Investigating gas furnace control practices with reinforcement learning

M Renault, J Viquerat, P Meliga, G.-A Grandin, N Meynet, E Hachem

Investigating gas furnace control practices with reinforcement learning

M. Renault[a], J. Viquerat[a], P. Meliga[a], G.-A. Grandin[b], N. Meynet[b], E. Hachem[a,*]

[a]*Mines Paris, PSL University, Centre for material forming (CEMEF), UMR CNRS, 06904 Sophia Antipolis, France*
[b]*ENGIE Lab CRIGEN, 4 rue Joséphine Baker 93240 Stains, France*

Abstract

Gas furnaces are the most widely used means of heating in industry, and with the growing concern for environmental issues, and a global energy crisis at our doorstep, the optimization of the processes related to them becomes a key challenge. This paper aims at introducing a new way of practicing gas furnace control involving simulations, virtual sensors and deep reinforcement learning (DRL) techniques. In order to do so we designed a set of simulations of conjugate heat transfer systems governed by the coupled Navier–Stokes and heat equations for single-step control. The DRL algorithm used in this paper is the policy-based optimization (PBO) algorithm specialized in single-step (or open-loop) control. We explore its ability to find global maxima in different situations and under various constraints. Therefore, various 2D and 3D cases are tackled, in which the position of the work piece, the flow rate, and other parameters are controlled. The obtained results highlight the potential of the DRL framework combined with computational fluid dynamics (CFD) conjugate heat transfer systems for optimizing searches in large parameter spaces. For the 2D case, PPO achieved an increase of 89% in temperature homogeneity, and for the 3D case an increase of 7% in final temperature with the same total input.

Keywords: Deep Reinforcement Learning; Artificial Neural Networks; Conjugate heat transfer; Computational fluid dynamics; Thermal control; Serpentine.

1. Introduction

Just like cooling, properly heating a part is a challenge that manufacturers take up every day to obtain the desired properties in the part, whether they are mechanical, electrical, optical or aesthetic. To achieve that, gas furnaces have been studied for a long time and their control has evolved to arrive today at the cohabitation of many techniques, including manual control, fuzzy control [1], proportional-integral (PI)/proportional-integral-derivative (PID) single-loop control or cascade control [2, 3] among others. The reason for this flourishing of techniques is the difficulty to accurately perform temperature control, whether it is because of the difficulty to measure the temperature accurately inside the chamber, or because of the different characteristics of temperature variation through flow control such as non linearity, inertia and time delay.

Temperature control has been addressed by a lot of articles in the literature, and with the emergence of Artificial Neural Networks (ANNs) taking advantage of the most recent advances in computational power and data analysis, new control techniques that are more robust and can deal with non linearity, inertia and time delay have been developed : image processing and clustering to control gas burners [4, 5], recurrent neural networks (RNNs) with electrical furnaces [6, 7], neural networks for metal quality control [8] and for flame stability control [9–11], and radial basis function neural networks (RBFNNs) to control coke furnaces [12]. In the aerospace industry, burners are also very thoroughly studied to prevent any incident inside aircraft engines. They use neural networks in the active control of the burning chambers and find ways to prevent oscillating patterns to occur during combustion [13]. Such patterns are generally responsible for a premature

*elie.hachem@minesparis.psl.eu

fatigue of the materials composing the chamber, as well as disturbances to the good functioning of the combustion. This is also true for industrial gas furnaces ; even though these issues are less explored in the field of furnace control, they become more and more important with the use of mixed fuels, and the alternating use of natural gas and dihydrogen.

In the realm of optimal control with constraints on the state of the system, the combination of deep neural networks (DNNs) and reinforcement learning (RL) algorithms (a formal framework in which an agent learns by interacting with an environment and learns by gathering experience) has brought new cards to the table in such a way that the best performing algorithms in a wide variety of tasks (*e.g.*, games [14, 15], cooling control [16], autonomous cars [17], medicine [18], energy [19, 20]). In fluid dynamics, this so-called deep reinforcement learning (deep RL, or DRL) has also been used with success for flow control and shape optimization with success (Refs. [21, 22] and the references therein) by taking advantage of its robustness to non linearity and to high dimensional spaces. This field of application is still at an early stage, as evidenced by the scarce literature dedicated to DRL-based thermal control [23, 24], but it shows great promise for the future of fluid related topics.

This work aims at introducing DRL into the field of gas furnace control. More specifically, it assesses the performances of proximal policy optimization (PPO [25]) for the one-step control of a heating chamber. We use an algorithm introduced in [26] and whose relevance for open-loop flow control problems is assessed in [27] that is a degenerate version of the classical PPO algorithm. The choice for PPO is driven by its data efficiency (a decisive criteria for computationally expensive simulations), ease of implementation and already widely assessed performance. Several problems of conjugate heat transfer in two and three dimensions are used as testbed to push forward the development of this novel approach. To the best of the authors knowledge, this constitutes the first attempt to achieve DRL-based control of conjugate forced convection heating processes.

2. Governing equations for fluid mechanics

The focus of this research is on conjugate heat transfer and laminar, incompressible fluid flow problems in two and three-dimensions, for which the conservation of mass, momentum and energy is described by the nonlinear, coupled Navier–Stokes and heat equations

$$\nabla \cdot \boldsymbol{u} = 0\,, \tag{1}$$

$$\rho(\partial_t \boldsymbol{u} + \boldsymbol{u} \cdot \nabla \boldsymbol{u}) = \nabla \cdot (-p\mathbf{I} + 2\mu\boldsymbol{\varepsilon}(\boldsymbol{u}))\,, \tag{2}$$

$$\rho c_p(\partial_t T + \boldsymbol{u} \cdot \nabla T) = \nabla \cdot (\lambda \nabla T)\,, \tag{3}$$

where $\boldsymbol{u}$ is the velocity field, p is the pressure, T is the temperature, $\boldsymbol{\varepsilon}(\boldsymbol{u}) = (\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T)/2$ is the rate of deformation tensor, we assume here constant fluid density ρ, dynamic viscosity μ, thermal conductivity λ, and specific heat c_p, and we have neglected buoyancy and radiative heat transfer, on behalf of the focus being on conjugate forced convection heat transfer.

This is solved here with an in-house stabilized finite elements environment cast in the Variational Multiscale (VMS) framework. This allows using equal order linear approximations for all variables (very desirable due to its simplicity of implementation and affordable computing cost) by enhancing the stability of the Galerkin method via a series of additional derived residual based terms evaluated over element interior. The solid is treated as an immersed body, using the Immerse Volume Method to compute the amount of heat exchanged between the solid and the fluid only from the individual material properties on either side of it (which in turn removes the need for a heat transfer coefficient, a limiting issue for the present numerical experiments where we vary the position of the solid). For details about the numerical framework, including the interface capturing method used to generated strongly anisotropic meshes adapted at the fluid-solid boundary (to ensure that the fluid properties are distributed as accurately and smoothly as possible over the smallest possible thickness around the interface), the reader can refer to Refs. [24, 28]. The relevance of this numerical method has been validated on multiple benchmarks here [29] and in particular for heat transfer here [30]. For the sake of simplicity we will not discuss it here.

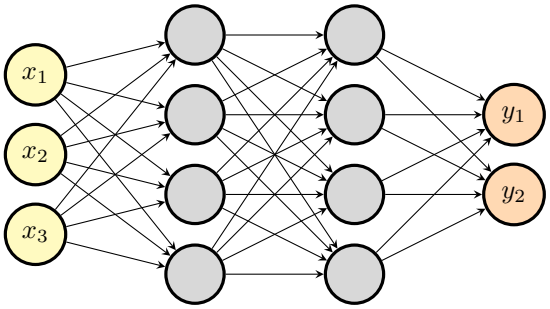

Figure 1: Fully connected neural network with two hidden layers, modeling a mapping from data living in $\mathbb{R}^3$ to data living in $\mathbb{R}^2$.

3. Deep reinforcement learning and proximal policy optimization

3.1. Neural networks

A neural network (NN) is a connected collection of non-linear functions. It is extremely useful when trying to mimic the relationship between multiple highly non-linear phenomena. A fully connected network is sketched in figure 1, it has layers (represented as columns in the sketch) and within each of them the neurons are connected to all of those contained in the next layer. The input layer is the one that receives information from the outside, the output layer is the one that gives the result and between them are hidden layers. Building an efficient neural network requires a relevant architecture (e.g., type of network, depth, width of each layer), finely tuned hyper-parameters (i.e., parameters that cannot be learned directly by the neural network, e.g., optimizer, learning rate, batch size) and an adequate amount of data to learn from. More details here Ref. [31] and the references therein.

3.2. Deep reinforcement learning

Reinforcement learning (RL) is a type of machine learning in which an agent learns the actions to do in an environment in order to get the best reward possible. Such method is often formulated as a Markov Decision Process for which a full loop looks like this :

- Assume the environment is in state $s_t \in S$ at iteration t, where S is a set of states;
- The agent uses w_t, an observation of the current environment state (and possibly a partial subset of s_t) to take action $a_t \in A$, where A is a set of actions;
- The environment reacts to the action by transitioning from s_t to state $s_{t+1} \in S$;
- The agent is fed with a reward $r_t \in R$, where R is a set of rewards, and a new observation w_{t+1}.

This repeats until a steady state is reached. The succession of states and actions defines a trajectory $\tau = (s_0, a_0, s_1, a_0, \dots)$ for which the agent will try and maximize the cumulative reward at each step, by choosing an action. That is why the most common quantity of interest in RL is the discounted cumulative reward :

$$R(\tau) = \sum_{t=0}^{T} \gamma^t r_t, \tag{4}$$

where T is the final time, and $\gamma \in [0, 1]$ is the discount factor that weights the importance of rewards according to their distance to the present (the agent being short-sighted in the limit where $\gamma \to 0$, and far-sighted in the limit where $\gamma \to 1$).

There exist two main types of RL algorithms : model-based methods either have access to the environment and therefore know the probability distribution of the states they end up in, or try to build an approximation of it ; model-free methods don't try to understand the environment, but only communicate with it and try to find the best actions to take, these methods are prominent in the DRL community.

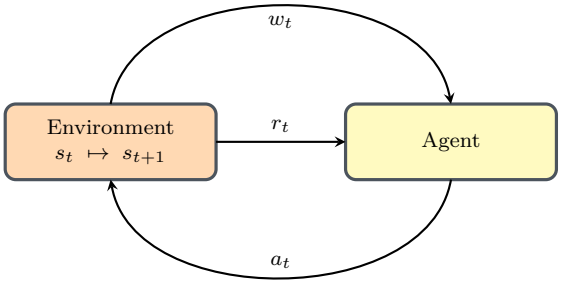

Figure 2: RL agent and its interactions with its environment.

