REINFORCEMENT LEARNING FOR CFD APPLICATIONS

D. Huergo [a*], D. Soler [a], L. Alonso [a], O. Mariño [a], M. de Frutos [a], S. Joshi [a], A. Juanicoteca [a], G. Rubio [a,b], E. Ferrer [a,b]

[a] ETSIAE-UPM-School of Aeronautics, Universidad Politécnica de Madrid, Plaza Cardenal Cisneros 3, E-28040 Madrid, Spain.
[b] Center for Computational Simulation, Universidad Politécnica de Madrid, Campus de Montegancedo, Boadilla del Monte, 28660 Madrid, Spain.

ABSTRACT

Reinforcement learning (RL) has emerged as a promising approach to automating decision processes in CFD simulations. It allows to improve the performance of the simulations (higher accuracy and lower computational time) and to find optimum operation points for changing environments (e.g., maximum power generation in wind turbines...). We explore different optimization problems, from a RL point of view, to improve our CFD simulations.

First, we explore the application of RL techniques to optimize the polynomial order in the computational mesh when using high-order h/p solvers [1]. Mesh adaptation plays a crucial role in improving the efficiency of numerical simulations by improving accuracy while reducing the cost. Here, actor-critic RL models based on Proximal Policy Optimization offer an approach for agents to learn optimal mesh modifications based on evolving conditions. We provide a strategy for p-adaptation in high-order solvers, including insights into the main aspects of RL-based mesh adaptation: the formulation of appropriate reward structures and the interaction between the RL agent and the simulation environment. The proposed strategy does not require a high-fidelity solution during the training process and the formulation is general for any computational mesh and partial differential equation, solved in a discontinuous Galerkin solver.

Also, we explore the use of RL, specifically Proximal Policy Optimization (PPO), to accelerate the solution of advection-diffusion equations on a multigrid hierarchy [2]. Multigrid methods are a well-established class of algorithms that exploit the idea of solving a problem on multiple grids of different sizes, called a hierarchy of discretizations, to accelerate the convergence of iterative solvers for partial differential equations. Multigrid methods have parameters that must be chosen for stability and efficiency. We propose combining RL with multigrid methods to further improve the efficiency of the numerical solution of advection-diffusion equations.

Finally, we propose a RL strategy to control wind turbine energy generation, by actively changing the rotor speed, the rotor yaw angle and the blade pitch angle [3]. A double deep Q-Learning with prioritized experience replay agent is coupled with a blade element momentum model and is trained to allow control for changing winds. The agent is trained to decide the best control (speed, yaw, pitch) for simple steady winds and is subsequently challenged with real dynamic turbulent winds, showing good performance and great adaptability.

