier.pironneau@sorbonne-universite.fr*, LJLL, Sorbonne Université, Paris, France.

Ideally a fisherman may want to catch a quantity Q_t on day t . Imposing a quota Q_M means $Q_t < Q_M$.

2. The fishing site model

Consider a fishing site with d types of fish in interaction, in the sense that some depend on others for food. Let $\mathbf{X}(t) \in \mathbb{R}^d$ be the fish biomasses at time t , $E(t)$ the fishing effort – interpreted as the number of boats at sea – and the capacity to catch type i is $qX_i(t)$, with the catchability q constant over types for simplicity.

An optimal strategy with quotas for each species i is given to each fisherman, to impose the maximum weight of fish, $Q_i(t) \in \mathbb{R}^d$, of that species caught on a day t . Hence the total amount of fish of type i caught on a day is $E(t) \min(qX_i(t), Q_i(t))$, $i = 1, \dots, d$.

The logistic equation for $\mathbf{X}(t)$ says that the biomasses is a consequence of the natural growth or decay rate \mathbf{r} , the long time limit $\underline{\kappa}^{-1}\mathbf{r}$ of X , where $\underline{\kappa}$ is the $d \times d$ species interaction matrix, and the depletion due to fishing:

$$\frac{d\mathbf{X}}{dt}(t) = \mathbf{X}(t) \cdot \underline{\Lambda}[\mathbf{r} - \underline{\kappa}\mathbf{X}(t) - \min(q\mathbf{X}(t), \mathbf{Q}(t))E(t)]. \quad (1)$$

The operator $\mathbf{r} \mapsto \underline{\Lambda}[\mathbf{r}]$ transforms a vector $\mathbf{r} \in \mathbb{R}^d$ into a diagonal $d \times d$ matrix. For example, with $d = 2$ and $\kappa_{12} < 0, \kappa_{21} > 0$, then, in (1),

$$\mathbf{X}(t) \cdot \underline{\Lambda}[\mathbf{r} - \underline{\kappa}\mathbf{X}(t)] = [X_1(r_1 - \kappa_{11}X_1 + |\kappa_{12}|X_2), X_2(r_2 - |\kappa_{21}|X_1 - \kappa_{22}X_2)]^T$$

which means that the first species lives on its own but profit from the second species because it eats it as shown by the equation for the second species which has a death rate augmented by $|\kappa_{21}|X_1 > 0$.

Let $\mathbf{F}(t) = \min(q\mathbf{X}(t), \mathbf{Q}(t))$. The fishing effort E is driven by the profit $\mathbf{p} \cdot \mathbf{F}$ minus the operating cost of a boat c , where p_i is the price of fish of species i :

$$\frac{1}{E} \frac{dE}{dt} = \mathbf{p} \cdot \mathbf{F} - c. \quad (2)$$

The price is driven by the difference between the demand $\mathbf{D}(\mathbf{p})$ and the resource $\mathbf{F}E$:

$$\Phi \frac{d\mathbf{p}}{dt} = \mathbf{D}(\mathbf{p}) - \mathbf{F}E \quad \text{with the demand } \mathbf{D}(\mathbf{p})_i = \frac{a'_i}{1 + \gamma p_i} \quad (3)$$

where Φ is the inverse time scale at which the fish market price adjusts. When $\Phi \ll 1$, (3) may be approximated by:

$$\mathbf{D}(\mathbf{p}) - \mathbf{F}E = 0 \Rightarrow \gamma p_i F_i E = a'_i - F_i E \Rightarrow \mathbf{p} \cdot \mathbf{F}E = \frac{1}{\gamma} \text{tr}[\mathbf{a}' - E\mathbf{F}],$$

where the trace operator is defined by $\text{tr}[\mathbf{a}] := \sum_{i=1}^d a_i$. Let us denote $a = \sum_1^d a'_i/\gamma^2$, $\tilde{q}_i = q_i\gamma$, $\tilde{Q}_i = Q_i\gamma$, and $\tilde{E} = E/\gamma$. Then the whole system (1)–(2) rewrites:

$$\frac{d\mathbf{X}}{dt} = \mathbf{X} \cdot \underline{\Lambda}[\mathbf{r} - \underline{\kappa}\mathbf{X} - \min(\tilde{q}\mathbf{X}, \tilde{\mathbf{Q}})\tilde{E}], \quad \frac{d\tilde{E}}{dt} = a - (\text{tr}[\min(\tilde{q}\mathbf{X}, \tilde{\mathbf{Q}})] + c)\tilde{E}, \quad (4)$$

where $\mathbf{X} \cdot \underline{\Lambda}[\cdot]$ is matrix-vector product $\underline{\Lambda}[\cdot]\mathbf{X}$. Since we are not going to use the original variables in the sequel, we drop the tildas and write q_i, Q_i , and E instead of \tilde{q}_i, \tilde{Q}_i , and \tilde{E} .

Finally we use another change of variables to get rid of the q variable: we replace t by t/q , \mathbf{Q} by $q\mathbf{Q}$ and $(\mathbf{r}, \underline{\kappa}, \mathbf{a}, c)$ by $q(\mathbf{r}, \underline{\kappa}, \mathbf{a}, c)$. Then the above system (4) is identical but now with $q = 1$. In the end, with $a = \text{tr}[\mathbf{a}] := \sum_1^d a_i$, the whole system for the evolution of the fish biomass \mathbf{X} and the fishing effort E is:

$$\frac{d\mathbf{X}}{dt} = \mathbf{X} \cdot \underline{\Lambda}[\mathbf{r} - \underline{\kappa}\mathbf{X} - \min(\mathbf{X}, \mathbf{Q})E], \quad \frac{dE}{dt} = a - (\text{tr}[\min(\mathbf{X}, \mathbf{Q})] + c)E. \quad (5)$$

2.1. The Stochastic vector control problem

To prevent fish extinction, a constraint is set on the total catch $\min(qX_i(t), Q_i(t))E(t)$, $i = 1, \dots, d$, per species. The value of $\mathbf{Q}(t)$ is found by solving an optimisation problem described below. It is expected that $Q_i(t) < qX_i(t)$ otherwise the policy of quota is irrelevant in the sense that the fisherman is given a maximum allowed catch which is greater than what he could mechanically catch.

Let us add a constraint $\mathbf{Q}(t) \leq \mathbf{Q}_M$ so that $\min(q\mathbf{X}(t), \mathbf{Q}(t)) = \mathbf{Q}(t)$. Denote $u_i(t) = E(t)Q_i(t)/X_i(t)$ the optimal fishing policy for species i per unit mass. We formulate the optimal control problem in terms of u_i rather than Q_i . Then E no longer appears and the problem becomes:

$$\min_{\mathbf{u} \in \mathcal{U}} \left\{ \bar{J} := \int_0^T |\mathbf{X}(t) - \mathbf{X}_d(t)|^2 dt : \frac{d\mathbf{X}}{dt} = \mathbf{X} \cdot \underline{\Lambda}[\mathbf{r} - \mathbf{u} - \underline{\kappa}\mathbf{X}], \quad \mathbf{X}(0) = \mathbf{X}^0 \right\} \quad (6)$$

where the control is in the constraint space

$$\mathcal{U} = \{u_m \leq u_i(t) \leq u_M, \quad i = 1, \dots, d, t \in [0, T]\},$$

which reflects a desire to ensure a minimal fishing u_m and a maximum one u_M so as to guarantee that $u_i < qE(t)$ at all time.

2.2. More constraints on quotas

To avoid small fishing quotas we use penalty and add to the criteria $-\int_0^T \sum_1^d \alpha_i u_i(t)$. Moreover, to avoid too many daily changes we penalise the quadratic variation of \mathbf{u} over the time period $(0, T)$, *i.e.*, to \bar{J} we add $\beta \cdot \mathbb{E}[\mathbf{u}]_t^{0,T}$, where the quadratic variation is defined as:

$$[\mathbf{u}]_t^{0,T} = \lim_{\|P\|} \sum_1^K |\mathbf{u}_{t_k} - \mathbf{u}_{t_{k-1}}|^2$$

where P ranges over partitions of the interval $(0, T) = \cup_k (t_{k-1}, t_k)$ and the limit is in probability when $\max_k |t_k - t_{k-1}| \rightarrow 0$. Here Itô calculus [7] tells us that:

$$\mathbb{E}[\mathbf{u}]_t^{0,T} = \sigma^2 \int_0^T \mathbb{E}[|X_t \cdot \nabla_{\mathbf{X}} \mathbf{u}_i|^2] dt.$$

Hence, an optimal policy in the presence of noise is a solution of

$$\min_{\mathbf{u} \in \mathcal{U}} \left\{ \bar{J} := \mathbb{E} \left[\int_0^T [|\mathbf{X}(t) - \mathbf{X}_d(t)|^2 dt - \boldsymbol{\alpha} \cdot \mathbf{u} + \boldsymbol{\beta} \cdot [\mathbf{u}]_t^{0,T}] dt \right] : \right.$$

$$d\mathbf{X}_t = \mathbf{X} \cdot \underline{\Lambda} [(\mathbf{r} - \mathbf{u} - \underline{\kappa}\mathbf{X})dt + \underline{\sigma}d\mathbf{W}_t], \quad \mathbf{X}(0) = \mathbf{X}^0 \}. \quad (7)$$

For the sake of clarity we assume $\beta_i = \beta$ for all i .

Remark 1. Let $\mathbf{Y} = \underline{\kappa}\mathbf{X}$. A multiplication of the SDE (7) by $\underline{\kappa}$ leads to

$$d\mathbf{Y}_t = \mathbf{Y}_t \cdot \underline{\kappa} \underline{\Lambda} [(\mathbf{r} - \mathbf{u} - \mathbf{Y}_t)dt + \underline{\kappa} \otimes \underline{\sigma}d\mathbf{W}_t] \underline{\kappa}^{-T}.$$

Thus, if all the terms on the diagonal of $\underline{\Lambda} [(\mathbf{r} - \mathbf{u} - \mathbf{Y}_t)dt + \underline{\kappa} \otimes \underline{\sigma}d\mathbf{W}_t]$ are equal and if the initials conditions $\underline{\kappa}\mathbf{X}(0)$ are equal, then the components of \mathbf{Y} are independent and equal because $\underline{\kappa}$ and $\underline{\Lambda}$ commute. It happens only if $\underline{\kappa} \otimes \underline{\sigma}d\mathbf{W}_t = \mathbf{1}dW_t$.

2.3. Existence of solution

First, the solution of the SDE exists and is positive. For clarity the proof is done for $d = 1$:

Note that $X \mapsto (r - \kappa X - u_t)X$ is locally Lipschitz, uniformly in t since $u_t \in [u_m, u_M]$. Hence for every realization $W_t(\omega)$ there is a unique strong solution until a blow-up time τ which is a stopping time for the filtration $\mathcal{F}_t^w = \sigma(W_s, s \leq t, \mathcal{N}_s)$.

On $[0, \tau[$ by Itô calculus,

$$X_t = X^0 \exp \left(\left(r - \frac{\sigma^2}{2} \right) t - \int_0^t (\kappa X_s + u_s) ds + \sigma W_t \right),$$

so:

$$X_t \leq X^0 e^{S_t(u_m)}, \forall t \in (0, \tau], \quad \text{where } S_t(v) := \left(r - \frac{\sigma^2}{2} \right) t - vt + \sigma W_t.$$

Hence $\int_0^\tau X_s ds = +\infty$ is impossible unless $\tau = +\infty$, \mathcal{P} a.s. Therefore

$$X^0 e^{S_t(u_M)} \exp \left(-\kappa \int_0^t e^{S_s(u_M)} ds \right) \leq X_t \leq X^0 e^{S_t(u_m)}.$$

Let $y = \log x$ and $v = r - \frac{\sigma^2}{2} - \kappa e^y - u(e^y, t)$. From [19], if $v \in W_{loc}^{1,1}(\mathbb{R})$, $v/(1+|y|) \in L^1(\mathbb{R}) \cap L^\infty(\mathbb{R})$, $\partial_y v \in L^\infty(\mathbb{R})$, then Y_t has a PDF $\rho \in L^\infty(L^2(\mathbb{R}) \cap L^\infty(\mathbb{R})) \cap L^2(H^1(\mathbb{R}))$ and the following *equivalent* control problem has a solution:

$$\begin{aligned} \min_{v \in \mathcal{V}} J(v) &:= \int_{-R}^R dy \int_0^T \left[(e^y - X_d)^2 - \alpha u(y, t) + \beta |\partial_t v(y, t)|^2 \right] \rho(y, t) dt : \\ \partial_t \rho + \partial_y(v\rho) - \partial_{yy} \left[\frac{\sigma^2}{2} \rho \right] &= 0, \quad \rho(y, 0) = \rho^0(y), \quad \forall y \in \mathbb{R}, \quad \forall t \in]0, T[. \end{aligned}$$

where $\mathcal{V} = \{v : v_m \leq v \leq v_M, \|\partial_t v\|_{L^2(\mathcal{Q})} \leq K\}$.