Inside the model-free family of algorithms are also two main techniques : value-based methods that learn to predict the future rewards of a set of actions in order to pick the best one, and policy-based methods that keep in memory a policy that maps states to actions and learn to obtain the best rewards by modifying this policy. Many DRL algorithms in the community, including PPO, the one used in this paper, use gradient ascent to optimize a parameterized policy with respect to the expected return and therefore belong to the family of policy gradient methods. For a more thorough introduction to the taxonomy of RL methods (together with their respective pros and cons) please refer to Ref. [32].

3.3. From policy methods to Proximal policy optimization

This section is intended for non-expert readers and provides an overview of the basic principles and prerequisites of the Policy Gradient Method and the various steps taken to improve it.

- *Policy methods.* A policy method maximizes the expected discounted cumulative reward of a decision policy π mapping states to actions. It doesn't use a value function as explained before, but a probability distribution to determine which actions are best at any given state. Policies being often stochastic, the following notations are introduced:

- $\pi(s, a)$ is the probability of taking action a in state s under policy π,
- $Q^\pi(s, a)$ is the expected value of the discounted cumulative reward after taking action a in state s (also termed state-action value function or Q-function)

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[R(\tau) | s, a \right], \tag{5}$$

  where $\mathbb{E}_\pi$ is the expected value $\mathbb{E}$ under policy π.

- $V^\pi(s)$ is the expected value of the discounted cumulative reward in state s (also termed value function or V-function)

$$V^\pi(s) = \mathbb{E}_\pi \left[R(\tau) | s \right]. \tag{6}$$

  The V and Q functions are thus related such that

$$V^\pi(s) = \sum_a \pi(s, a) Q^\pi(s, a), \tag{7}$$

  so $V^\pi(s)$ can also be understood as the probability-weighted average of discounted cumulative rewards over all possible actions in state s.

- *Policy gradient methods.* A policy gradient method aims at optimizing a parameterized policy π_θ, where θ denotes the free parameters whose values can be learnt from data (as opposed to the hyper parameters). In practice, one defines an objective function based on the expected discounted cumulative reward

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[R(\tau)\right], \tag{8}$$

and looks for θ^* maximizing $J(\theta)$:

$$\theta^* = \arg\max_\theta \mathbb{E}_{\pi_\theta}\left[R(\tau)\right]. \tag{9}$$

One can try to do this by estimating the policy gradient $\nabla_\theta J(\theta)$ and using a gradient ascent algorithm. This is certainly a difficult task as one is seeking a gradient that depends on the policy parameters, but also on the whole space of state-action pairs, in a context where the effects of policy changes on the state probability distribution are unknown (since modifying the policy will most likely modify the probability distribution over the set of visited states). One commonly used estimator, derived in [32] using the log-probability trick, reads

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log\left(\pi_\theta(s_t, a_t)\right) R(\tau)\right] \sim \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log\left(\pi_\theta(s_t, a_t)\right) \widehat{A}^\pi(s_t, a_t)\right], \tag{10}$$

where $\widehat{A}^\pi$ is some biased estimate (here its normalization to zero mean and unit variance) of the advantage function

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s), \tag{11}$$

that measures the improvement (if $A^\pi > 0$, otherwise the lack thereof) associated with taking action a in state s (Q-function) compared to taking the average over all possible actions (V-function). This is possible because the V-function doesn't depend on the action, and therefore doesn't change the expected value, but it shows experimentally that it also reduces the variance and speeds up the learning. When the policy π_θ is represented by a neural network (in which case θ simply represents the network parameters), we tend to estimate the policy loss

$$L(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \log\left(\pi_\theta(a_t|s_t)\right) \widehat{A}^\pi(s_t, a_t)\right], \tag{12}$$

whose gradient is equal to the (approximated) policy gradient (10) and is computed with respect to each parameter of the neural network by using the chain rule at each layer with the back-propagation algorithm [33].

- *Trust regions.* The learning rate of the policy gradient methods, i.e. the size of the steps taken at each learning iteration, has a large impact on their performances. Too small, and the learning will never end, too large, and it will be difficult to get out of degenerate regions where the gradient is already high or noisy. Fine-tuning the learning rate could be a solution, but it asks sometimes for too much work finding the right balance. One way to stay in the range of improvement is to define a maximum distance between the new policy and the old one, this way, even when the gradient becomes too high, the trust region clips the distance and avoid the aforementioned issues. We will not dwell on the intricate details of the many algorithms developed to solve such trust region optimization problems, e.g., natural policy gradient (NPG [34]), or trust region policy optimization (TRPO [35]). Suffice it to say that they use the MinMax algorithm to maximize iteratively a surrogate policy loss (i.e. a lower bound approximating locally the actual loss at the current policy), but are difficult to implement and can be computationally expensive, as they rely on an estimate of the second-order gradient of the policy log probability.

- *Proximal policy optimization.* Proximal policy optimization (PPO) is a similar approach to TRPO (on which it is based) but with a simpler heuristic that uses a probability ratio between the two policies to maximize improvement without the risk of performance collapse [25]. The focus here is on the PPO-clip algorithm[1], that optimizes the surrogate loss

$$L(\theta) = \mathbb{E}_{\pi_\theta}\left[\min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}, g(\epsilon, \widehat{A}^\pi(s,a))\right)\widehat{A}^\pi(s,a)\right], \tag{13}$$

where

$$g(\epsilon, A) = \begin{cases} 1+\epsilon & A \geq 0\,, \\ 1-\epsilon & A < 0\,, \end{cases} \tag{14}$$

and $\epsilon \in [0.1, 0.3]$ is the clipping range, a small hyper parameter defining how far away the new policy is allowed to go from the old. Its range is adapted from the paper on Proximal Policy Optimization [36] and was confirmed in our own implementation. The general picture is that a positive (resp. negative) advantage increases (resp. decreases) the probability of taking action a in state s, but always by a proportion smaller than ϵ, otherwise the min kicks in (13) and its argument hits a ceiling of $1+\epsilon$ (resp. a floor of $1-\epsilon$). This prevents stepping too far away from the current policy, and ensures that the new policy will behave similarly but hopefully in a better way.

3.4. Single-step PPO

We now come to single-step PPO, a "degenerate" version of PPO introduced in [26] and intended for situations where the optimal policy to be learnt by the neural network is state-independent, i.e. $\pi_\theta(a,s) = \pi_\theta(a)$, as is notably the case in optimization and open-loop control problems (closed-loop control problems conversely require state-dependent policies for which standard PPO is best suited). The main difference between standard and single-step PPO can be summed up as follows: where standard PPO seeks the optimal set of parameters $\theta^\star$ leading to the largest possible cumulative reward over one episode, single-step PPO seeks the optimal parameters $\theta^\star$ such that $a^\star = \pi_{\theta^\star}(s_0)$, where s_0 is some input state (usually a constant vector of zeros) consistently fed to the agent for the optimal policy to eventually embody the transformation from s_0 to $a^\star$. The agent initially implements a random initial policy determined by the free parameters θ_0, after which it gets only one attempt per learning episode at finding the optimal (i.e., it interacts with the environment only once per episode). This is illustrated in figure 3 showing the agent draw a population of actions a_t from the current policy, and being returned incentives from the associated rewards to update the free parameters for the next population of actions $a_{t+1} = \pi_{\theta_{t+1}}(s_0)$ to yield larger rewards.

In practice, the agent outputs a policy parameterized by the mean and variance of the probability density function of a d-dimensional multivariate normal distribution, with d the dimension of the action required by the environment. Actions drawn in $[-1,1]^d$ are then mapped into relevant physical ranges, a step deferred to the environment as being problem-specific. The resolution essentially follows the process described in section 3.3, only a normalized averaged reward substitutes for the advantage function. This is because classical PPO is actor-critic, i.e., it improves the learning performance by updating two different networks, a first one called actor that controls the actions taken by the agent, and a second one called critic, that learns to estimate the advantage from the value function as

$$A(s_t, a_t) = r_t + \gamma V(s_{t+1}) - V(s_t)\,. \tag{15}$$

In single-step PPO, the trajectory consists of a single state-action pair, so the discount factor can be set to $\gamma = 1$ with no loss of generality. In return, the advantage reduces to the whitened reward since the two rightmost terms cancel each other out in (15). This means that the approach can do without the value-function evaluations of the critic network, i.e., it is not actually actor-critic.

[1] There is also a PPO-Penalty variant which uses a penalization on the average Kullback–Leibler divergence between the current and new policies, but PPO-clip performs better in practice.

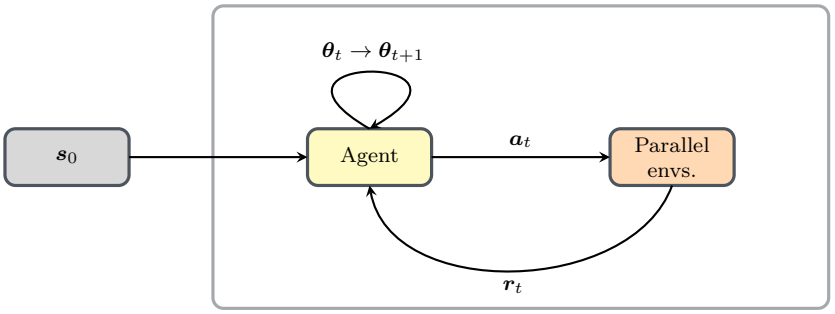

Figure 3: Action loop for single-step PPO. At each episode, the input state $\boldsymbol{s}_0$ is provided to the agent, which in turn provides n actions to n parallel environments. The latter return n rewards, that evaluate the quality of each action taken. Once all the rewards are collected, an update of the agent parameters is made using the PPO loss (13).

(a) (b)

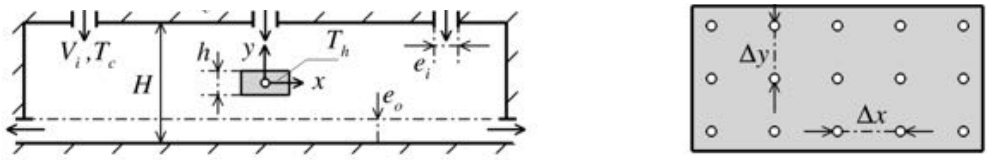

Figure 4: (a) Schematic of the 2D forced convection set-up. (b) Sensors positions in the solid domain.

