



HAL
open science

On the coupling of deep reinforcement learning and computational fluid dynamics

Hassan Ghraieb

► **To cite this version:**

Hassan Ghraieb. On the coupling of deep reinforcement learning and computational fluid dynamics. Fluid mechanics [physics.class-ph]. Université Paris sciences et lettres, 2022. English. NNT : 2022UPSLM037 . tel-04043187

HAL Id: tel-04043187

<https://pastel.hal.science/tel-04043187v1>

Submitted on 23 Mar 2023

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

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

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à MINES Paris

**Sur le Couplage de l'Apprentissage par Renforcement
Profond et de la Mécanique des Fluides Numérique**

**On the Coupling of Deep Reinforcement Learning and
Computational Fluid Dynamics**

Soutenue par

Hassan Ghraieb

Le 24 Juin 2022

École doctorale n°364

**Sciences Fondamentales et
Appliquées**

Spécialité

**Mathématiques Numériques,
Calcul Intensif et Données**

Composition du jury :

Ramon CODINA Professor, Universitat Politècnica de Catalunya, Spain	<i>Rapporteur, Président du jury</i>
Anil Anthony BHARATH Professor, Imperial College London, England	<i>Rapporteur</i>
Nissrine AKKARI Doctor, Safran Tech	<i>Examineur</i>
Anca BELME Doctor, Sorbonne Université, France	<i>Examineur</i>
Jonathan VIQUERAT Doctor, Université PSL, Mines Paris-Tech, France	<i>Examineur, Maître de thèse</i>
Philippe MELIGA Doctor, Université PSL, Mines Paris-Tech, France	<i>Examineur, Co-Directeur de thèse</i>
Elie HACHEM Professor, Université PSL, Mines ParisTech, France	<i>Examineur, Directeur de thèse</i>

Contents

Contents	i
1 Introduction	1
1.1 Flow control	6
1.2 Systematical approaches for flow control	7
1.3 The emergence of Deep Reinforcement learning	9
1.4 Objectives of the thesis	11
1.5 Layout of the thesis	13
1.6 Achievements	13
2 Methodologie - Deep Reinforcement Learning	15
2.1 Introduction	17
2.2 Reinforcement learning framework	17
2.2.1 Value-based methods	20
2.2.2 Policy-based methods	23
2.3 Deep Reinforcement Learning	24
2.3.1 Deep Q-Network (DQN)	25
2.3.2 Vanilla deep policy gradient	26
2.3.3 Advantage actor-critic (A2C)	26
2.3.4 Deep Deterministic Policy gradient DDPG	27
2.3.5 Trust-region and proximal policy optimization (TRPO and PPO)	27
2.4 Single-step Deep Reinforcement Learning	29
2.4.1 PPO-1	29
2.4.2 Policy-Based Optimization	30
2.5 Conclusion	34
3 Methodology - Computational Fluid Dynamics	35
3.1 Introduction	37
3.2 State of the art	37
3.3 Flow Finite Element Solver	38

3.3.1	The incompressible Navier–Stokes equations	38
3.3.2	The Variational Multiscale approach (VMS)	39
3.4	Turbulence modeling	43
3.4.1	Reynolds Averaged Navier–Stokes (RANS)	44
3.4.2	The Spalart-Allmaras SA turbulence model	44
3.5	Immersed volume method	47
3.5.1	The Level-Set Method	47
3.5.2	Anisotropic mesh adaptation	48
3.5.3	Mixing Laws	49
3.6	Conclusion	50
4	Single-step DRL for open-loop control of laminar and turbulent flows	53
4.1	Introduction	56
4.2	Methodology	58
4.2.1	Deep reinforcement learning	58
4.2.2	Proximal policy optimization	58
4.2.3	Single-step PPO	60
4.2.4	Computational fluid dynamics environment	61
4.2.5	Numerical implementation	63
4.3	Application to flow optimization	65
4.3.1	Flow past a NACA 0012 airfoil	65
4.3.2	Flow past an arrangement of two side-by-side circular cylinders	67
4.4	Application to open-loop flow control	72
4.4.1	Optimal cylinder drag reduction using a smaller control cylinder	72
4.4.2	Optimal drag reduction of a triangular bluff-body using rotating cylinders	83
4.5	Discussion	90
4.5.1	Adjoint methods	90
4.5.2	Evolution strategies	92
4.6	Conclusion	93
5	DRL for the control of conjugate heat transfer	97
5.1	Introduction	101
5.2	Computational fluid dynamics	103
5.2.1	The immersed volume method	104
5.2.2	Variational multi-scale approach (VMS)	106
5.3	Deep reinforcement learning and proximal policy optimization	108
5.3.1	Neural networks	108
5.3.2	Deep reinforcement learning	108
5.3.3	From policy methods to Proximal policy optimization	110

5.3.4	Single-step PPO	113
5.3.5	Numerical implementation	115
5.4	Control of natural convection in 2-D closed cavity	115
5.4.1	Case description	115
5.4.2	Control	118
5.4.3	Results	119
5.5	Control of forced convection in 2-D open cavity	122
5.5.1	Case description	122
5.5.2	Control	123
5.5.3	Results	124
5.5.4	Discussion	133
5.6	Extension to 3-D forced convection	135
5.6.1	Case description	135
5.6.2	Control strategy	136
5.6.3	Results	138
5.7	Conclusion	140
6	Single-step DRL for two- and three-dimensional optimal shape design	143
6.1	Introduction	146
6.2	Methodology	149
6.2.1	Deep reinforcement learning	149
6.2.2	Single-step deep reinforcement learning	149
6.2.3	Policy based optimization	150
6.2.4	Computational fluid dynamics environment	151
6.2.5	Numerical implementation	153
6.3	Validation	156
6.3.1	Test case description	156
6.3.2	Results	159
6.3.3	Discussion	162
6.4	Application to optimal aerodynamic design	164
6.4.1	Test case description	164
6.4.2	Laminar regime at $Re = 250$	170
6.4.3	Turbulent (transitional) regime at $Re = 5000$	171
6.5	Extension to 3-D shape optimization.	172
6.6	Conclusion	173
7	Multi-step Deep Reinforcement Learning	181
7.1	Introduction	183
7.2	Conjugate heat transfer control for homogeneous cooling	184
7.2.1	Uncontrolled flow	187

7.2.2	Open-loop control using single-step DRL	188
7.2.3	Closed-loop control using multistep-step DRL	190
7.3	Drag reduction in the flow past a circular cylinder	194
7.3.1	Uncontrolled flow	195
7.3.2	Open-loop control using single-step DRL	195
7.3.3	Closed-loop control using multi-step DRL	198
7.4	Conclusion	202
8	Conclusion	205
8.1	Summary	206
8.2	Perspectives	208

Chapter 1

Introduction

Contents

1.1	Flow control	6
1.2	Systematical approaches for flow control	7
1.3	The emergence of Deep Reinforcement learning	9
1.4	Objectives of the thesis	11
1.5	Layout of the thesis	13
1.6	Achievements	13

Le contrôle de l'écoulement est la capacité d'adapter un écoulement à un état différent, plus souhaité, servant un avantage technique idéalement important, comme la réduction de la traînée, l'augmentation de la portance, l'amélioration du mélange ou la réduction du bruit. Il s'agit à la fois d'un domaine d'une importance sociétale et économique considérable et d'une technologie de pointe pour les avancées en mécanique des fluides, car la disponibilité de stratégies de contrôle pertinentes pour les ingénieurs de conception et de production représente un avantage concurrentiel décisif, qui peut avoir un impact considérable sur d'autres applications, voire sur plusieurs disciplines scientifiques. Le contrôle de l'écoulement est souvent évalué dans le contexte de la réduction de la traînée des corps flottants, pour laquelle de nombreuses approches ont été mises en œuvre [1], qui utilisent soit des appendices passifs, soit des dispositifs actifs [2]. La commande d'actionnement peut être prédéterminée (contrôle en boucle ouverte) ou reposer sur une détection appropriée de l'état du flux (contrôle en boucle fermée). Dans le contexte des applications industrielles réelles, les stratégies de contrôle ont longtemps été (et le sont encore souvent) déterminées par essais et erreurs à l'aide de campagnes expérimentales ou numériques longues et coûteuses. Cela a motivé le développement de formalismes mathématiques rigoureux capables de fournir une conception optimale à des coûts réduits, qui sont généralement classés comme basés sur le gradient et sans gradient, selon que la méthode requiert des informations sur le gradient, en plus des évaluations de fonction, pour déterminer les directions de recherche adéquates pour de meilleures conceptions pendant les itérations d'optimisation. Les méthodes basées sur le gradient et celles sans gradient peuvent bénéficier de modèles de substitution peu coûteux à évaluer pour évaluer les fonctions d'objectif et de contrainte coûteuses en calcul sans recourir systématiquement aux simulations numériques. Plusieurs approches existent pour construire de tels modèles de substitution, comme les surfaces de réponse, les méthodes bayésiennes et les réseaux de neurones artificiels.

Malgré les efforts considérables déployés dans la théorie du contrôle de l'écoulement, les approches ci-dessus se heurtent à des difficultés considérables lorsqu'elles tentent de concevoir des stratégies complexes, et la plupart des études se contentent traditionnellement d'une entrée harmonique ou constante simpliste (et probablement sous-optimale) [3]. Par conséquent, il reste nécessaire de développer des méthodes de contrôle efficaces capables d'effectuer un contrôle complexe et de tirer pleinement parti des possibilités d'actionnement. Une option prometteuse pour ce faire consiste à s'appuyer sur les méthodes d'apprentissage automatique, et plus particulièrement sur les réseaux neuronaux profonds (DNN), une famille d'outils paramétriques polyvalents capables d'apprendre à extraire hiérarchiquement des caractéristiques informatives à partir de données. Les réseaux de neurones ont atteint des niveaux de performance étonnants dans divers domaines, comme la classification d'images [4], la reconnaissance vocale [5] ou les tâches génératives [6]. L'accès généralisé aux ressources

de calcul des GPU grâce à du matériel moins cher ou à l'informatique en nuage a permis de réaliser des progrès considérables dans le domaine des techniques de prise de décision, grâce au couplage des DNN avec des algorithmes d'apprentissage par renforcement (appelé apprentissage par renforcement profond, ou DRL). Ces avancées ont permis de lever plusieurs obstacles majeurs qui freinaient l'apprentissage par renforcement classique, en permettant l'utilisation d'espaces d'états de haute dimension et en exploitant les capacités d'extraction de caractéristiques des DNN. En retour, une efficacité sans précédent a été atteinte dans de nombreux domaines tels que la robotique [7], le traitement du langage [8], les jeux [9], bien que le DRL se soit également révélé utile dans des applications industrielles, par exemple les voitures autonomes [10] ou le refroidissement des centres de données [11]. Il existe également un grand potentiel d'application de DRL à la mécanique des fluides, pour laquelle les efforts sont en cours mais encore à un stade précoce, avec une poignée d'études pionnières donnant un aperçu des améliorations de performance à fournir dans le domaine. L'engagement soutenu de la communauté de l'apprentissage automatique a permis d'élargir le champ d'application, depuis les réductions de faible dimension, peu coûteuses en calcul, de la dynamique des fluides sous-jacente [12–14] jusqu'aux systèmes complexes de Navier-Stokes, en passant par les montages expérimentaux [15]. Néanmoins, seule une gamme limitée d'applications a été considérée jusqu'à présent (qui comprend, par ordre chronologique, les nageurs, la réduction de la traînée et l'optimisation de la forme) tandis que d'autres ont reçu une attention marginale (microfluidique [16], transfert de chaleur par convection libre [17]) ou ont été essentiellement laissées de côté (écoulements multiphasiques ou interactions fluide-structure).

Le premier objectif de cette thèse est d'approfondir les capacités du Deep Reinforcement Learning (DRL) pour le contrôle en boucle fermée dans un contexte de CFD. Les études mentionnées ci-dessus montrent le fort potentiel de cette approche sur des problèmes classiques, notamment les problèmes de réduction de la traînée à faible nombre de Reynolds. Néanmoins, des efforts considérables sont nécessaires pour (i) consolider les connaissances acquises, (ii) réduire l'écart entre le DRL et les méthodes numériques avancées pour la dynamique des fluides numérique (CFD) multi-échelle et multi-physique, et (iii) répondre aux attentes inévitablement soulevées par la nécessité de s'attaquer aux problèmes à l'échelle industrielle. Deux ingrédients clés sont nécessaires pour atteindre cet objectif : un algorithme de DRL efficace pour l'apprentissage, et un solveur CFD évolutif pour calculer avec précision les solutions numériques coûteuses en temps de calcul à partir desquelles on peut extraire la récompense fournie à l'agent DRL. Nous cherchons ici à coupler Proximal policy optimization (PPO [18]) et CimLib-CFD dans un contexte de calcul haute performance. Le premier est un nouveau venu qui s'est rapidement imposé comme l'un des algorithmes de DRL les plus utilisés pour les problèmes de contrôle de flux,

en raison de l'efficacité de ses données, de sa simplicité d'implémentation et de ses performances fiables. Le second est une bibliothèque d'éléments finis C++ massivement parallèle et multiphysique développée par l'équipe CFL au centre de recherche CEMEF. L'espoir est que cela permettra de s'attaquer à des problèmes en suspens dans divers domaines, par exemple, l'optimisation des coefficients aérodynamiques stationnaires et instationnaires, la charge des structures, le contrôle des processus thermiques industriels, le mélange optimal des écoulements multiphasiques, et bien d'autres.

