AO4ELT7 - June 2023 - Avignon

PAPYRUS at OHP: Predictive control with reinforcement learning for improved performance

Raissa Camelo[a], Jalo Nousiainen[b,c], Cedric Taïssir Heritier[a,d], Morgan Gray[a], and Benoit Neichel[a]

[a]Aix-Marseille Université, CNRS, CNES, LAM, Marseille, France
[b]Department of Computational and Process Engineering, Lappeenranta–Lahti University of Technology, Finland
[c]European Southern Observatory, Karl-Schwarzschild-Str. 2, 85748 Garching bei München, Germany
[d]DOTA, ONERA, F-13661 Salon cedex Air - France

ABSTRACT

In this paper, we implement and study a model-based reinforcement learning (MBRL) method called Policy Optimization for Adaptive Optics (PO4AO) in numerical simulations. We use the Object-Oriented Python Adaptive Optics (OOPAO) simulation tool to simulate the Provence Adaptive Optics Pyramid Run System (PAPYRUS) optical bench and provide a real-time model of the system. In particular, we demonstrate the predictive capability of the method since the temporal error dominates the error budget of PAPYRYS. We first present a detailed description of the reinforcement learning framework, including our definition of the state space, the action space, and the reward function. The experiment section compares PO4AO to a well-tuned integrator in different atmospheric conditions. In conclusion, we discuss the importance of experiments in numerical simulations before applying the method to a real telescope and possible avenues for future work.

Keywords: Adaptive Optics, Reinforcement Learning, Neural Networks, Simulation, wavefront sensing, Predictive Control

1. INTRODUCTION

Astronomical observations from ground-based telescopes are deeply impacted by atmospheric turbulence, which introduces phase distortions to the incoming light and, consequently, causes blurred and distorted images. Adaptive Optics (AO) systems have emerged as a way to mitigate these effects, allowing telescopes to correct wavefront aberrations and generate better observations. A standard AO system composes two main components:

Further author information:
R. Camelo : E-mail: raissa.camelo@lam.fr,
J. Nousiainen.: E-mail: jalo.nousiainen@lut.fi

a wavefront sensor (WFS) that measures the phase aberrations caused by the atmosphere and a deformable mirror (DM) that corrects for them. Most AO systems are controlled in a closed-loop configuration, where the WFS measures the wavefront distortions after DM correction. Traditional control algorithms have been effective but are often limited by their dependence on predetermined models of atmospheric turbulence and errors in the control system models. More recently, the incorporation of Reinforcement Learning (RL) on AO systems has shown promising results to overcome these shortcomings – RL is shown to predict the temporal evolution of turbulence and adjust common modeling errors such as misregistration and optical gains [7, 6, 4, 10, 11]. Besides Reinforcement learning, recent advances in AO techniques have brought up the use of artificial intelligence as an aid to other wavefront sensing-related tasks [2], such as focal-plane wavefront sensing [1] and Point Spread function (PSF) reconstruction [12].

Between the few RL techniques implemented for adaptive optics, Nousiainen et al.'s PO4ELT [6] consists of a model-based policy optimization algorithm that estimates the voltages to be applied in the Deformable Mirror (DM) from wavefront sensor (WFS) data. This algorithm has shown promising results in extensive numerical simulations and in lab setups [6, 8], indicating suitability for on-sky tests. This paper takes the first steps towards testing the method on-sky with the PAPYRUS bench [5] at the Observatoire Haute de Provence (OHP). We implement and test the method on numerical simulation mimicking the PAPYRUS. This step allows us to have a better understanding of the RL algorithm and how it communicates with the bench. It also leaves us room for experimenting with different optimizations before deciding on a final version to go on-sky. Moreover, with the increase of AI-based approaches for AO, the need for robust and compatible simulated platforms has escalated. Object-Oriented Python Adaptive Optics (OOPAO) [3] is a Python-based tool created to perform end-to-end AO simulations. In this paper, we also demonstrate how OOPAO's availability as an open-source project and the fact that it's written in Python make it a suitable option to serve as an experimenting ground for RL approaches.