4. Control of forced convection heating in a 2D open cavity

4.1. Case description

This test case is based on the second test case of this paper [24] with a few twists. Here we address the control of conjugate heat transfer for the heating of a piece by injection of a hot fluid in the chamber. We use a Cartesian coordinate system with origin at the center of the chamber. The solid has a rectangular shape with height h and aspect ratio 2:1, and is initially at the cold temperature T_c. It can be fixed or move around the chamber according to the parameters we wish to control. Its density, thermal conductivity and heat capacity can take two values, depending on the type of solid we are trying to emulate in order two compare the two : brick or steel. The chamber itself has a rectangular shape with height H and aspect ratio 4:1, and its walls are isothermal at temperature T_w. The north wall of the chamber has three identical inlets of width e, each of which models the exit plane of an injector blowing hot air with constant temperature T_h and velocities $V_i \in [0.01, 0.99]$ subjected to

$$\sum_{i=1}^{3} V_i = 1\,, \tag{16}$$

to emulate a constant input to the chamber. The fluid is released through two identical outlets on each side of the chamber with height e_0, and positioned against the south wall.
In the absence of buoyancy, temperature evolves as a passive scalar to the Navier–Stokes equations. All parameters named above are provided in 1, along with the material properties used to model the composite fluid, that yield fluid values of the Reynolds and Prandtl numbers

$$\mathrm{Re} = \frac{\rho e \max_{i\in\{1,2,3\}} V_i}{\mu} \in [67, 200]\,, \qquad \mathrm{Pr} = \frac{c_p \mu}{\lambda} = 2\,. \tag{17}$$

Note the very high value of the solid to fluid viscosity ratio, meant to ensure that the velocity inside the solid is zero and that the no-slip condition on the boundary is satisfied. Thus, only pure conduction occurs in the solid. The governing equations are solved with no-slip isothermal conditions $\mathbf{u} = \mathbf{0}$ and $T = T_w$ on $\partial\Omega$, except at the inlets where $\mathbf{u} = -V_i \mathbf{e}_y$ and $T = T_h$, and at the exhausts where a zero-pressure condition is imposed : $p = \partial_x u = \partial_x T = 0$. No thermal condition is imposed at the interface, where heat exchange is implicitly driven by the difference

H	h	e	T_w	T_c	T_h	μ	ρ	λ	c_p	
						0.001	1	0.5	1000	Fluid
1	0.2	0.2	10	10	300	1000	1800	1	600	Brick
						1000	7800	50	500	Steel

Table 1: Geometric and numerical parameters used in the 2-D forced convection setup. All values in SI units, with the exception of temperature given in Celsius.

in the individual material properties, as intended in the Immersed Volume Method. The mesh on which the computation is performed is defined at the beginning only around the interface to ensure its good definition, then remeshing is performed every 5 time steps based on velocity gradients to also capture accurate velocity profiles. We aim at 20000 elements, with $h_{min} = 0.0001$. More about our remeshing technique here [37].

4.2. Control

The quantities subjected to optimization are the three inflow velocities $V_{i\in\{1,2,3\}}$, plus the insertion angle of the piece being heated, and the position of its center of mass, hence six control parameters (three for the inflow distribution, one for the angle and two for the position). We could have used two parameters to control the inflow distribution since the last one is constrained by the total inflow, but in order to avoid asymmetry in the learning process we decided to control each of the injectors in the same way. In practice, each injector is given a value between 0.01 and 0.99 which is then scaled in order to obtain their speed while following (16).

Just like in [24], we distribute 15 probes uniformly in the workpiece to compute the reward used by the DRL algorithm. The probes are arranged in an array of $n_x = 5$ columns and $n_y = 3$ rows with resolutions $\Delta x = 0.09$ and $\Delta y = 0.075$, respectively; see figure 4(b). The following formula gives an estimate of the tangential heat flux by averaging the norm of the temperature gradient across rows and columns respectively :

$$\langle||\nabla_\parallel T||\rangle_i = \frac{2}{n_y - 1}|\sum_{j\neq 0}\mathrm{sgn}(j)||\nabla T||_{ij}|\,, \qquad \langle||\nabla_\parallel T||\rangle_j = \frac{2}{n_x - 1}|\sum_{i\neq 0}\mathrm{sgn}(i)||\nabla T||_{ij}|\,, \tag{18}$$

where subscripts i, j and ij denote quantities evaluated at $x = i\Delta x$, $y = j\Delta y$ and $(x, y) = (i\Delta x, j\Delta y)$, respectively, and symmetrical numbering is used for the center probe to sit at the intersection of the zero-th column and row. The reward $r_t = -\langle||\nabla_\parallel T||\rangle$ fed to the DRL agent is given by the average of the quantities calculated before

$$r_t = -\frac{1}{n_x + n_y}\sum_{i,j}\langle||\nabla_\parallel T||\rangle_i + \langle||\nabla_\parallel T||\rangle_j\,, \tag{19}$$

which especially yields $r_t = 0$ for a perfectly homogeneous heating.

A second reward is also tested in this paper to assess the feasibility of controlling both the homogeneity and efficiency of the furnace. This is expressed as

$$\psi_t(w_1, w_2) = \frac{w_1}{n_x n_y}\sum_{i,j} T_{ij} + w_2 r_t\,, \tag{20}$$

where the first term is the right-hand side measures the solid temperature averaged across all sensors, and $w_{1,2}$ are scalar-valued factors weighing the priority given to each objective. In practice, a single point concurrently minimizing both objectives usually does not exist. The optimal solutions are thus to be understood as Pareto-efficient solutions [38] that best manage trade-offs between the two criteria, in the sense that further optimizing one objective decreases the performance of the other one (after which the final decision is made by the practitioner based on subjective preferences).

The agent is a fully-connected network with two hidden layers, each holding 2 neurons. The resolution process uses 8 environments and 2 steps mini-batches to update the network for 32 epochs, with learning rate set to 5×10^{-3}, and PBO loss clipping range to $\epsilon = 0.3$.

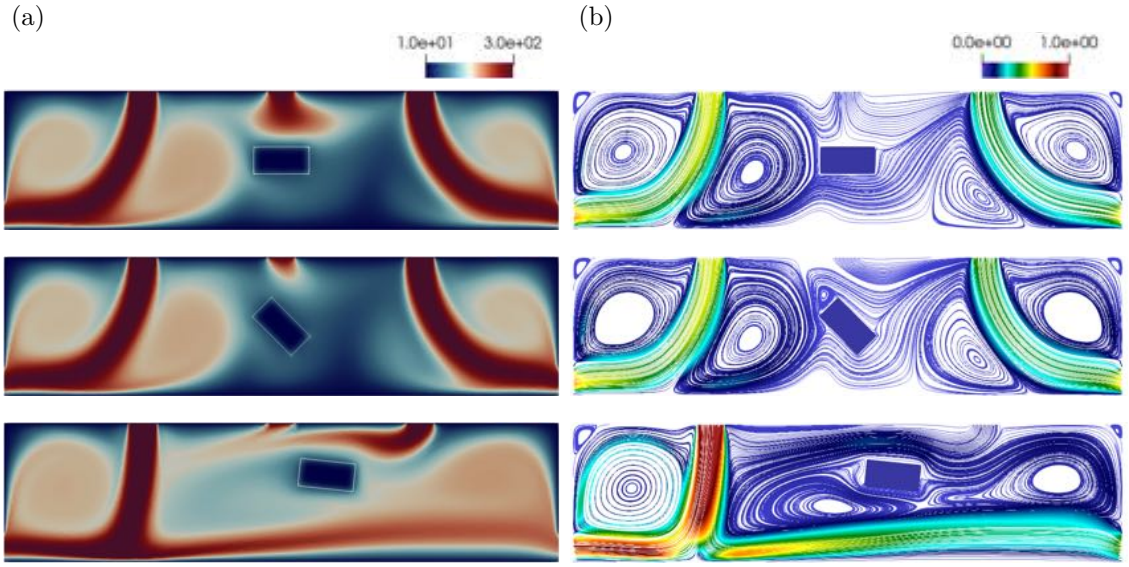

Figure 5: Control of heating homogeneity for the brick-like workpiece. (a) Optimal temperature distribution found by the DRL algorithm by controlling (from top to bottom) : the inflow velocities under constant workpiece angle and position, the inflow velocities and workpiece angle under constant position, and the inflow velocities, workpiece angle and position. (b) Corresponding streamlines colored by the magnitude of velocity.

4.3. Results

4.3.1. Control of heating homogeneity

We evaluate first the performance of the algorithm in several scenarios of increasing complexity, using the homogeneity-based reward r_t. Two different workpieces are used, one that has the properties of a brick-like material, and one that has the properties of a steel-like material. For each workpiece, three different cases are considered, in which the DRL agent is tasked with optimizing either the inflow velocities (under constant workpiece angle and position), or the inflow velocities and the workpiece angle (under constant position), or the inflow velocities, angle and position (the most general case).

For each case, 150 episodes have been run (1200 simulations), each of which performs 2000 iterations with time step $\Delta t = 0.1$ (hence a heating time of 200), starting from an initial condition consisting of zero velocity and uniform temperature (except in the solid domain), and using the level set, velocity and temperature as multiple-component criterion to adapt the mesh (initially pre-adapted using the sole level set) every 5 time steps under the constraint of a fixed number of elements $n_{el} = 15000$. This represents 1200 simulations, each of which is performed on 8 cores and lasts 10mn, hence 200h of total CPU cost. We can clearly see in figures 5 and 6 the flow patterns that develop when the blown fluid travels through the cavity. Moreover, it clearly depends on the inflow distribution and position of the piece, and features complex rebound phenomena (either fluid/solid, when a jet impinges on the workpiece, or fluid/fluid, when a deflected jet meets the crossflow of another jet), leading to the formation of multiple recirculations varying in number, position and size.

The results of the various optimization scenarios are shown in figure 5 for the brick-like material and figure 6 for the steel-like material. The use of the reward r_t has a great effect on the aspect of these results in any control configuration. Even though they are all different, we can spot some common features : the workpiece is kept in colder areas of the chamber, which allows for lower temperature gradients as prescribed by the reward, and it is also generally well surrounded by streamlines that ensure symmetry in the heating. In the simplest case where the agent has to control only the inflow velocities, it has no problem finding the best solution possible, which is true also when we add the workpiece angle (not shown here for conciseness), and they both present the features we mentioned.