Un deuxième objectif découle de l'observation que, bien que les réseaux de neurones soient utilisés depuis longtemps dans les problèmes d'optimisation (où la politique à apprendre est indépendante de l'état, et où une simple entrée de contrôle harmonique ou constante est effectivement pertinente), ils sont le plus souvent exploités comme des substituts entraînés pour la fonction objective réelle [19], mais ont longtemps été laissés en dehors du processus d'optimisation central, à l'exception d'une poignée d'études [20, 21]. Une prémisse de cette thèse est que les algorithmes DRL peuvent également être utilisés comme optimiseurs efficaces en boîte noire pour de telles situations également pertinentes pour les problèmes de contrôle de flux en boucle ouverte. Pour ce faire, il faut modifier un algorithme classique pour que le réseau neuronal apprenne une correspondance simple entre un état d'entrée constant et une action optimale en n'interagissant qu'une seule fois par épisode avec son environnement (d'où les épisodes à une seule étape, et par extension, le DRL à une seule étape, contrairement au DRL classique dans lequel la récompense évalue, non pas une seule, mais une série d'actions prises au cours du même épisode, d'où le surnom de multi-étapes). Nous mesurons ici la capacité du DRL à une seule étape à optimiser de manière fiable les systèmes de flux complexes, cette approche a été considérée comme ayant un fort potentiel, mais reste à être analysée en profondeur (nous n'avons pas connaissance d'autres travaux appliquant le DRL de cette manière, bien qu'un concept similaire de "DRL sans état" ait été ébauché dans [22] à des fins de validation, mais n'a pas été poursuivi). Nous verrons qu'essentiellement, les méthodes DRL à une étape héritent des algorithmes de gradient de politique profonde dans le sens où les paramètres pertinents de la fonction de densité de probabilité sont obtenus à partir de réseaux neuronaux entraînés en utilisant une perte de type gradient de politique. Pourtant, elles sont également héritières des stratégies évolutionnaires (ES), car leurs étapes successives suivent une nomenclature génération/individu, exploitant les informations des générations précédentes afin de mettre à jour les paramètres d'une fonction de densité de probabilité.

Cette thèse est divisée en 8 chapitres, y compris la présente introduction. Le chapitre 2 passe en revue les principaux algorithmes de DRL qui ont été utilisés dans le contexte de la mécanique des fluides, et introduit le concept de DRL à étape unique développé au cours de ce travail. Le chapitre 3 décrit les méthodes CFD

utilisées pour calculer la récompense numérique fournie à l'agent de DRL. Cela comprend les formulations d'éléments finis stabilisés par VMS utilisées pour résoudre les équations de Navier-Stokes et de Navier-Stokes moyennées de Reynolds, et la méthode des volumes immergés couplée à l'adaptation du maillage de la couche limite anisotrope, utilisée pour résoudre avec précision un ensemble unique d'équations relatives aux phases solide et fluide sur un domaine de calcul unique. Le chapitre 4 évalue la pertinence de l'approche de DRL à une seule étape en tant qu'optimiseur en boîte noire pour les problèmes de contrôle en boucle ouverte régis par les équations de Navier–Stokes. Plusieurs cas laminaires et turbulents sont utilisés comme banc d'essai pour la mise en œuvre et la validation de l'approche, avec un accent particulier sur la réduction de la traînée. Le chapitre 5 se concentre sur le contrôle des systèmes de transfert de chaleur conjugués régis par les équations couplées de Navier–Stokes et de la chaleur, et présente la toute première tentative d'utilisation de la DRL pour contrôler la convection forcée en deux et trois dimensions. Le chapitre 6 étend le champ d'application de DRL à une étape à l'optimisation de forme par l'optimisation directe des paramètres de forme indépendants de l'état. Le chapitre 7 évalue la capacité de DRL multi-étapes pour la réduction active de la traînée et le contrôle thermique. Enfin, le chapitre conclusion fournit la conclusion et discute l'extension possible du présent travail à des problèmes de physique plus complexe.

1.1 Flow control

Flow control is the ability to tailor a flow into a different, more desired state serving an ideally large engineering benefit, like drag reduction, lift increase, mixing enhancement or noise reduction. It is at once a field of tremendous societal and economical importance, and a pacing technology for advances in fluid mechanics, as the availability of relevant control strategies for design and production engineers represents a decisive competitive advantage, that can greatly impact other applications or even several scientific disciplines. For instance, reducing drag by just a few percent while maintaining lift can help reducing fossil fuel consumption and CO₂ emission while saving several billion dollars annually in applications such as ocean shipping (figure 1.1) or airline traffic [23], while thermal control using heat/cool exchangers is key to regulate process temperatures, which in turn ensures that machinery, chemicals, water, gas, and other substances remain within safe operating conditions, or to regulate aircraft cabin temperature and humidity under substantial variations of the ambient conditions, which is mandatory to provide a high level of comfort for passengers.

Flow control is often benchmarked in the context of bluff body drag reduction, for which numerous approaches have been implemented [1], that use either passive appendices or active devices [2]. Passive appendices are steady and require no energy by definition, typical examples being the rods, end/splitter plates or flexible tails placed on the rear surface or downstream of a cylinder, or the roughness elements placed on a line parallel to the leading edge of a flat plate (figure 1.2). Active devices requires actuators that may be driven in a time-dependent manner and require energy, for instance sweeping jets and plasma actuators (figure 1.3). The actuation command may be pre-determined (open-loop control) or rely on appropriate sensing of the flow state (closed-loop control).

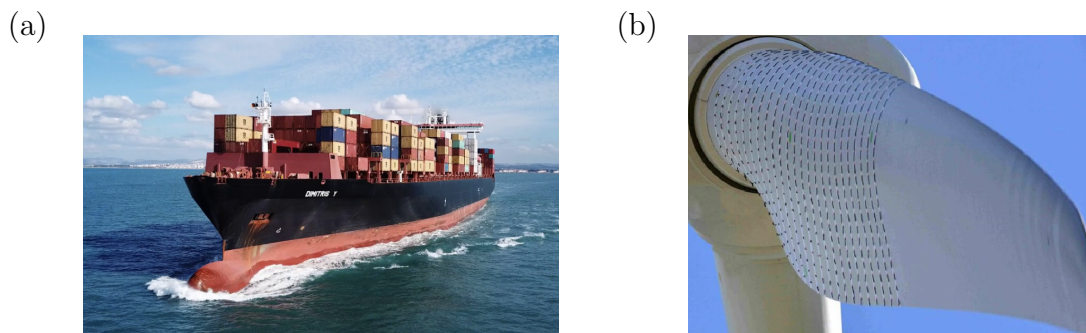


Figure 1.1: Candidate systems for flow control and optimization. (a) ocean shipping [24] and (b) wind turbine blade design [25].

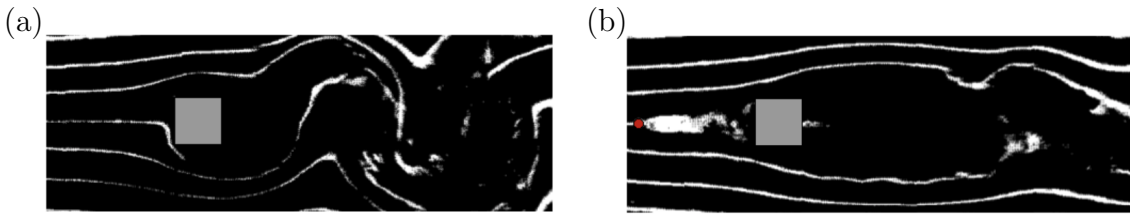


Figure 1.2: *Passive control: smoke visualization of the turbulent flow past a square cylinder, (a) without and (b) with a small, upstream, passive rod (in red). Adapted from [26].*

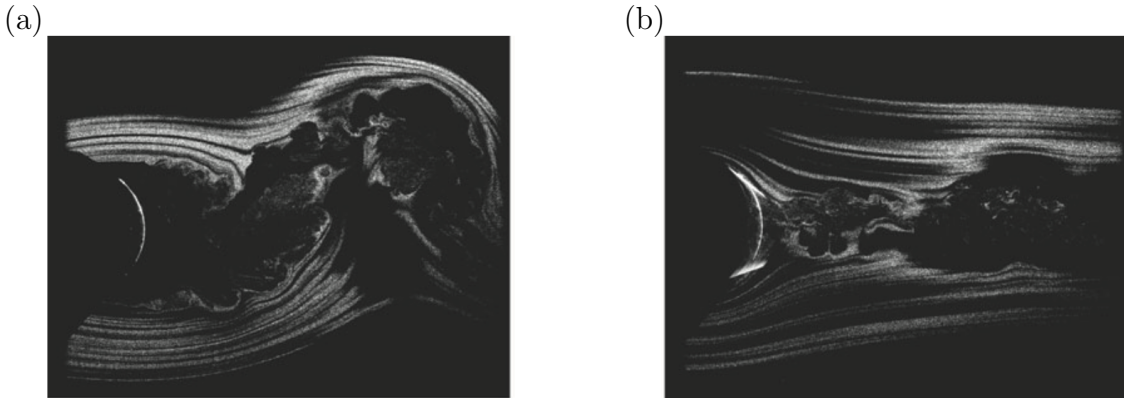


Figure 1.3: *Active, open-loop control: particle-image-velocimetry images of the turbulent flow past a circular cylinder with plasma actuators (a) off and (b) on, on the lee side of the cylinder. Adapted from [27].*

1.2 Systematical approaches for flow control

In the context of real-world, industrial applications, control strategies have long been (and often still are) determined by trial and error using extensive, costly experimental or numerical campaigns. This has motivated the development of rigorous mathematical formalisms capable of delivering optimal design at reduced costs, that are generally classified as gradient-based and gradient-free, depending on whether the method requires gradient information, in addition to function evaluations, to determine adequate search directions for better designs during optimization iterations. Both gradient-based and gradient-free methods can benefit from cheap-to-evaluate surrogate models to evaluate computationally expensive objective and constraint functions without resorting systematically to numerical simulations [28]. Several approaches exist for building such surrogate models, such as response surfaces, Bayesian methods, and artificial neural networks.

Apart from being prone to being trapped in local optimal (hence a possibly high sensitivity to the initial guess), gradient methods are effective in large optimization spaces, especially when the gradient is computed by the adjoint method. The lat-

ter has progressively gained prominence in a number of applications ranging from atmospheric sciences [29] to shape optimization [30], to flow control [31], and comes in two forms, namely continuous and discrete. In the continuous approach, control theory is applied to the continuous form of the linearized governing partial differential equations (PDEs), which yields analytical PDEs for the adjoint variable, to be discretized and solved. In the discrete approach, the control theory is applied to the discrete form of the governing equations, which yields a linear system of equations for solving the discretized adjoint variable. The main advantage of the continuous adjoint method is that it provides a continuous analytical equation that does not depend on the solver for the forward equation, can be solved separately, and gives insight into the physical interpretation to be given to the adjoint numerical solutions. The main disadvantage is that such adjoint equations must be deriving and implementing manually on a case by case basis, which is time consuming and error prone in the absence of suitable benchmark test cases. Also, difficulties have been reported when the observed in situations where adjoint equation features sharp discontinuities in the source terms [32]. The discrete approach is conceptually simpler, as a discrete adjoint operator for the Euclidian inner product is the transpose of the matrix form of the associated linear operator. Nonetheless, the difficulty to perform exact linearization of the underlying sophisticated numerical schemes and turbulence models often leads to approximations or simplifications (such as the neglect of the differentiation of artificial dissipation or the assumption of constant eddy-viscosity in turbulent flows) that can lead to inaccurate discrete gradient of the objective function, and may in turn affect the optimization process [33]. A possibility to address such implementation issues is to use automatic differentiation [34] to calculate all required derivatives in an automatic manner, but this requires to provide the source code of the original flow solver, and can yield a high computational complexity.

