Development of a Q-Learning Algorithm for Model-free Autonomous Soaring

Erwan Lecarpentier[1], Sebastian Rapp[2], Marc Melo and Emmanuel Rachelson[3]

***Abstract*— Autonomous unpowered flight is a challenge for control and guidance systems: all the energy the aircraft might use during flight has to be harvested directly from the atmosphere. We investigate the design of an algorithm that optimizes the closed-loop control of a glider's bank and sideslip angles, while flying in the lower convective layer of the atmosphere in order to increase its mission endurance. Using a Reinforcement Learning approach, we demonstrate the possibility for real-time adaptation of the glider's behavior to the time-varying and noisy conditions associated with thermal soaring flight. Our approach is online, data-based and model-free, hence avoids the pitfalls of aerological and aircraft modeling and allow us to deal with uncertainties and non-stationarity. Additionally, we put a particular emphasis on keeping low computational requirements in order to make on-board execution feasible. This article presents the stochastic, time-dependent aerological model used for simulation, together with a standard aircraft model. Then we introduce an adaptation of a Q-learning algorithm and demonstrate its ability to control the aircraft and improve its endurance by exploiting updrafts in non-stationary scenarios.**

I. INTRODUCTION

The number of both civil and military applications of small unmanned aerial vehicles (UAVs) has augmented during the past few years. However, as the complexity of their tasks is increasing, extending the range and flight duration of UAVs becomes a key issue. Since the size, and thus the energy storage capacity, is a crucial limiting factor, other means to increase the flight duration have to be examined. A promising alternative is the use of atmospheric energy in the form of gusts and upwinds. This could significantly augment the mission duration while simultaneously save fuel or electrical energy. For this reason, there is a great interest in the development of algorithms that optimize the trajectories of soaring UAVs by harvesting the energy of the atmosphere. Since the atmospheric conditions are changing over time, it is crucial to develop an algorithm able to find an optimal compromise between exploring and exploiting convective thermal regions, while constantly adapting itself to the changing environment.

In this work we introduce a method to increase the flight duration of UAVs in exchange of light computational cost and no fuel consumption. Furthermore, our method is model-free, therefore suitable for a large range of environments and aircrafts. Additionally, it does not need pre-optimization or pre-training and works in real-time (*and can be applied online?). Although the gap towards a fully autonomous physical demonstrator has not been bridged yet, our main contribution in this work is the *proof of concept* that a model-free reinforcement learning approach can efficiently enhance a glider's endurance. We start by reviewing the state of the art in UAV static soaring and thermal modeling in Section II and position our contributions within previous related work. Then, in Section III, we present the specific atmospheric model we used and its improvements over previous contributions, along with the thermals scenario used in later experiments. Section IV details the aircraft dynamics model. We introduce our implementation of a Reinforcement Learning algorithm in Section V and discuss its strengths, weaknesses and specific features. Simulation results are presented in Section VI and argue about the limitations in Section VII. We finally conclude and introduce some perspectives in Section VIII.

II. RELATED WORK

During the last decade, several possibilities to efficiently utilize atmospheric energy for soaring aircrafts have been proposed. For a general introduction to static and dynamic soaring, refer to [7] for instance. For a more specific review on thermal centering and soaring in practice, see [16].

Most approaches to thermal soaring rely on the identification of some model of the wind field surrounding the aircraft. [1] detects thermals by monitoring accelerations and pressure, and identifying a predefined thermal model. Predefined trajectory patterns are then applied. [3] estimate thermal parameters and relative position to the glider which are in turn used to track an optimal trajectory inside a thermal. In their work, a mode logic was developed to switch between soaring and searching flight. [12] use Gaussian Processes regression to calculate (estimate ?) the mean and variance of a wind field. They exploit this knowledge to rank trajectories with respect to mission goals. Based on similar ideas, in [11], a path planning architecture for autonomous soaring flight in unknown wind fields has been developed. Similar ideas can be found in [5] or [8], whose approach showed energy savings of up to 90% in simulation compared to conventional flights. In the path planning literature, [6] adapt an A^* algorithm to the search for an optimal trajectory in a known wind field. Finally, to cope with model uncertainties, [10] introduce a robust adaptive control algorithm taking into

[1]Erwan Lecarpentier is with the DTIS (Department of Information Processing and Systems), ONERA, 2 avenue Edouard Belin, 31000 Toulouse, France erwan.lecarpentier@isae.fr