Remark 2. This result also gives a computational method, by discretizing the above and solving it as an optimisation problem. Of the four methods discussed in this article, this is the most expensive. It was tested in [18] and shown not to be superior in precision to other methods.

3. Time discretization

We consider a uniform grid with $M + 1$ points in time. Let $h = \frac{T}{M}$, $t^m = mh$ and, for any f defined on $[0, T]$, f^m an approximation of $f(t^m)$. Define the Euler scheme:

$$\mathbf{X}^{m+1} = \mathbf{X}^m (1 + \underline{\mathbf{A}} [(\mathbf{r} - \mathbf{u}^m - \underline{\kappa}\mathbf{X}^m)h + \underline{\sigma}\delta\mathbf{W}^m]), \quad (8)$$

where $\delta W_i^m = W_i^{(m+1)h} - W_i^{mh} \sim \sqrt{h}\mathbb{N}_{0,1}$, $i = 1, \dots, d$. Note that positivity may not be preserved by this scheme, but in practice negative values do not seem to appear.

A Monte-Carlo method is used with K sample solutions of (8) to compute the cost:

$$J_K(\mathbf{u}) := \sum_{m=1}^M \frac{h}{K} \sum_{k=1}^K \left[|\mathbf{X}_k^m - \mathbf{X}_d|^2 - \boldsymbol{\alpha} \cdot \mathbf{u}_k^m + \frac{\beta}{h} |\mathbf{u}_k^{m+1} - \mathbf{u}_k^m|^2 \right],$$

with the convention that $\mathbf{u}_k^{M+1} = \mathbf{u}_k^M$.

Remark 3. Let $\mathbf{X} \mapsto \mathbf{u}^{m+1}(\mathbf{X})$ and $\mathbf{X} \mapsto \mathbf{u}^m(\mathbf{X})$ be two given differentiable functions. Let $\sigma \in \mathbb{R}^+$. The following holds (reading hint: $u(X + a)$ is u at $X + a$, not u times $(X + a)$):

$$\begin{aligned} \mathbf{u}^{m+1}(\mathbf{X}^{m+1}) &= \mathbf{u}^{m+1}(\mathbf{X}^m (1 + h(r - \kappa\mathbf{X}^m - \mathbf{u}^m) + \sigma\delta W^m)) \\ &\approx \mathbf{u}^{m+1}(\mathbf{X}^m (1 + h(r - \kappa\mathbf{X}^m - \mathbf{u}^m))) + \mathbf{u}^{m+1}\sigma\mathbf{X}^{m+1}\delta W^m, \end{aligned}$$

where the derivative \mathbf{u}^{m+1} is evaluated at \mathbf{X}^m or at $\mathbf{X}^m (1 + h(r - \kappa\mathbf{X}^m - \mathbf{u}^m) + \sigma\delta W^m)$. Hence:

$$\begin{aligned} \mathbb{E}[|\mathbf{u}^{m+1} - \mathbf{u}^m|^2 | \mathbf{X}^m] &\approx \left| \mathbf{u}^{m+1}(\mathbf{X}^m (1 + h(r - \kappa\mathbf{X}^m - \mathbf{u}^m))) - \mathbf{u}^m \right|^2 \\ &\quad + h\sigma^2 |\mathbf{X}^{m+1} \cdot \nabla \mathbf{u}^{m+1}|^2, \end{aligned}$$

so:

$$\begin{aligned} &\frac{\beta}{h} \mathbb{E} \left[\sum_{m=1}^M h |\mathbf{u}^{m+1} - \mathbf{u}^m|^2 | \mathbf{X}^m \right] \\ &\approx \frac{\beta}{h} \sum_{m=1}^M h \left[\left| \mathbf{u}^{m+1}(\mathbf{X}^m (1 + h(r - \kappa\mathbf{X}^m - \mathbf{u}^m))) - \mathbf{u}^m \right|^2 + h\sigma^2 |\mathbf{X}^{m+1} \cdot \nabla \mathbf{u}^{m+1}|^2 \right]. \quad (9) \end{aligned}$$

Even though the first term in (9) is dominated by the second term, we may keep it for numerical convenience.

4. Stochastic Dynamic Programming (SDP)

Consider the value function

$$\begin{aligned} V(\mathbf{X}, t) &= \min_{\mathbf{u} \in \mathcal{U}} \left\{ \mathbb{E} \left[\int_t^T [|\mathbf{X}(t) - \mathbf{X}_d(t)|^2 dt - \boldsymbol{\alpha} \cdot \mathbf{u} + \beta [\mathbf{u}]_t^{t,T}] dt \right] : \right. \\ &\quad \left. d\mathbf{X}_t = \mathbf{X} \cdot \underline{\mathbf{A}} [(\mathbf{r} - \mathbf{u} - \underline{\kappa}\mathbf{X})dt + \underline{\sigma}d\mathbf{W}_t], \mathbf{X}(t) = \mathbf{X} \right\}. \quad (10) \end{aligned}$$

Let $\zeta_h(\mathbf{X}, \mathbf{u}^m, \mathbf{z})$ be the next iterate of a numerical scheme for the SDE starting at \mathbf{X} . For instance:

$$\zeta_h(\mathbf{X}, \mathbf{u}, \mathbf{z}) = \mathbf{X}(1 + \underline{\Lambda} [(\mathbf{r} - \mathbf{u} - \underline{\kappa}\mathbf{X})h + \underline{\sigma}\mathbf{z}\sqrt{h}]), \quad \mathbf{z} \text{ being one realisation of a } \mathbb{N}_{0,1}^d \text{ r.v.}$$

Bellman's dynamic programming principle tells us that the optimal control of the problem is the minimiser, u^m , in:

$$\begin{aligned} v^m(\mathbf{X}) &:= \min_{\mathbf{u} \in \mathcal{U}} \left\{ \int_{mh}^{(m+1)h} \mathbb{E} [|\zeta_h(\mathbf{X}, \mathbf{u}, \mathbf{z}) - \mathbf{X}_d|^2 - \underline{\alpha} \cdot \mathbf{u}] d\tau \right. \\ &\quad \left. + \beta[\mathbf{u}]_t^{mh, (m+1)h} + \mathbb{E}[v^{m+1}(\zeta_h(\mathbf{X}, \mathbf{u}, \mathbf{z}))] \right\} \\ &\approx \min_{\mathbf{u} \in \mathcal{U}} \left\{ h\mathbb{E} [|\zeta_h(\mathbf{X}, \mathbf{u}, \mathbf{z}) - \mathbf{X}_d|^2] - h\underline{\alpha} \cdot \mathbf{u} + \beta\mathbb{E} [|\mathbf{u}^{m+1}(\zeta_h(\mathbf{X}, \mathbf{u}, \mathbf{z})) - \mathbf{u}|^2] \right. \\ &\quad \left. + \mathbb{E}[v^{m+1}(\zeta_h(\mathbf{X}, \mathbf{u}, \mathbf{z}))] \right\}, \quad \text{with } v^M(\mathbf{X}) = 0. \end{aligned} \quad (11)$$

Evidently,

$$\mathbb{E} [|\zeta_h(\mathbf{X}, \mathbf{u}, \mathbf{z}) - \mathbf{X}_d|^2] = |\mathbf{X} - \mathbf{X}_d + h\underline{\Lambda}\underline{\Lambda}[\mathbf{r} - \underline{\kappa}\mathbf{X} - \mathbf{u}]|^2 + h|\underline{\sigma}\mathbf{X}|^2.$$

For every component, to compute $\mathbb{E}[v^{m+1}(\zeta_h(\mathbf{X}, \mathbf{u}, \mathbf{z}))]$, we use a quadrature formula with Q points $\{z_q\}_1^Q$ and weights $\{w_q\}_1^Q$ based on optimal quantization of the normal distribution $\mathbb{N}_{0,1}^d$ (see [25] or [22, Chapter 5] and the website www.quantize.maths-fi.com for download of grids) so that

$$\mathbb{E}[v^{m+1}(\zeta_h(\mathbf{X}, \mathbf{u}, \mathbf{z}))] \approx \sum_{q=1}^Q w_q v^{m+1}(\zeta_h(\mathbf{X}, \mathbf{u}, \mathbf{z}_q)) \quad (12)$$

Finally at every time step and every $X_j = jL/J$, $j = 0, \dots, J$ (with $L \gg 1$), the result is minimised with respect to $\mathbf{u} \in \mathcal{U}$ by a dichotomic search. In this fashion $\{\mathbf{u}^m(\mathbf{X}_j)\}_{j=1}^J$ is obtained on a grid and a piecewise linear interpolation is constructed to prepare for the next time step t^{m-1} .

4.1. Numerical results for a single species

For the numerical tests we have chosen: $r = 2$, $\kappa = 1.2$, $X_d = 1$, $T = 2$,

$$\alpha = 0.01, \quad \beta = 0.1, \quad \sigma = 0.1, \quad L = 3, \quad J = 40, \quad M = 50, \quad K = 100$$

and $X_0 = 0.5 + 0.1j$, $j = 0, \dots, 9$. Figures 1 and 5 show the control surface $(X, t) \mapsto u(X, t)$ and the value function $(X, t) \mapsto v(X, t)$. Although the control seems to be either 0.5 or 1 everywhere, there is a small interval near $X = X_d = 1$ in which it is not bang-bang. It is an important region, as seen on the sample solutions which are shown on Figure 3 and 4 where it is clear that the control is not always 0.5 or 1. For these an initial condition X_0 is chosen, a noise dW is generated and the SDE are integrated by the Euler scheme with the approximately optimal control $u(X_t, t)$ obtained by the SDP method described above. Then, by definition of the cost function, X_t should tend to be equal to X_d as much as possible without too many jumps for u . The trajectory is compared with one on which $u = u_M$, *i.e.*, no quota.

Finally on Figure 2 the cost function of the control problem is plotted for 100 samples and various values of X_0 . Away from X_d , it increases, which makes sense because $X_0 = 1$ means that we start with the optimal value in terms of fish biomass.

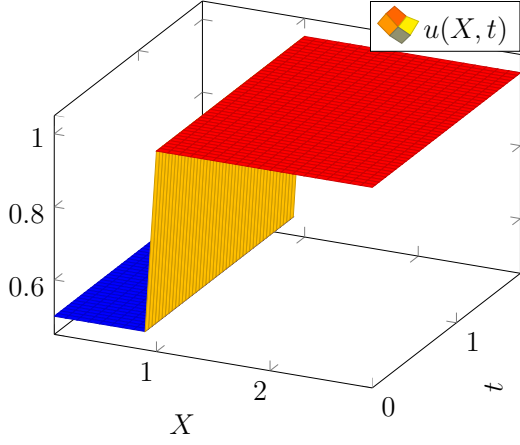


Figure 1: Solution $u(X, t)$ from SDP.

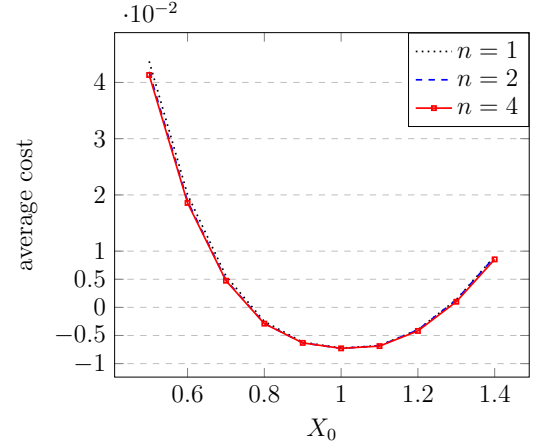


Figure 2: Values of the average cost versus X_0 for 100 realizations of dW and with the SDP, for the 3 time meshes $50n$, showing that the results are independent of n .