When tasked with simultaneously optimizing all six parameters (inflow velocities, angle and position), we show in figure 7 that the algorithm learns quite fast up to episode 60, but pursues

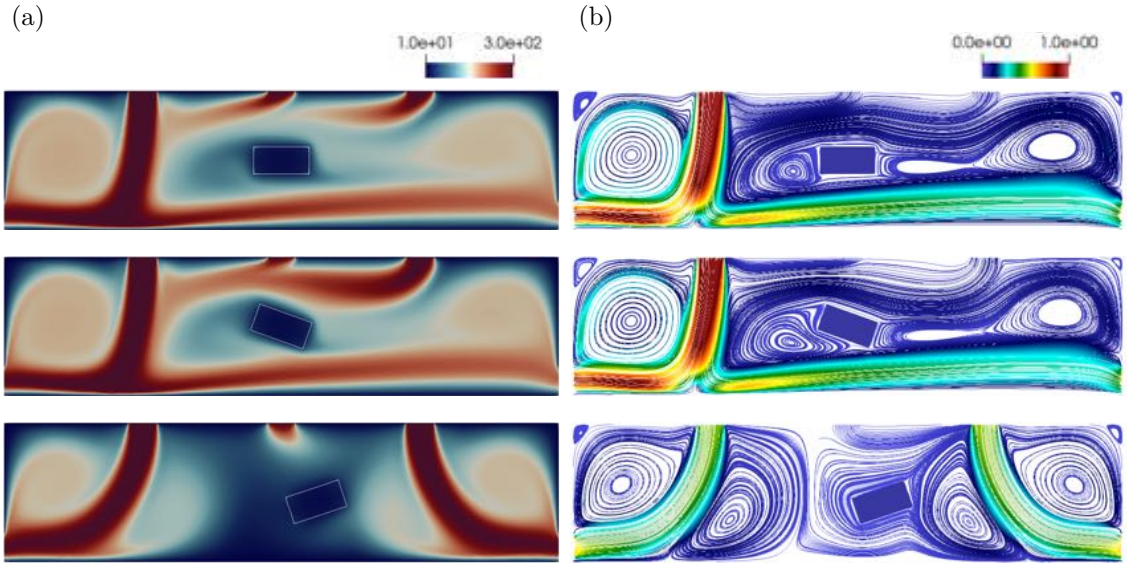

Figure 6: Control of heating homogeneity for the steel-like workpiece. (a) Optimal temperature distribution found by the DRL algorithm by controlling (from top to bottom) : the inflow velocities under constant workpiece angle and position, the inflow velocities and workpiece angle under constant position, and the inflow velocities, workpiece angle and position. (b) Corresponding streamlines colored by the magnitude of velocity.

the iteration to handle the position on the y axis and the angle, as it restarts an exploration phase around episode 85, which allows to reach even better rewards in the end (see also figure 8 for an illustration of temperature distributions at the final step randomly sampled over the course of optimization). Indeed, with a larger space to explore, some parameters may be optimized more easily than others, for example if they have a greater impact on the reward. In this case, this may be a good idea to optimize the last parameters individually, since the first reward chosen may not be the best fitted to their optimization. The reward obtained from the original simulation (centered piece, same speed at each inlet) has a mean value of -0.450, figure 7 indicates that PPO converged around a reward of -0.050 which yields an improvement of 89% of the temperature homogeneity.

4.3.2. Control of heating efficiency

As explained in 4.2, we introduce here a second reward ψ_{t,w_1,w_2} to assess the feasibility of controlling the efficiency of heating of the piece while keeping the gradient homogeneous. The PPO algorithm should consider a new solution space in which minima are decided by both the temperature and the gradient. This should lead to different solutions than the ones before, especially concerning the average of the temperature values at the probes.

Only the steel-like workpiece is considered here (as it allows for a better conduction and therefore larger temperature differences) under the third optimization scenario, *i.e.*, inflow velocities, workpiece angle and position as free parameters). We tested two sets of weight, $(w_1, w_2) = (1, 1)$ and $(10, 1)$, with the second one giving more priority to the averaged temperature component, and compare in figure 9 the obtained average temperature at the final time step, to those obtained under the three scenario presented earlier for pure homogeneous control (that corresponds to $(w_1, w_2) = (0, 1)$). For the simplest homogeneous control cases presented in figures 9(a) and (b), the temperature is constrained by the position of the workpiece, and the algorithm quickly converges. Adding in the workpiece position as free parameter increases the complexity, (the possibility to move the workpiece anywhere in the chamber yields much higher variance in the space of achievable temperature distributions). In return, the average temperature in figure 9(c) drops to a lower value value slightly above 26°, to be understood as an indirect consequence of the optimization of the homogeneity reward r_t. The scenario shown in figure 9(d) corresponds to the optimization of the compound reward ψ_t using all six control parameters $(w_1, w_2) = (1, 1)$, for which the temperature again drops, similar to the previous case. Finally, by using the reward $\psi_{t,10,1}$ with $(w_1, w_2) = (10, 1)$, the algorithm reaches a temperature of 26.5°. Consistently, the optimal temperature distributions for this two cases are somewhat similar to those obtained by

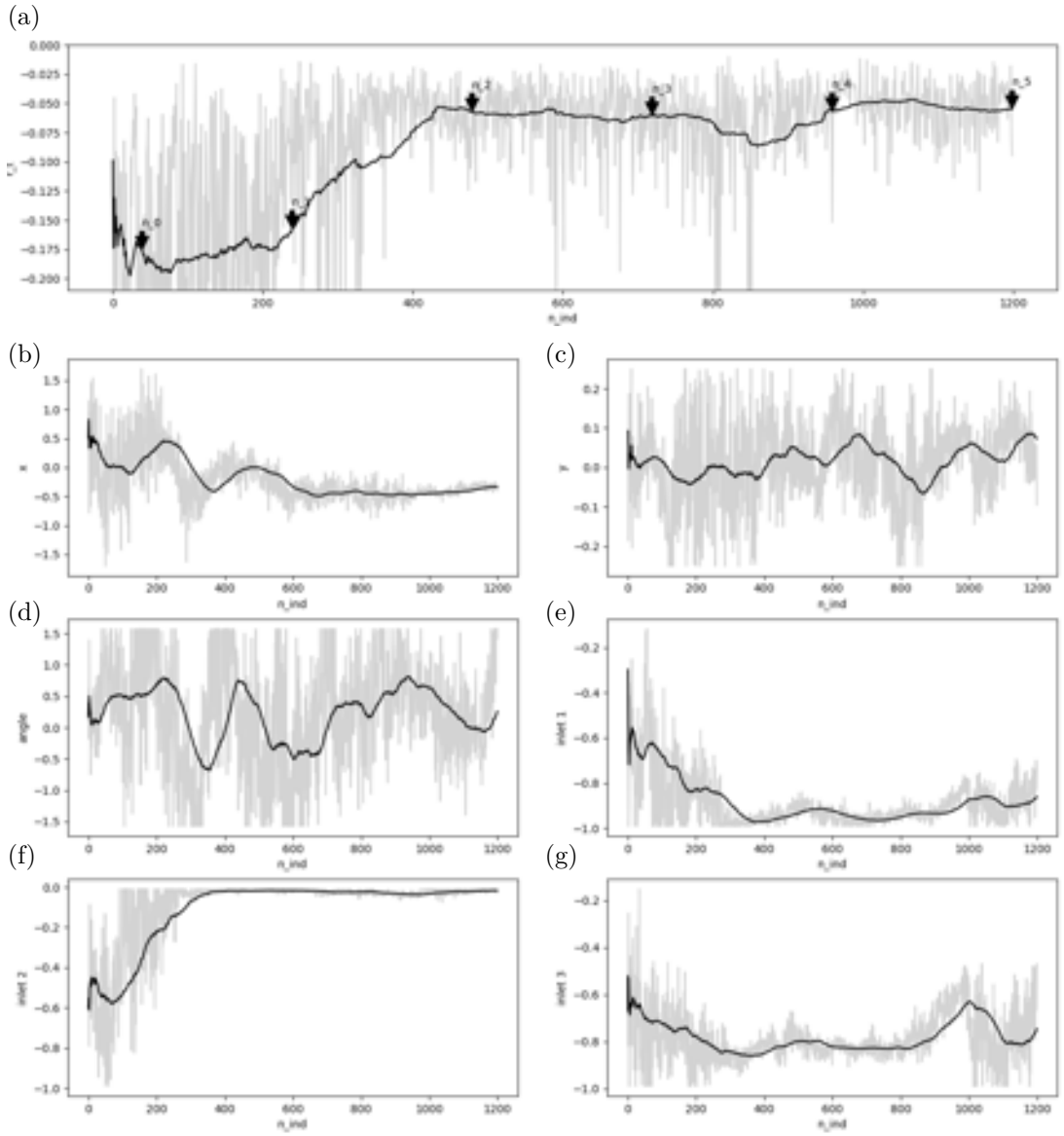

Figure 7: Control of heating homogeneity for the steel-like workpiece using the inflow velocities, workpiece angle and position as free parameters, corresponding to the scenario at the bottom of figure 6. Evolution per episode of the (a) reward, and (b-g) control parameters. Black curves are the moving averages.

controlling the heating homogeneity, but with a workpiece that is closer to the heat sources as the value of w_1 in the reward ψ_t increases, to give more priority to lowering the averaged temperature.

4.4. Discussion

Whether it be for the brick-like or the steel-like workpiece, the algorithm finds, in each scenario, a relevant local minimum that satisfies the conditions we imposed on it. One point worth noticing is the wide variety of solutions the algorithm comes up with, best illustrated by comparing figures 5, 6 and 10. This may be because the complexity of this case gives room for a lot of equivalent solutions, and the algorithm struggles to find the global minimum and always ends up in local minima. Numerical approximations can also create noise and give an information too imprecise to be processed by the PPO method, especially since the reward is calculated from point-wise data interpolated from the simulation (similar to experimental measurements). By running the same simulation 1000 times, we noticed a fluctuation in the reward, with estimated relative standard deviation by 4.1%.

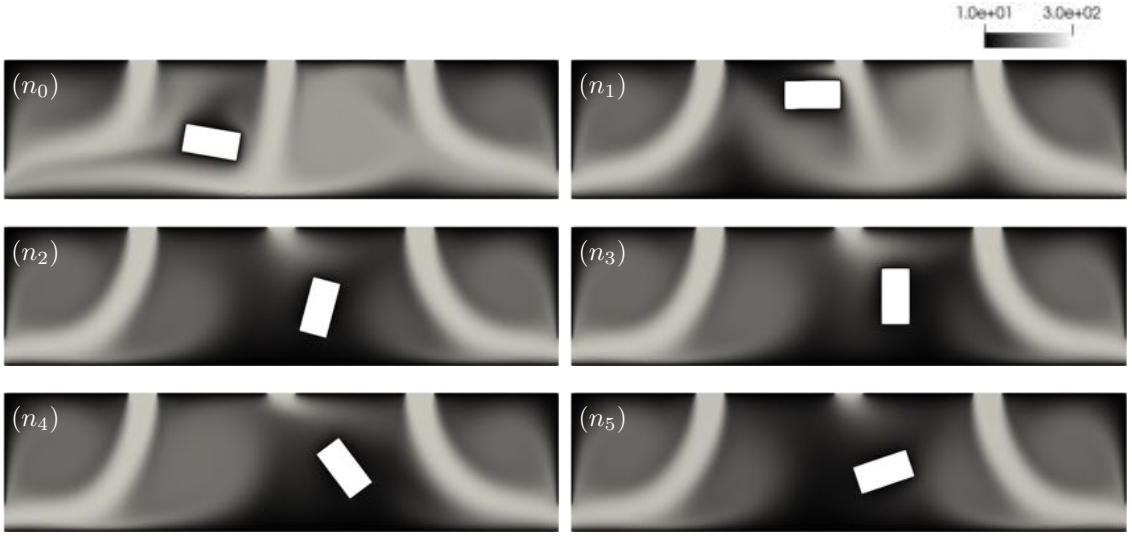

Figure 8: Control of heating homogeneity for the steel-like workpiece using the inflow velocities, workpiece angle and position as free parameters, corresponding to the scenario at the bottom of figure 6. Temperature distribution at the final time step for randomly sampled episodes marked by the arrows in figure 7(a).