[2]Sebastian Rapp is with the Department of Aerodynamics, Wind Energy & Propulsion, TU Delft, Building 62, room B62-5.07, Kluyverweg 1, 2629 HS Delft, Netherlands s.rapp@tudelft.nl

[3]Emmanuel Rachelson is with the Department of Complex Systems Engineering, ISAE – Supaero, 10 avenue Edouard Belin, 31055 Toulouse, France emmanuel.rachelson@isae.fr

account the changing flight and environmental conditions. Here again, the thermal strength and position are supposed to be known, using sensor measurements and ground base data. Overall, one major disadvantage of such model-based approaches is the dependence on a thermal identification algorithm, which may not be robust enough against model inaccuracy.

In this paper, we reconsider the possibility to use a *Reinforcement Learning* (RL, [18]) approach to optimize the trajectory. Using RL to exploit thermals has already been examined in [20]. In his work, he used a neural-based thermal center locator for the optimal autonomous exploitation of the thermals. After each completed circle, the algorithm memorizes the heading where the lift was the strongest and moves the circling trajectory towards the lift. However, this thermal locator was too time consuming for real time on-board applications.

We introduce a *Q-Learning* algorithm using a *linear function approximation*, which is simple to implement and demands less computational resources. We interface this online learning algorithm (Section V) with a simulation model that couples the aircraft dynamics (Section IV) with an improved local aerological model (Section III). We use the model to test our algorithm in several scenarios and show that it yields a significant endurance improvement. Our algorithm's main feature lies in its complete independence of the characteristics of the aerological environment, which makes it robust against model inaccuracy and estimation noise. Moreover we do not explicitly estimate the thermal center position and updraft magnitude, thus saving valuable computational time.

III. ATMOSPHERIC MODEL

The updraft model we develop is inspired by [2]. The original model contained three desirable features: dependence of the updraft distribution in the vertical direction, explicit modeling of downdrafts at the thermal's border and at every altitude, and finally the use of an environmental sink rate to ensure conservation of mass. Although a complete literature review on modeling the convective boundary layer is beyond the scope of this paper, it should be noted that [2] is the first reference that includes these three modeling aspects.

We describe a thermal updraft as a symmetrical, bell-shaped distribution as illustrated in Figure 1. This distribution is characterized by two radii r_1 and r_2. At a given altitude z, if r denotes the distance to the thermal center, for $r < r_1$ the updraft has a quasi-constant value of w_{peak}, then for $r_1 < r < r_2$ this value drops smoothly to zero, and between r_2 and $2r_2$ appears a downdraft. The thermal has no influence further than $2r_2$.

The maximum updraft velocity w_{peak} evolves altitude-wise proportionally to $w^* \left(\frac{z}{z_i}\right)^{\frac{1}{3}} \left(1 - 1.1\,\frac{z}{z_i}\right)$, where w^* is an average updraft velocity and z_i is a scaling factor indicating the convective boundary layer thickness. Above $0.9 z_i$ all velocities are assumed to be zero.

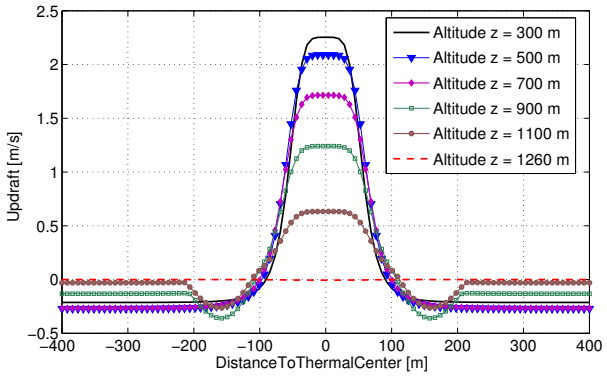

Fig. 1. Updraft distribution with altitude

Finally, based on the conservation of mass principle, an altitude-dependent global environmental sink rate is calculated and applied everywhere outside the thermals. For specific equations, we refer the reader to [2].