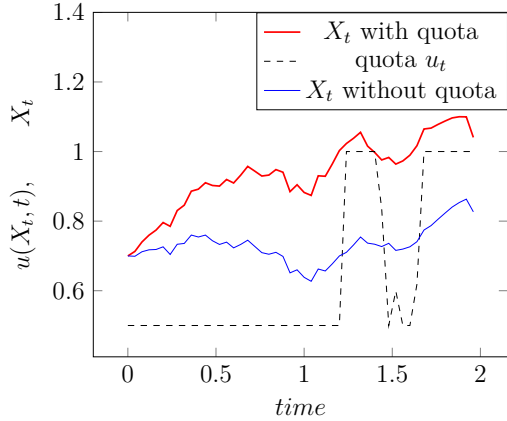


Figure 3: Simulation of the fishing model with a quota function computed by SDP and $X_0 = 0.7$.

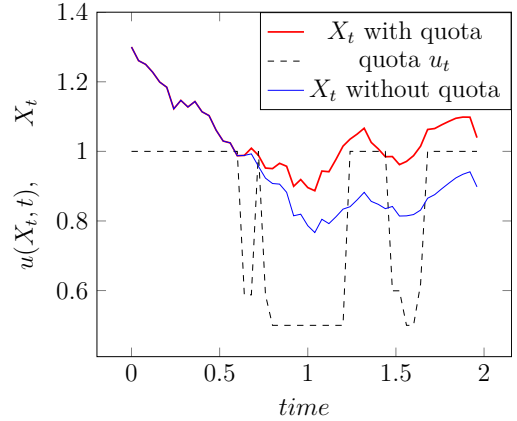


Figure 4: Simulation of the fishing model with a quota function computed by SDP and $X_0 = 1.3$.

5. Hamilton-Jacobi-Bellman solutions (HJB)

Let us return to (11) and note that:

$$v^m(\mathbf{X}) \approx \min_{\mathbf{u} \in \mathcal{U}} \left\{ |\mathbf{X} - \mathbf{X}_d|^2 - \boldsymbol{\alpha} \cdot \mathbf{u} + \frac{\beta}{h} \mathbb{E} \left[|\mathbf{u}^{m+1}(\zeta_h(\mathbf{X}, \mathbf{u}, \mathbf{z})) - \mathbf{u}(\mathbf{X})|^2 \right] + \mathbb{E}[v^{m+1}(\zeta_h(\mathbf{X}, \mathbf{u}, \mathbf{z}))] \right\}, \quad (13)$$

with $v^M = 0$. According to Remark 3

$$\begin{aligned} \mathbb{E} \left[|\mathbf{u}^{m+1}(\zeta_h(\mathbf{X}, \mathbf{u}, \mathbf{z})) - \mathbf{u}|^2 \right] &\approx \mathbb{E} \left[\left| \mathbf{u}^{m+1} \left(\mathbf{X} + \mathbf{X} \underline{\Lambda} \left[h(\mathbf{r} - \underline{\kappa} \mathbf{X} - \mathbf{u}) + \sqrt{h} \boldsymbol{\sigma} \mathbf{z} \right] \right) - \mathbf{u} \right|^2 \right] \\ &\approx \left| \mathbf{u}^{m+1}(\mathbf{X} + \mathbf{X} \underline{\Lambda} [h(\mathbf{r} - \underline{\kappa} \mathbf{X})]) - \mathbf{u} \right|^2 + \frac{h}{2} |\boldsymbol{\sigma} \mathbf{X} \mathbf{u}^{m+1}|^2. \end{aligned} \quad (14)$$

Also, by Itô formula,

$$\mathbb{E}[v^{m+1}(\zeta_h(\mathbf{X}, \mathbf{u}^m, \mathbf{z}))] = \mathbb{E}[v^{m+1}(\mathbf{X} + h \mathbf{X} \underline{\Lambda} [\mathbf{r} - \underline{\kappa} \mathbf{X} - \mathbf{u}^m] + \mathbf{X} \boldsymbol{\sigma} \sqrt{h} \mathbf{N}_{0,1}^d)]$$

$$= v^{m+1}(\mathbf{X} + h\mathbf{X}\underline{\Lambda}[\mathbf{r} - \underline{\kappa}\mathbf{X} - \mathbf{u}^m]) + \frac{h}{2}|\underline{\sigma}\mathbf{X}|^2 v''^{m+1}(\mathbf{X}). \quad (15)$$

Consequently, with v''^{m+1} replaced by v''^m for numerical convenience, (11) becomes

$$v^m(\mathbf{X}) \approx \min_{\mathbf{u}^m \in \mathcal{U}} \left\{ v^{m+1}(\mathbf{X} + h\mathbf{X}\underline{\Lambda}[\mathbf{r} - \underline{\kappa}\mathbf{X} - \mathbf{u}^m(\mathbf{X})]) + \frac{h}{2}|\underline{\sigma}\mathbf{X}|^2 v''^{m+1}(\mathbf{X}) + h|\mathbf{X} - \mathbf{X}_d|^2 - h\boldsymbol{\alpha} \cdot \mathbf{u}^m(\mathbf{X}) + \beta|\mathbf{u}^{m+1}(\mathbf{X} + h\mathbf{X}\underline{\Lambda}[\mathbf{r} - \underline{\kappa}\mathbf{X} - \mathbf{u}^m]) - \mathbf{u}^m(\mathbf{X})|^2 + h\frac{\beta}{2}|\underline{\sigma}\mathbf{X}\mathbf{u}^{m+1}|^2 \right\}. \quad (16)$$

The last line is an approximation motivated by the search for a stable implicit numerical scheme for v^m :

$$\frac{1}{h}v^m(\mathbf{X}) + \frac{|\underline{\sigma}\mathbf{X}|^2}{2}v''^m(\mathbf{X}) = \frac{1}{h}v^{m+1}(\mathbf{X} + h\mathbf{X}\underline{\Lambda}[\mathbf{r} - \underline{\kappa}\mathbf{X} - \mathbf{u}^m]) + |\mathbf{X} - \mathbf{X}_d|^2 - \boldsymbol{\alpha} \cdot \mathbf{u}^m(\mathbf{X}) + \frac{\beta}{h}|\mathbf{u}^{m+1}(\mathbf{X} + h\mathbf{X}\underline{\Lambda}[\mathbf{r} - \underline{\kappa}\mathbf{X} - \mathbf{u}^m]) - \mathbf{u}^m(\mathbf{X})|^2 + \frac{\beta}{2}|\underline{\sigma}\mathbf{X}|^2|\mathbf{u}^{m+1}|^2. \quad (17)$$

Furthermore, \mathbf{u}^m is given by minimising the right hand side of (17):

$$\mathbf{u}^m = \mathbf{u}^{m+1}(\zeta_h(\mathbf{X}, \mathbf{u}^m, 0) + \frac{h}{2\beta}(\boldsymbol{\alpha} + \underline{\Lambda}[\mathbf{X}]\nabla v^{m+1}(\eta_h(\mathbf{X}, \mathbf{u}^m, 0)) + o(h, \frac{h}{\beta})), \quad (18)$$

capped by the bounds u_m and u_M . Boundary conditions are not needed at $\mathbf{X} = 0$ because the PDE is self starting but for large \mathbf{X} , cancelation of the quadratic terms requires $v''^m(\mathbf{X}) \sim -2/|\underline{\sigma}|^2$.

Remark 4. Notice that (18) makes an important use of the comment in Remark 3.

5.1. Numerical results for a single species by HJB

The finite element method of degree 1 on intervals was used to approximate spatially (17). The linear systems were solved using the FreeFem++ software [15]. The range of \mathbf{X} was approximated by the segment $(0, 3)$ discretized into 160 intervals; 200 time steps were used.

With the same parameters as in Section 4.1 the results are shown on Figures 7, 6, 9, 10 and 8. These should be compared, respectively, with Figures 1, 5, 3, 4 and 2.

Except for the singularity at final time, which is not handled in the same way, the results are similar. HJB seems to handle better the penalty term with β as $t \mapsto u(t)$ is not bang-bang.

6. Optimisation with Markovian feedback neural network controls

In this section, we look for Markovian feedback controls, *i.e.*, functions $u : [0, T] \times \mathbb{R}^+ \rightarrow \mathbb{R}$, adapted to X_t , satisfying $u_m \leq u(x, t) \leq u_M$ for every (x, t) . We restrict our attention to such functions which are encoded by neural networks. This strategy has been used previously in the optimal control literature, see *e.g.* [12],[14],[10],[3].

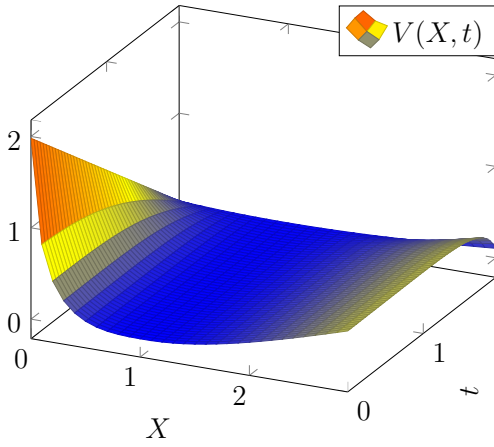


Figure 5: SDP solution: $V(X,t)$.

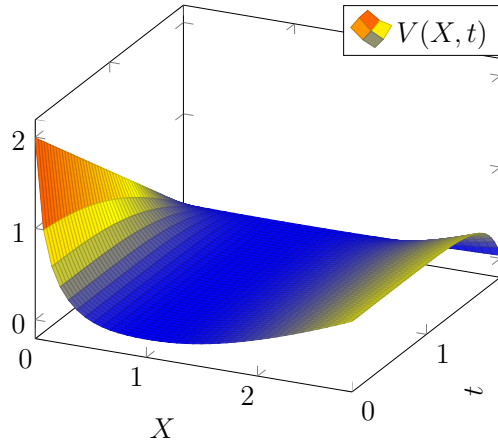


Figure 6: HJB solution: value function $V(X,t)$.

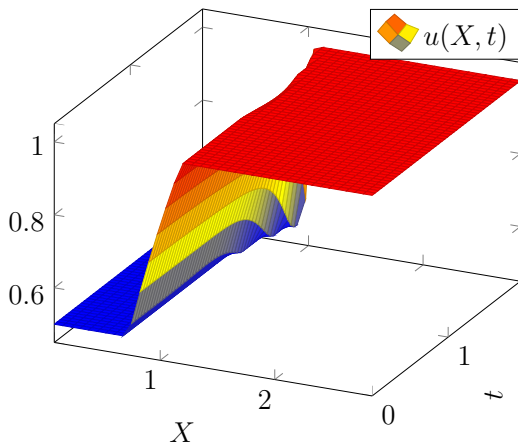


Figure 7: Solution $u(X,t)$ from HJB.

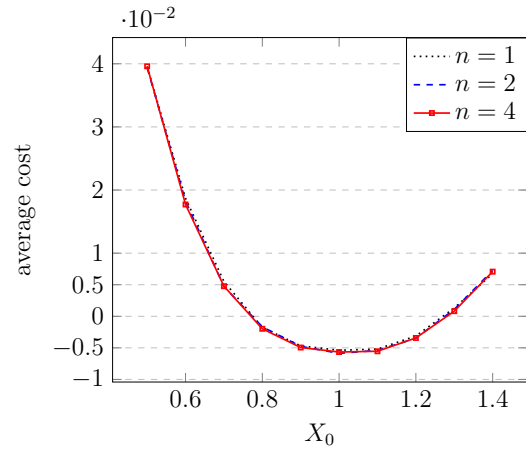


Figure 8: Values of the cost functions for the 3 space \times times meshes $40n \times 50n$, to solve the control problem by a HJB algorithm.

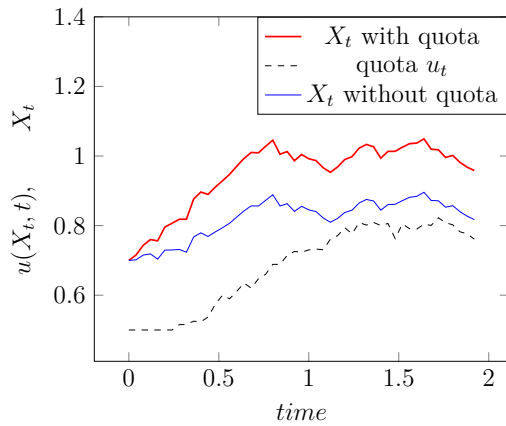


Figure 9: Simulation of the fishing model with a quota function computed by HJB and $X_0 = 0.7$.