This has been confirmed by running 8 times the same learning experiment (steel workpiece, homogeneous reward r_t, control of the inflow velocities, workpiece angle and position), for which the algorithm found local minima more or less sensitive to perturbations, and the relative standard deviation computed for the moving average over the 50 latest values is 9.9%. As interpreted before, the algorithm seems to struggle finding a global maximum of our reward function r_t, likely because there exists a sensitivity to noise and system uncertainty. There are a few possible reactions to that. Firstly, it could be useful to improve the balance between exploration and exploitation, using improved search distributions to effectively encourage the policy to explore more on the potential valuable actions, no matter whether they were preferred by the previous policies or not (for instance, using the recently introduced PBO algorithm [39], that uses three separate neural networks to learn the mean, variance and correlation parameters of a multivariate normal search distribution, while single-step PPO updates the mean and variance, the same for all variables, from a single neural network). Another possibility is to fine-tune the architecture of the neural network. As the number of parameters to control increases, the number of neurons that model the policy should increase too, but there is no guideline for that. Finally, the reward itself could be changed, along with the constraints on the parameters to control. These aspects have a great impact on the shape of the solution space, and therefore on the ease for the algorithm to find a global maximum. In real world applications, a lot of constraints can come into consideration when choosing the reward, or the parameters to operate on. Finding the right way to communicate our needs to the algorithm is a whole topic in itself. We leave the exploration of these reactions to future works.

5. Single-step control of a 3D serpentine heater

5.1. Case description

We propose in this test case to apply the same DRL-CFD framework to a three-dimensional case of industrial interest: the serpentine heater. It consists of a 3D simulation of two fluids, a liquid which is cold at first, and the hot gas that is distributed inside the chamber, and should come out colder due to the transfer of heat from the gas to the liquid (this resembles a heat exchanger but with a configuration closer to a gas burner).

The control objective here is to find the best flow distribution between an array of gas burners to heat a liquid in a pipe. We use Cartesian coordinate system with origin at the center of the chamber. The chamber is a simple parallelepiped with size H on x and z axes and h on the y axis. The pipe is made from the extrusion of a circle of radius R and a center at $(-H/2,\ 0,\ -H/4)$, with three 180 degrees bends around the y axis. It has two planes of symmetry Oxz and Oxy. The longer straight part of the pipe has a length of L, the second straight part has a length of l and

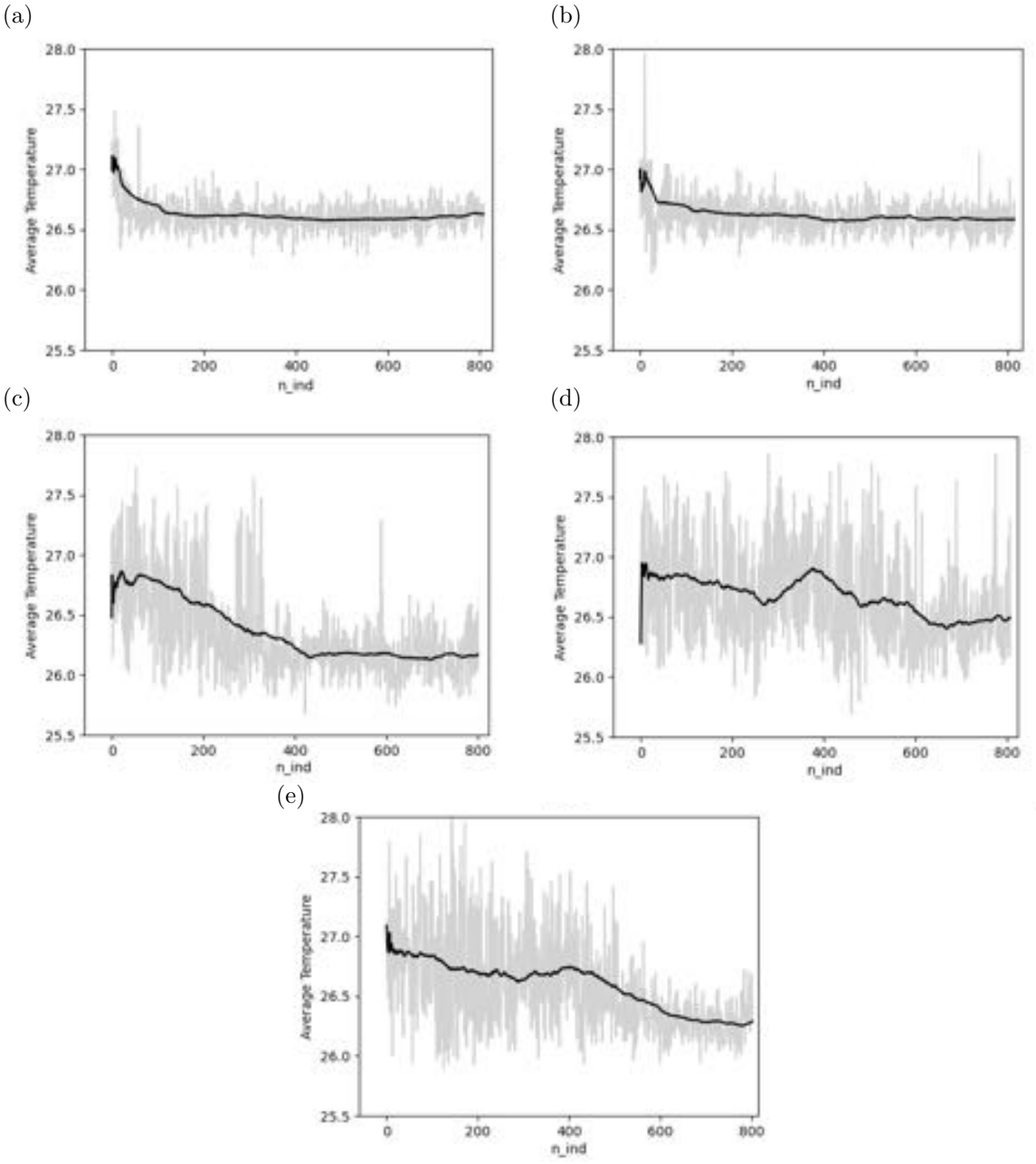

Figure 9: Evolution per episode of the final temperature averaged across all sensor positions in the steel-like workpiece, using (a-c) the homogeneity reward and (d-e) the compound homogeneity/efficiency reward with (d) $(w_1, w_2) = (1, 1)$ and (e) $(w_1, w_2) = (10, 1)$. The free parameters subjected to optimization are (a) the inflow velocities under constant workpiece angle and position, (b) the inflow velocities and workpiece angle under constant position, and (c-e) the inflow velocities, workpiece angle and position.

the rest of the pipe can be deducted from symmetry with the condition that all three bends have exactly the same radius. The hot gas comes out of 6 circular inlets of radius r, 4 on the y^- side and two on the y^+ side. They are disposed in an array aligned on the center of the pipe at each of its bends, see figure 12 (a) et (c). Since this case has no moving parts, remeshing is not needed, hence a unique mesh is used, as seen in figure 12 (d). It is composed of 35021 nodes and 197949 tetrahedral elements. The elements aspect ratio is shown in the bar chart 13.

The fluid in the pipe comes in through the lower hole with a parabolic speed profile corresponding to a flow rate D, and a temperature T_c. At $t = 0$ the temperature in the chamber is T_c, and hot gas comes through the 6 inlets at temperature T_h, with speed $V_i \in [0.1, 0.9]$. In the same manner

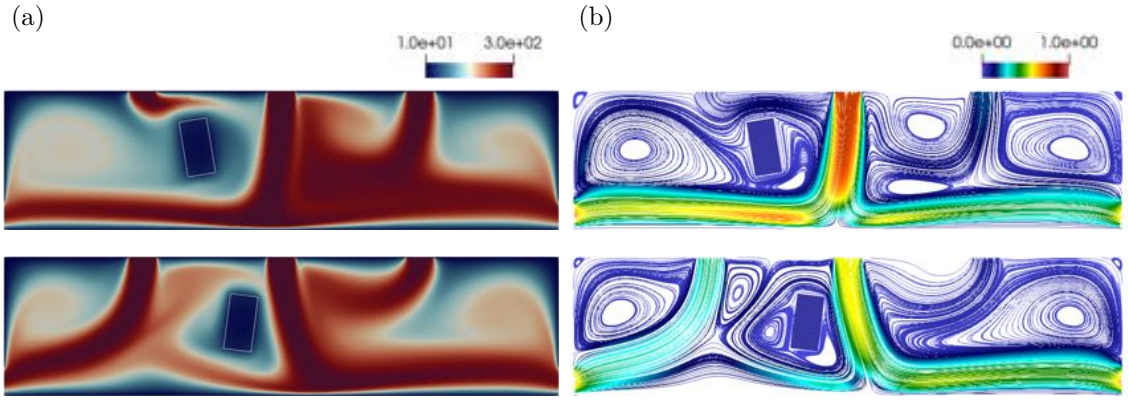

Figure 10: Control of efficiency for the steel-like workpiece. (a) Optimal temperature distribution found by the DRL algorithm by controlling the inflow velocities, workpiece angle and position, using reward ψ_t with weighs (w_1, w_2) set to (from top to bottom) $(1, 1)$ and $(10, 1)$. (b) Corresponding streamlines colored by the magnitude of velocity.