We introduce three additional features that bring our simulation model closer to a real-life description namely thermal drift, life-cycle and noise. First, in order to account for local winds, we let the thermals drift in the horizontal plane with a velocity $(\bar{v}_x, \bar{v}_z)$. Usually, the root point of a thermal is a fixed location and the thermal leans with the wind, so introducing a thermal drift is a poor description of this phenomenon. Nevertheless, for our simulations, it approximates the practical phenomenon of drift given that the aircraft model is reduced to a single point-mass. Thermals also have a finite life. We decompose a thermal's life in a latency phase of duration t_{off} and a growth, maturity and fade-off phase of duration t_{life}. After $t_{off} + t_{life}$ the thermal dies. The life-cycle of a thermal is described by the updraft coefficient $c_\xi(t)$ shown in Figure 2, using a shape parameter ξ. This $c_\xi(t)$ coefficient is used as a multiplier on the total updraft. Finally, it is well-known among cross-country pilots that thermals are rarely round and present a great variety of shapes and much noise. In order to account for this fact and to model real-life uncertainties we added a Gaussian distributed noise n to the wind velocity.

We maintain a constant number N of thermals in the flight area, although some might be in their latency phase. Consequently, whenever a thermal dies, a new thermal is generated with randomly drawn parameters $\{x_{th}, y_{th}, w^*, z_i, \bar{v}_x, \bar{v}_y, t_{off}, t_{life}, \xi\}$.

J'ai interverti le paragraphe [We maintain ...] avec le paragraphe [Finally it is well-known ... to the wind velocity.] pour faire un seul bloc avec l'enumeration des 3 contrib que l'on fait au niveau du modele.

IV. AIRCRAFT MODEL

To model the dynamical behavior of our aircraft, we used the equations derived in [4], which consider the aircraft as a point-mass, 6 degrees of freedom system, and take into account the three dimensional wind velocity vector of the

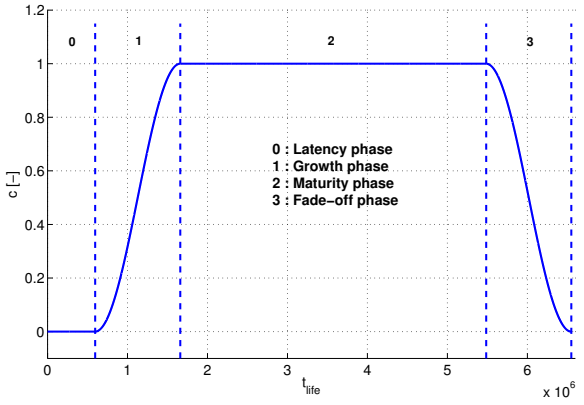

Fig. 2. Evolution of the updraft coefficient $c_\xi(t)$

atmosphere as well as a parametric model for the aircraft's aerodynamics:

$$\dot{x} = V\cos(\chi)\cos(\gamma) \qquad \dot{z} = V\sin(\gamma)$$
$$\dot{y} = V\sin(\chi)\cos(\gamma) \qquad \dot{V} = -\frac{D}{m} - g\sin(\gamma)$$

$$\dot{\gamma} = \frac{1}{mV}\left(L\cos(\mu) + C\sin(\mu) - \frac{g}{V}\cos(\gamma)\right)$$
$$\dot{\chi} = \frac{1}{mV\cos(\gamma)}\left(L\sin(\mu) - C\cos(\mu)\right)$$

The first three equations describe the kinematics and position rates in an earth-based coordinate system. The last three equations define the dynamics of our glider aircraft. Let m be the glider's mass and g the gravity acceleration. The variables are:

- V the absolute value for the aircraft velocity;
- γ the angle of climb;
- χ the course angle;
- μ the bank angle;
- β the side-slip angle;
- L, D and C the lift, drag and side-force.

For a detailed presentation of the aerodynamic parameters and forces, refer to [4].

V. ADAPTATIVE CONTROLLER

Reinforcement Learning (RL, [18]) is a branch of Discrete-time Stochastic Optimal Control that aims at designing optimal controllers for non-linear, noisy systems, using only interaction data and no *a priori* model. The only hypothesis underlying RL algorithms is that the system to control can be modeled as a Markov Decision Process (MDP, [15]), even if this model is not available. An MDP is given by a set of system states $s \in S$, a set of control actions $a \in A$, a discrete-time transition function $p(s'|s,a)$ denoting the probability of ending up in state s' given that action a was undertaken in state s, and finally a reward model $r(s,a,s')$ indicating how valuable the (s,a,s') transition was with respect to the criterion one wants to maximize.