In the following sections, we provide a brief explanation of important RL concepts, such as Markov Decision Processes used to design and implement the solution and how it is outlined for AO control. We proceed by describing the PO4ELT algorithm and its training process alongside the neural network's structure, illustrating how it is integrated into the bench simulation. In the results section, we display the simulation parameters used in the experiments with OOPAO followed by a comparison of the integrator's performance under different seeing and wind conditions, demonstrating PO4ELT's capability to adapt to different optical gains along with the predictive control.

2. REINFORCEMENT LEARNING FOR AO PREDICTIVE CONTROL

Reinforcement Learning (RL) is an area of machine learning focused on intelligent agents and how they take actions to change the state of the environment they are in, with the objective of maximizing the cumulative reward [13]. It differs from traditional supervised learning, where the labeled input and output pairs are given, and the model can learn from its mistakes by comparing its predictions to the labeled ground-truth. In RL no labeled data is given and the model optimizes itself by maximizing the rewards received for each action. The reward function is defined according to the problem being solved.

RL is often used to solve problems that require real-time computation since the algorithms can be trained "online", as it solves the task. RL problems are usually described through Markov Decision Process (MDP) formulation. An MDP is a mathematical framework that consists of a set of states, actions, transition probabilities, and rewards. The state is the current state of the environment; the actions are the control signals given by the agent, the transition probabilities are the probabilities of the system moving from one state to the other with given actions; and the reward is a user-designed measure of performance.

At each time instance, an agent takes an action (from the set of all possible actions), leading to a new state according to transition probabilities. The agent receives a reward for each action taken. The agent aims to learn an optimal control strategy, e.g., a function (Policy) that maps an observation of the state to actions that maximize the cumulative reward by interacting with the system.

Next, we describe how the AO framework is defined as an MDP following the approach of Nousiainen et al. [7]. There are different ways to define an MDP for Adaptive Optics control (see, e.g.,[4, 10, 11]), but in this paper, we constraint ourselves to explain the approach used in PO4AO's.

We denote the control voltages applied to a DM at a given time instance t as v_t and the WFS measurements, pre-processed to slopes as w_t. As such, an action effectuated at time step t is defined as the differential control voltages applied to the DM in that instant t:

$$a_t = \Delta v_t, \tag{1}$$

while the the full control voltages are $\Delta v_t + v_{t-1}$. At each time step t, the WFS measurement w_t is observed. The measurements are projected into voltage space by operating the reconstruction matrix, denoted as C. Since the system in our simulations is a closed loop, it corresponds to the residual voltages detected by the WFS. We define an observation (of the state) at time instant t as

$$o_t = Cw_t. \tag{2}$$

To ensure the Markovian property, each state S_t is represented by a concatenation of previous observations and actions, that is,

$$S_t = (o_t, o_{t-1}, .., o_{t-k}, a_{t-1}, a_{t-2}, ..., a_{t-m}), \tag{3}$$

where $k = m$, including data from the previous m time steps and the reconstruction matrix C.

The reward function is defined as the residual voltages' negative squared norm as stated below:

$$r_t(s_{t+1}, s_t, a_t) = -1 * ||o_{t+1}||^2. \tag{4}$$

In the next section, we explain the PO4AO algorithm using the MDP notation.

3. PO4AO

Reinforcement Learning algorithms can be roughly divided into two sub-classes: model-based algorithms and model-free [9]. In model-based reinforcement learning, the agent builds an explicit model of the environment, called the *dynamics model*, which approximates the real transition dynamics of the environment (i.e., AO system, bench). The dynamics model is then used for model predictive control or, like in this work, for optimizing a *policy model*, a model that dictates the actions the agent will take (i.e., controlling the DM). In contrast, model-free reinforcement learning doesn't rely on an explicit model of the environment but learns directly from interaction with it, acquiring a policy function without a dynamics model (see, e.g., [4, 10]).