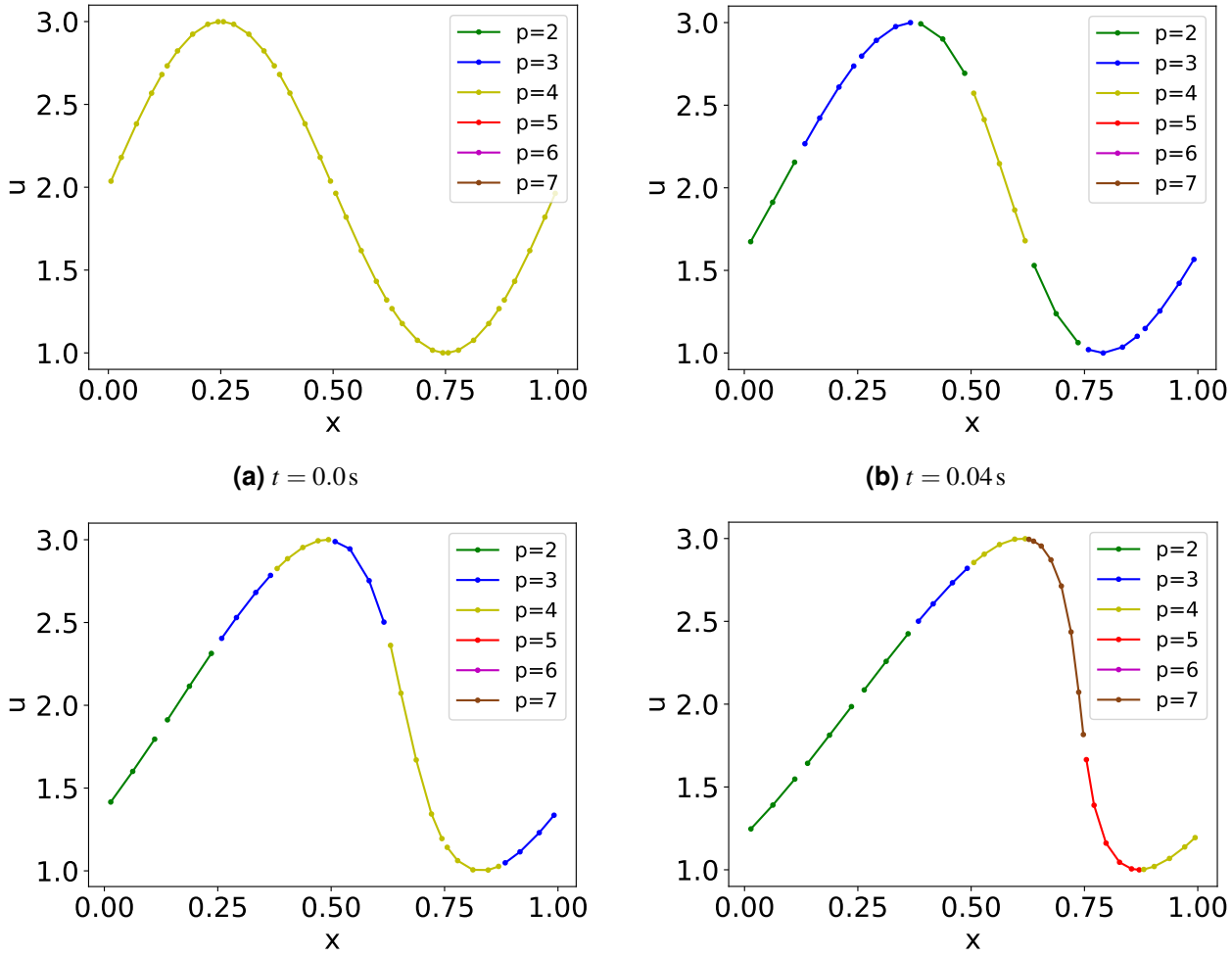
All these test cases show an overview of the capabilities of Reinforcement Learning and how it can highly improve our CFD solvers by reducing the required computational time, leading to more accurate simulations and providing an alternative solution for optimization problems based on fluid dynamics.

*Correspondence to david.huergo.perea@upm.es

A REINFORCEMENT LEARNING STRATEGY FOR P-ADAPTATION

To demonstrate the effectiveness of the proposed methodology, the trained RL agent is used to improve the accuracy of a Burgers' equation. In a separate phase, the RL agent has learnt how to increase or decrease the polynomial order of each element of a given mesh to obtain an optimum performance; that is, a very low error at the lowest computational cost.

The solution for $t \in [0, 0.12]$ s is represented in Figure 1. The adaptation process is performed each 0.01 s of simulation time, that is each 1000 iterations, for every element in the mesh. In this case, the RL model is able to dynamically modify the polynomial order, reducing the order when possible and increasing the order near regions with strong gradients.



(a) $t = 0.0$ s **(b)** $t = 0.04$ s

(c) $t = 0.08$ s **(d)** $t = 0.12$ s

FIGURE 1: 1D inviscid Burgers' equation solved with a DGSEM spatial scheme and an explicit Euler temporal scheme, using a RL p-adaptation algorithm. The polynomial of each element of the computational mesh has been adapted once every 0.01 s of simulation time, starting with a polynomial order $p = 4$ in every element. The solution is shown in four different timestamps.