Gradient-free methods usually increases the likelihood of finding a global optimum, and does not require continuity over the design space, but can be more complex to implement and to use than gradient-based methods. Popular gradient-free techniques include genetic algorithms [35], a population-based optimisation technique implementing an evolutionary loop in which each iteration corresponds to a generation, selected upon principles derived from genetics and natural evolution mechanisms (crossing, mutation, selection), and particle swarm optimization, a bio-inspired algorithm mimicking the movement of organisms in a bird flock or fish school, in which a population (called a swarm) of candidate solutions (called particles) moves around in the search-space from its own knowledge of the best-known position, as well as the entire swarm's best-known position (in a way such that when improved positions are being discovered, they will then come to guide the movements of the swarm). These methods are generally considered powerful, general-purpose optimizers, capable of easily handling different types of variables

and functions with no adaptation necessary [36]. Moreover, they are less sensitive than gradient methods to the numerical noise that may be present in the computations, and high-fidelity CFD codes can be used without any modifications. The main drawbacks associated with these algorithms are the high computational cost, the poor ability to handle geometrical and/or operational constraints, the requirement for problem specific tuning and the limited number of design parameters that can be tackled simultaneously [32].

1.3 The emergence of Deep Reinforcement learning

Despite the considerable efforts put into the theory of flow control, the above approaches face considerable difficulties when attempting to design complex strategies, and most studies traditionally settle for simplistic (and probably suboptimal) harmonic or constant input [3]. Therefore, there remains a need to develop efficient control methods capable of perform complex control and to take full advantage of actuation possibilities. A promising option for doing so is to rely on machine learning methods, and more specifically deep neural network (DNN), a family of versatile parametric tools that can learn how to hierarchically extract informative features from data. Neural networks have reached astonishing performance levels in various domains, such as image classification [4], speech recognition [5] or generative tasks [6]. With generalized access to GPU computational resources through cheaper hardware or cloud computing, this has yielded considerable progress in the domain of decision-making techniques, by the coupling of DNNs with reinforcement learning algorithms (called deep reinforcement learning, or DRL). These advances have lifted several major obstacles that had so far hindered classical reinforcement learning, by allowing the use of high-dimensional state spaces and exploiting the feature extraction capabilities of DNNs. In return, unprecedented efficiency has been achieved in many domains such as robotics [7], language processing [8], games [9], although DRL has also proven useful in industrial applications, for instance autonomous cars [10] or data center cooling [11].

There is also great potential for applying DRL to fluid mechanics, for which efforts are ongoing but still at an early stage, with a handful of pioneering studies providing insight into the performance improvements to be delivered in the field. Sustained commitment from the machine learning community has allowed expanding the scope from computationally inexpensive, low-dimensional reductions of the underlying fluid dynamics [12–14] to complex Navier–Stokes systems, all the way to experimental set-ups [15]. Nonetheless, only a limited range of applications has been considered so far (that includes, by chronological order, swimmers, drag reduction, and shape optimization) while others have either received marginal attention (microfluidics [16], free convection heat transfert [17]) or been essentially left aside

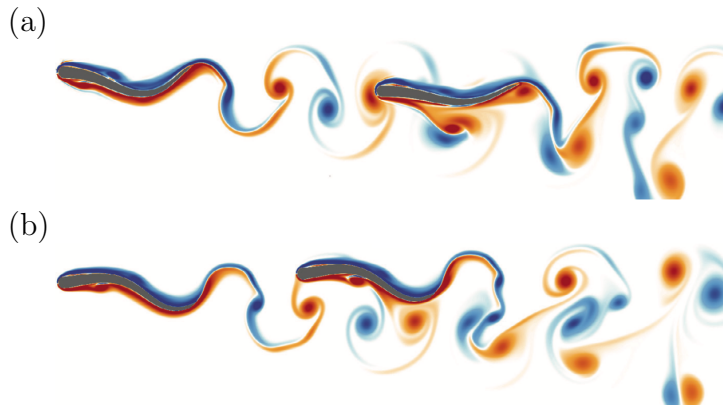


Figure 1.4: Vorticity contours for two self-propelled swimmers arranged in a leader-follower configuration. (a) DRL-based smart-follower. (b) Uncontrolled follower (below). Adapted from [37].

(multiphase flows or fluid-structure interactions).

The control of swimmers (figure 1.4) has been a pioneering field for applying deep reinforcement learning to fluid mechanics problems, with a couple of early publications seeking to optimize the kinematics of two self-propelled swimmers arranged in a leader-follower configuration from 2-D and 3-D simulations of viscous incompressible flows [37, 38]. Namely, the first fish (leader) swims a steady gait, while the second fish (follower) using DRL to adapt its behavior dynamically to account for the effects of the wake encountered. A simple reward that increasingly penalizes the follower when it strays too far away from the leader path allows identifying an optimal arrangement yielding up to about 30% reduction in energy expenditure for the follower, provided it keep its position in the center of the leader's wake, and synchronizes its head with the vortices shed by the leader.

Since then, drag reduction has been by far the class of problems that has received the most attention. Ten or so different papers have considered prototypical, two-dimensional (2-D) incompressible flows past span-wise infinite cylinders, which is mostly because the open-source diffusion of the seminal work from Rabault [39] shown in figure 1.5 has been re-used in several follow-up works. Various control strategies have been implemented, including zero-mass-flow-rate jets [40, 41], rotating cylinders [42, 43] or plasma actuators [44], with reported reductions by up to 20-30%. Since performing a relevant network update requires evaluating a sufficient number of actions drawn from the current policy (which in turn requires computing the same amount of rewards from resource expensive numerical simulations), most DRL agents learn at a faster pace by interacting simultaneously with multiple environments. This has become customary for DRL-based control of CFD problems, after Rabault and Kuhnle [40] have teased an almost perfect speedup up to 20 parallel environments, and a decent performance improvement up to 60. Regarding

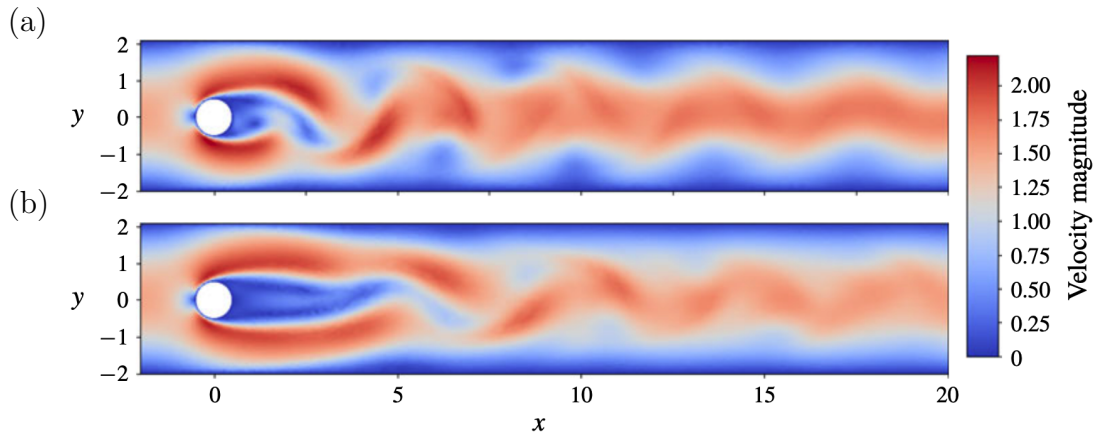


Figure 1.5: Snapshots of the velocity magnitude (a) without and (b) with DRL-based active flow control occurring via actuation zero-mass-flow-rate jets on the lee side of the cylinder. Reproduced from [39].

the flow regimes, almost all contributions assume laminar conditions with Reynolds numbers in a range of a few hundred. A weakly turbulent case at an intermediate Reynolds number $Re = 1000$ is explicitly targeted in [45], but makes it harder to achieve successful drag reduction, as evidenced by the increased number of episodes needed to learn an efficient policy.

Shape optimization is another field inherently associated with flow control, that can seem as a natural domain application for the DRL techniques covered above. Nonetheless, it is worth noticing that shape optimization generally consists in determining a fixed shape meeting a set of required criteria (e.g. high lift-to-drag ratio, low pressure loss). This is not per se the original purpose of DRL, that aims at identifying optimal state-to-action relations (by means of a neural network) and is thus best suited to dynamically manipulate a deformable shape. In return, all dedicated studies in the literature rely on incremental shape transformations, meaning that an initial shape is incrementally modified into an optimal one [21, 46, 47].

1.4 Objectives of the thesis

The thesis is part of the MINDS project (Mines Initiative for Numeric and Data Science), that federates 15 research centers from the Carnot M.I.N.E.S. Institute, and is interested in the convergence of high performance computing and data science. It thus aims at developing a joint digital research and development platform bringing together advances in both fields. The long-term ambition is to revisit the current design of control strategies for computationally-expensive flow problems representative of industrial applications (high Reynolds numbers, complex geometries, multiple phases and/or non-Newtonian fluids), which requires flexible tools that can be easily

integrated in a tuning-free, ready-to-use, near-real time optimization procedure.

The first objective is to further shape the capabilities of Deep Reinforcement Learning (DRL) for closed-loop control in a CFD context. The aforementioned studies tout the high potential of this approach on textbook problems, prominently low Reynolds number drag reduction problems. Nonetheless, considerable efforts are needed to (i) consolidate the acquired knowledge, (ii) bridge the gap between DRL and advanced numerical methods for multiscale, multiphysics computational fluid dynamics (CFD), and (iii) meet the expectations inevitably raised by the need to tackle industrial scale problems. Two key ingredients are mandatory for this purpose: an efficient DRL algorithm for learning, and a scalable CFD solver to accurately predict the computationally expensive numerical solutions from which to extract the reward fed to the DRL agent. We aim here at coupling Proximal policy optimization (PPO [18]) and CimLib-CFD in a high-performance computing context. The former is a relative newcomer that has quickly gained momentum as one of the go-to DRL algorithms for flow control problems due to its data efficiency, simplicity of implementation and reliable performances. The latter is a massively parallel, multiphysics C++ finite element library developed by the CFL team at the CEMEF research center. The expectation is that this will allow tackling pending problems in various scopes, e.g., optimization of steady and unsteady aerodynamics coefficients, load of structures, control of industrial thermic processes, optimal mixing of multiphase flows, and many others.

A second objective stems from the observation that, while neural networks have long been used in optimization problems (where the policy to be learnt is independent on state, and simple harmonic or constant control input is indeed relevant), they are most often exploited as trained surrogates for the actual objective function [19], but have long been mostly left out of the central optimization process, with the exception of a handful of studies [20, 21]. A premise of this thesis is that DRL algorithms can also be used as efficient black-box optimizers for such situations also relevant to open-loop flow control problems. This requires tweaking a classical algorithm for the neural network to learn a simple mapping from a constant input state to an optimal action by interacting only once per episode with its environment (hence, single-step episodes, and by extension, single-step DRL, in contrast with classical DRL in which the reward evaluates, not a single, but a series of actions taken over the course of the same episode, hence the multi-step moniker). We gauge here the ability of single-step DRL to reliably optimize complex flow systems, that such approach has been speculated to hold a high potential, but remains to be analyzed in full depth (we are not aware of any other work applying DRL this way, although a similar concept of “stateless DRL” has been early sketched in [22] for validation purpose, but has not been further pursued). We shall see that in essence, single-step DRL methods inherit from deep policy gradient algorithms in

the sense that relevant probability density function parameters are obtained from neural networks trained using a policy gradient-like loss. Yet, they also fall heir of evolutionary strategies (ES), as their successive steps follow a generation/individual nomenclature, exploiting informations from previous generations in order to update the parameters of a probability density function.

1.5 Layout of the thesis

This thesis is divided into 8 chapters, including the present introduction. Chapter 2 reviews the main DRL algorithms that have been used in the context of fluid mechanics, and introduces the concept of single-step DRL developed over the course of this work. Chapter 3 describes the CFD methods used to compute the numerical reward fed to the the DRL agent. This includes the VMS-stabilized finite element formulations used to solve the Navier–Stokes and Reynolds–averaged Navier–Stokes equations, and the immersed volume method coupled to anisotropic boundary layer mesh adaptation, used to solve accurately a single set equations pertaining to both solid and fluid phases on a unique computational domain. Chapter 4 assesses relevance of the single-step DRL approach as a black box optimizer for open-loop control problems governed by the Navier–Stokes equations. Several laminar and turbulent cases are used as testbed for implementing and validating the approach, with special emphasis on drag reduction. Chapter 5 focuses on the control of conjugate heat transfer systems governed by the coupled Navier–Stokes and heat equations, and presents the very first attempt of using DRL to control two- and three-dimensional forced convection. Chapter 6 further extends the scope of single-step DRL to shape optimization by direct optimization of state-independent shape parameters. Chapter 7 assesses the capability of multi-step DRL for active drag reduction and thermal control. Finally, Chapter 8 provides the conclusion and discusses the possible extension of the present work to more complex flow physics.