The overall goal of an RL algorithm is to derive a control policy $\pi(s) = a$ that maximizes the expected cumulative sum of rewards $\mathbb{E}\left(\sum_{t=0}^{\infty}\eta^t r_t\right)$ from any starting state s ($\eta \in [0;1[$ is a discount factor over future rewards). We focus on model-free RL algorithms that do not commit to the knowledge of the transition and reward models of the underlying MDP but use *samples* of the form (s,a,r,s') to learn an optimal policy. In our case, that means that an RL algorithm controlling the glider with an overall goal of gaining energy will use sensor data to build π^* online, without relying on a (possibly approximate) model of the atmosphere, or the aircraft's flight dynamics. [J'ai un peu de mal avec cette phrase]

Q-learning, introduced by [19], is one of the most simple and popular online RL algorithms. It aims at estimating the optimal action-value function $Q^*(s,a)$ in order to asymptotically act optimally. The latter is the expected gain of applying action a from state s, and then applying an optimal control policy π^*:

$$Q^*(s,a) = \mathbb{E}\left(\sum_{t=0}^{\infty}\eta^t r_t | s_0 = s, a_0 = a, a_t = \pi^*(s_t)\right)$$

The key idea behind Q-learning is that the optimal action in state s is the one that maximizes $Q^*(s,a)$. Thus the optimal policy is greedy with respect to Q^* in every state. Estimating Q^* from (s,a,r,s') samples is actually a stochastic approximation problem which can be solved with a procedure known as *temporal differences*. The Q-learning algorithm is summarized in algorithm 1.

Algorithm 1: Q-learning

Initialize $Q(s,a)$ for all (s,a),
$s_t \leftarrow s_0$.
repeat
  With probability $1-\epsilon_t$, apply $a_t = \arg\max(s,a)$,
  otherwise apply a random action a_t
  Observe s_{t+1} and r_t
  $\delta_t = r_t + \eta \max_{a' \in A}\left(Q\left(s_{t+1}, a'\right)\right) - Q\left(s_t, a_t\right)$
  Update $Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha_t\delta_t$
  $s_t \leftarrow s_{t+1}$
until *simulation end*

It is important to note that Q-learning is an *off-policy* method, that is it estimates Q^* regardless of the chosen actions when interacting with the system. It relies on an ϵ-*greedy* exploration strategy, choosing a random action to apply with probability ϵ_t. As ϵ_t tends to zero, if Q has converged to Q^*, the Q-learning agent tends to act optimally. As long as all state-action pairs are visited infinitely often when $t \to \infty$, Q is guaranteed to converge to Q^* if the sequence of learning rates α_t satisfies the conditions of [17]:

$$\sum_{t=0}^{\infty}\alpha_t = \infty, \quad \sum_{t=0}^{\infty}\alpha_t^2 < \infty$$

Recall that the state of the aircraft, as defined in Section IV, or the state of the atmospheric model (Section III) are not fully observable to our learning agent. So it would be unrealistic to define the state space S as the observations of these values. Instead, we suppose that a state only defined by $(\dot{z}, \dot{\gamma}, \mu, \beta)$ is accessible and that its dynamics still defines an MDP. Such a state is easily measurable with usual sensors such as pressure sensors, accelerometers or gyrometers. This key assumption is crucial to the success of our method since it reduces the size of the state space, easing the approximation of $Q^*(s, a)$. We shall see later that this choice of state variables has other advantages.

The actions consist in directly controlling the aircraft's aerodynamic angles: $a_t = (\mu_t, \beta_t)$. [faire un point sur les espaces d'etat-action : l'espace d'action est $\{-\delta\mu, 0, \delta\mu\} \times \{-\delta\beta, 0, \delta\beta\}$ et (μ, β) ne font pas partie de l'espace d'etat, si ? Je ne me souviens plus]

The goal of our learning algorithm is to maximize the glider's endurance. This boils down to maximizing the expected total energy gain, so we wish that $Q(s, a) = \mathbb{E}\,(\text{total energy at } t = \infty)$. To achieve this, we choose:

$$r_t = \dot{E}_{aircraft} = \frac{d}{dt}\left(z + \frac{V^2}{2g}\right) \tag{1}$$

Thus we assume that a reward signal r_t is provided to the learning algorithm at each time step, representing the (possibly noisy) total energy rate of the aircraft.