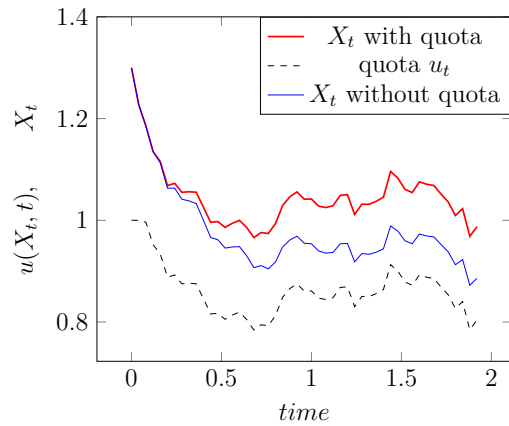


Figure 10: Simulation of the fishing model with a quota function computed by HJB and $X_0 = 1.3$.

Given an *activation* function $\psi : \mathbb{R} \rightarrow \mathbb{R}$ (e.g. a sigmoid or ReLU), let us introduce some helpful notation:

$$\mathbf{L}_{d_1, d_2} = \left\{ \phi : z \in \mathbb{R}^{d_1} \mapsto \{\psi(\beta_i + w_i \cdot z)\}_{i=1}^{d_2} \in \mathbb{R}^{d_2} \mid \beta \in \mathbb{R}^{d_2}, w \in \mathbb{R}^{d_2 \times d_1} \right\} \quad (19)$$

is the set of layer functions with input dimension d_1 and output dimension d_2 .

Let $\mathbf{d} = \{d_0, \dots, d_{\ell+1}\}$; define the set of feedforward fully connected neural networks with ℓ hidden layers and one output layer,

$$\mathbf{N}_{\mathbf{d}} = \left\{ u : \mathbb{R}^{d_0} \rightarrow \mathbb{R}^{d_{\ell+1}}, u = \phi^{(\ell)} \circ \phi^{(\ell-1)} \circ \dots \circ \phi^{(0)} \mid \phi^{(i)} \in \mathbf{L}_{d_i, d_{i+1}}, i = 0, \dots, \ell \right\} \quad (20)$$

where \circ denotes the composition of functions. Denote by Θ the set of parameters:

$$\Theta = \left\{ \theta := (\beta^{(0)}, w^{(0)}, \beta^{(1)}, w^{(1)}, \dots, \beta^{(\ell)}, w^{(\ell)}) \mid \beta^{(i)} \in \mathbb{R}^{d_i}, w^{(i)} \in \mathbb{R}^{d_i \times d_{i+1}} \right\}.$$

For each $\theta \in \Theta$, the corresponding network function will be denoted by \mathbf{u}_θ . Hence (20) may be rewritten as:

$$\mathbf{N}_{\mathbf{d}} = \{\mathbf{u}_\theta : \theta \in \Theta\}$$

For the fishing control problem $d_0 = d + 1$ (d species for the space variable, plus the time variable) and $d_{\ell+1} = d$ (the dimension of the control variable \mathbf{u}). Furthermore, to ensure the constraint $\mathbf{u}_m \leq \mathbf{u}(x, t) \leq \mathbf{u}_M$, it is sufficient to take the last activation function which satisfies this constraint. In the present implementation we have used $\mathbf{u}_m + (\mathbf{u}_M - \mathbf{u}_m)\sigma(x)$ where σ is the sigmoid function. Figure 11 shows a fully connected 2 layers neural network with 2 inputs, one output and 10 neurons in each layer.

The control problem is approximated by the minimisation over $\theta \in \Theta$ of:

$$\mathbb{J}(\theta) = \frac{h}{K} \sum_{k=1}^K \sum_{m=1}^M \|\mathbf{X}_k^m - \mathbf{X}_d\|^2 - \boldsymbol{\alpha} \cdot \mathbf{u}_\theta(\mathbf{X}_k^m, t^m) + \frac{\beta}{h} |\mathbf{u}_\theta(\mathbf{X}_k^m, t^m) - \mathbf{u}_\theta(\mathbf{X}_k^{m-1}, t^{m-1})|^2 \quad (21)$$

where \mathbf{X}_k^m is the result of one step of the Euler scheme (8) with $\mathbf{u}_\theta(\mathbf{X}_k^{m-1}, t^{m-1})$ and a realization of the noise.

Noting that (21) is a sum of terms, we can run a Stochastic Gradient Descent or one of its variants like ADAM. At each iteration, we sample a mini-batch of initial positions and realizations of the noise which allow us to compute random realizations of trajectories and compute a partial sum of (21). It is then used to compute a gradient and back-propagate it to adjust θ .

The stochastic gradient algorithm ADAM is used to update θ towards a local minimum of $\mathbb{J}(\theta)$ needs the gradient of $\mathbb{J}(\theta)$. A mini-batch is a random subset $B \subset \{1, \dots, K\}$; the gradients with respect to θ are computed as follows. Let $\mathbb{J}_B(\theta) := \frac{h}{K} \sum_{k \in B} \mathbf{j}_k(\theta)$ where

$$\mathbf{j}_k(\theta) = \sum_{m=1}^M \|\mathbf{X}_k^m - \mathbf{X}_d\|^2 - \boldsymbol{\alpha} \cdot \mathbf{u}_\theta(\mathbf{X}_k^m, t^m) + \frac{\beta}{h} |\mathbf{u}_\theta(\mathbf{X}_k^m, t^m) - \mathbf{u}_\theta(\mathbf{X}_k^{m-1}, t^{m-1})|^2.$$

The gradient of $\mathbf{j}_k(\theta)$ with respect to $\theta = \{\theta_n\}_n$ is made of:

$$\frac{d\mathbf{j}_k}{d\theta_n} = \sum_m \frac{d\mathbf{j}_k}{d\mathbf{u}^m} \frac{d\mathbf{u}^m}{d\theta_n} \quad \text{with} \quad \frac{d\mathbf{j}_k}{d\mathbf{u}^m} = P_k^m \mathbf{X}_k^m - \boldsymbol{\alpha} - 2\frac{\beta}{h} (\mathbf{u}^{m+1} - 2\mathbf{u}^m + \mathbf{u}^{m-1}), \quad (22)$$

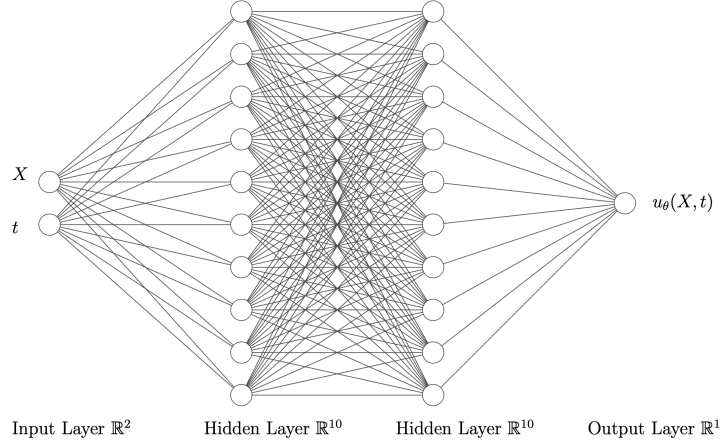


Figure 11: Sketch of a two layers neural network with 2 inputs, X, t and one output u . Each layer here has 10 neurons while for this article we used 100 neurons in each layers.

where P_k^m is defined by $P_k^{M-2} = 0$ and

$$P_k^{m-1} = P_k^m (1 + h(r - 2\kappa \mathbf{X}_k^m - \mathbf{u}^m) + \sigma dW_k^m,) - 2(\mathbf{X}_k^m - \mathbf{X}_d), \quad m = M - 2, \dots, 1. \quad (23)$$

Softwares for neural networks such as `tensorflow` provide automatic differentiation tools (back propagation) to compute $\frac{d\mathbf{u}^m}{d\theta_n}$.

Recall that ADAM algorithm consists in choosing B randomly and then

1. Let $\alpha = 0.001$, $m_1 = 0.9$, $m_2 = 0.999$, $\varepsilon = 10^{-8}$.

Loop on i :

2. Set $\mathbf{g}^i = \left\{ \frac{d\mathbb{J}_B}{d\theta_n^i} \right\}_{n=1}^N$
3. Set $\mathbf{p}^i = m_1 \mathbf{p}^{i-1} + (1 - m_1) \mathbf{g}^i$
4. Set $q^i = m_2 q^{i-1} + (1 - m_2) |\mathbf{g}^i|^2$
5. Set $m_j^i = m_j \cdot m_j^{i-1}$, $j = 1, 2$, $\hat{\mathbf{p}}^i = \frac{\mathbf{p}^i}{1 - m_1^i}$, $\hat{q}^i = \frac{q^i}{1 - m_2^i}$
6. Update $\theta_n^i = \theta_n^{i-1} - \frac{\alpha \cdot \hat{p}_n^i}{\sqrt{\hat{q}^i} + \varepsilon}$, $n = 1, \dots, N$, the number of parameters.

6.1. Numerical results using a neural network for a single species

Notice that we use the tools of AI but there is no learning phase from known solutions, rather the learning phase is simply to find the best parameters to achieve the minimum of a given cost function.

To check the results we implemented also a single layer neural network with 15000 neurons directly in C++ using the automatic differentiation in reverse mode of the library `adept` [16] and a conjugate gradient algorithm instead of ADAM. Convergence is fast (20 to 40 iterations) but this simple method does not work for multiple layered neural network because of memory limitation.

For the single species numerical test, the results are the same whether obtained by the one layer or the two layers networks.

Results are shown on Figures 12, 14, 15 and 13 and should be compared, respectively, with Figures 7, 9, 10 and 8 and/or, respectively, with Figures 1, 3, 4 and 2.

Note that the results are closer to those obtained with HJB than those obtained with SDP.

Our general impression is that all 3 methods are equal, with a slight reserve for SDP for which the control oscillates much more than with the other two methods, implying that it does not handle as well the β -penalisation term.

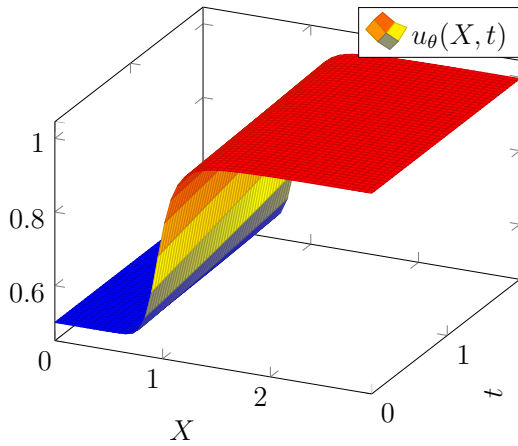


Figure 12: Dynamic feedback control, $u_\theta(X, t)$ computed by the Markovian Neural Network.

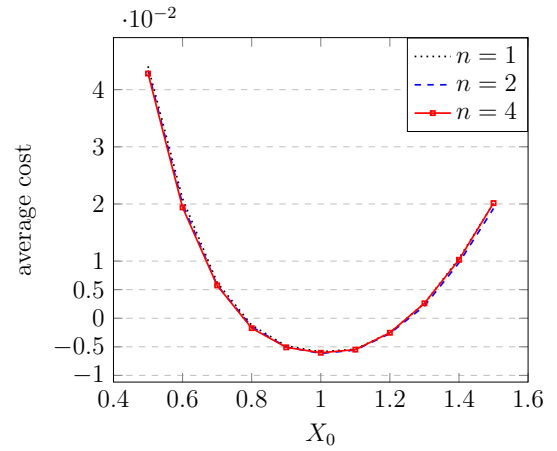


Figure 13: Single species: Values of the cost versus X_0 for 100 realizations with the Markovian neural network, for the 3 time meshes $50 \times n$.

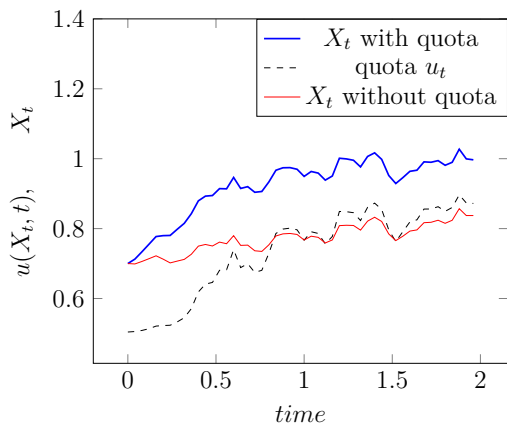


Figure 14: Simulation for a single fish species computed by the Markovian Neural Network and $X_0 = 0.7$. Performance with and without quota u_t .