REINFORCEMENT LEARNING TO ACCELERATE P-MULTIGRID

The PDEs in which this method has been tested are the one-dimensional linear advection-diffusion (1) and the Burgers' (2) equations:

$$\frac{\partial u}{\partial t} = \nu \frac{\partial^2 u}{\partial x^2} - a \frac{\partial u}{\partial x} \tag{1}$$

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2} + f \tag{2}$$

where $\nu \in \mathbb{R}$ is the kinematic viscosity and $a \in \mathbb{R}$ is the constant advection velocity, $u(x,t) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ the scalar quantity of interest; $t \in \mathbb{R}^+$ is the time and $x \in [0,1]$ the spatial coordinate. f is a source-term added to achieve a steady-state solution. We will impose Dirichlet boundary conditions and the initial condition of a seven-frequency-sine function.

In Table 1, the performance of a PPO agent trained in an hp-environment is compared to the performance of the original solver in a hp-multigrid context with varying coefficients. The use of RL and PPO enables the development of an agent that learns to exploit the hierarchical structure of the multigrid method, leading to reduced computational effort and faster convergence. The agent's policy parameters are optimized iteratively through interactions with the environment, allowing for the discovery of effective strategies for navigating the multigrid hierarchy and reducing the run-time.

REINFORCEMENT LEARNING TO MAXIMISE WIND TURBINE ENERGY GENERATION

The performance of the different agents in the environments is assessed. To do so, we consider four agents:

1. A RL agent with optimized hyperparameters,
2. A RL agent with arbitrary hyperparameters (not optimised),
3. A classic PID controller,
4. Fixed operating conditions (or uncontrolled agent).

Figure 2 shows the performance of each agent across the three validation environments. In this plot, the optimized RL agent (orange) is the most efficient for all environments. Its performance (although eclipsed in the *narrow* environment by the perfect control achieved by the RL agent with arbitrary hyperparameters) is consistently better than that the classic PID control and, of course, much better than the uncontrolled agent. Moreover, the optimized RL agent experiences a less severe drop in performance as the environments become harder, proving its adaptability. The difference in performance between RL agents is remarkable and emphasizes the relevance of carefully tuning the agent's hyperparameters.

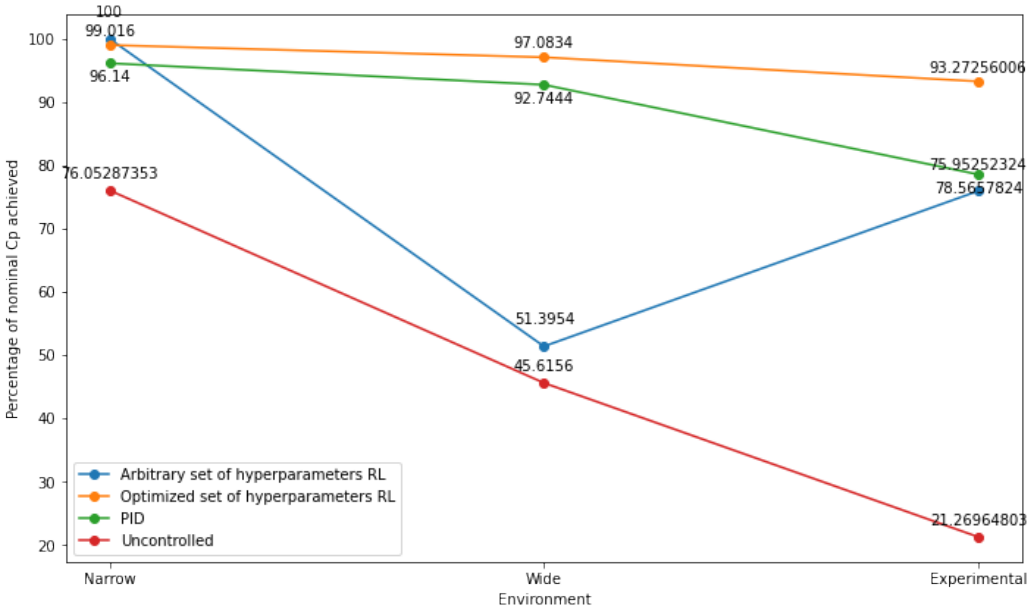


FIGURE 2: Performance of four control strategies (including RL and PID) for three wind environments.