1.6 Achievements

The coupling between state-of-the art DRL algorithms and CFD solvers (both components developed in-house by the CFL team at the CEMEF research center) into a unified environment, as well as their implementation, validation and assessment in a context of high performance computing, make for the main novelty of this thesis. Ultimately, the emphasis has been on assessing the relevance and efficiency of the single-step DRL methodology, with later efforts dedicated to the validation and application of the classical multi-step approach in the context of thermal control. The conducted work has contributed to the publications, oral communications and prizes presented below:

Publications

- Ghraieb, Viquerat, Larchet, Meliga & Hachem, Single-step deep reinforcement learning for two- and three-dimensional optimal shape design, submitted to *Phys. of Fluids*, 2022
- Hachem, Ghraieb, Larchet, Viquerat & Meliga, Deep reinforcement learning for the control of conjugate heat transfer, *J. Comp. Phys.* 436:110317, 2021
- Ghraieb, Viquerat, Larchet, Meliga & Hachem, Single-step deep reinforcement learning for open-loop control of laminar and turbulent flows, *Phys. Rev. Fluids* 6:053902, 2021
- Viquerat, Rabault, Kuhnle, Ghraieb, Larcher & Hachem, Direct shape optimization through deep reinforcement learning, *J. Comp. Phys.* 428:110080, 2021

Conferences

- Ghraieb, Viquerat, Larcher, Meliga & Hachem, Single-step deep reinforcement learning for open-loop control of laminar and turbulent flows. *Mechanistic Machine Learning and Digital Twins for Computational Science, Engineering & Technology*, september 26-29, 2021, San Diego CA.

Awards

- Finalist for the fourth edition of Pierre Laffite Award, "Couplage Mécanique numérique et Intelligence Artificielle pour le design et l'optimisation de forme", October 2020.

Chapter 2

Methodologie - Deep Reinforcement Learning

Contents

2.1	Introduction	17
2.2	Reinforcement learning framework	17
2.2.1	Value-based methods	20
2.2.2	Policy-based methods	23
2.3	Deep Reinforcement Learning	24
2.3.1	Deep Q-Network (DQN)	25
2.3.2	Vanilla deep policy gradient	26
2.3.3	Advantage actor-critic (A2C)	26
2.3.4	Deep Deterministic Policy gradient DDPG	27
2.3.5	Trust-region and proximal policy optimization (TRPO and PPO)	27
2.4	Single-step Deep Reinforcement Learning	29
2.4.1	PPO-1	29
2.4.2	Policy-Based Optimization	30
2.5	Conclusion	34

Ce chapitre représente une introduction simple aux concepts de l'apprentissage par renforcement RL. Il présente la formulation générale des problèmes RL et ses deux principales classes d'algorithmes, puis la combinaison des réseaux neuronaux profonds DNN avec les algorithmes RL, appelée apprentissage par renforcement profond DRL. Ensuite, il passe en revue les principaux algorithmes DRL qui ont été utilisés dans le contexte de la mécanique des fluides et dans ce travail. Enfin, le concept de DRL à une seule étape est décrit, y compris les deux algorithmes développés au cours de ce travail.

This chapter represents a simple introduction to the Reinforcement learning RL concepts. It introduces the general formulation of RL problems and its two major classes of algorithms, followed by the combination of Deep Neural Networks DNN with RL algorithms, called Deep Reinforcement Learning DRL. Then it reviews the main DRL algorithms that have been used in the context of fluid mechanics and in this work. Finally, the concept of single-step DRL is described including the two algorithms developed over the course of this work.

2.1 Introduction

When talking about learning, the idea to learn by trial-and-error while interacting with the surrounding environment first comes to mind. For example, a young child learns new motor skills by moving his arms and feet without any teacher, but with a direct sensorimotor connection to his environment. Thanks to this connection, he gathers many informations about actions and their short- and long-term consequences, and how to arrange a sequence of actions to achieve various goals.

Similarly, reinforcement learning encompasses methods able to solve sequential decision-making problems, in which an agent interacts with an environment to maximize a reward signal. The environment provides observations (a partial representation of its current state) to the agent, which will, in turn, return actions that will modify the state of the environment. For each action taken, the environment also provides a reward value that corresponds to a measure of the quality of the said action. With these informations, the goal of the agent is to learn to perform adequate sequences of actions in order to maximize reward signal in the long term. Indeed, in most cases, actions will affect the immediate reward and the following situation and, through that, all subsequent rewards. Consequently, the Reinforcement Learning (RL) problem is formalized using the concepts of optimal control of Markov Decision Processes (MDPs), which details will be introduced in next section.

In the course of the past decade, the coupling of DNNs with RL algorithms, called Deep Reinforcement Learning (DRL), has resulted in significant progress in decision-making techniques thanks to the DNNs' feature extraction capabilities and capacity to handle high-dimensional spaces. Several major obstacles to classical RL have been overcome, and remarkable efficiency has been achieved in many domains such as robotics [7], language processing [8], or games [9, 48]. Moreover, DRL has also proven useful in many industrial applications, such as autonomous cars or data center cooling, and recently in fluid mechanics [10, 42, 49].

In the first section, the general formulation of RL problems in the context of MDPs is given, and the two major classes of algorithms (called value-based and policy-based methods) are described. Then, the combination of DNN with RL algorithms, called DRL, is introduced, along with a review of several well-known DRL algorithms. As this chapter represents a simple introduction to the RL concepts, the interested reader is referred to [50] for additional details.

2.2 Reinforcement learning framework

The general RL problem is formalized as an MDP, a classical approach for resolving sequential decision-making problems. It consists in learning from the interaction between an agent that makes decisions and an environment involving everything

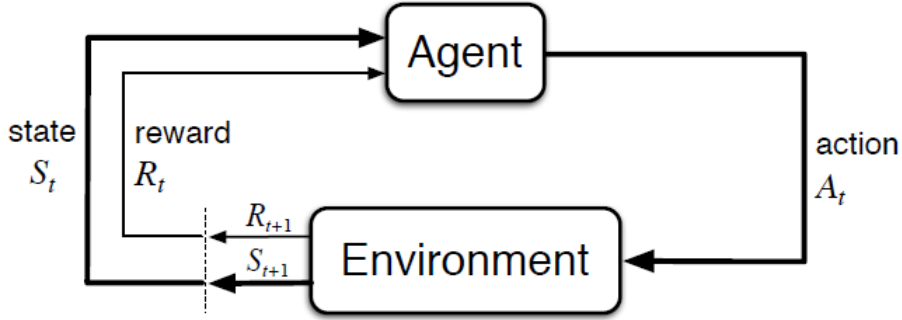


Figure 2.1: The agent–environment interaction in a Markov decision process [50]

outside the agent. At each time step t , the agent receives an observation of the environment’s state, $S_t \in \mathcal{S}$, and selects an action $A_t \in \mathcal{A}$ based on the latter (with \mathcal{S} and \mathcal{A} being the sets of valid states and actions). Action A_t is provided to the environment, which then returns a numerical reward, $R_{t+1} \in \mathcal{R}$ (where \mathcal{R} is the set of valid rewards), and observations of its new state, S_{t+1} . A sketch of the process is presented in figure (2.1). The succession of state and actions defines a *trajectory* τ :

$$\tau = (S_0, A_0, S_1, A_1, \dots, S_n)$$

In the finite MDP framework, a particular case is considered where state and action spaces are both discrete, and R_t and S_t are random variables having discrete probability distributions dependent on the previous state and action. That is, given particular values of the preceding state s and action a at time t , the probability of reaching the new state s' with reward r at time $t + 1$ is:

$$p(s', r | s, a) = \Pr\{S_{t+1} = s', R_{t+1} = r | S_t = s, A_t = a\}, \quad (2.1)$$

for all $s', s \in \mathcal{S}, r \in \mathcal{R}$, and $a \in A(s)$. Here, p defines the dynamics of the environment, or, put differently, it characterizes the dynamics of the MDP. Therefore, the past agent–environment interaction information must be included in the state S_{t-1} . The latter statement implies that the considered process has the *Markov property*, meaning that the process’s future only depends on the current observation, and the agent has no interest in looking at the entire history.

Formally, the goal of the agent is to maximize the expected discounted return $\mathbb{E}[G_t]$, where G_t is defined as:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots = \sum_{k=t+1}^T \gamma^{k-t-1} R_k, \quad (2.2)$$

with $\gamma \in [0, 1]$ being a discount factor weighting the relative importance of present and future rewards. Beyond the agent and its environment, an RL agent includes one or more of the following components:

- A *policy* π that defines the agent's way of choosing actions given a set of states. It may be deterministic, meaning that actions are directly obtained from the policy as $a = \pi(s)$, or stochastic, in which case the policy outputs define a probability density function from which actions are then sampled as $a \sim \pi(s)$;
- One or multiple *value functions* that estimate the expected, cumulative, discounted future reward, starting from a given state. In essence, the latter help measure the goodness of each state or each state-action pair;
- A *model* defining the environment's dynamic in conjunction with a planning algorithm.

We distinguish two categories of RL methods, namely *model-based* and *model-free*. An algorithm said to be *model-based* when the latter component is used, providing the probability $p(s', r|s, a)$ defined in equation (2.1) to compute the estimated state-transition probabilities and the expected reward for each state-action pair. In this case, the agent does not only learn from real experience, but also plans with simulated trajectories from the environment model. In particular, Dynamic programming (DP) is a class of general RL methods that assume complete knowledge of the transition and reward models of the MDP. Given the high induced computational cost, and the strong assumption of having a perfect model of the environment at hand, such methods are rarely exploited in practice.

Contrarily, in *model-free* algorithms, the agent interacts directly with the environment and learns from trial-and-error experience. These approaches are currently the most commonly used within the DRL community, mainly for their ease of application and implementation. Model-free methods are also divided into two categories: *value-based* and *policy-based*. Although both methods aim to maximize their expected return, *policy-based* methods directly optimize the parameterized policy, while *value-based* methods learn to estimate the expected value of a state-action pair optimally, which specifies the best action to take in each state. In the following sections, the concept of the value-based approaches is introduced and followed by its two fundamental classes of algorithms, named Monte Carlo methods and Temporal Difference Learning. Then, the basic concepts of policy-gradient methods are presented, where a parameterized policy is learned without relying on a value function for action selection.

2.2.1 Value-based methods

Most RL algorithms rely on the use of value functions to solve decision-making problems. The *state-value function* $v_\pi(s)$ estimates the expected discounted cumulative reward starting in state s , then following trajectory τ according to policy π :

$$v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s],$$

where $\mathbb{E}_\pi[\cdot]$ indicates the expected value of any random variable when following policy π , and t is any time step. Similarly, the *action-value function* $q_\pi(s, a)$ determines the expected return when starting from a given state s , taking action a , and following the policy π :

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a].$$

A fundamental characteristic of value functions is that they satisfy recursive relations. The following consistency condition between the value of s and the value of its possible successor states s' is called the *Bellman equation*, and holds for any policy π and any state s :

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s', r} p(s', r | s, a) [r + \gamma v_\pi(s')] \quad (2.3)$$

Value functions also define a partial ordering over policies: simply put, π_1 is better than or equal to π_2 if $v_{\pi_1}(s) \geq v_{\pi_2}(s)$ for all $s \in S$. Hence, at least one policy is always better than or equal to all other policies, and is called the *optimal policy* (although it is not necessarily unique). All optimal policies share the same optimal state-value function v_* and the same optimal action-value function q_* , defined as:

$$\begin{aligned} v_*(s) &= \max_{\pi} v_\pi(s), \forall s \in S. \\ q_*(s, a) &= \max_{\pi} q_\pi(s, a), \forall s \in S, \forall a \in A(s). \end{aligned}$$

Thus, q_* can be written in terms of v_* as:

$$q_*(s, a) = \mathbb{E} [R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a].$$

Given that $v_*(s)$ is the optimal value function, it satisfies the consistency condition (2.3) for the state value function, which is called, in this case, the *Bellman optimality equation*. The latter expresses that, from state s , the state value function under an optimal policy must equal the expected return while choosing the best action:

$$v_*(s) = \max_{a \in A(s)} \sum_{s', r} p(s', r | s, a) [r + \gamma v_*(s')]. \quad (2.4)$$

Similarly, the *Bellman optimality equation* for q_* can be expressed as:

$$q_*(s) = \sum_{s',r} p(s', r | s, a) [r + \gamma \max_{a'} q_*(s', a')]. \quad (2.5)$$

Finally, the *Bellman optimality equation* must be solved, which is not simple, as it requires an exhaustive search over all possible actions, including their probabilities and expected rewards. That is why specific algorithms are needed to find the optimal policy in order to select the best sequence of actions for a given RL problem. Generally, value-based RL algorithms are described as *generalized policy iteration* (GPI) algorithms, referring to the idea of alternating between two main processes, *policy evaluation* (also called the *prediction problem*), and *policy improvement*, in a sequence to converge to an optimal policy:

$$\pi_0 \xrightarrow{PE} v_{\pi_0} \xrightarrow{PI} \pi_1 \dots \xrightarrow{PI} \pi_* \xrightarrow{PE} v_*$$

where the policy is being continually improved based on the value function. In policy evaluation, the value functions of the current policy π_i are estimated to obtain $V_{\pi_i}(s)$ (or $Q_{\pi_i}(s, a)$). Then, the current value function is used to generate a better policy π_{i+1} in the policy improvement step, for example, by selecting actions greedily with respect to the value function.