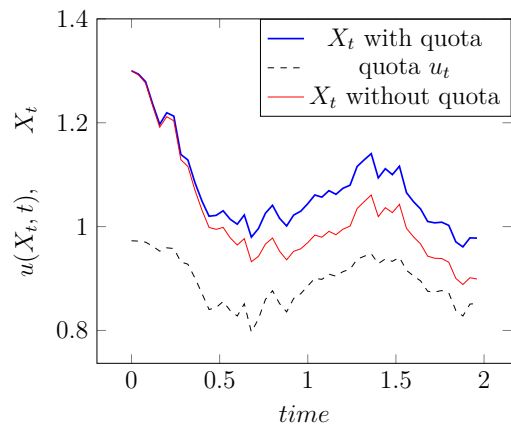


Figure 15: Simulation for a single fish species computed by the Markovian Neural Network and $X_0 = 1.3$. Performance with and without quota u_t .

7. Numerical results: 3 species

We now turn our attention to an example with 3 species. Consider the following interaction matrix between species:

$$\kappa = \begin{pmatrix} 1.2 & -0.1 & 0 \\ 0.2 & 1.2 & 0 \\ 0.1 & 0.1 & 1 \end{pmatrix}$$

All other parameters are as in Section 4.1 and $X_d = (1, 1, 1)^T$.

With SDP (Stochastic Dynamic Programming) there is a difficulty: the minimisation in (12) is now 3 dimensional and cannot be obtained by dichotomy. So the method was not tested. We have compared HJB and Neural Network optimisation with 1 and then with 2 layers. The results differ; according to the last subsection none of the methods found the true solution.

7.1. Numerical results for 3 species with HJB

The computation is done with 80 time steps. Here too a finite element discretisation was used with P^1 elements on tetraedra. The mesh for the cube $(0, 3)^3$ is obtained from an automatic mesh generator from a $30 \times 30 \times 30$ surface mesh resulting into 29791 vertices and 16200 tetraedra. It took 7 min on an M1 Apple laptop. The vector valued optimal control $\mathbf{X}, t \in \mathbb{R}^3 \times (0, T) \mapsto \mathbf{u} \in (0.5, 1)^3$ computed with HJB is shown on Figures 16 to 24. In these, two of the 3 coordinates of \mathbf{X} , are fixed at their X_d values.

Then the fishing model is integrated with the optimal \mathbf{u} and a random realization of $d\mathbf{W}$ for 2 values of \mathbf{X}_0 : $\mathbf{X}_0 = (0.7, 0.7, 0.7)^T$ or $(1.3, 1.3, 1.3)^T$. The results are compared with a similar simulation without quota, *i.e.*, $\mathbf{u} \equiv 1$. See Figures 25, 26, 27. The quality of the optimisation is seen visually when X_t is closest to X_d and numerically from the lowest value of the cost function, plotted versus \mathbf{X}_0 on Figure 28.

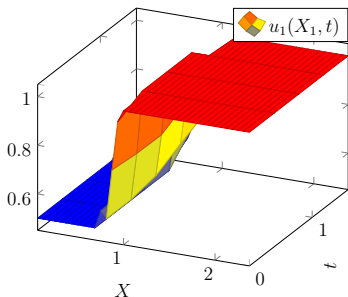


Figure 16: HJB: $X_1, t \mapsto u_1$.

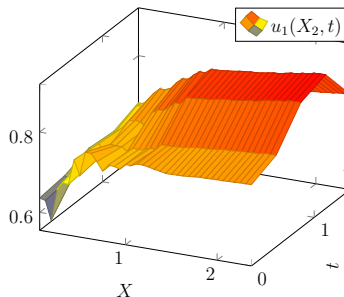


Figure 17: HJB: $X_2, t \mapsto u_1$.

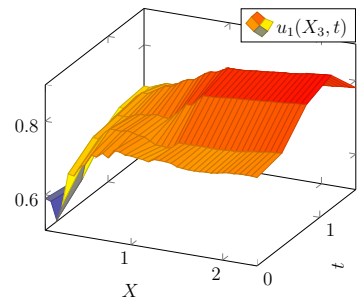


Figure 18: HJB: $X_3, t \mapsto u_1$.

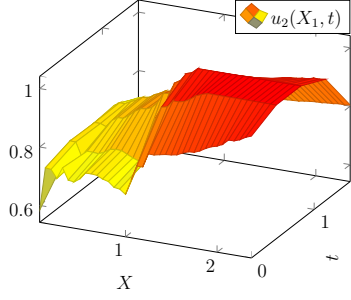


Figure 19: HJB: $X_1, t \mapsto u_2$.

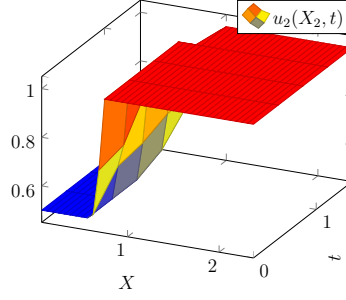


Figure 20: HJB: $X_2, t \mapsto u_2$.

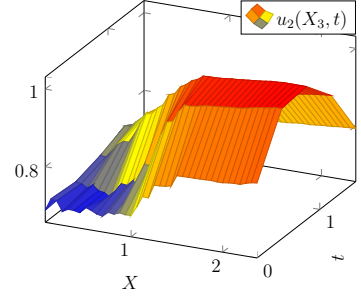


Figure 21: HJB: $X_3, t \mapsto u_2$.

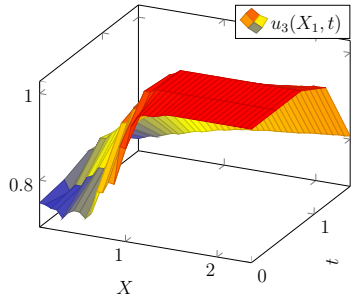


Figure 22: HJB: $X_1, t \mapsto u_3$.

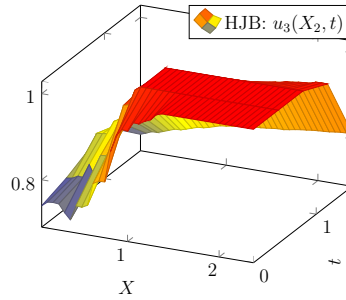


Figure 23: HJB: $X_2, t \mapsto u_3$.

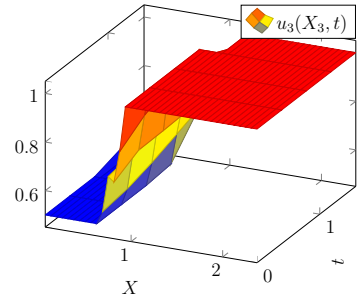


Figure 24: HJB: $X_3, t \mapsto u_3$.

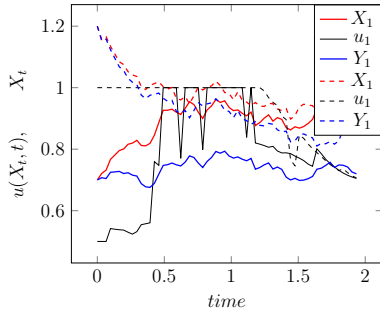


Figure 25: HJB: Optimal biomass and quota function when $X_1(0) = 0.7$ and $X_1(0) = 1.3$.

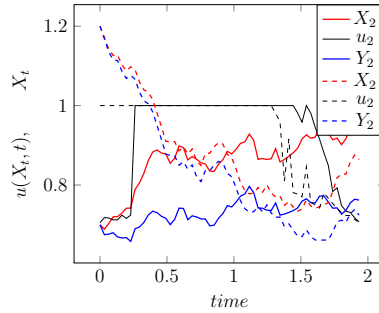


Figure 26: HJB: Optimal biomass and quota function when $X_2(0) = 0.7$ and $X_2(0) = 1.3$.

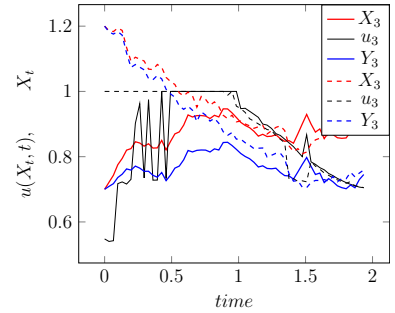


Figure 27: HJB: Optimal biomass and quota function when $X_3(0) = 0.7$ and $X_3(0) = 1.3$.

7.2. Optimisation with a Neural Network with one layer of 15000 Neurons

The parameters are the same. The neural network has one layer with 15000 neurons. Conjugate gradient with optimal step size is used and the derivatives are computed with automatic differentiation in reverse mode.

Convergence of the conjugate gradient algorithm is seen on Figure 29. The norm of the gradient is reduced from 0.1 to 0.00017. Then the results are displayed as above: first the optimal quota vector function, with 9 surfaces on Figures 30 to 38. The fact that the surface $\{t, X_2\} \mapsto u_2(1, X_2, 1, t)$ is flat is an obvious indication that the solution found by the NN with one layer is only suboptimal.

Then two types of random trajectories, one starting at $X_0 = (0.7, 0.7, 0.7)^T$, the other at $(1.3, 1.3, 1.3)^T$. Results are shown on Figures 39, 40 and 41. The section ends with a display of the cost function versus \mathbf{X}_0 on Figure 42. The results seem less accurate than with HJB. Figure 34 is surprising! It shows also on Figure 40 where the optimal quota

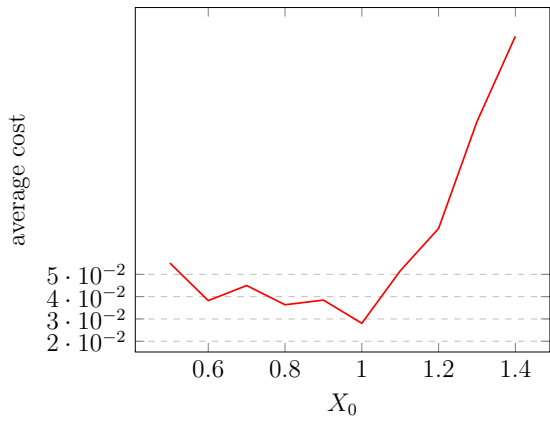


Figure 28: Values of the average cost versus X_0 for 20 realizations by HJB for the 3 species problem.

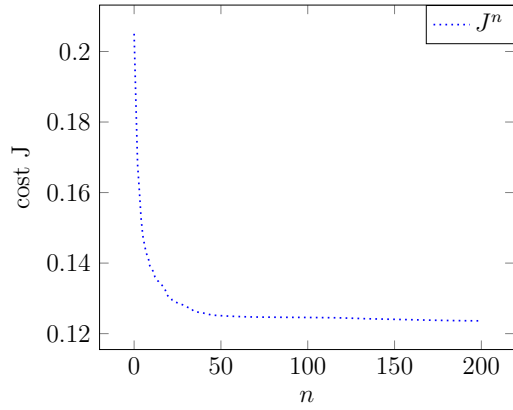


Figure 29: convergence of the cost function during the optimisation process to solve the 3D problem with a NN with one layer of 150000 neurons.

does not steer \mathbf{X} anywhere near \mathbf{X}_d .

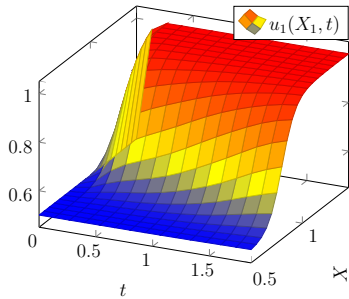


Figure 30: $X_1, t \mapsto u_1$.

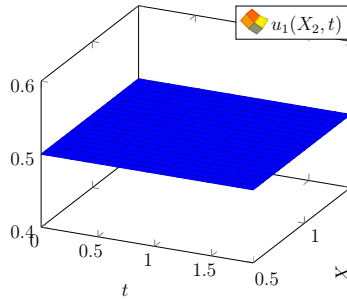


Figure 31: $X_2, t \mapsto u_1$.

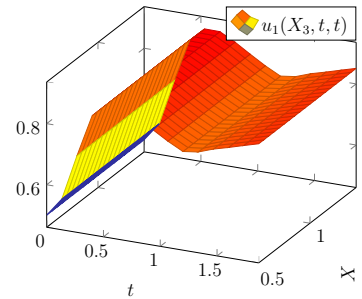


Figure 32: $X_3, t \mapsto u_1$.