REFERENCES

[1] D. Huergo, G. Rubio, E. Ferrer *A reinforcement learning strategy for p-adaptation in high order solvers,* arXiv:2306.08292 (preprint).

[2] L. Alonso, S. Joshi, A. Juanicotena, D. Huergo, E. Ferrer *A reinforcement learning strategy to accelerate p-multigrid in high-order flux reconstruction solvers,* under review.

[3] D. Soler, O. Marino, D. Huergo, M. de Frutos, E. Ferrer *Reinforcement learning to maximise wind turbine energy generation,* under review.

TABLE 1: Run-time and number of iterations of the runs of the original solver compared to PPO for order 2, 3 and 5. For clarity, the shorter run-times are in bold.

hp-multigrid, non-uniform mesh

Original		PPO		
Runtime (s)	**Nº iterations**	**Runtime (s)**	**Nº iterations**	**Final residual**
Linear advection-diffusion - Order 2				
a= 1., ν = 0.01				
69.72	197	**31.03**	626	9.67e-9
a= 0.5., ν = 0.01				
80.01	207	**31.81**	651	8.33e-9
a= 0.5., ν = 0.5				
808.60	2178	**33.21**	652	9.31e-9
a= 0.4, ν = 0.6				
634.27	3166	**31.52**	654	9.29e-9
a= 0.2, ν = 0.8				
1,476.48	8063	**31.48**	648	9.98e-9
Linear Advection-Diffusion - Order 3				
a= 1., ν = 0.01				
51.73	228	**33.98**	658	9.33e-9
a= 0.5., ν = 0.01				
54.73	237	**35.34**	659	9.87e-9
a= 0.5., ν = 0.5				
2,168.68	6094	**130.78**	1602	7.89e-9
a= 0.4, ν = 0.6				
1,861.64	8915	**178.78**	2201	9.87e-9
a= 0.2, ν = 0.8				
2,467.52	9444	**157.62**	1958	8.62e-9
Linear Advection-Diffusion - Order 5				
a= 1., ν = 0.01				
142.62	479	1,081.12	7094	9.37e-9
a= 0.5., ν = 0.01				
2,500.01	8053	**1,203.32**	7126	9.98e-9
a= 0.5., ν = 0.5				
17,976.47	37647	**1,200.36**	7102	9.77e-9
a= 0.4, ν = 0.6				
18,003.09	37647	**1,213.74**	7115	9.06e-9
a= 0.2, ν = 0.8				
2,467.52	9444	**1,194.44**	7101	9.62e-9
Burgers - Order 2				
v = 0.05				
296.45	799	**90.81**	1524	8.64e-9
ν = 0.3				
1,022.65	3109	**79.14**	1328	9.43e-9
ν = 0.5				
1,593.30	5222	**82.47**	1388	8.77e-9
ν = 0.8				
2,544.08	8853	**84.51**	1400	9.32e-9
ν = 1				
3,102.94	11209	**99.00**	1644	9.92e-9
Burgers - Order 3				
ν = 0.05				
453.92	1351	**84.66**	1405	9.72e-9
ν = 0.3				
2,729.82	7371	**86.48**	1411	8.10e-9
ν = 0.5				
4,990.60	14402	**86.03**	1400	9.18e-9
ν = 0.8				
10,067.14	31683	**94.25**	1534	9.99e-9
ν = 1				
13,590.96	47423	**74.06**	1217	9.98e-9
Burgers - Order 5				
ν = 0.05				
1,290.16	3681	**830.71**	7621	1.00e-8
ν = 0.3				
8,566.44	21486	**1,602.69**	7680	9.57e-9
ν = 0.5				
4,990.60	14402	**1,536.88**	7343	9.98e-9
ν = 0.8				
10,067.14	31683	**1,545.23**	7573	9.41e-9
ν = 1				
13,590.96	47423	**1,640.68**	8007	9.94e-9