In the following, we consider two fundamental *model-free* approaches of value-based RL algorithms: Monte Carlo (MC) methods, and temporal-difference (TD) learning. MC updates rely on full trajectories, and therefore are not suited for step-by-step incremental computation. Contrarily, TD algorithms are able to perform updates using incremental updates. Hence, they present different strengths and weaknesses and differ in their efficiency and performance.

2.2.1.1 Monte Carlo methods

The Monte Carlo (MC) methods exploit full trajectories sampled from the environment to perform updates of the agent [51]. They solve the RL problem by averaging the returns from each state-action pair and averaging the reward for each action. These methods are incremental in an episode-by-episode way where value estimates (policy evaluation) and policy improvements are made only at the end. Because the environment model is not available, it is particularly useful to predict the action-value functions, which is why MC-based approaches usually estimate $q(s, a)$ from experience by averaging the returns observed after several visits to each state-action pair. As more returns are observed, the average should converge towards the expected value. So MC methods estimate $q_\pi(s, a)$ for all state-action pairs, given a set of episodes collected following π , by averaging the returns of the doublet (s, a) at each occurrence in the episodes.

To approximate optimal policies, MC control algorithm follows a generalized policy iteration GPI algorithm, which consists in alternating between the policy evaluation step to compute each $q_\pi(s, a)$ and the policy improvement step to improve the actual policy based on the action-value function. There are two approaches for control: the *on-policy* and the *off-policy* methods. The *on-policy* methods evaluate or improve the actual policy used to make decisions. This method uses $\varepsilon - greedy$ policies, meaning that at each step, the agent either selects the action with the maximal estimated action-value function (greedily) with probability $1 - \varepsilon$, or selects a random action with probability ε . In contrast, a more straightforward approach is the *off-policy*, where the agent learns the optimal policy using two policies: one that is learned about and becomes the optimal policy, and one that is more exploratory and used to produce behavior. The *target policy* π is being learned about, while the *behavior policy* b is being used to generate behavior. A benefit of this split is that the policy π may be deterministic (greedy), while the policy b keeps sampling all possible actions. Usually, *off-policy* methods use the importance sampling approach [52], a known technique to estimate the expected values (v_π or q_π) given samples obtained following another policy b ($b \neq \pi$).

2.2.1.2 Temporal-difference learning

Temporal difference (TD) learning is a mixture of MC approaches and DP approaches. TD methods learn from experience without an environment model like MC. Moreover, they make updates based on other learned value functions without waiting for the end of episodes; in other words, they bootstrap [53].

In the policy evaluation step, MC methods must wait until the end of episodes, then get G_t and use it as a target for the value function v_π (or q_π), as shown in the following MC update formula:

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)],$$

where α is a constant step-size parameter (learning rate). On the other hand, TD methods can perform updates with an arbitrary regularity. When updating at every time step, the update formula is called the *one-step TD update* (noted TD(0)), and reads:

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

Note that the term in brackets is an error between the estimated value of S_t and its target $R_{t+1} + V(S_{t+1})$. This quantity appears in various forms through RL, and it is called the TD error:

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$$

As in MC methods, two approaches exist for control: *on-policy* and *off-policy* methods. For on-policy methods, q_π is estimated using the original behavior policy π for all state-action pairs (s, a) . This step can be done using TD prediction described previously, and the corresponding update rule is:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)] \quad (2.6)$$

Every time an update is made, this rule uses the quintuple $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$, which gave its name to the SARSA algorithm. Then, for control, it keeps estimating q_π using the behavior policy π and changing π toward new ε -greedy policies in the policy improvement step. On the other hand, *Q-learning* was one of the breakthroughs in RL as an off-policy TD algorithm [54]. It is defined by:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \quad (2.7)$$

where the learned action-value function Q directly approximates the optimal action-value function q_* without considering the followed policy.

It must be noted that the algorithms discussed above are based on a one-step return (TD(0), Q(0), and SARSA(0)), but variants with multi-step returns are also commonly exploited. For example, in the n -step update, the target of the update rule is the n -step return:

$$G_{t:t+n} = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n})$$

2.2.2 Policy-based methods

So far, all the presented approaches belonged to the class of value-based methods, *i.e.* methods where the agent learns one or multiple value functions (using tables or approximate functions), thus implicitly defining a policy by selecting the action of highest value. Contrarily, *policy-gradient* methods directly learn a parameterized policy $\pi_\theta(s, a)$ (with θ representing the policy parameters), and do not rely on a value function to select actions (implying that value functions can still be used for other purposes). Policy-based methods yield several advantages compared to value-based methods, one of them being that they can naturally handle high dimensional action spaces. Also, they are able to learn stochastic policies, and display better convergence properties. These methods are based on the estimation of the gradient of a scalar objective function, called performance measure $J(\theta)$, with respect to θ . In the episodic tasks, the performance measure J is defined based on the expected discounted cumulative reward or as the value function of the start state of the episode:

$$J(\theta) = \mathbb{E}_{\pi_\theta}[G_t]. \quad (2.8)$$

The policy-based methods aim at maximizing J_θ by performing gradient ascent updates using an approximate value of the policy gradient:

$$\theta_{t+1} = \theta_t + \alpha \widehat{\nabla} J_{\theta_t}, \quad (2.9)$$

where $\widehat{\nabla} J_{\theta_t} \in \mathbb{R}^{d'}$ is a stochastic estimate approximating the gradient of J with respect to θ_t . The estimation of $\nabla J(\theta)$ is obtained thanks to the *policy gradient theorem*, which provides an analytic definition for this gradient with respect to the policy parameter (needed to approximate for the general scheme of stochastic gradient ascent):

$$\nabla J(\theta) = \mathbb{E}_\pi \left[\sum_{t=0}^T \nabla \log \pi(A_t | S_t, \theta) G_t \right] \quad (2.10)$$

where the gradients are column vectors of the partial derivatives. In this way, the stochastic gradient-ascent update formula algorithm can be obtained. Moreover, it represents the REINFORCE algorithm, the base of policy gradient methods, which includes a complete return (waits until the end of the episode as in MC methods [55]).

2.3 Deep Reinforcement Learning

As seen in previous sections, several approaches are available for the function approximations required in RL problems with large state or action spaces. Yet, during the past decade, the use of *deep neural networks* (DNNs) as approximators in RL algorithms (called deep reinforcement learning, or DRL) has witnessed a great success, leading to multiple breakthroughs in the domain of active control. Among those are the mastering of games, with trained agents displaying super-human levels on various board and Atari games [48][56], but also applications to robotics [57], finance [58], computer vision [59], self-driving cars [60], optimal control [61], and recently fluid mechanics [13][39][38].

A neural network is a collection of artificial neurons, *i.e.* connected computational units with universal approximation capabilities, that can be trained to arbitrarily well approximate the mapping function between input and output spaces. Each connection provides the output of a neuron as an input to another neuron. Each neuron performs a weighted sum of its inputs, to assign significance to the inputs with regard to the task the algorithm is trying to learn. It then adds a bias to better represent the part of the output that is actually independent of the input. Finally, it feeds a non-linear activation function that determines whether and to what extent the computed value should affect the ultimate outcome. As sketched

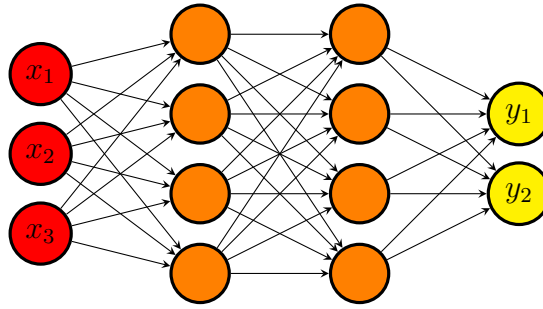


Figure 2.2: Fully connected neural network with two hidden layers, modeling a mapping from \mathbb{R}^3 to \mathbb{R}^2 .

in figure (2.2), a fully connected network is generally organized into layers, with the neurons of one layer being connected solely to those of the immediately preceding and following layers. The layer that receives the external data is the input layer, the layer that produces the outcome is the output layer, and in between them are zero or more hidden layers. Once a forward pass has been achieved from the input to the output through each layer, gradients of the loss with respect to the network parameters are obtained using the backpropagation algorithm [62], after what a stochastic gradient ascent algorithm is used to adjust the weights and biases of the network. An efficient neural network design requires a relevant architecture (type of network, depth, activation functions, etc.), adequate hyper-parameters (optimizer, learning rate, batch size, etc.), and a sufficiently large amount of data to learn from are all critical ingredients for success, according to the vast literature on the subject (see, for example, [63] and the references therein).

In the context of RL, DNNs can be exploited for the approximation of value functions ($\hat{v}(s, w)$ or $\hat{q}(s, a, w)$) and policies ($\pi(a|s, \theta)$), where w and θ are the network's parameters, and the learning process updates these connection weights and biases to reduce the well-chosen loss function. The rest of this section will briefly present some popular DRL methods and their main features.

2.3.1 Deep Q-Network (DQN)

In 2015, a novel algorithm combining Q-learning methods with DNNs, called *Deep Q-Network* (DQN), was proposed by Mnih *et al.* [64]. DQN offered several contributions to solve the problems of convergence instabilities, and was shown outperform human players on a set of 49 Atari games [65]. The ingredients of DQN include a Q-network and a target network to temper learning instabilities, as well as experience replay [66]. The loss function used to train Q-networks is obtained from the update expressions seen in section 2.2.1.2 for the Q-learning method:

$$L(\theta) = \mathbb{E}_{(S_t, A_t, R_t, S_{t+1}) \sim \mathcal{D}} [R_{t+1} + \gamma \max_a \hat{Q}(S_{t+1}, a, \theta_t^T) - \hat{Q}(S_t, A_t, \theta_t)]^2 \quad (2.11)$$

where \mathcal{D} is the set of transitions collected from the environment, and θ_t^T are the parameters of the target network, while θ_t are the Q-network parameters at step t . Note that the target network is updated periodically.

2.3.2 Vanilla deep policy gradient

A stochastic gradient algorithm is used in policy methods to perform network updates from a policy loss:

$$L(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^T \log \pi(A_t | S_t, \theta) G_t \right], \quad (2.12)$$

whose gradient equals the policy gradient 2.10. The latter is computed one layer at a time from the output to the input layer using the back-propagation algorithm with respect to each weight and bias by the chain rule. Because the loss (2.12) takes the form of an expected value that can be numerically computed using an empirical average over a set of complete trajectories, this method is also known as Monte Carlo policy gradient. However, if some low-quality actions are taken along the path, their negative impact will be masked by the positive actions and will go undetected. This problem can be solved using actor-critic methods, which combine a Q function evaluation with policy optimization.

2.3.3 Advantage actor-critic (A2C)

Indeed, different strategies exist to reduce the high variance of training the agent from (2.12), with the discounted cumulative reward being replaced by the advantage function:

$$A(S_t, A_t) = Q(S_t, A_t) - V(S_t) \quad (2.13)$$

which represents the improvement in the expected cumulative reward when taking action a in state s compared to the average of all possible actions taken in state s . As a result, the loss function reads:

$$L(\theta) = \mathbb{E}_\pi \left[\sum_0^T \log \pi(A_t | S_t, \theta) A_\pi(S_t, A_t) \right] \quad (2.14)$$

In practice, the classical policy network (known as *actor*) is used in conjunction with a second network (known as *critic*) that learns to predict the state-value function

$V(s)$. To avoid having to learn a third network to predict the state-action value $Q(s, a)$, the advantage function is approximated as:

$$A(S_t, A_t) \simeq R(S_t, A_t) + \gamma V(S_{t+1}) - V(S_t) \quad (2.15)$$

Thus, the actor-critic algorithm allows training the policy network in a temporal-difference manner, allowing updates to be performed multiple times during an episode.