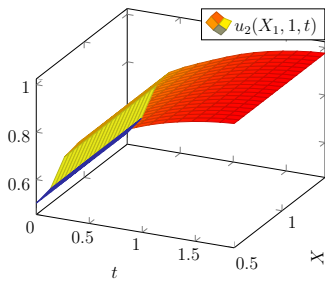


Figure 33: $X_1, t \mapsto u_2$.

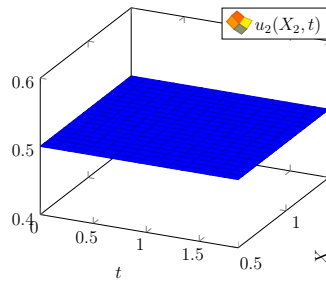


Figure 34: $X_2, t \mapsto u_2$.

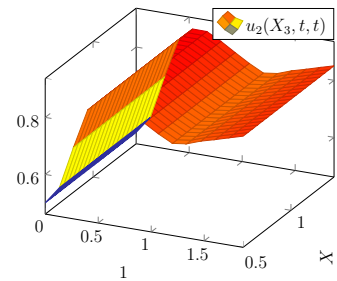


Figure 35: $X_3, t \mapsto u_2$.

7.3. With two layers

The same problem is solved with a two layers neural network with 100 neurons on each layer.

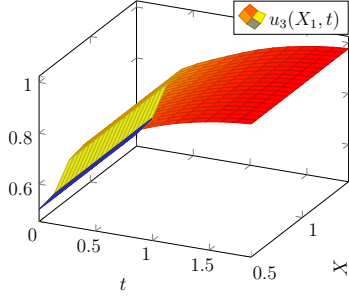


Figure 36: $X_1, t \mapsto u_3$.

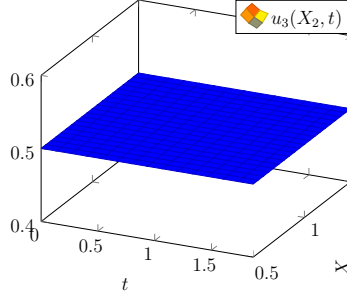


Figure 37: $X_2, t \mapsto u_3$.

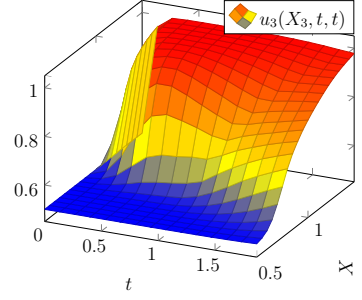


Figure 38: $X_3, t \mapsto u_3$.

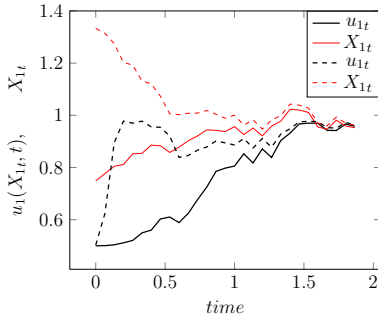


Figure 39: Optimal biomass and quota function computed with the single layer NN when $X_0 = 0.7$ and $X_0 = 1.3$.

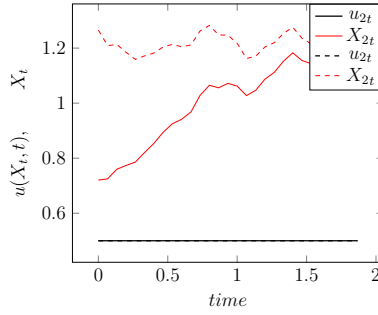


Figure 40: Optimal biomass and quota function computed with the single layer NN when $X_0 = 0.7$ and $X_0 = 1.3$.

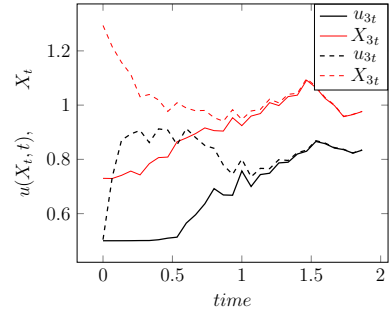


Figure 41: Optimal biomass and quota function computed with the single layer NN when $X_0 = 0.7$ and $X_0 = 1.3$.

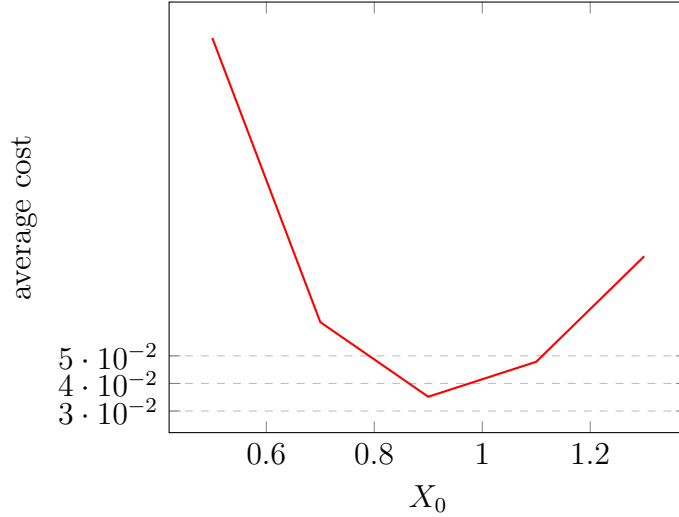


Figure 42: Values of the average cost versus X_0 for 10 realizations of dW and with the Markovian neural network with 15000 neurons and 10 conjugate gradient iterations.

The results are displayed as above: first the optimal quota vector function, with 9 surfaces on Figures 43 to 51.

Then two types of random trajectories, one starting at $X_0 = (0.7, 0.7, 0.7)^T$, the other at $(1.3, 1.3, 1.3)^T$. Results are shown on Figures 52, 53 and 54. The section ends with a

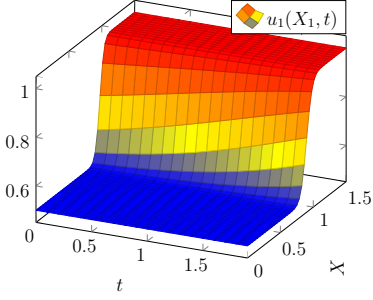


Figure 43: $X_1, t \mapsto u_1$.

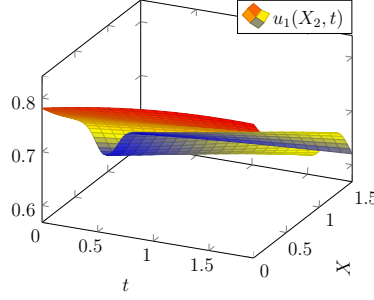


Figure 44: $X_2, t \mapsto u_1$.

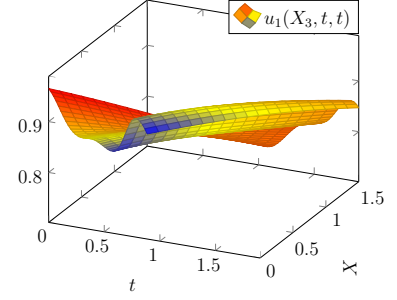


Figure 45: $X_3, t \mapsto u_1$.

display of the cost function versus \mathbf{X}_0 on Figure 55.

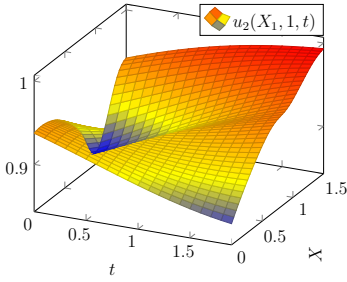


Figure 46: $X_1, t \mapsto u_2$.

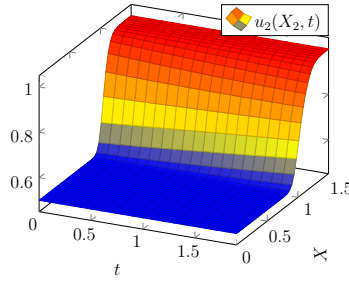


Figure 47: $X_2, t \mapsto u_2$.

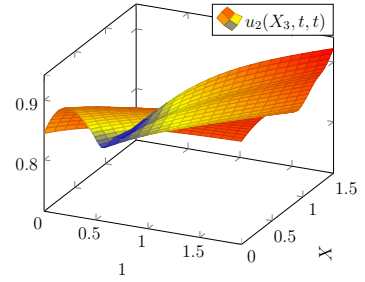


Figure 48: $X_3, t \mapsto u_2$.

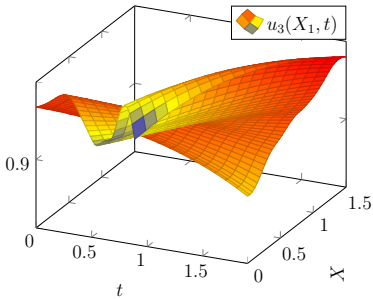


Figure 49: $X_1, t \mapsto u_3$.

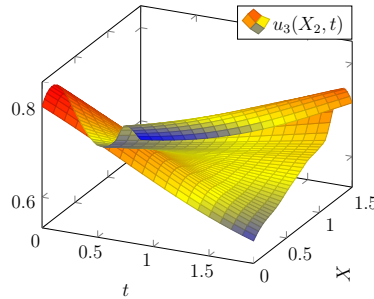


Figure 50: $X_2, t \mapsto u_3$.

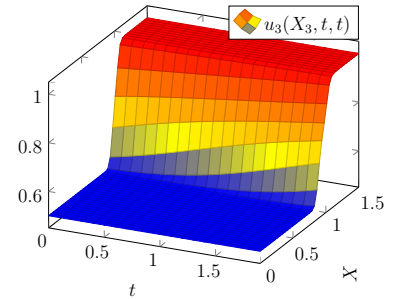


Figure 51: $X_3, t \mapsto u_3$.

Consequently, the neural network with 2 layers give decent results, better than HJB. The neural network with one layer only gives poor results.

7.4. Assessment of the results

Remark 1 allows the construction of a simple 3D solution from a 1D solution.

Let v, y be solution of

$$\min_v \left\{ \int_0^T \mathbb{E} \left[|y - y_d|^2 - \alpha v + \beta |v'|^2 \right] dt : dy_t = y_t(r - y + v + \sigma dw_t), y(0) = y_0 \right\}$$

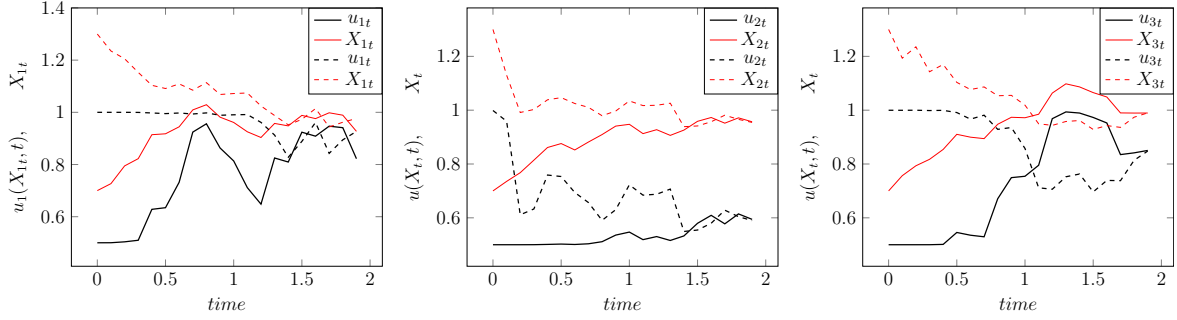


Figure 52: Optimal biomass and Figure 53: Optimal biomass and Figure 54: Optimal biomass and quota function computed with quota function computed with quota function computed with the two layers NN when $X_0 = 0.7$ the two layers NN when $X_0 = 0.7$ the two layers NN when $X_0 = 0.7$ and $X_0 = 1.3$. and $X_0 = 1.3$. and $X_0 = 1.3$.