The previous requirements on ϵ_t and α_t for convergence hold if the environment can indeed be modeled as an MDP. However, in the studied case, the environment is non-stationary since the thermals have a time-varying magnitude (thermal coefficient) and location (drift). Consequently, our learning agent evolves in a constantly changing environment and we rely on its ability to learn and adapt quickly to changing conditions. In order to allow this quick adaptation, we need to force a permanent exploration of the state-action space and to constantly question the reliability of Q. This corresponds to make use of constant α_t and ϵ_t values, which need to be well-chosen in order to retain a close-to-optimal behavior while quickly adapting to the changes in the environment. Furthermore, the choice of a simplified low-dimensional state space made earlier make the adaptation to a non-stationary environment feasible. Indeed, the chosen variables evolve slowly through the time, making the evolution of the optimal action value slow as well. This allows to make maximal use of the collected samples since only a local approximation around the current state is required to compute the optimal current action. The success of the method is therefore due to the capacity of the Q-learning algorithm to track the optimal action quickly enough in comparison to the environment's dynamic.

In order to avoid the discretization of the state and action spaces ["of the state space" uniquement ?] in the description of Q, we adopt a linear function approximation of $Q(s, a)$. We introduce sigmoid-normalized versions of the state space variables and define our basis functions ϕ as the monomials of these normalized variables of order zero to two (15 basis functions). Then, by writing $Q(s, a) = \theta^T \phi(s, a)$, the update equation of Q-learning becomes $\theta_{t+1} = \theta_t + \alpha_t \delta_t \phi(s_t, a_t)$. There is abundant literature on choice of feature functions in RL, we refer the reader to [14], [9], or [13] for more details.

Since the Q-learning algorithm solves a maximization problem at each time step over the continuous variable a, we chose to define incremental changes $\delta\mu$ and $\delta\beta$ in order to simplify this maximization. Consequently, at each time step, the algorithm chooses the estimated best action among all 9 combinations of $\{-\delta\mu, 0, \delta\mu\} \times \{-\delta\beta, 0, \delta\beta\}$. We chose the values of $\delta\mu$ and $\delta\beta$ so that, given a certain control frequency, the cumulated effect of a constant action does not exceed the admissible dynamics of the aircraft. This results in a steady state change, representative of the actual behavior of the actuators.

In the next Section, we shall assume a control frequency of 1kHz which is physically unrealistic. We argue however that even though this assumption is irrelevant, it can be leveraged by a better tuning of Q-learning, and, more importantly, it accounts for the many exogenous perturbations experienced by our aircraft (gusts, etc.) that generate relevant training data for Q-learning.

To summarize, our glider is controlled by a Q-learning algorithm with fixed learning and exploration rates (α and ϵ). The optimal action-value function Q^* is approximated with a linear architecture of quadratic features defined over a set of observation variables (observations of the state and action variables) $(\dot{z}, \dot{\gamma}, \mu, \beta)$ [idem a propos de l'espace etat-action]. Finally, at each time step, the chosen action is picked among a set of 9 possible increments on the (μ, β) current values. [n'est-ce pas repetitif/evident ?]

One key feature of our algorithm lies in the fact that the difficulty of learning an optimal soaring control policy (namely the continuous state-action space and action-value function, the non-observability of some variables and the time-dependent dynamics) can be compensated by the quick adaptation of the learning method to its changing environment. This quick adaptation is made possible thanks to the careful choice of state variables. In fact, with these variables, on the short term, the learning agent observes a quasi-constant state and the optimal action in this state is almost constant also. This allows to make maximal use of the samples collected by Q-learning since only a local approximation around the current state is required to compute the optimal current action. As long as Q-learning tracks this optimal action quickly enough to react to the changes in the environment, the glider shall be able to adapt to the changing conditions. [J'ai pense que ce paragraphe devait etre mis a la suite de celui qui commence par [The previous requirements on ϵ_t ...] et je l'ai un peu modifie]

VI. SIMULATION RESULTS

We identify three scenarios designed to illustrate the convergence of the algorithm and the overall behavior of the glider. These scenarios take place within a 1100m wide circular flight arena. Whenever the glider exits the arena, an autopilot steers it back in. The aircraft is initialized at