2.3.4 Deep Deterministic Policy gradient DDPG

Policy gradient methods are widely used in problems with continuous action spaces. The basic idea is to represent the policy by a stochastic parametric probability distribution $\pi_\theta(a|s)$. However, in 2014, a deterministic policy gradient (DPG) algorithm was introduced, that instead considers deterministic policies $a = \mu_\theta(s)$ [67]. This algorithm belongs to actor-critic methods, and exploits an off-policy approach that uses a behavior policy to learn a deterministic target policy. Moreover, in 2016, a *Deep Deterministic Policy Gradient* (DDPG) [61] algorithm was implemented, extended from DQN and DPG algorithms. With actor-critic as in DPG, DDPG avoids the optimization of action value function at every time step. Meanwhile, it makes the learning stable and robust by deploying experience replay and an idea similar to target network as in DQN. In DDPG, the loss function is:

$$L(\theta) = \mathbf{E}_{(S_t, A_t, R_t, S_{t+1}) \sim \mathcal{D}} [R_{t+1} + \gamma \hat{Q}^T(S_{t+1}, \mu^T(S_{t+1}, \theta_t^{\mu^T}), \theta_t^{Q^T}) - \hat{Q}(S_t, A_t, \theta_t^Q)]^2 \quad (2.16)$$

where \mathcal{D} is the set of transitions collected from the environment, and μ^T, Q^T refer to the target actor and critic, while μ, Q are the behavior actor and critic.

Finally, in order to achieve an efficient balance between exploration and exploitation, gaussian noise is usually applied to the predicted actions.

2.3.5 Trust-region and proximal policy optimization (TRPO and PPO)

The performance of policy gradient methods is hurt by the high sensitivity to the learning rate, *i.e.*, the size of the step to be taken in the gradient direction. Indeed, as too small learning rates are detrimental to learning, too large learning rates can also be detrimental and lead to a definitive performance collapse. To overcome this issue, Schulman *et al.* [68] proposed in 2015 the trust-region policy optimization TRPO algorithm, in which policy updates are based limited to a trust region defined by a maximal deviation between the previous policy and the current one.

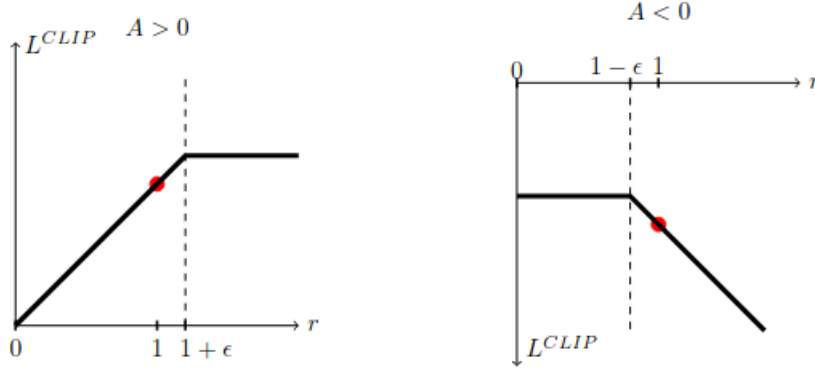


Figure 2.3: Clipping the probability ratio r_t [18]

Although efficient, the TRPO algorithm was soon overshadowed by the proximal policy optimization PPO algorithm, also proposed by Schulman *et al.*[18] in 2017. The policy updates of the PPO method are based on a clipped surrogate objective, defined as:

$$L^{\text{clip}}(\theta) = \hat{\mathbb{E}}_t \left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t) \right], \quad (2.17)$$

where $r_t(\theta)$ is the probability ratio $r_t(\theta) = \frac{\pi(A_t|S_t,\theta)}{\pi(A_t|S_t,\theta_{\text{old}})}$ and ϵ is a small clip range parameter, usually in the range $[0.1, 0.3]$. In this expression, the probability ratio is clipped to avoid r_t from moving outside the interval $[1 - \epsilon, 1 + \epsilon]$ (see figure (2.3)). Due to its improved learning stability and its relatively robust behaviour with respect to hyper-parameters, the PPO algorithm has received considerable attention in the DRL community. The pseudo-algorithm is provided below, where at each iteration, T timesteps of data (states, actions, returns, advantages...) are collected by N parallel actors. Then the surrogate loss is constructed on these $N \times T$ timesteps of data and optimized using a minibatch SGD for K epochs.

Algorithm 1 PPO Algorithm [18]

```

for iteration=1, 2,... do
  for actor=1, 2, ..., N do
    Run policy  $\pi_{\theta_{\text{old}}}$  in environment for  $T$  timesteps
    Compute advantage estimates  $\hat{A}_1, \dots, \hat{A}_T$ 
  Optimize surrogate  $L$  w.r.t.  $\theta$ , with  $K$  epochs and minibatch size  $M \leq NT$ 
   $\theta_{\text{old}} \leftarrow \theta$ 

```

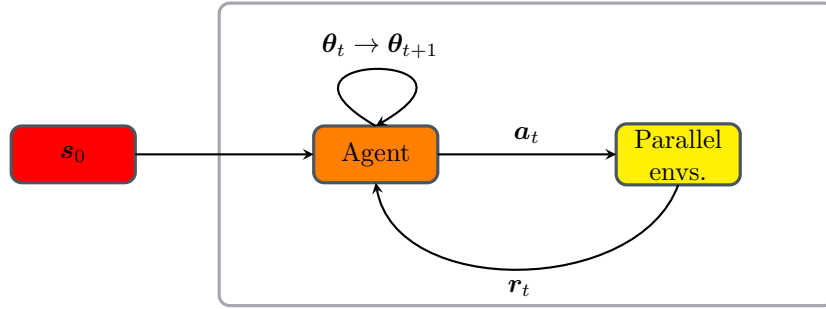


Figure 2.4: PPO-1 [69].

2.4 Single-step Deep Reinforcement Learning

In the present manuscript, a considerable part of the applications consider the use of a particular class of algorithms, adapted from standard DRL, and labeled as *single-step* DRL. These algorithms are degenerate versions of the policy gradient RL algorithm. Their premise is that single-step episodes are enough for the policy to be learned in a state-independent mode as in optimization and open-loop control problems. Indeed, single-step DRL algorithms seek the optimal parametrization θ^* that performs the optimal action $a^* = \pi_{\theta^*}(s_0)$ maximizing the instantaneous immediate reward instead of a discounted cumulated reward, with s_0 being some input state (usually a constant vector) constantly fed to the agent. The following sections are devoted to the presentation of two algorithms used in this thesis. The first one, called PPO-1, relies on the classical PPO algorithm, that has quickly gained prominence in the DRL community, and is by far the most common algorithm exploited in the context of DRL-based control for fluid mechanics. The second one, called PBO, is an in-house algorithm developed by the CFL team under the premise that convergence to the optimal can be made faster if actions are sampled anisotropically from full covariance matrices.

2.4.1 PPO-1

The PPO-1 algorithm is a single-step version of the proximal policy optimization (PPO) algorithm, where the optimal policy is independent of the input state, and is therefore adapted to the optimization of open-loop control laws. In this method, the optimal mapping from a constant input state S_0 is sought, using only one action per episode (the situation is summed up in figure (2.4)). In practice, the agent is provided with a constant input state, and outputs the mean and variance d -dimensional normal distribution with an isotropic covariance matrix, d being the number of actions sent to the environment by the agent.

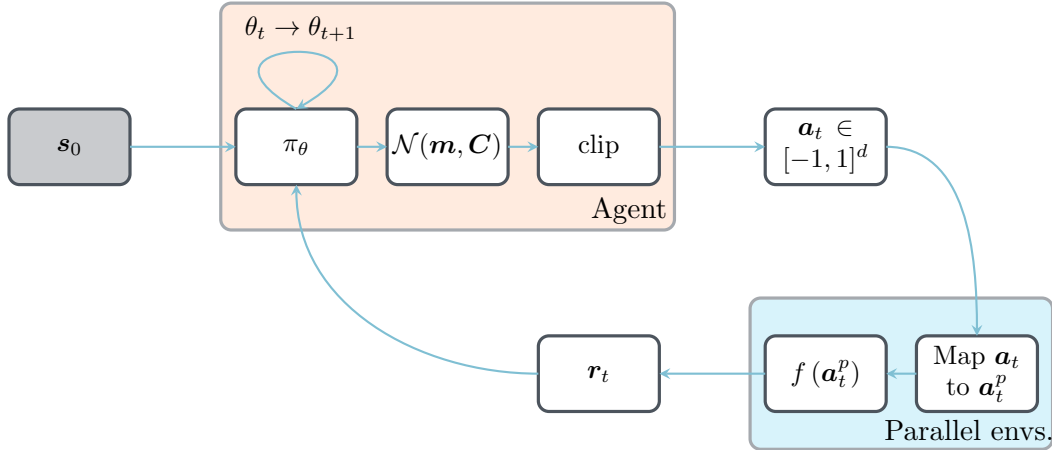


Figure 2.5: Action loop for the PBO method [70].

2.4.2 Policy-Based Optimization

Another black-box optimization method called *Policy-Based Optimization* (PBO) [70] has been developed for solving single-step optimization problems. It is based on the classical DRL policy gradient algorithm, and shares similarities with evolution strategy (ES) methods, in particular with the covariance matrix adaptation evolution strategy method (CMA-ES). In the latter, the parameters of a multivariate normal law with full covariance matrix are evolved with successive generations using pre-defined update laws, yielding a powerful and flexible optimization technique. Similarly, the PBO method relies on a normal law with full covariance matrix being generated from neural network outputs, the successive optimization of which using the policy gradient loss eventually lead to an optimum of the cost function. Figure (2.5) shows a sketch of the functioning of the PBO algorithm, where the agent draws a population of actions from the current policy and is incentivized to update the policy parameters for the next population of actions in order to yield larger rewards. Note that the agent performs only a transformation from a constant input state to a given action and not a complex state-action relation; therefore, PBO uses smaller policy networks (compared to other DRL contributions networks in the literature).

2.4.2.1 Gradient ascent update rule

In practice, PBO generates actions using a probability density function. Three independent neural networks are used to output a d -dimensional multivariate normal distribution $\mathcal{N}(\mathbf{m}, \mathbf{C})$ with mean \mathbf{m} and full covariance matrix \mathbf{C} as shown in figure (2.6), with hyperbolic tangent and sigmoid activation functions on the output layers. Actions are then clipped in $[-1, 1]^d$ before being mapped to their relevant physical ranges \mathbf{a}^p (a step deferred to the environment as problem-specific, see figure (2.5)).

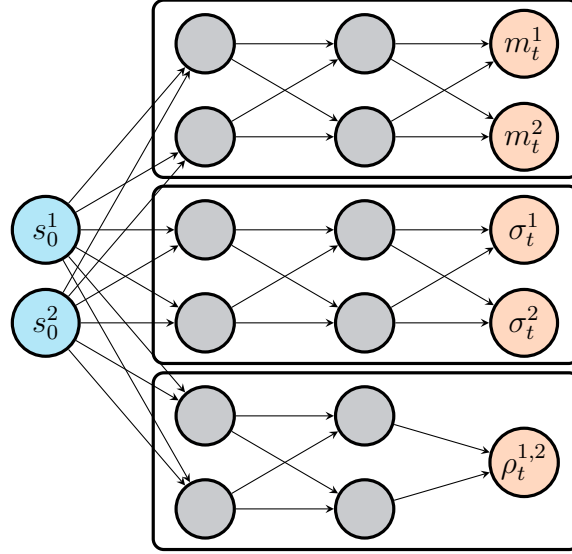


Figure 2.6: Policy networks used in PBO to map states to policy.

Ultimately, the Adam algorithm performs stochastic gradient ascent on the policy parameters using the modified loss:

$$L(\theta) = \mathbb{E}_{a \sim \pi_\theta} \left[\log \pi_\theta(a) \hat{R}(a) \right], \text{ with } \hat{R}(a) = \max \left(\frac{r(a) - \mu_r}{\sigma_r}, 0 \right). \quad (2.18)$$

In the latter equality, μ_r (resp. σ_r) represents the reward average (resp. standard deviation) over the current generation in the PBO loss, which is formally identical to the classical PG loss (2.12), except for the discounted cumulative reward substituted with a clipped generation-wise whitened reward. Since PBO is a single-step algorithm (single state-action pair), the discount factor can be set to $\gamma = 1$, in which case the advantage function (in PG loss) is reduced to the reward, as explained in [69]. In addition, the normalization to zero mean and unit standard deviation introduces bias but reduces variance, and consequently the number of actions required to estimate the expected value. Finally, when performing multiple mini-batch gradient steps with the same data, the max allows discarding negative-advantage actions that can destabilize learning.