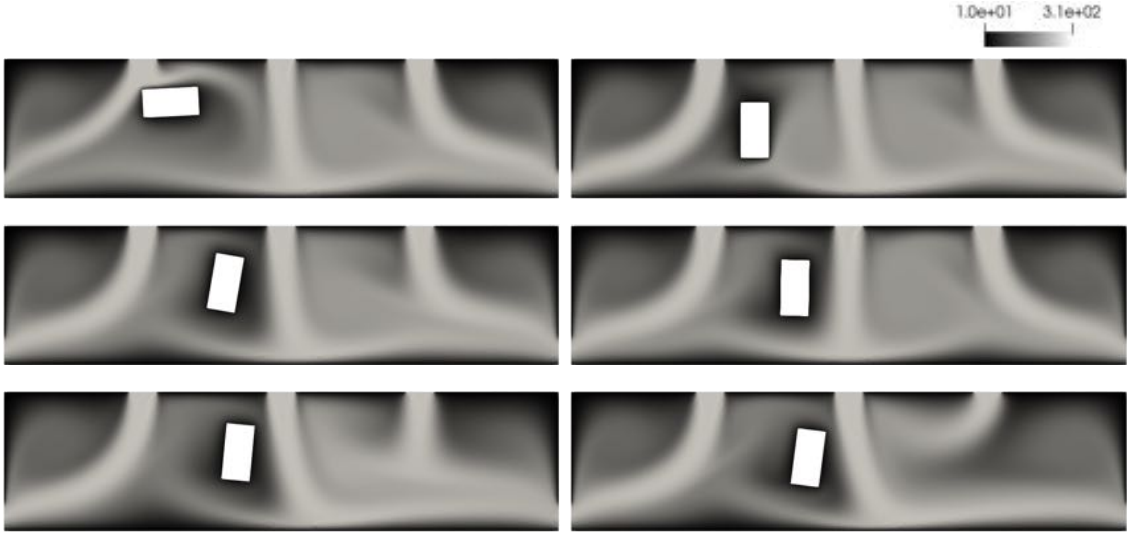

Figure 11: Temperature distribution at the final time step for randomly sampled episodes of heating efficiency control for the steel-like workpiece. The free parameters subjected to optimization are the inflow velocities, workpiece angle and position, corresponding to the scenario at the bottom of figure 10.

as for the previous case, we emulate a constant input by imposing

$$\sum_{i=1}^{6} V_i = 1\,. \tag{21}$$

The exhaust is positioned in the center of the top face of the chamber. It has a rectangular shape and is half the size of the top face. We consider in this case that there is no solid material separating the inside of the pipe from the gas. Only a no slip and isothermal condition is applied on the walls of the chamber (with the exception of the inlets and exhausts), with $T = T_w$. A no slip condition is also applied on the 2D interface representing the pipe. At the exhausts a zero-pressure condition is imposed : $p = \partial_x u = \partial_x T = 0$. No thermal condition is imposed at the interface, where heat exchange is implicitly driven by the difference in the individual material properties. Again, in the absence of buoyancy, the temperature evolves as a passive scalar for the Navier–Stokes equations. This allows us to solve the Navier–Stokes equations separately for the gas and the fluid in the pipe and then find the temperature field in the whole domain. Using the parameters for the liquid and the gas provided in Table 2, the Reynolds and Prandtl numbers in the hot gas domain are estimated to be $\mathrm{Re} \in [5, 30]$ and $\mathrm{Pr} = 20$, and $\mathrm{Re} \in [320]$ and $\mathrm{Pr} = 400$ in the liquid.

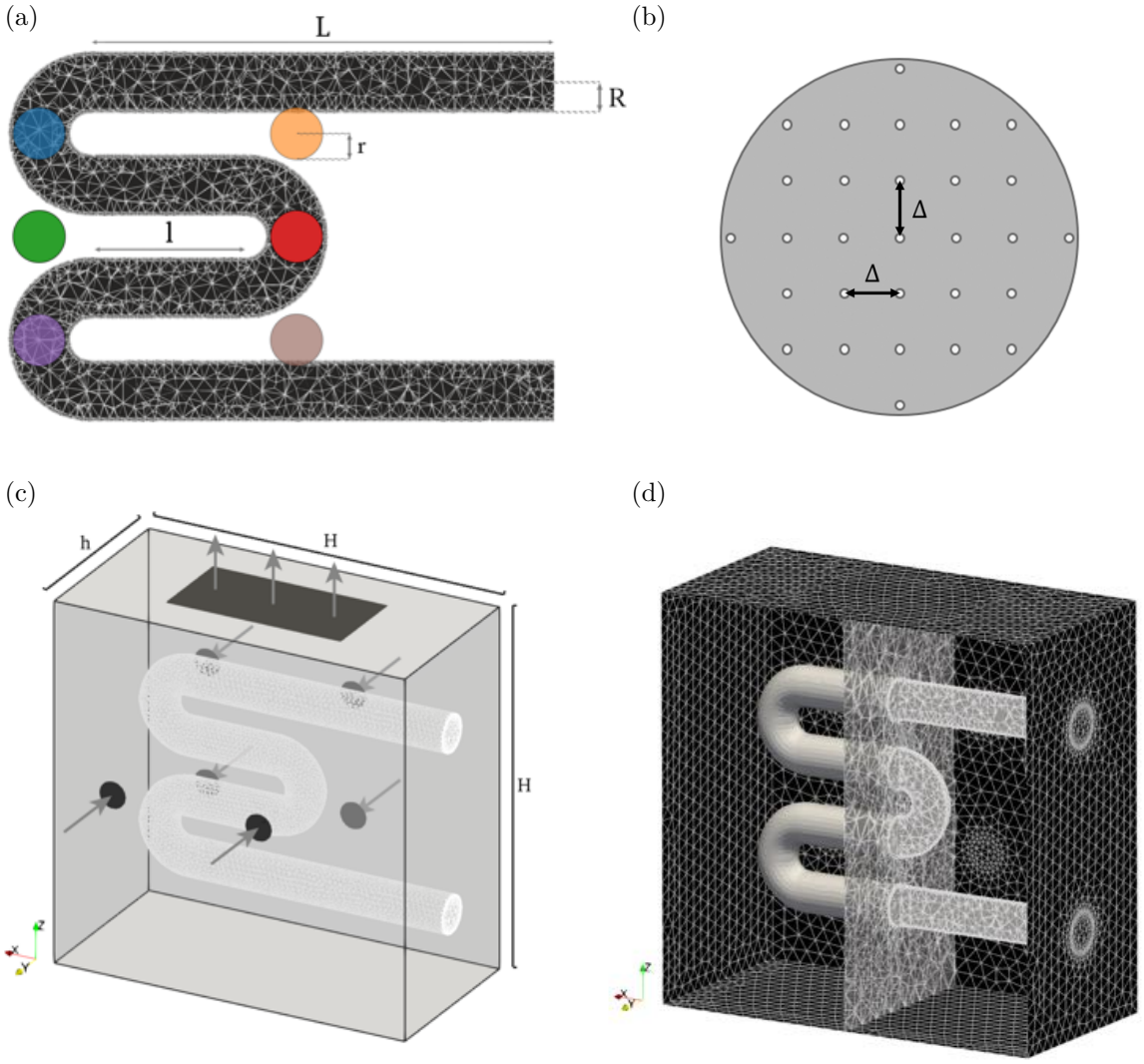

Figure 12: (a) Schematic of the pipe with positions of the gas inlets projected along the y axis. Fully opaque circles are inlets in front of the pipe, i.e. on the y^+ side of the chamber, and transparent circles are inlets on the other side. (b) Position of the sensors on the outlet of the pipe. (c) Schematic of the whole setup with arrows representing the direction of the gas flow. (d) Representation of the mesh used.

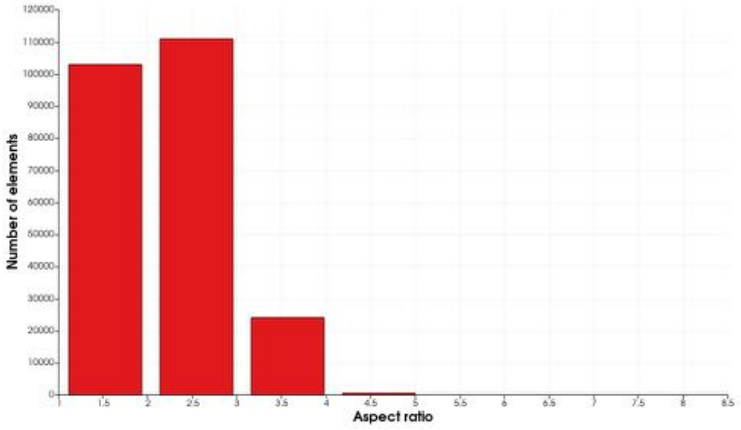

Figure 13: Bar chart of the aspect ratio of tetrahedral elements.

H	h	R	r	L	l	D	T_w	T_c	T_h	μ	ρ	λ	c_p	
4	2	0.2	0.15	3	1	0.05	150	20	300	0.01	1	0.5	1000	Gas
										0.1	100	1	4000	Pipe

Table 2: Geometric and numerical parameters used in the 3D serpentine heater setup. All values in SI units, with the exception of temperature given in Celsius.

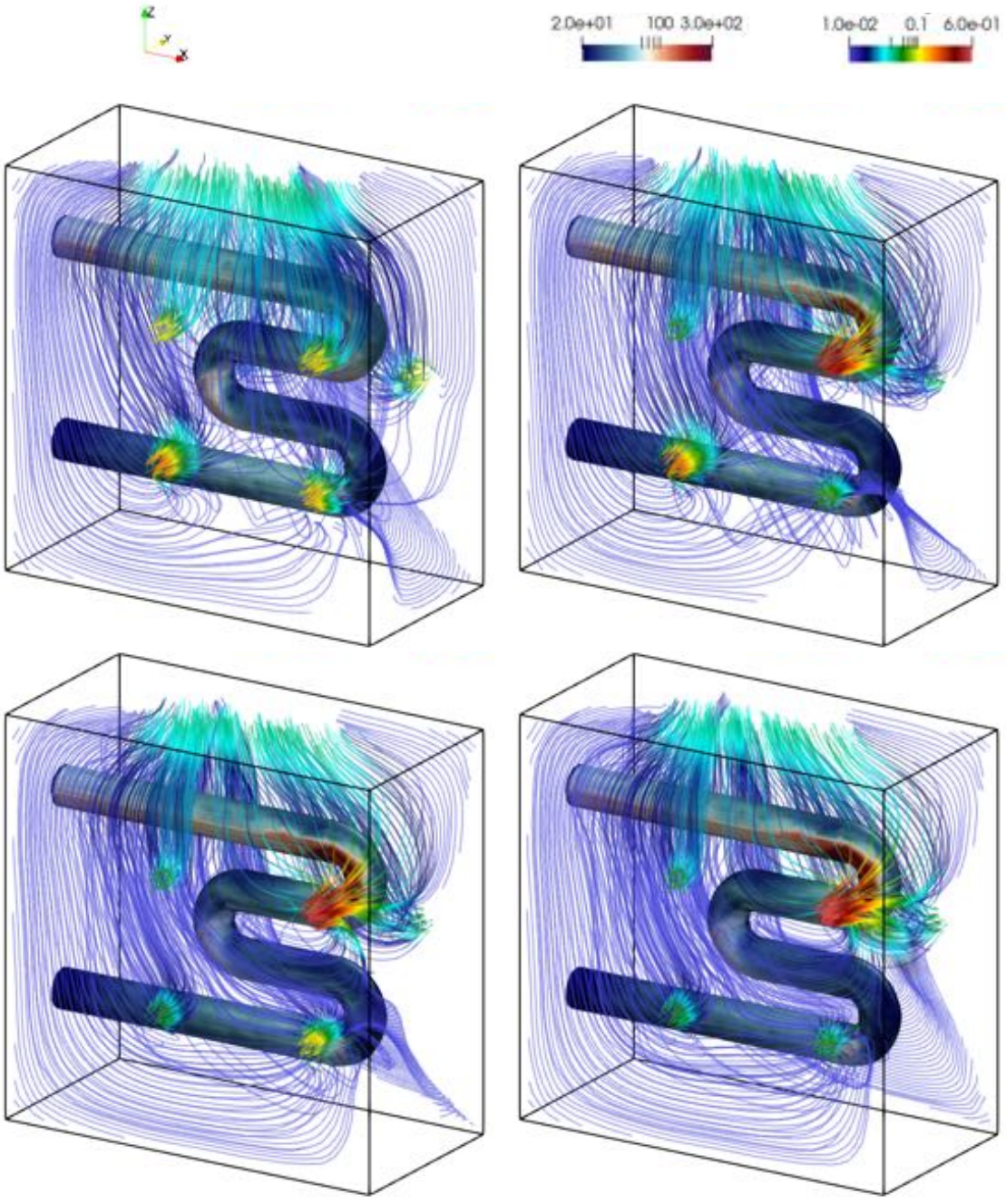

Figure 14: Representative temperature fields sampled on the course of optimization on the surface of the pipe together with streamlines colored by the magnitude of velocity, both in logarithmic scales.