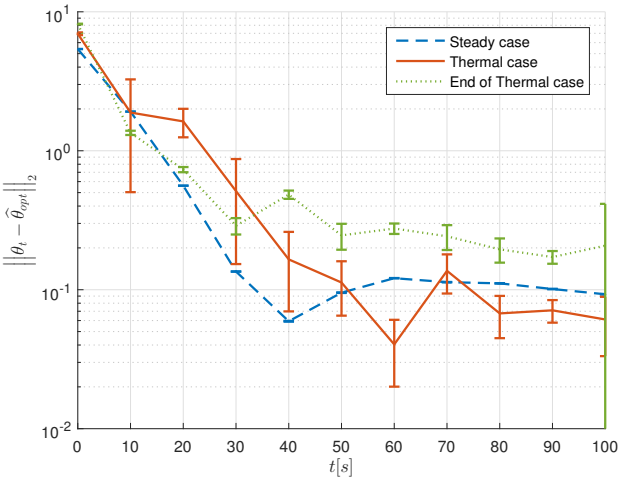

Fig. 3. Convergence of the action-value function

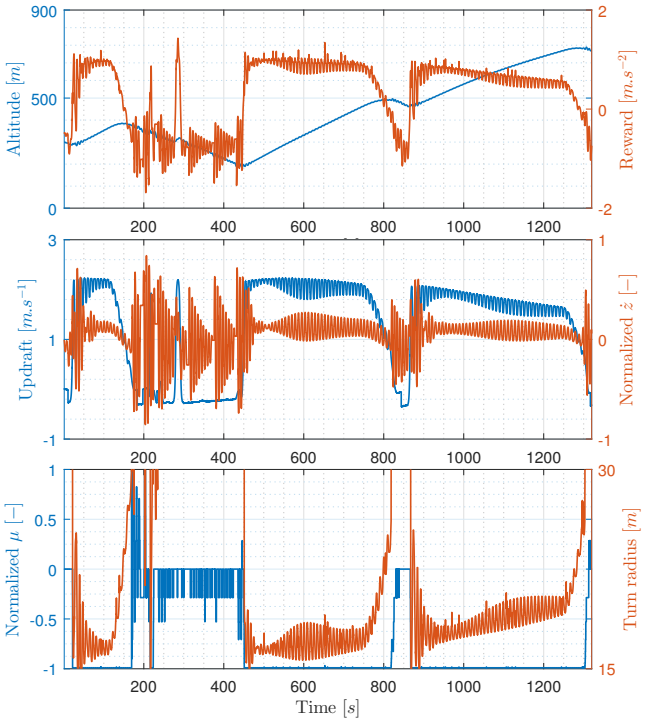

Fig. 4. Evolution of the aircraft variables with time

$z = 300$m and $V = 15$m/s. According to [2], we set $w^* = 2.56$m/s and $z_i = 1401$m. The algorithm parameters were $\epsilon = 0.01$; $\alpha = 0.001$; $\eta = 0.99$; $\delta\beta = 0.003\,\text{deg}$; $\delta\mu = 0.003\,\text{deg}$; $\beta_{max} = 45\,\text{deg}$; $\mu_{max} = 25\,\text{deg}$.

The three scenarios are the flight in still air with noisy downdraft and no thermal, the flight inside a thermal, and the death of a thermal (when the aircraft has to come back to a still air flight configuration). In each scenario, we refer to the optimal action-value function parameters as θ_{opt}. In order to analyse the convergence of the algorithm, we built an estimate $\widehat{\theta_{opt}}$ by letting the value function converge on multiple simulations, and monitored $\|\theta_t - \widehat{\theta}_{opt}\|_2$ along 50 roll-outs of the system (Figure 3, error bars indicate the standard deviation). One can see that the time required to adjust the parameters to each situation ranges between 30 and 40 seconds, which is compatible with the change rates of the glider's environment. Note in particular that the glider's behaviour might be optimal long before θ converges to θ_{opt} since what matters is the ranking between actions in s due to $Q(s,a)$, rather than the actual associated values. Configurations vary between the three studied cases and the exploratory feature of the ϵ-greedy policy allows to permanently adapt the Q-function to the situation.

The performance reached by the control algorithm can be measured via the total energy of the aircraft, capturing the reached altitude and the velocity. In the three aforementioned scenarios, the expected results are not the same. Indeed, in a steady atmosphere, the optimal policy only allows to minimize the loss of altitude by setting $\beta = \mu = 0$. Such a configuration is optimal since no thermals can be found and the glider can only maximize its long term energy by flying straight and avoiding sharp manoeuvres. Then, when a thermal is reached, the algorithm's exploratory behavior allows to captures the information that it is worth changing β and μ, and adapts the trajectory to maximize the long-term return. In the third situation, when the glider flies inside a dying thermal, the algorithm brings back the parameters to a steady atmosphere configuration and again minimizes the expected loss of energy.