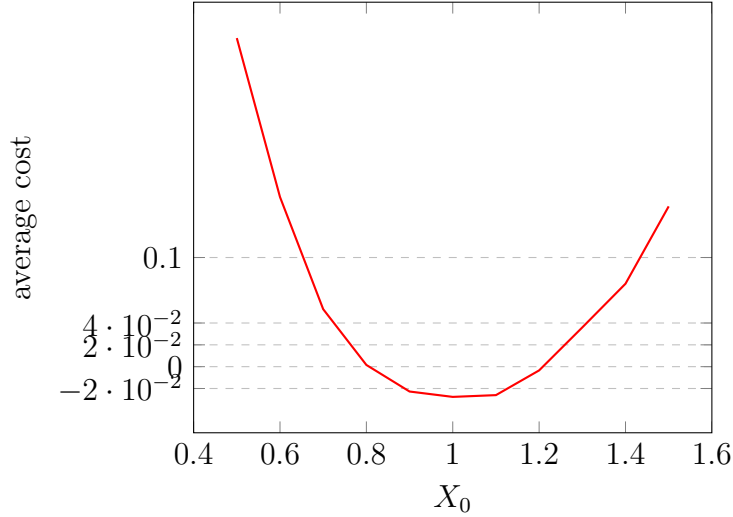


Figure 55: Values of the average cost versus X_0 for 100 realizations of dW and with the Markovian neural network with two layers of neurons.

Let $\underline{\kappa} \in \mathbb{R}^{3 \times 3}$, $\mathbf{u} = (v, v, v)^T$, $\mathbf{r} = (r, r, r)^T$, $\boldsymbol{\alpha} = (\alpha, \alpha, \alpha)^T$, $\underline{\boldsymbol{\sigma}} = \underline{\boldsymbol{\Lambda}}[(\sigma, \sigma, \sigma)^T]$, $\mathbf{Y} = (y, y, y)^T$ and $d\mathbf{W}_t = (dw_t, dw_t, dw_t)^T$. Then

$$d\mathbf{Y}_t = \underline{\boldsymbol{\Lambda}} [(\mathbf{r} - \mathbf{u} - \mathbf{Y})dt + \underline{\boldsymbol{\sigma}}d\mathbf{W}_t] \mathbf{Y}$$

Let $\mathbf{X} = \underline{\kappa}^{-1}\mathbf{Y}$. It implies

$$d\mathbf{X}_t = \underline{\kappa}^{-1}\underline{\boldsymbol{\Lambda}} [(\mathbf{r} - \mathbf{u} - \underline{\kappa}\mathbf{X})dt + \underline{\boldsymbol{\sigma}}d\mathbf{W}_t] \underline{\kappa}\mathbf{X}.$$

The matrix $\underline{\boldsymbol{\Lambda}} [(\mathbf{r} - \mathbf{u} - \underline{\kappa}\mathbf{X})dt + \underline{\boldsymbol{\sigma}}d\mathbf{W}_t]$ is diagonal and all terms are equal, so it commutes with $\underline{\kappa}$. Therefore

$$d\mathbf{X}_t = \underline{\boldsymbol{\Lambda}} [(\mathbf{r} - \mathbf{u} - \underline{\kappa}\mathbf{X})dt + \underline{\boldsymbol{\sigma}}d\mathbf{W}_t] \mathbf{X}.$$

and \mathbf{u} is solution of

$$\min_{u \in \mathcal{U}} \bar{J} := \mathbb{E} \left[\int_0^T \left[|\underline{\kappa}(\mathbf{X} - \underline{\kappa}^{-1}\mathbf{Y}_d)|^2 dt - \boldsymbol{\alpha} \cdot \mathbf{u} + \beta \left| \frac{d\mathbf{u}}{dt} \right|^2 \right] dt \right]$$

because this is 3 times the cost function of the problem in y, v .

Let \mathbf{X}, \mathbf{u} be the solution of this problem, then by construction $\mathbf{X} = \underline{\boldsymbol{\kappa}}^{-1}\mathbf{Y}$ and

$$\mathbf{u}(X_1, X_2, X_3) = (v(Y_1), v(Y_2), v(Y_3))^T = (v(y), v(y), v(y))^T.$$

For instance, $\mathbf{u}_1(1, X_2, 1) = v((\underline{\boldsymbol{\kappa}}(1, X_2, 1)^T)_1)$.

Now we build on the fact that v is approximatively bang-bang and equal to 0.5 when $y < 1$ and 1 otherwise.

Example

$$\underline{\boldsymbol{\kappa}} = \begin{pmatrix} 1.2 & -0.1 & 0 \\ 0.2 & 1.2 & 0 \\ 0.1 & 0.1 & 1 \end{pmatrix} \quad \underline{\boldsymbol{\kappa}}^{-1} = \begin{pmatrix} 0.822 & 0.0685 & 0 \\ -0.137 & 0.822 & 0 \\ -0.0685 & -0.089 & 1 \end{pmatrix} \quad \underline{\boldsymbol{\kappa}}^{-1} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1.16 \\ 0.89 \\ 1.09 \end{pmatrix} \times 0.77$$

The scaling 0.77 plays no role.

$$\mathbf{u}(X_1, X_{d2}, X_{d3}) = v \left(\underline{\boldsymbol{\kappa}} \begin{pmatrix} X_1 \\ 0.89 \times 0.77 = 0.685 \\ 1.09 \times 0.77 = 0.839 \end{pmatrix} \right) = v \left(\begin{pmatrix} 1.2X_1 - 0.69 \\ 0.2X_1 + 0.82 \\ 0.1X_1 + 0.91 \end{pmatrix} \right)$$

Hence $\mathbf{u}_1(X_1, X_{d2}, X_{d3})$ is expected to be bang-bang at $X_1 = 0.56$, $\mathbf{u}_2(X_1, 1, 1)$ is expected to be bang bang at $X_1 = 0.9$, and $\mathbf{u}_3(X_1, 1, 1)$ is expected to be bang bang at $X_1 = 0.9$.

$$\mathbf{u}(X_{d1}, X_2, X_{d3}) = v \left(\underline{\boldsymbol{\kappa}} \begin{pmatrix} 1.16 \times 0.77 = 0.89 \\ X_2 \\ 1.09 \times 0.77 = 0.89 \end{pmatrix} \right) = v \left(\begin{pmatrix} 1.07 - 0.1X_2 \\ 0.18 + 1.2X_2 \\ 0.98 + 0.1X_2 \end{pmatrix} \right)$$

Hence $\mathbf{u}_1(X_{d1}, X_2, X_{d3})$ is expected to be bang bang at $X_2 = 0.7$, and $\mathbf{u}_2(X_{d1}, X_2, X_{d3})$ is expected to be bang-bang at $X_2 = 0.68$ and $\mathbf{u}_3(X_{d1}, X_2, X_{d3})$ is expected equal to be bang bang at $X_2 = 0.2$.

$$\mathbf{u}(X_{d1}, X_{d2}, X_3) = v \left(\underline{\boldsymbol{\kappa}} \begin{pmatrix} 1.16 \times 0.77 = 0.89 \\ 0.89 \times 0.77 = 0.685 \\ X_3 \end{pmatrix} \right) = v \left(\begin{pmatrix} 1.0 \\ 1.0 \\ 0.16 + X_3 \end{pmatrix} \right)$$

Hence $\mathbf{u}_1(X_{d1}, X_{d2}, X_3)$ and $\mathbf{u}_2(X_{d1}, X_{d2}, X_3)$, are expected equal to 1 everywhere and $\mathbf{u}_3(X_{d1}, X_{d2}, X_3)$ is expected to be bang-bang at $X_3 = 0.84$.

7.5. Optimisation with HJB when $X_d = \underline{\boldsymbol{\kappa}}^{-1}\mathbf{1}$

The same numerical test was done with HJB but with $\mathbf{X}_d = (1.16, 0.089, 1.09)^T$. The results are shown on Figures 56 to 67. The lowest cost is 0.04, for $X_0 = 1$.

The solution is nearer to the one constructed above but still fairly different. The Neural Networks performances are poor on this test. It is likely that they produced a local minimum.

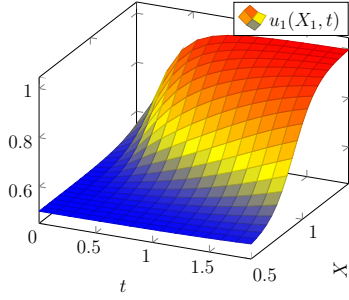


Figure 56: 3 species: $X_1, t \mapsto u_1$.

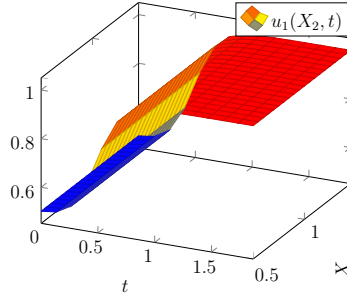


Figure 57: 3 species: $X_2, t \mapsto u_1$.

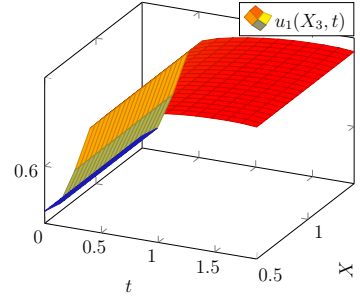


Figure 58: 3 species: $X_3, t \mapsto u_1$.

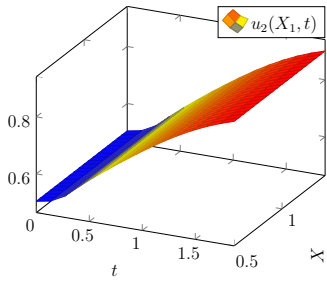


Figure 59: 3 species: $X_1, t \mapsto u_2$.

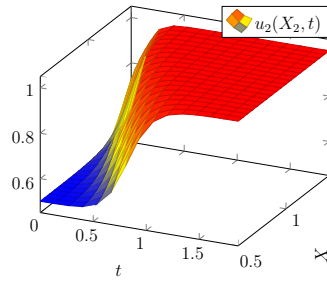


Figure 60: 3 species: $X_2, t \mapsto u_2$.

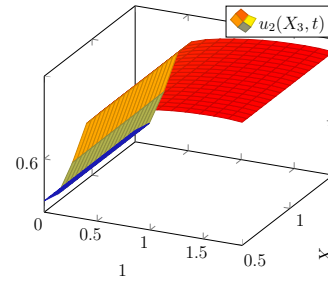


Figure 61: 3 species: $X_3, t \mapsto u_2$.

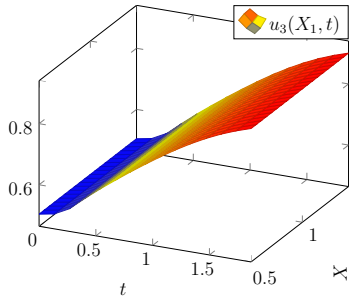


Figure 62: 3 species: $X_1, t \mapsto u_3$.

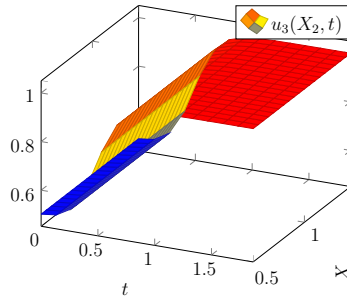


Figure 63: 3 species: $X_2, t \mapsto u_3$.

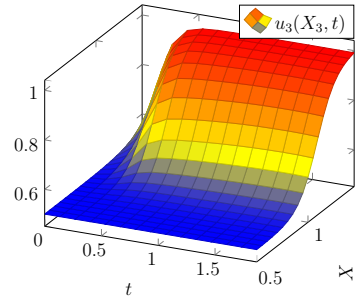


Figure 64: 3 species: $X_3, t \mapsto u_3$.

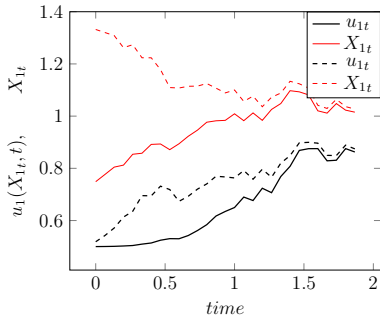


Figure 65: 3 species: Optimal biomass and quota function computed with the single layer NN when $X_0 = 0.7$ and $X_0 = 1.3$.

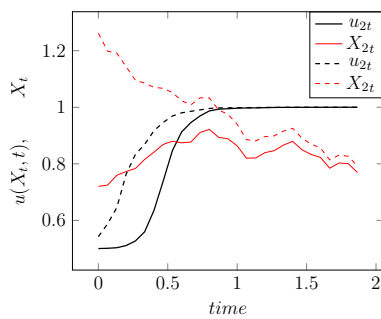


Figure 66: 3 species: Optimal biomass and quota function computed with the single layer NN when $X_0 = 0.7$ and $X_0 = 1.3$.