The Policy Optimization for Adaptive Optics (PO4AO) algorithm is a model-based reinforcement learning method, introduced and detailed in J. Nousiainen et al. [6], that utilizes two neural networks, the dynamic model and the policy. The dynamics neural network model is trained on previously saved states and actions and learns to infer the next state for the previous state and a given action, $Dyn(s_t, a_t) = s'_{t+1}$, where s'_{t+1} is the predicted next state. The policy (π) is a neural network that takes the current state s_t as input and outputs an action a_t to be taken by the DM. Both the dynamics and the policy model are small convolutional neural networks containing three layers each.

In order to train the models, the data from the simulation (a tuple containing: the next state, previous states, and the actions taken) is collected and stored in a dataset, $\mathcal{D} = \{(s_t^i, a_t^i)s_{t+1}^i\}_{i=1}^N$, where N is number of data samples. The dynamics model, $Dyn(s_t, a_t)$, is trained on the collected dataset as a standard supervised learning task. Later, the policy model (π) is trained (i.e., the parameters are optimized) using the dynamics model. The policy training loop starts by having the policy predicting the next action, from a state t collected from $\mathcal{D}$, $\pi(s_t) = a'_t$, with a'_t being the action given by the policy (π) in the instant t. Next, the predicted action (a'_t) is fed to the trained dynamics model, along with the corresponding state (s_t); the dynamics model then predicts the next state (s_{t+1}), which is fed again to the policy, repeating this procedure for a fixed number of instances T (called the planning horizon). At each time instance (inside the planning horizon), the rewards are collected and finally back-propagated to policy parameters, optimizing the policy to maximize the rewards.

In our implementation of PO4AO, we define each episode as 500 frames of the simulation, and we run all the experiments for a total of 20 episodes. We start the simulation by running the bench and applying random voltages to the DM for one episode, collecting the state-action-next_state tuples (s_t, a_t, s_{t+1}) for each frame in $\mathcal{D}$. This initial episode is called the **warm-up** step. After the warm-up the policy will then be used to estimate

the control voltages. This process of saving the tuples in $\mathcal{D}$ and updating the dynamics and the policy models continues through all 20 episodes. Every subsequent training round starts after the end of every episode. In our simulations, we trained the policy for 60 instances during the warm-up and for 7 in the subsequent training rounds. These values were manually chosen during the experimentation phase.

4. EXPERIMENTS

In this section, we compare the simulation results obtained both with PO4AO and a classical integrator. We run the experiments in 4 different turbulence settings, varying the loop gain in order to optimize the integrator. The Strehl ratio was chosen as a comparative metric and was computed on each frame of every experiment. Finally, we compare the best integrator result with PO4AO's after each episode (500 frames). In the following subsection, we specify the OOPAO simulation parameters used in the experiments. The results are shown and discussed subsequently.

4.1 OOPAO Simulation Parameters

We used OOPAO to simulate the PAPYRUS bench and telescope T152 at OHP. The simulation parameters for the system are provided in the table below. We simulated the atmospheric turbulence as a sum of five frozen flow layers with Von Karman power spectra. The same system configuration was used for all the experiments, with PO4AO and the integrator, and the only changing parameters between experiments were the atmospheric parameters (seeing and wind speeds), which are provided in the label of each result separately. We note here that numerical simulations were rather simple; for example, no central obstruction and spiders were present. A more precise simulation is left for future work.

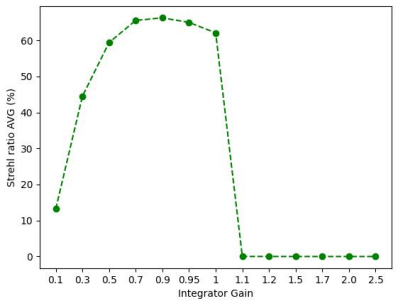
Table 1: PAPYRUS Simulation parameters for OOPAO

N°Actuators	17*17
Telescope Diameter	1.52m (OHP)
Sampling time	2ms (500 Hz)
Delay (PO4AO)	1 Frame
WFS	Pyramid (Modulated)
WFS Modulation	3 λ/D
Modal Basis	Zernike (50 modes)
DM Mechanical Coupling	35%
Wind Speed*	10 (m/s) / layer
Magnitude (NGS)	8
Sensing Wavelength	"I" (806 nm)