Figure 4 shows the evolution of altitude and instantaneous rewards through time in a typical scenario with multiple thermal crossings. Each altitude pike shows the entry of the aircraft into a thermal. First the trajectory is bent in order to maximize the altitude gain and when the thermal dies, the glider goes back to the steady flight configuration. Clearly, each gain-of-altitude phase corresponds to a positive reward and, conversely, a loss-of-altitude phase to a negative one. A 3D display of the trajectory inside a thermal is presented in Figure 5.

The Q-learning controller yields an overall behaviour close to the one of a human pilot. While flying in still air, the glider remains in "flat" flight attitude, thus maximizing its flight time expectancy. Whenever an updraft is spotted, it engages in a spiral, as shown in Figure 4. If the updraft dies, the aircraft comes back to the first configuration. This results in an overall trajectory composed with straight lines and circles as displayed in Figure 6.

Figure 4 also illustrates the reaction times of the glider and the overall command behaviour. It appears that the glider starts to circle up the thermals long before the value function has converged. Similarly, the convergence to a steady air optimal behaviour is faster than the Q-function convergence illustrated on Figure 3. When the glider reaches the thermal's top, the updraft naturally decreases. Consequently one can notice the reduction of the bank angle (enlargement of the turning radius) computed by the algorithm in order to stay

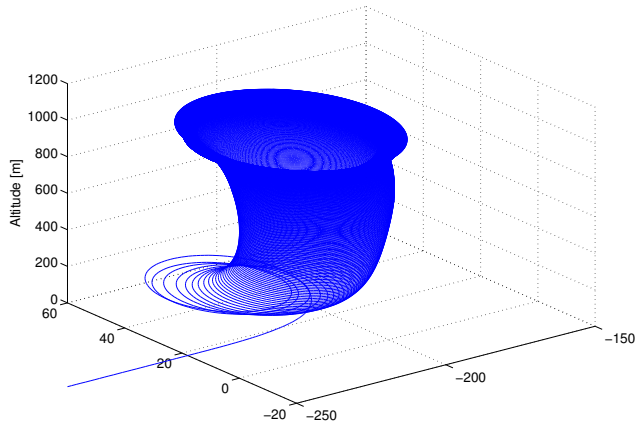

Fig. 5. Trajectory of the aircraft inside a thermal

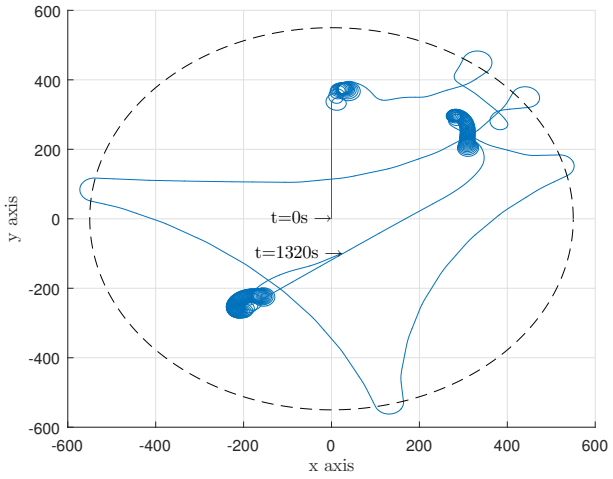

Fig. 6. An example of trajectory

in the thermal while reaching a zero vertical velocity.

VII. LIMITATIONS

One weakness of our current implementation is the *bouncing* phenomenon, i.e. the fact that sometimes the glider literally bounces on a thermal instead of circling inside. This is due to the random action exploration strategy that initiates the turn in the wrong direction. For instance, if the thermal is on the right and the aircraft turns left without trying to turn right first, it starts gaining energy but quickly flies away and comes back to the steady configuration. This pitfall is well known to glider pilots that adapt their trajectory accordingly while our controller currently just flies away.