2.4.2.2 Off-policy updates

Before the algorithm can proceed to update the agent parameters, it must sample a large number of state-action-reward triplets to accurately compute the expected value in the policy loss (2.18). At each generation, a set of actions derived from the current policy π_θ is distributed to n environments running in parallel; each computes

and returns a reward associated with its input action. This cycle can be repeated until the agent has a sufficient number of state-action-reward triplets. Still, because computing the reward can be a computationally intensive task, using an enormous value of n in many cases is not tractable, limiting the number of state-action-reward triplets available from the current policy. Therefore, like CMA-ES, PBO improves the loss evaluation's reliability by incorporating data from previous generations. The loss expression (2.18) must account for updating policy π_θ with samples generated under previous policies. So, given the off-policy π_b , the loss obtained using π_b 's samples is:

$$L_{\text{off}}(\theta) = \mathbb{E}_{a \sim \pi_b} \left[\frac{\pi_\theta(a)}{\pi_b(a)} \hat{R}(a) \right], \quad (2.19)$$

where $\frac{\pi_\theta(a)}{\pi_b(a)}$ is the *importance* term [71]. The objective function resulting gradient is, therefore:

$$\nabla_\theta J(\theta) = \mathbb{E}_{a \sim \pi_b} \left[\frac{\pi_\theta(a)}{\pi_b(a)} \nabla_\theta \log \pi_\theta(a) \hat{R}(a) \right], \quad (2.20)$$

and the original loss is retrieved for $\pi_b = \pi_\theta$. However, it was discovered that using (2.19) resulted in unstable updates in the final steps of the optimization process, particularly in the vicinity of local or global minima, producing a significant reduction in the algorithm's overall performance. That is why, currently, a decay parameter ETA is used in conjunction with loss expression (2.18) to give recent generations more weight by exponentially decreasing the reward from previous generations. In contrast, this issue is deferred to a future contribution for a more detailed study. Thus, a rule of thumb for the decay factor is used in this algorithm and given by:

$$\eta = 1 - e^{-\alpha d}, \quad (2.21)$$

with $\alpha > 0$ to keep a longer memory of the previous individuals as the problem dimensionality d increases.

In practice, each network is updated for n_e epochs with a learning rate of λ_r and history of n_g generations shuffled and organized in n_b mini-batches. Finally, PBO has the advantage of allowing all three networks to use different meta-parameters and network architectures, which can significantly impact the convergence rate.

2.4.2.3 Generating valid covariance matrices from neural network outputs

To have physical significance, matrices that represent correlations between variables must meet four basic properties: (i) all entries must be in $[-1, 1]$, (ii) all diagonal entries must be equal to 1, (iii) the matrix must be symmetric, and (iv) the matrix

must be positive semi-definite (PSD). The above naive approach of directly having a neural network outputting a set of adequate correlation parameters is doomed to fail, as there is no guarantee that the resulting matrix will be PSD. Furthermore, while it is theoretically possible to have the neural network output correlation coefficients until a PSD matrix is obtained, this quickly becomes inefficient, as the chances of finding a valid matrix for $d > 3$ are meagre. PBO solves this problem using hypersphere decomposition [72], a technique that generates valid correlation matrices from a set of angular coordinates on a unit-radius hypersphere. We should notice that the method parameterizes a lower triangular elementary matrix \mathbf{B}_d with entry:

$$b_{ij} = \begin{cases} 1 & \text{for } i = j = 1 \\ \cos \varphi_{ij} & \text{for } i > 1, j = 1 \\ \cos \varphi_{ij} \prod_{k=1}^{j-1} \sin \varphi_{ik} & \text{for } i > 1, j < i \\ \prod_{k=1}^{j-1} \sin \varphi_{ik} & \text{for } i > 1, j = i \\ 0 & \text{for } j > i \end{cases} \quad (2.22)$$

from a set of so-called correlative angles $\boldsymbol{\varphi} \in [0, \pi]^D$, with $D = \frac{d(d-1)}{2}$. For instance, the corresponding matrix for $d = 4$ reads:

$$\mathbf{B}_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \cos \varphi_{2,1} & \sin \varphi_{2,1} & 0 & 0 \\ \cos \varphi_{3,1} & \cos \varphi_{3,2} \sin \varphi_{3,1} & \sin \varphi_{3,2} \sin \varphi_{3,1} & 0 \\ \cos \varphi_{4,1} & \cos \varphi_{4,2} \sin \varphi_{4,1} & \cos \varphi_{4,3} \sin \varphi_{4,2} \sin \varphi_{4,1} & \sin \varphi_{4,3} \sin \varphi_{4,2} \sin \varphi_{4,1} \end{bmatrix} \quad (2.23)$$

Because it is symmetric and PSD by construction, the product of this matrix with its transpose is guaranteed to be a valid correlation matrix, with all entries in $[-1, 1]$ (since all B_{ij} are products of cosine and sine functions) and unit diagonal.

The following is the procedure for effectively doctoring neural network outputs into valid parameterization of a multivariate normal distribution: using a hyperbolic tangent activation function on the output layer, the first network outputs the mean \mathbf{m} in $[-1, 1]^d$. Then, using a sigmoid activation function on the output layer, the second network outputs the standard deviations $\boldsymbol{\sigma}$ in $[0, 1]^d$. Finally, the third network uses a sigmoid activation function on the output layer to produce a set of coefficients $\boldsymbol{\rho}$ in $[0, 1]^D$. These are then mapped into correlative angles $\boldsymbol{\varphi} = \pi \boldsymbol{\rho}$ and assembled into the above elementary matrix \mathbf{B}_d , after which the covariance matrix is built as:

$$\mathbf{C} = \mathbf{S} (\mathbf{B}\mathbf{B}^t) \mathbf{S}, \quad (2.24)$$

with $\mathbf{S} = \text{diag}(\boldsymbol{\sigma})$. The complete PBO pseudo-code is given below.

Algorithm 2 PBO Algorithm [70]

```

Initialize  $\pi_\theta$ ,  $n_i$  parallel environments
for  $g = 0, n_g^{\max} - 1$  do
  Sample  $n_i$  actions/individuals  $a_i \in [-1, 1]^d$  from  $\pi_\theta$ 
  for  $i = 0, n_i - 1$  do ▷ This loop is executed in parallel
    Provide action  $a_i$  to environment  $i$ 
    Retrieve reward  $r_i$  from environment  $i$  ▷ End of "single-step" episode
  Compute clipped normalized reward  $\hat{R}_g$  ▷ Modified reward vector for generation  $g$ 
  for  $\sigma$ ,  $\rho$  and  $\mu$  networks do
    for  $e = 0, n_e - 1$  do ▷  $n_e$  can be specific to each network
      Shuffle data from most recent  $n_g$  generations ▷  $n_g$  can be specific to each network
      for  $b = 0, n_b - 1$  do ▷  $n_b$  can be specific to each network
        Generate mini-batch  $b$  from shuffled data
        Update current network with loss (2.18) ▷  $\lambda$  can be specific to each network

```

2.5 Conclusion

This chapter has introduced the basic concepts of Deep Reinforcement Learning, with a brief review of the popular value-based and policy-based methods used so far in fluid mechanics. The algorithms selected for this thesis have then been presented in further details, including the Proximal Policy Optimization (PPO) algorithm that will be used in chapter 7 for closed-loop control, its tweaked, single-step version (PPO-1) that will be used in chapters 4-5 for open-loop control, and the more sophisticated single-step algorithm called Policy Based Optimization (PBO), used in chapter 6 for shape optimization.

Chapter 3

Methodology - Computational Fluid Dynamics

Contents

3.1	Introduction	37
3.2	State of the art	37
3.3	Flow Finite Element Solver	38
3.3.1	The incompressible Navier–Stokes equations	38
3.3.2	The Variational Multiscale approach (VMS)	39
3.4	Turbulence modeling	43
3.4.1	Reynolds Averaged Navier–Stokes (RANS)	44
3.4.2	The Spalart-Allmaras SA turbulence model	44
3.5	Immersed volume method	47
3.5.1	The Level-Set Method	47
3.5.2	Anisotropic mesh adaptation	48
3.5.3	Mixing Laws	49
3.6	Conclusion	50

Ce chapitre décrit les méthodes de calcul numérique de la dynamique des fluides CFD utilisées pour calculer la récompense fournie à l’agent de DRL. Cela comprend les formulations d’éléments finis variationnels multi-échelles stabilisées par VMS utilisées pour résoudre les équations de Navier-Stokes et de Navier-Stokes moyennées de Reynolds RANS, ainsi que la méthode des volumes immergés couplée à l’adaptation du maillage de la couche limite anisotrope, utilisée pour résoudre avec précision un ensemble unique d’équations relatives aux phases solide et fluide sur un domaine de calcul unique.

This chapter describes the computational fluid dynamics CFD methods used to compute the numerical reward fed to the the DRL agent. This includes the variational multiscale VMS-stabilized finite element formulations used to solve the Navier-Stokes and Reynolds-Averaged Navier-Stokes RANS equations, and the immersed volume method coupled to anisotropic boundary layer mesh adaptation, used to solve accurately a single set equations pertaining to both solid and fluid phases on a unique computational domain.

3.1 Introduction

While the breakthrough in Computational Fluid Dynamics (CFD) has first been made in the context of the finite difference and the finite volume methods, Finite Element (FE) methods have grown significantly over the past decades, and are now widely used to accurately and efficiently solve the Navier–Stokes and the Convection-Diffusion-Reaction (CDR) equations while handling complex geometries relevant for a wide range of engineering applications. This chapter introduces the Variational Multiscale method (VMS) developed at CEMEF by the CFL team for solving such CFD problems in the context of FE methods.

3.2 State of the art

The incompressible Navier–Stokes equations are used to model many important physical phenomena, such as turbulent flow around airfoils or in cooling ducts, arterial blood flow. Nonetheless, the classical Galerkin formulation has been found to lack robustness to model complex flows dominated by convection. Indeed, the stability of the discrete formulation depends on the choice of compatible Finite Element spaces for the velocity and the pressure (this is known as the Babuska-Brezzi inf-sup condition). In particular, using elements with continuous equal order linear/linear interpolation, albeit efficient from a CPU cost perspective, does not comply with the inf-sup condition, and is known to yield an unstable discretization that manifests in uncontrollable oscillations that pollute the solution.

The need to bypass the limitations of the Galerkin approach has led a large body of research [73–75]. Among the many different approaches that have emerged in the literature, we consider here the Stabilized Finite Element Method (SFEM) and the Variational Multiscale method (VMS), two closely connected methods that both add weighted residual terms to the classical weak formulation of the problem to improve stability while maintaining consistency, the SFEM by modifying directly the variational formulation, and the VMS by modifying the Finite Element basis.

A survey of residual based stabilization methods, as summarized in the book by Donea and Huerta [76] is as follows: the ground work for PSPG (Pressure Stabilized Petrov Galerkin) methods and multiscale methods has been laid almost simultaneously in the 90s by Tezduyar [77] and Hugues *et al.* [78], respectively. Since then, fresh developments by Codina and co-workers [79, 80] based on orthogonal subscales and time-dependent subscales have paved the way for an application to turbulence modeling, as evidenced by the three scale separation method presented in [81–83].

3.3 Flow Finite Element Solver

Computing numerical flow solutions requires solving the time-dependent Navier–Stokes equations by Direct Numerical Simulation (DNS) [84], with very fine mesh resolutions and adequate time steps being mandatory in the context of turbulent flows. In return, the required computing resources may not always be affordable especially when simulating complex industrial processes. This is the main reason why most engineering computations involving turbulent flows rely on a certain degree of turbulence modeling, with a wide range of models varying in sophistication and ease of implementation having been developed and analyzed in the last few decades. Those are classified here into three different categories, VMS, Large eddy simulation (LES), and Reynolds-Averaged Navier–Stokes (RANS). As one progresses from DNS (no modelling) to RANS, an increasing part of the turbulent motion is lumped into the model, which decreases the computational resources needed to perform the computations. This is especially convenient when dealing with industrial application since the user may decide which method is most relevant for the considered application, being understood that each method comes with its own trade-off between loss of accuracy and gain in computational costs. The Finite Element implementation of the Navier–Stokes and Convection-Diffusion-Reaction equations (ubiquitous in most turbulent models) will be briefly described and analyzed in the following sections.