5.2. Control strategy

The quantity being optimized is the distribution of the inflow between the 6 injectors $V_{i\in[\![1,6]\!]}$. In order to avoid asymmetry in the learning process we decided to control each of the injectors in

the same way, that is to assign each injector a value between 0.1 and 0.9, scaled a posteriori to recover a valid velocity satisfying (21). In order to compute the reward r_t we distribute 29 probes uniformly on the outlet of the pipe each at a distance $\Delta = 0.06$ of their closest neighbour, see figure 12 (b). These probes allow us to monitor the temperature out of the pipe, and the reward given the PPO algorithm (computed at the final simulation time) is defined as

$$r_t = \frac{1}{29} \sum_{i=1}^{29} T_i \,, \tag{22}$$

with T_i being the temperature at probe i.

In a second stage, we add a strong constraint to this reward. By looking at the highest temperature in the pipe across all time steps, we penalize the solutions that reach a temperature above a limit $T_l = 200$. We chose the limit by looking at the solutions found by the PPO algorithm with the first reward and their highest temperature inside the pipe. The new reward is thus

$$\rho_t = \begin{cases} r_t, & \text{if } \max_\Pi T < T_l \,, \\ 0, & \text{if } \max_\Pi T \geq T_l \,, \end{cases} \tag{23}$$

where Π is the inner pipe domain. This is meant to avoid concentrating too much energy in the same spot and thus protecting the materials used in the heating process.

5.3. Results

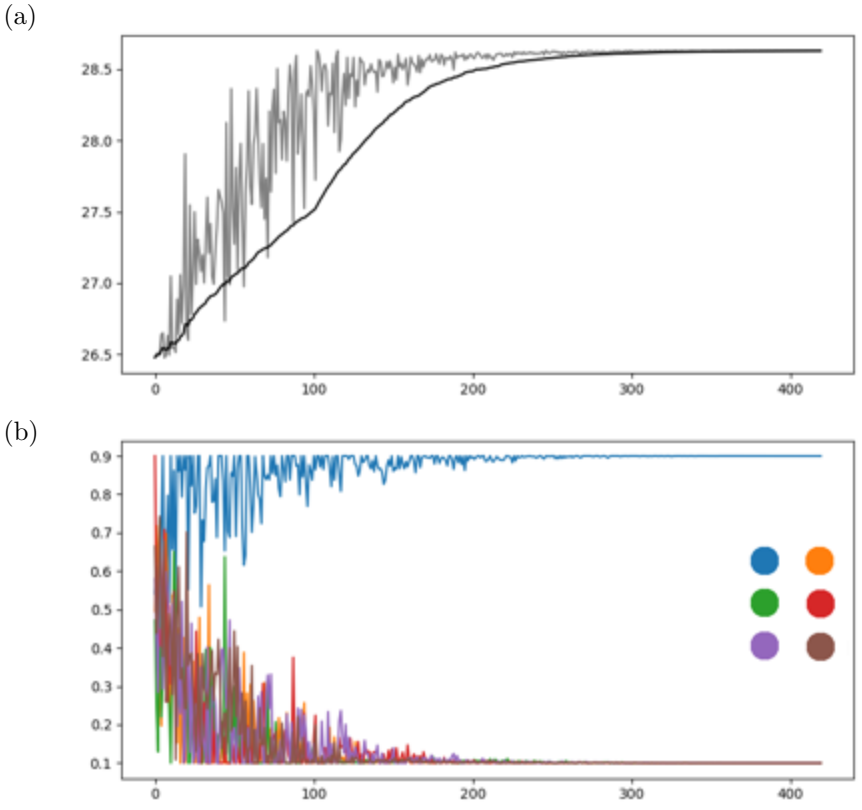

Figure 15: Evolution per episode of the (a) reward r_t, and (b) control velocity at each gas burner, and (c) maximum temperature $\max_\Pi T$ in the pipe. Black curves are the moving averages. The color code of the inlets is the same as in fig.12

For this case, 150 episodes have been run, each composed of 8 environment, each of which performs 1300 iterations with time step $\Delta t = 0.1$ to march in time the same initial condition (consisting of zero velocity and uniform temperature, except in the pipe domain). This represents

1200 simulations, each of which is performed on 8 cores and lasts 40mn, hence 800h of total CPU cost; see figure 14 for representative temperature and velocity distributions sampled of the course of optimization.

Figure 15 shows the evolution of the reward r_t and the values given to each inlet at each individual. The colors correspond to each inlet and are kept also in figure 16. The trend of these curves demonstrate a fast convergence of PBO in this context, with a transitional phase during which it tries to explore other possibilities. The solution found by the algorithm puts all the energy in a single, well positioned hot gas inlet (which is rather intuitive except that the location of said may not be), as the last 200 episodes all yield the same distribution among the inlets : all for the top left inlet and nothing for the rest. As can be seen in figure 12 (a), this particular inlet is near the end of the tube, right in front of large area thanks to the bend in the tube. This position allows for a large part of the heat to be transferred to the tube without too much energy being lost in perturbations. The results of the learning process under the constrained reward ρ_t are shown in figure 16, where the red line marks the limit temperature T_l that we try not to top (which the single-step PPO algorithm successfully achieves, as the temperature converges to a value slighly below). No moving average is given here since the hard constraint forces the reward to be zero when the limit temperature is topped. A trend is however visible since the reward still goes up in general and the frequency of constraint violation reduces along the training. Colored curves can be linked to the figures 12 (a) and 15. After convergence, the DRL still puts most of the weight on the top left inlet, with the rest of energy mainly given to the top right inlet, right next to the other, and closer to the output of the pipe. This shows the ability of the method to find optimal solutions under the constraint of safe operating conditions (here the pipe maximum temperature).

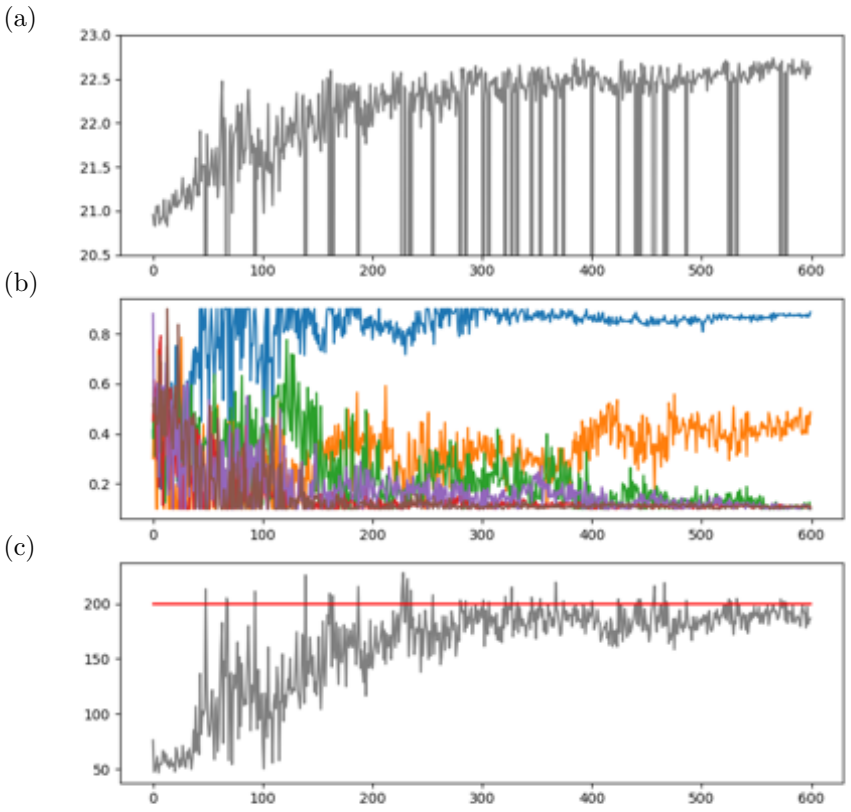

Figure 16: Evolution per episode of the (a) reward ρ_t, (b) control velocity at each gas burner, and (c) maximum temperature $\max_\Pi T$ in the pipe. Black curves are the moving averages. The color code of the inlets is the same as in fig.12

5.4. Discussion

The main interest behind this comparison is to assess how the algorithm performs when a hard constrained is applied directly to the reward. The first control experiment are somehow consistent with intuition, but the second one provides with more relevant and exploitable results in a real context where the ability to not exceed a certain temperature can be essential for the industrial process. On another note, for this case, it could be more useful for this case to investigate the feasibility to learn active control strategies (by adjusting dynamically the control velocities to appropriate sensing of flow variables). As the temperature of one spot approches the maximum authorized value, this would allow redistributing some of the power to the other spots to balance everything properly, and to converge rapidly to an optimal configuration. This would also allow for some evolution of the input flow of the pipe, and a reaction to it.

6. General discussion

Several points deserve further consideration to keep pushing forward the development of DRL in the context of such real-life applications. First, improving efficiency and convergence (by fine-tuning the hyper parameters and comparing with different DRL algorithms allowing for increased exploration, as PPO prevents by design large updates of the policy to avoid performance collapse). Second, designing improved reward construction strategies, as approximating the reward from point-wise temperature data has been shown to yield a certain sensitivity to system and numerical uncertainty, which in turn may trap the algorithm in local optimal. Third, enriching the description of the test cases using multi-physics modeling, *e.g.*, radiative heat transfer and thermo-mechanical coupling to encompass the solid deformations. Finally, investigating the case of active flow control in which the control parameters are dynamically adjusted from measurements in the workpiece and the furnace, and has often proved to be of great importance in industrial contexts.

7. Conclusion

Optimization of heating processes is achieved here training a fully connected network with the PBO deep reinforcement algorithm, in which it gets only one attempt per learning episode at finding the optimal. The numerical reward fed to the network is computed with a stabilized finite elements CFD environment solving the coupled Navier–Stokes and heat equations, using a combination of variational multi-scale modeling, immerse volume method, and multi-component anisotropic mesh adaptation.