It should also be noted that no low-level dampening control of the angle of attack was performed in our simulations. This resulted in the oscillations appearing in Figure 4, typical of a phugoid mode. Future work could include such attitude stabilization systems to increase the efficiency of the glider's flight.

VIII. CONCLUSION

Model-free online reinforcement learning approaches are convenient control methods that present a great adaptivity feature and do not need pre-computation. This proof-of-concept demonstrated their applicability to the problem of maximizing the flight autonomy of a soaring UAV for a low computational cost and no energy consumption. Simulations showed the benefit of an adapted Q-learning algorithm in an unknown, unsteady, noisy environment with time and space-dependent updraft velocities. Our intention was to stick to a realistic scenario where updrafts positions are unknown and the policy has to trade-off exploration of the map and exploitation of thermals while maximizing the altitude of the aircraft.

Limits in terms of exploration were met in what we called the *bouncing* phenomenon, suggesting to improve the information collected by the system or their processing, for instance by computing a belief over the environment configuration. This opens the door to a whole span of perspectives, from the left/right disambiguation in the bouncing phenomenon to more advanced function approximation methods in RL, from softer constraints on the control frequencies to different control algorithms and architectures (oscillation damping, simulation-based open-loop online control, actor-critic architectures, policy gradients, etc.). Moreover, future work should explore the interaction between low-level piloting such as the one presented here and the issues in navigation and in mission planning.

REFERENCES

[1] Allen, M.J. (2005). Autonomous soaring for improved endurance of a small uninhabited air vehicle. Technical report, NASA Dryden Research Center.

[2] Allen, M.J. (2006). Updraft model for development of autonomous soaring uninhabited air vehicles. Technical report, NASA Dryden Flight Research Center.

[3] Allen, M.J. and Lin, V. (2007). Guidance and control of an autonomous soaring uav. Technical report, NASA Dryden Flight Research Center.

[4] Beeler, S., Moerder, D., and Cox, D. (2003). A flight dynamics model for a small glider in ambient winds. Technical report, NASA.

[5] Bencatel, R., de Sousa, J.T., and Girard, A. (2013). Atmospheric flow field models applicable for aircraft endurance extension. *Prog. in Aerospace Sciences*, 61.

[6] Chakrabarty, A. and Langelaan, J. (2010). Flight path planning for uav atmospheric energy harvesting using heuristic search. In *AIAA Guidance, Navigation, and Control Conference*.

[7] Chen, M. and McMasters, J. (1981). From paleoaeronautics to altostratus - a technical history of soaring. In *AIAA Aircraft Systems and Technology Conference*.

[8] Chen, W. and Clarke, J.H.A. (2011). Trajectory generation for autonomous soaring UAS. In *17th International Conference on Automation and Computing*.

[9] Hachiya, H. and Sugiyama, M. (2010). Feature selection for reinforcement learning: Evaluating implicit state-reward dependency via conditional mutual information. In *European Conference on Machine Learning and Knowledge Discovery in Databases*, 474–489.

[10] Kahveci, N. and Mirmirani, M. (2008). Adaptive LQ control with anti-windup augmentation to optimize UAV performance in autonomous soaring application. In *IEEE Transactions on Control System Technology*.

[11] Lawrance, N. (2011). *Autonomous Soaring Flight for Unmanned Aerial Vehicle*. Ph.D. thesis, The University Of Sydney.

[12] Lawrance, N. and Sukkarieh, S. (2011). Path planning for autonomous soaring flight in dynamic wind. In *IEEE International Conference on Robotics and Automation*.

[13] Nguyen, T., Li, Z., Silander, T., and Leong, T.Y. (2013). Online feature selection for model-based reinforcement learning. In *Int. Conf. on Machine Learning*.

[14] Parr, R., Li, L., Taylor, G., Painter-Wakefield, C., and Littman, M.L. (2008). An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In *International Conference on Machine Learning*.

[15] Puterman, M.L. (2005). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc.

[16] Reichmann, H. (1993). *Cross-Country Soaring*. Soaring Society of America.

[17] Robbins, H. and Monro, S. (1951). A stochastic approximation method. *Ann. Math. Statist.*, 22(3), 400–407.

[18] Sutton, R.S. and Barto, A.G. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

[19] Watkins, C.J.C. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8, 279–292.

[20] Wharington, J. (1998). *Autonomous Control of Soaring Aircraft by Reinforcement Learning*. Ph.D. thesis, Royal Melbourne Institute of Technology.