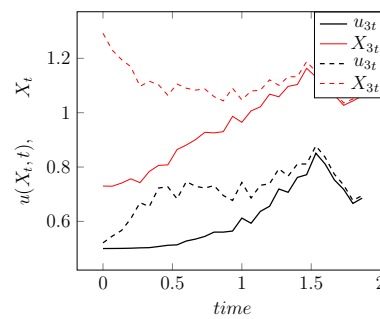


Figure 67: 3 species: Optimal biomass and quota function computed with the single layer NN when $X_0 = 0.7$ and $X_0 = 1.3$.

8. Numerical results: 5 species

The great advantage of Neural Network optimisation is that it scales well with dimensions. So to show that it is possible, with the same computer code with very few modifications we computed with the one-layer NN with 150000 neurons a case with 5 species. 20 iterations of conjugate gradients were done.

All parameters are as above except the species correlation matrix:

$$\boldsymbol{\kappa} = \begin{pmatrix} 1.2 & -0.1 & 0.0 & 0.0 & -0.1 \\ 0.2 & 1.2 & 0.0 & 0.0 & -0.1 \\ 0.0 & 0.2 & 1.2 & -0.1 & 0.0 \\ 0.0 & 0.0 & 0.1 & 1.2 & 0.0 \\ 0.1 & 0.1 & 0.0 & 0.0 & 1.2 \end{pmatrix}$$

Only components 1,2,5 are shown see Figures 68 to 76. On Figures 77, 78 and 79 trajectories with optimal quotas show that the NN solution is in general driving \mathbf{X} towards \mathbf{X}_d .

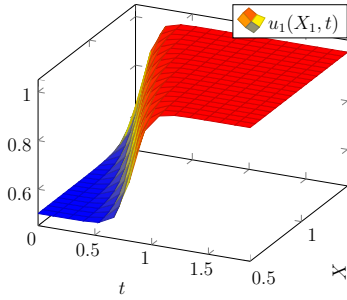


Figure 68: 5 species: $X_1, t \mapsto u_1$.

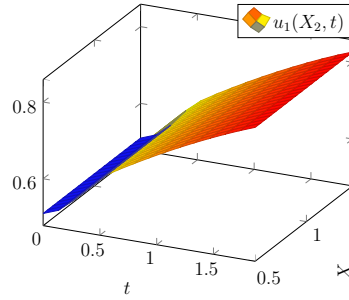


Figure 69: 5 species: $X_2, t \mapsto u_1$.

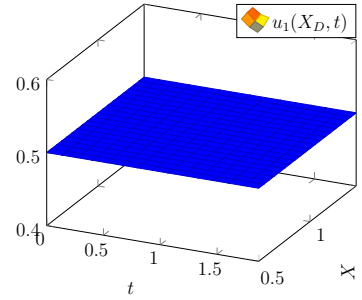


Figure 70: 5 species: $X_D, t \mapsto u_1$.

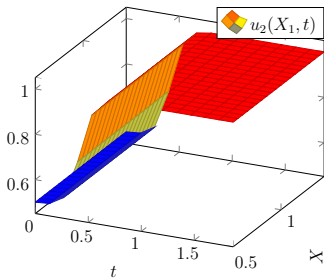


Figure 71: 5 species: $X_1, t \mapsto u_2$.

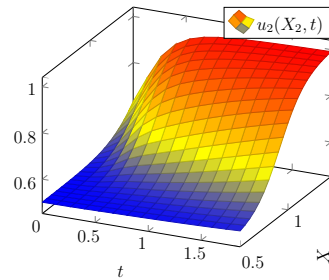


Figure 72: 5 species: $X_2, t \mapsto u_2$.

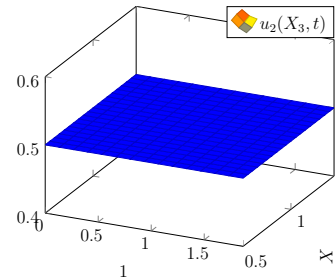


Figure 73: 5 species: $X_3, t \mapsto u_2$.

The minimum of the cost function is 0.038 for $\mathbf{X}_0 = 1$.

9. Conclusion

Control of the biomass of a fishing site has been here a mathematical opportunity to test the numerical methods at hand. One of the advantage of the model is that it is

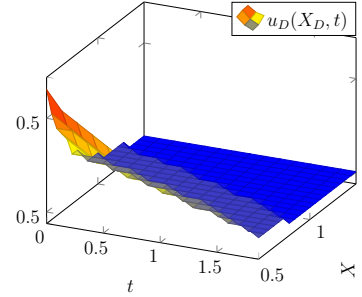
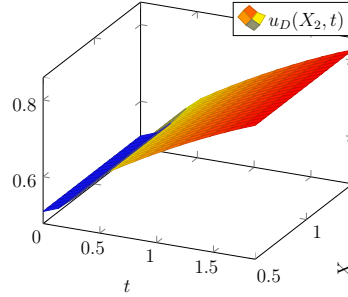
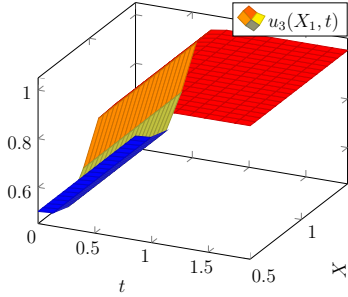


Figure 74: 5 species: $X_1, t \mapsto u_3$. Figure 75: 5 species: $X_2, t \mapsto u_3$. Figure 76: 5 species: $X_3, t \mapsto u_3$.

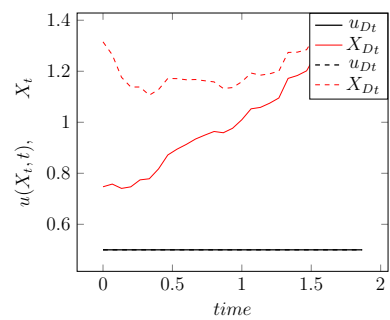
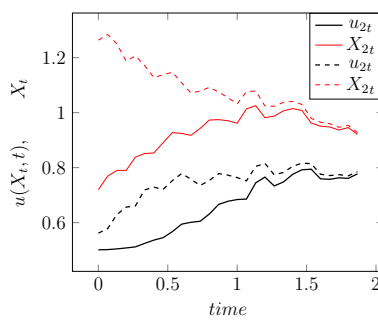
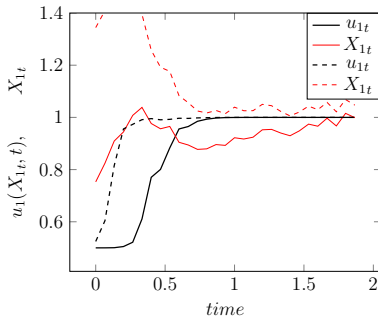


Figure 77: 5 species: Optimal biomass and quota function computed with the single layer NN when $X_0 = 0.7$ and $X_0 = 1.3$. Figure 78: 5 species: Optimal biomass and quota function computed with the single layer NN when $X_0 = 0.7$ and $X_0 = 1.3$. Figure 79: 5 species: Optimal biomass and quota function computed with the single layer NN when $X_0 = 0.7$ and $X_0 = 1.3$.

meaningful in any dimension, the number of species. Thus it is a testbed for stochastic control numerical methods.

We have compared Hamilton-Jacobi-Bellman solutions and stochastic dynamic programming with two implementation of a neural network based optimisation.

The later is conceptually very simple and the computer libraries of AI and Automatic Differentiation can be used. But they need to be validated and it is the object of this article.

Stochastic dynamic programming is difficult to use numerically beyond dimension 2 and HJB solutions do not scale beyond dimension 4, unless sophisticated discretisation tools are used like sparse grids.

Neural network based optimisation can be used for large dimension problems but assessing the precision of the answer seems difficult. Here in dimension 3 the one layer network did not work well and with the two layers network we could observe discrepancies with HJB solutions. In dimension 5, the numerical solution seems reasonable but it is probably suboptimal.

References

- [1] P.M. Allen and J.M. McGlade: Modelling complex human systems: A fisheries example. European Journal of Operational Research 30 (1987) 147-167.
- [2] P. Auger and O. Pironneau, Parameter Identification by Statistical Learning of a

Stochastic Dynamical System Modelling a Fishery with Price Variation. Comptes-Rendus de l'Académie des Sciences. May 2020.

- [3] A. Bachouch, C.Huré, N. Langrené and Huyên Pham; Deep neural networks algorithms for stochastic control problems on finite horizon: numerical applications; arXiv:1812.05916v3 [math.OC] 27 Jan 2020.
- [4] S. Balakrishnan and V. Biega. Adaptive-critic-based neural networks for aircraft optimal control. *Journal of Guidance, Control and Dynamics*, 19(4), 893–898. 1996.
- [5] R. Bellman, *Dynamic Programming*, Princeton, NJ: Princeton University Press, 1957.
- [6] D. Bertsekas, *Reinforced Learning and Optimal Control*. Athena Scientific, Belmont Mass. 2019.
- [7] A. Bick. Quadratic-Variation-Based dynamic strategies. *Management Sciences*, vol 41, No 4, p722-732 (1995).
- [8] S. Boyd and C. Barratt: *Linear Controller Design – Limits of Performance*. Prentice-Hall, 1991.
- [9] T. Brochier, P. Auger, D. Thiao, A. Bah, S. Ly, T. Nguyen Huu, P. Brehmer. Can overexploited fisheries recover by self-organization? Reallocation of the fishing effort as an emergent form of governance. *Marine Policy*, 95 (2018) 46-56.
- [10] R. Carmona and M. Laurière: Convergence Analysis of Machine Learning Algorithms for the Numerical Solution of Mean Field Control and Games: II–The Finite Horizon Case. arXiv preprint arXiv:1908.01613, 2019. To Appear in *Annals of Applied Probability*.
- [11] F. Chollet : *Deep learning with Python*. Manning publications (2017).
- [12] E Gobet and R Munos. Sensitivity Analysis Using Itô–Malliavin Calculus and Martingales, and Application to Stochastic Optimal Control. *SIAM Journal on control and optimisation*, 2005
- [13] I. Goodfellow, Y. Bengio and A. Courville (2016): *Deep Learning*, MIT-Bradford.
- [14] J. Han and W. E. Deep learning approximation for stochastic control problems. *Deep Reinforcement Learning Workshop, NIPS (2016)*
- [15] F. Hecht (2012): New development in FreeFem++, *J. Numer. Math.*, 20, pp. 251-265. (see also www.freefem.org.)
- [16] R. G. Hogan : Fast reverse-mode automatic differentiation using expression templates in C++. *ACM Trans. Math. Softw.*, 40, 26:1-26:16 (2014) : www.met.reading.ac.uk/clouds/adept/
- [17] R. Kamalapurkar and P. Walters and J. Rosenfeld and W. Dixon, *Reinforcement Learning for Optimal Feedback Control*. Springer 2018.

- [18] M. Laurière and O. Pironneau : Dynamic Programming for mean-field type control J. Optim. Theory Appl. 169 (2016), no. 3, 902–924.
- [19] C. Le Bris and P.L. Lions, Existence and uniqueness of solutions to Fokker-Planck type equations with irregular coefficients. Comm. Partial Differential Equations, 33, 1272-1317 , 2008.
- [20] A. Moussaoui and P. Auger: A bioeconomic model of a fishery with saturated catch and variable price: Stabilizing effect of marine reserves on fishery dynamics. Ecological Complexity 45 (2021) 100906.
- [21] A. Moussaoui, M. Bensenane, P. Auger, A. Bah, On the optimal size and number of reserves in a multi-site fishery model, Journal of Biological Systems. Vol. 23, No. 01, pp. 31-47 (2015)
- [22] G. Pagès. *Numerical Probability: An Introduction with Applications to Finance*. Springer, Berlin, 2018, 574p.
- [23] G. Pagès, H. Pham and J. Printems: An Optimal Markovian Quantization Algorithm For Multi-Dimensional Stochastic Control Problems, *Stochastics and Dynamics* , 4(4):501–545, 2004.
- [24] G. Pagès and O. Pironneau, Protection of a fishing Site with Optimal Quotas: Dynamic Programming versus Supervised Learning. Encyclopedia, E. Trélat ed. (to appear)
- [25] G. Pagès, J. Printems. Optimal quadratic quantization for numerics: the Gaussian case, *Monte Carlo Methods and Appl.*, 9(2):135–165, 2003.
- [26] J. Yong and X. Y. Zhou: *Stochastic Controls Hamiltonian Systems and HJB Equations* Application of Mathematics series vol 43. Springer 1991.