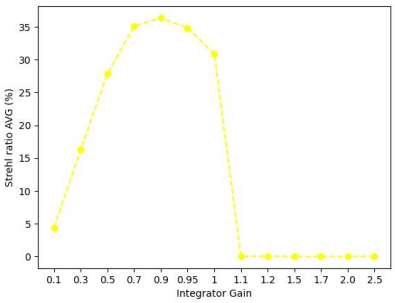
4.2 Results

The initial experimental wind speed (WS) chosen was 10m/s for each of the 5 simulated atmospheric layers, combining an r0 of 0.13m @500nm. In the second experiment, the wind speed was doubled, and the r0 was kept the same. For the third experiment, the wind speed was kept at its original value, and the r0 was set to $\approx$ 0.086m @500nm, 2/3 of the original value. Lastly, in the fourth experiment, the wind speed was doubled, and the value of the r0 was 0.086m @500nm.

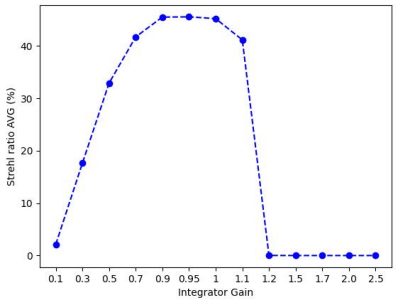
In Figures 1, we display the integrator performance for each turbulence setting with different gain values. For each gain value, the Strehl ratio (SR) was averaged over 10.000 frames. A total of 10.000 frames (20 episodes) was chosen as it was enough for the algorithm and the integrator to converge and stabilize the results. For atmospheric conditions, the optimal integrator gain was either 0.9 or 0.95, both providing very similar performance (see Figure 1). The gain values bigger than 1 were unstable. We chose the gain value of 0.9 for the comparison to PO4AO (Figure 2). Further, we note here that the gain value of 0.9 is above traditional stability criteria, and here, we aimed to exploit the bench's optical gain by going above 0.5 in an attempt to optimize the integrator for a fair



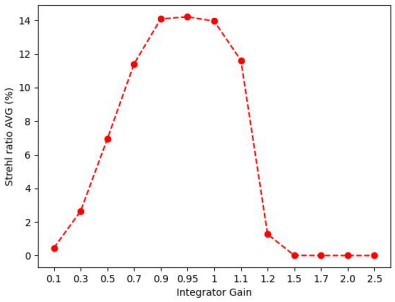
(a) WS = 10m/s, r0 = 0.13m @500nm



(b) WS = 20m/s, r0 = 0.13m @500nm



(c) WS = 10m/s, r0 ≈ 0.086m @500nm

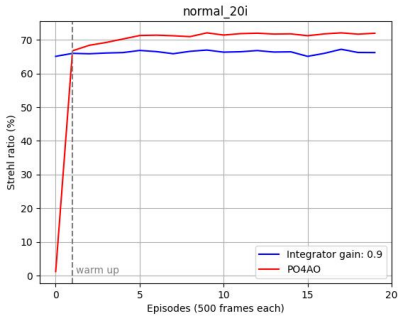


(d) WS = 20m/s, r0 ≈ 0.086m @500nm

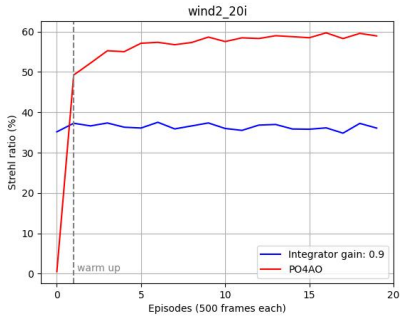
Figure 1: Integrator experiments: Average Strehl ratio for each gain in 10.000 frames

comparison against PO4AO. Overall, augmenting the gain had a significant impact going from the standard gain value of 0.5 up to approximately 0.9.