The approach succeeds at improving the homogeneity of temperature (or a blend of homogeneity and absolute temperature) across the surface of two-dimensional cold workpieces under jet impingement heating. Several control scenarios have been considered (that can be considered different levels of design constraint), from the simple case where only the inflow velocity of the hot air injectors is optimized relative to a fixed workpiece position, up to the most complex case where the DRL agent also optimizes the position and insertion angle of the workpiece itself. The potential of the approach for industrial configurations of engineering interest is also showcased by optimizing the inflow of multiple gas burners in a three-dimensional serpentine heater.

The present results highlight the capabilities of coupling DRL and computational fluid dynamics in the context of industrial manufacturing processes in general, and heating processes inside industrial furnaces in particular.

References

[1] S. Dequan, G. Guili, G. Zhiwei, X. Peng, Application of Expert Fuzzy PID Method for Temperature Control of Heating Furnace, Procedia Engineering 29 (2012) 257–261. doi:10.1016/j.proeng.2011.12.703.
URL https://www.sciencedirect.com/science/article/pii/S1877705811065404

[2] M. Tóthová, M. Balara, J. Dubják, Simulation Model of Cascade Control of the Heating System, International Journal of Engineering Research in Africa 18 (2015) 20–27, conference Name: International Journal of Engineering Research in Africa Vol. 18 ISBN: 9783038356790 Publisher: Trans Tech Publications Ltd. doi:10.4028/www.scientific.net/JERA.18.20.
URL https://www.scientific.net/JERA.18.20

[3] N. Philip, S. Sahlan, A. P. I. D. N. Wahab, Application of Auto-Tuner Fuzzy PID Controller on Industrial Cascade Control, ELEKTRIKA- Journal of Electrical Engineering 19 (2020) 61–65.

[4] E. Rafajłowicz, W. Rafajłowicz, Image-Driven Decision Making with Application to Control Gas Burners, in: K. Saeed, W. Homenda, R. Chaki (Eds.), Computer Information Systems and Industrial Management, Lecture Notes in Computer Science, Springer International Publishing, Cham, 2017, pp. 436–446. doi:10.1007/978-3-319-59105-6_37.

[5] E. Rafajłowicz, H. Pawlak-Kruczek, W. Rafajłowicz, Statistical Classifier with Ordered Decisions as an Image Based Controller with Application to Gas Burners, in: L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. A. Zadeh, J. M. Zurada (Eds.), Artificial Intelligence and Soft Computing, Lecture Notes in Computer Science, Springer International Publishing, Cham, 2014, pp. 586–597. doi:10.1007/978-3-319-07173-2_50.

[6] N. Nguyen, D. Tran, Neural Network based Model Reference Control for Electric Heating Furnace with Input Saturation, 2019, pp. 111–114. doi:10.1109/ICA-SYMP.2019.8646052.

[7] D.-D. Wen, Decoupling control of electric heating furnace temperature based on DRNN neural network, 2010, pp. 261–264. doi:10.1109/ICECTECH.2010.5479934.

[8] V. R. Radhakrishnan, A. R. Mohamed, Neural networks for the identification and control of blast furnace hot metal quality, Journal of Process Control 10 (6) (2000) 509–524. doi:10.1016/S0959-1524(99)00052-9.
URL https://www.sciencedirect.com/science/article/pii/S0959152499000529

[9] L. Zhang, Y. Xue, Q. Xie, Z. Ren, Analysis and neural network prediction of combustion stability for industrial gases, Fuel 287 (2021) 119507. doi:10.1016/j.fuel.2020.119507.
URL https://www.sciencedirect.com/science/article/pii/S0016236120325035

[10] J. Matthes, P. Waibel, M. Vogelbacher, H. J. Gehrmann, H. B. Keller, A new camera-based method for measuring the flame stability of non-oscillating and oscillating combustions, Experimental Thermal and Fluid Science 105 (2019) 27–34. doi:10.1016/j.expthermflusci.2019.03.008.
URL https://www.sciencedirect.com/science/article/pii/S0894177718317503

[11] J. Matthes, P. Waibel, M. Vogelbacher, H. B. Keller, H. J. Gehrmann, D. Stapf, A camera-based flame stability controller for non-oscillating and forced-oscillating combustion, in: Proceedings of the 13TH EUROPEAN CONFERENCE ON INDUSTRIAL FURNACES AND BOILERS, 2022.

[12] J. Tao, Z. Yu, R. Zhang, F. Gao, RBF neural network modeling approach using PCA based LM–GA optimization for coke furnace system, Applied Soft Computing 111 (2021) 107691. doi:10.1016/j.asoc.2021.107691.
URL https://www.sciencedirect.com/science/article/pii/S1568494621006128

[13] D. Zhao, Z. Lu, H. Zhao, X. Y. Li, B. Wang, P. Liu, A review of active control approaches in stabilizing combustion systems in aerospace industry, Progress in Aerospace Sciences 97 (2018) 35–60. doi:10.1016/j.paerosci.2018.01.002.
URL https://www.sciencedirect.com/science/article/pii/S0376042117300878

[14] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, D. Hassabis, Mastering the game of Go without human knowledge, Nature 550 (7676) (2017) 354–359, number: 7676 Publisher: Nature Publishing Group. doi:

10.1038/nature24270.
URL https://www.nature.com/articles/nature24270

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, Playing Atari with Deep Reinforcement Learning, arXiv:1312.5602 [cs]ArXiv: 1312.5602 (Dec. 2013).
URL http://arxiv.org/abs/1312.5602

[16] Google just gave control over data center cooling to an AI.
URL https://www.technologyreview.com/2018/08/17/140987/google-just-gave-control-over-data-center-cooling-to-an-ai/

[17] A. Gupta, A. Khwaja, L. Guan, B. Venkatesh, Policy-Gradient and Actor-Critic Based State Representation Learning for Safe Driving of Autonomous Vehicles, Sensors 20 (2020) 5991. doi:10.3390/s20215991.

[18] F. Wang, L. P. Casalino, D. Khullar, Deep Learning in Medicine-Promise, Progress, and Challenges, JAMA internal medicine 179 (3) (2019) 293–294. doi:10.1001/jamainternmed.2018.7117.

[19] D. Skrobek, J. Krzywanski, M. Sosnowski, A. Kulakowska, A. Zylka, K. Grabowska, K. Ciesielska, W. Nowak, Prediction of sorption processes using the deep learning methods (long short-term memory), Energies 13 (24) (2020). doi:10.3390/en13246601.
URL https://www.mdpi.com/1996-1073/13/24/6601

[20] J. Krzywanski, K. Sztekler, M. Szubel, T. Siwek, W. Nowak, Mika, A comprehensive three-dimensional analysis of a large-scale multi-fuel cfb boiler burning coal and syngas. part 1. the cfd model of a large-scale multi-fuel cfb combustion, Entropy 22 (9) (2020). doi:10.3390/e22090964.
URL https://www.mdpi.com/1099-4300/22/9/964

[21] P. Garnier, J. Viquerat, J. Rabault, A. Larcher, A. Kuhnle, E. Hachem, A review on deep reinforcement learning for fluid mechanics, Computers & Fluids 225 (2021) 104973. doi:10.1016/j.compfluid.2021.104973.
URL https://www.sciencedirect.com/science/article/pii/S0045793021001407

[22] J. Viquerat, P. Meliga, A. Larcher, E. Hachem, A review on deep reinforcement learning for fluid mechanics : an update, Phys. Fluids 34 (2022) 111301.

[23] G. Beintema, A. Corbetta, L. Biferale, F. Toschi, Controlling Rayleigh-B\'enard convection via reinforcement learning, arXiv preprint arXiv:2003.14358 (2020).

[24] E. Hachem, H. Ghraieb, J. Viquerat, A. Larcher, P. Meliga, Deep reinforcement learning for the control of conjugate heat transfer, Journal of Computational Physics 436 (2021) 110317. doi:10.1016/j.jcp.2021.110317.
URL https://www.sciencedirect.com/science/article/pii/S0021999121002126

[25] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal Policy Optimization Algorithms, arXiv:1707.06347 [cs]ArXiv: 1707.06347 (Aug. 2017).
URL http://arxiv.org/abs/1707.06347

[26] J. Viquerat, J. Rabault, A. Kuhnle, H. Ghraieb, A. Larcher, E. Hachem, Direct shape optimization through deep reinforcement learning, arXiv:1908.09885 [cs]ArXiv: 1908.09885 (Dec. 2020).
URL http://arxiv.org/abs/1908.09885

[27] H. Ghraieb, J. Viquerat, A. Larcher, P. Meliga, E. Hachem, Single-step deep reinforcement learning for open-loop control of laminar and turbulent flows, Phys. Rev. Fluids 6 (2021) 053902.

[28] E. Hachem, H. Digonnet, E. Massoni, T. Coupez, Immersed volume method for solving natural convection, conduction and radiation of a hat-shaped disk inside a 3d enclosure, International Journal of numerical methods for heat & fluid flow (2012).

[29] E. Hachem, B. Rivaux, T. Kloczko, H. Digonnet, T. Coupez, Stabilized finite element method for incompressible flows with high reynolds number, Journal of Computational Physics 229 (23) (2010) 8643–8665. doi:https://doi.org/10.1016/j.jcp.2010.07.030.
URL https://www.sciencedirect.com/science/article/pii/S0021999110004237

[30] E. Hachem, T. Kloczko, H. Digonnet, T. Coupez, Stabilized finite element solution to handle complex heat and fluid flows in industrial furnaces using the immersed volume method, International Journal for Numerical Methods in Fluids 68 (1) (2012) 99–121. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/fld.2498, doi:https://doi.org/10.1002/fld.2498.
URL https://onlinelibrary.wiley.com/doi/abs/10.1002/fld.2498

[31] I. Goodfellow, Y. Bengio, A. Courville, The Deep Learning Book, MIT Press, 2017.

[32] R. S. Sutton, A. G. Barto, Reinforcement learning: An introduction, MIT press, 2018.

[33] D. Rumelhart, G. Hinton, R. Williams, Learning representations by back-propagating errors, Nature 323 (1986) 533–536.

[34] A. Kakade, A natural policy gradient, Adv. Neural Inf. Process Syst. 14 (2001) 1531–1538.

[35] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, P. Abbeel, Trust Region Policy Optimization, arXiv e-prints (Feb. 2015). arXiv:1502.05477.

[36] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal Policy Optimization Algorithms, arXiv e-prints (Jul. 2017). arXiv:1707.06347.

[37] T. Coupez, L. Silva, E. Hachem, Implicit Boundary and Adaptive Anisotropic Meshing, Springer International Publishing, Cham, 2015, pp. 1–18. doi:10.1007/978-3-319-06053-8_1.
URL https://doi.org/10.1007/978-3-319-06053-8_1

[38] T. W. Athan, P. Y. Papalambros, A note on weighted criteria methods for compromise solutions in multi-objective optimization, Eng. Optim. 27 (1996) 155–176.

[39] J. Viquerat, R. Duvigneau, P. Meliga, A. Kuhnle, E. Hachem, Policy-based optimization: single-step policy gradient method seen as an evolution strategy, Neural Comput. Appl. (2022).