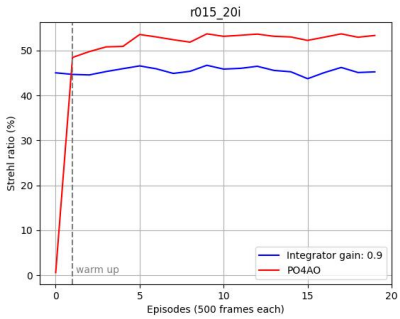
Figure 2 shows the comparison with the integrator and PO4AO. We run both controllers for 10,000 frames, i.e., 20 episodes. The integrator performance stays approximately constant (as expected). The PO4AO starts with the warm-up phase (i.e., random commands), after which the policy NN starts to control the system. PO4AO achieves better performance than the integrator already after the integrator warm-up and converges around 20 episodes. Figures indicate that PO4AO fared better than the optimal integrator, especially in the worst weather conditions. In experiments (a) and (c), the integrator had a marginally close result to the RL approach. However, in experiments (b) and (d), where the wind speed was doubled, the difference between PO4AO and the integrator is substantially more remarkable. This illustrates RL's predictive capability, indicating a significant potential for an On-sky experiment on PAPYRUS (error budget dominated by temporal error).



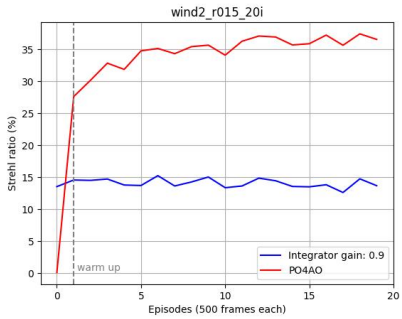
(a) WS = 10m/s, r0 = 0.13m @500nm



(b) WS = 20m/s, r0 = 0.13m @500nm



(c) WS = 10m/s, r0 ≈ 0.086m @500nm



(d) WS = 20m/s, r0 ≈ 0.086m @500nm

Figure 2: Integrator (gain = 0.9) vs PO4AO: Strehl ratio for each 10.000 frames (20 episodes)

5. DISCUSSION

As shown in the previous section, PO4AO demonstrates encouraging results. The PO4AO outperformed the optimized integrator in all test cases already after 500 frames of interaction. Moreover, simulating the PAPYRUS bench allowed us to test the reinforcement learning approach with the flexibility provided by OOPAO. We were able to experiment with different turbulence profiles and gain experience with the method, which is essential before moving on to PAPYRUS.

The neural network's capability to detect and exploit hidden features allied to the MDP formulation and implementation of the RL technique seems to allow an improved correction. The PO4AO approach consistently optimizes the residual WFS measurement (and consequently the SR) in different turbulence scenarios, with the advantage of being able to adapt to changing conditions through online learning. In general, neural networks indicate to be a promising solution for wavefront correction, emphasizing the potential of reinforcement learning to enable telescopes to dynamically optimize their corrections for atmospheric turbulence.

In order to offer a more accurate simulation, further experiments will be focused on improving the PAPYRUS simulation, such as the incorporation of a central obstruction and integration of telescope spiders. We also look forward to optimizing the neural network's parameters and overall trying to better the performance of the algorithm by fine-tuning the many hyperparameters of the method. Another future improvement we intend to make is to gradually change the turbulence conditions of the simulation during a single experiment instead of trying each turbulence configuration separately in a unique simulation. For example, increasing the wind speed and changing seeing conditions during the simulation. Misregistration and vibrations should also be added to the PAPYRUS model in order to make the simulation more realistic. In our upcoming experiments, we plan to enhance the PAPYRUS simulation model by upgrading it to a more recent and precise version. This upgrade will bring us closer to achieving results that mirror those obtained from the actual bench. Additionally, we aim to increase the magnitude of the calibration source while also calculating the error in the transfer function.

In our upcoming experiments, we plan to enhance the PAPYRUS simulation model by upgrading it to a more recent and precise version. This upgrade will bring us closer to achieving results that mirror those obtained from the actual bench.

Finally, we aim to continue this project by proceeding with the mentioned experiments on the simulated bench and then deploying the algorithm on the PAPYRUS, under calibration source and, subsequently, on-sky. In order to accomplish this goal, there are a few challenges related to PAPYRUS's hardware that need to be addressed. First, to successfully execute PO4AO on the bench, the system must be provided with an integrated GPU, giving the system the ability to execute the neural network. Lastly, the software must be adapted and integrated into the PAPYRUS's RTC, which requires changes to the software's current interface. We hope to manage these challenges in the near future and further to explore the capabilities of reinforcement learning for adaptive optics.

ACKNOWLEDGMENTS

This work benefited from the support of the the French National Research Agency (ANR) with WOLF (ANR-18-CE31-0018), APPLY (ANR-19-CE31-0011) and LabEx FOCUS (ANR-11-LABX-0013); the Programme Investissement Avenir F-CELT (ANR-21-ESRE-0008), the Action Spécifique Haute Résolution Angulaire (ASHRA) of CNRS/INSU co-funded by CNES, the ECOS-CONYCIT France-Chile cooperation (C20E02), the ORP-H2020 Framework Programme of the European Commission's (Grant number 101004719), STIC AmSud (21-STIC-09), the Région Sud and the french government under the France 2030 investment plan, as part of the Initiative d'Excellence d'Aix-Marseille Université -A*MIDEX, program number AMX-22-RE-AB-151. This research has made use of computing facilities operated by CeSAM data center at LAM, Marseille, France

References

[1] Maxime Dumont et al. "Deep learning for space-borne focal-plane wavefront sensing". In: *Space Telescopes and Instrumentation 2022: Optical, Infrared, and Millimeter Wave*. Vol. 12180. SPIE. 2022, pp. 1107–1114.

[2] J Fowler and Rico Landman. "Tempestas ex machina: A review of machine learning methods for wavefront control". In: *arXiv preprint arXiv:2309.00730* (2023).

[3] Cédric Taïssir Héritier et al. "Object oriented python adaptive optics (OOPAO)". In: *AO4ELT-7 proceedings* (2023), https://github.com/cheritier/OOPAO.

[4] Rico Landman et al. "Self-optimizing adaptive optics control with reinforcement learning for high-contrast imaging". In: *Journal of Astronomical Telescopes, Instruments, and Systems* 7.3 (2021), pp. 039002–039002.

[5] Eduard Muslimov et al. "Current status of PAPYRUS: the pyramid based adaptive optics system at LAM/OHP". In: *Optical Instrument Science, Technology, and Applications II*. Vol. 11876. SPIE. 2021, pp. 56–68.

[6] J Nousiainen et al. "Toward on-sky adaptive optics control using reinforcement learning-Model-based policy optimization for adaptive optics". In: *Astronomy & Astrophysics* 664 (2022), A71.

[7] Jalo Nousiainen et al. "Adaptive optics control using model-based reinforcement learning". In: *Optics Express* 29.10 (2021), pp. 15327–15344.

[8] Jalo Nousiainen et al. "Advances in model-based reinforcement learning for adaptive optics control". In: *Adaptive Optics Systems VIII*. Vol. 12185. SPIE. 2022, pp. 882–891.

[9] Aske Plaat. *Deep reinforcement learning*. Vol. 10. Springer, 2022.

[10] B Pou et al. "Adaptive optics control with multi-agent model-free reinforcement learning". In: *Optics express* 30.2 (2022), pp. 2991–3015.

[11] B Pou et al. "Model-free reinforcement learning with a non-linear reconstructor for closed-loop adaptive optics control with a pyramid wavefront sensor". In: *Adaptive Optics Systems VIII*. Vol. 12185. SPIE. 2022, pp. 945–958.

[12] Jeffrey Smith et al. "Image-to-image translation for wavefront and point spread function estimation". In: *Journal of Astronomical Telescopes, Instruments, and Systems* 9.1 (2023), pp. 019001–019001.

[13] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.