3.3.1 The incompressible Navier–Stokes equations

Let $\Omega \subset \mathbf{R}^n$ be the n -dimension spatial domain and Δ be the boundary of Ω . Consider unsteady flows governed by the incompressible Navier–Stokes equations in a time interval $[0, T]$:

$$\rho(\partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u}) + \nabla p - 2\mu \nabla \cdot \boldsymbol{\epsilon}(\mathbf{u}) = \mathbf{f} \quad \text{in } \Omega \times [0, T], \quad (3.1)$$

$$\nabla \cdot \mathbf{u} = 0 \quad \text{in } \Omega \times [0, T], \quad (3.2)$$

with ρ the density, μ is the dynamic viscosity, \mathbf{u} the velocity, p the pressure, \mathbf{f} a body force vector, and $\boldsymbol{\epsilon}(\mathbf{u})$ the strain-rate tensor defined as:

$$\boldsymbol{\epsilon}(\mathbf{u}) = \frac{1}{2}(\nabla \mathbf{u} + \nabla \mathbf{u}^T). \quad (3.3)$$

A divergence-free velocity field $\mathbf{u}_0(x)$ is imposed over Ω at $t = 0$, while the Dirichlet and natural boundary conditions are:

$$\mathbf{u}(x, 0) = \mathbf{u}_0(x), \quad (3.4)$$

$$\mathbf{u} = \mathbf{g} \quad \text{on } \Gamma_g \times [0, T], \quad (3.5)$$

$$-p\mathbf{n} + 2\mu \nabla \cdot \boldsymbol{\epsilon}(\mathbf{u}) \cdot \mathbf{n} = \mathbf{0} \quad \text{on } \Gamma_h \times [0, T], \quad (3.6)$$

where Γ_g and Γ_h are two complementary subsets of Γ and \mathbf{n} is the unit outward normal vector of Γ .

The function spaces for the velocity and the pressure are respectively defined by:

$$V = \{\mathbf{u}(x, t) | \mathbf{u}(x, t) \in H^1(\Omega)^n, \mathbf{u} = \mathbf{g} \text{ on } \Gamma_g\}, \quad (3.7)$$

$$P = \{p(x, t) | p(x, t) \in L^2(\Omega)\}, \quad (3.8)$$

while the test function space for the velocity is:

$$V_0 = \{\mathbf{u}(x, t) | \mathbf{u}(x, t) \in H^1(\Omega)^n, \mathbf{u} = \mathbf{0} \text{ on } \Gamma_g\}. \quad (3.9)$$

The weak form of the Navier–Stokes equations system (3.1-3.2) consists in finding $\mathbf{u} \in V$ and $p \in P$ such that:

$$\begin{aligned} (\rho \partial_t \mathbf{u}, \mathbf{w})_\Omega + (\rho \mathbf{u} \cdot \nabla \mathbf{u}, \mathbf{w})_\Omega \\ + (2\mu \boldsymbol{\epsilon}(\mathbf{u}) : \boldsymbol{\epsilon}(\mathbf{w}))_\Omega - (p, \nabla \cdot \mathbf{w})_\Omega = (\mathbf{f}, \mathbf{w})_\Omega \quad \forall \mathbf{w} \in V_0, \end{aligned} \quad (3.10)$$

$$(\nabla \cdot \mathbf{u}, q)_\Omega = 0 \quad \forall q \in P, \quad (3.11)$$

where we note $(a, b)_\Omega$ the canonical scalar product in $L^2(\Omega)$.

In practice, Ω is decomposed into N elements K covering the entire computational domain. They are either disjoint or share a complete edge (in 2D, a complete face in 3D). Using this partition \mathcal{K}_h , the function spaces defined in (3.7) and (3.8) are approached by finite-dimensional spaces spanned by continuous piecewise polynomials such that:

$$V_h = \{\mathbf{u}_h | \mathbf{u}_h \in C^0(\Omega)^n, \mathbf{u}_h|_K \in \mathcal{P}_h^n, \forall K \in \mathcal{K}_h\}, \quad (3.12)$$

$$P_h = \{p_h | p_h \in C^0(\Omega), p_h|_K \in \mathcal{P}_h, \forall K \in \mathcal{K}_h\}, \quad (3.13)$$

and the Galerkin discrete problem consists thus in finding a pair \mathbf{u}_h and p_h to solve the followed mixed problem such that $\forall (\mathbf{w}_h, q_h) \in V_{h,0} \times P_h$:

$$\begin{aligned} (\rho \partial_t \mathbf{u}_h, \mathbf{w}_h)_\Omega + (\rho \mathbf{u}_h \cdot \nabla \mathbf{u}_h, \mathbf{w}_h)_\Omega \\ + (2\mu \boldsymbol{\epsilon}(\mathbf{u}_h) : \boldsymbol{\epsilon}(\mathbf{w}_h))_\Omega - (p_h, \nabla \cdot \mathbf{w}_h)_\Omega = (\mathbf{f}, \mathbf{w}_h)_\Omega, \end{aligned} \quad (3.14)$$

$$(\nabla \cdot \mathbf{u}_h, q_h)_\Omega = 0. \quad (3.15)$$

3.3.2 The Variational Multiscale approach (VMS)

In order to solve the above variational formulation of the Navier–Stokes equations, Hughes [78] introduced the Variational Multiscale, that aims at numerically resolving the largest flow scales at play, while modeling the effects of the finest scales. First, all unknowns are split into coarse (large) and fine (small) components corresponding to different scales of resolution. Then, the fine scales are solved in an approximate manner and their effect on the large scale is modeled via appropriate residual-based terms.

3.3.2.1 Variational formulation

The solution spaces are split into $V_h \oplus V'$ and $P_h \oplus P'$, where the h subscript now denotes the coarse Finite Element component (solved), and the prime superscript stands for the fine, so-called subgrid-scale component (approximated). The scale decomposition reads:

$$\mathbf{u} = \mathbf{u}_h + \mathbf{u}' \in V_h \oplus V' \quad (3.16)$$

$$p = p_h + p' \in P_h \oplus P' \quad (3.17)$$

for the solution and:

$$\mathbf{w} = \mathbf{w}_h + \mathbf{w}' \in V_{h,0} \oplus V'_0 \quad (3.18)$$

$$q = q_h + q' \in P_{h,0} \oplus P'_0 \quad (3.19)$$

for the test functions. The weak formulation of (3.14-3.15) is then:

$$\begin{aligned} & (\rho \partial_t (\mathbf{u}_h + \mathbf{u}'), \mathbf{w}_h + \mathbf{w}') + (\rho (\mathbf{u}_h + \mathbf{u}') \cdot \nabla (\mathbf{u}_h + \mathbf{u}'), \mathbf{w}_h + \mathbf{w}') \\ & + (2\mu \boldsymbol{\epsilon}(\mathbf{u}_h + \mathbf{u}'), \boldsymbol{\epsilon}(\mathbf{w}_h + \mathbf{w}')) - (p_h + p', \nabla \cdot (\mathbf{w}_h + \mathbf{w}')) = (\mathbf{f}, \mathbf{w}_h + \mathbf{w}') \end{aligned} \quad (3.20)$$

$$(\nabla \cdot (\mathbf{u}_h + \mathbf{u}'), q_h + q') = 0 \quad (3.21)$$

Additional simplifications are necessary, that follow from the following classical approximations:

- Quasi-static subscales are considered here (no time tracking), meaning that the time-dependent term in the subscale equations can be omitted.
- Convection is assumed to occur at the large-scale velocity, meaning that the non-linear term can be approximated as

$$(\mathbf{u}_h + \mathbf{u}') \cdot \nabla (\mathbf{u}_h + \mathbf{u}') \approx \mathbf{u}_h \cdot \nabla (\mathbf{u}_h + \mathbf{u}'). \quad (3.22)$$

- Terms with subscales can be integrated by parts, and the subscales will be neglected on the element boundaries.

The coarse scales equations are obtained by setting the subscale test functions to zero in (3.20-3.21). Moreover, it is worth mentioning that because only linear elements are used in this work, all terms involving second derivatives can be neglected. Ultimately, we obtain:

$$\begin{aligned} & (\rho \partial_t \mathbf{u}_h, \mathbf{w}_h) + (\rho \mathbf{u}_h \cdot \nabla \mathbf{u}_h, \mathbf{w}_h) \\ & + (2\mu \boldsymbol{\epsilon}(\mathbf{u}_h), \boldsymbol{\epsilon}(\mathbf{w}_h)) - (p_h + p', \nabla \cdot \mathbf{w}_h) + \sum_K (\mathbf{u}', -\rho \mathbf{u}_h \cdot \nabla \mathbf{w}_h)_K = (\mathbf{f}, \mathbf{w}_h) \end{aligned} \quad (3.23)$$

$$(\nabla \cdot \mathbf{u}_h, q_h) - \sum_K (\mathbf{u}', \nabla q_h)_K = 0 \quad (3.24)$$

where \sum_K denotes the sum over all the elements of the Finite Element partition \mathcal{K}_h and $(\cdot, \cdot)_K$ is the L^2 product on element K .

The fine scales equations are obtained setting the coarse scales test functions to zero in (3.20-3.21) and using the three approximations introduced above. Based on the same analysis as in [85, 86], the subscales are expressed in each element $K \in \mathcal{K}_h$ as:

$$\mathbf{u}' = \tau_v \Pi'_v(\mathbf{R}_v) \quad p' = \tau_p \Pi'_p(R_p), \quad (3.25)$$

where Π'_v and Π'_p are the projections onto V' and P' , respectively, and \mathbf{R}_v and R_p are Finite Element residuals defined as

$$\mathbf{R}_v = \mathbf{f} - \rho \partial_t \mathbf{u}_h - \rho \mathbf{u}_h \cdot \nabla \mathbf{u}_h - \nabla p_h + \nabla \cdot (2\mu \boldsymbol{\epsilon}(\mathbf{u}_h)), \quad (3.26)$$

$$R_p = -\nabla \cdot \mathbf{u}_h. \quad (3.27)$$

Finally, τ_v and τ_p are stabilization parameters computed within each element as:

$$\tau_v = \left[\left(\frac{c_1 \mu}{\rho h_K^2} \right)^2 + \left(\frac{c_2 \|\mathbf{u}_h\|_K}{h_K} \right)^2 \right]^{-1/2}, \quad (3.28)$$

$$\tau_p = \left[\left(\frac{\mu}{\rho} \right)^2 + \left(\frac{c_2 \|\mathbf{u}_h\|_K h_K}{c_1} \right)^2 \right]^{-1/2}, \quad (3.29)$$

where c_1 and c_2 are two algorithmic constants fixed to 4 and 2 for linear elements (see [87]), h_K is the element size and $\|\mathbf{u}_h\|_K$ is a normed measure of \mathbf{u}_h in K . It is worth mentioning that τ_v often includes the time step size Δt of the temporal discretization as a mean to improve the algorithm convergence behavior when dealing with nonlinear problems, but this has several conceptual drawbacks further explained in [87, 88]. Therefore, as a mean to make τ_v more consistent over the computational domain and consequently enhance the behavior of the method, one may use:

$$\tau_v = \left[\frac{1}{\tau_0^2} + \left(\frac{c_1 \mu}{\rho h_K^2} \right)^2 + \left(\frac{c_2 \|\mathbf{u}_h\|_K}{h_K} \right)^2 \right]^{-1/2}, \quad (3.30)$$

where τ_0 is a τ_v reference value given by (3.28) and computed over the whole mesh, that should relate to the time discretization time step size.

Substituting for the subscales quantities in (3.23-3.24) yields

$$\begin{aligned}
& (\rho \partial_t \mathbf{u}_h, \mathbf{w}_h) + (\rho \mathbf{u}_h \cdot \nabla \mathbf{u}_h, \mathbf{w}_h) + (2\mu \boldsymbol{\epsilon}(\mathbf{u}_h), \boldsymbol{\epsilon}(\mathbf{w}_h)) - (p_h, \nabla \cdot \mathbf{w}_h) \\
& + \sum_K \tau_v (\rho \partial_t \mathbf{u}_h + \rho \mathbf{u}_h \cdot \nabla \mathbf{u}_h + \nabla p_h - \nabla \cdot (2\mu \boldsymbol{\epsilon}(\mathbf{u}_h)) - \mathbf{f}, \rho \mathbf{u}_h \cdot \nabla \mathbf{w}_h)_K, \\
& + \sum_K \tau_p (\nabla \cdot \mathbf{u}_h, \nabla \cdot \mathbf{w}_h) = (\mathbf{f}, \mathbf{w}_h)
\end{aligned} \tag{3.31}$$

$$(\nabla \cdot \mathbf{u}_h, q_h) - \sum_K \tau_v (\rho \partial_t \mathbf{u}_h + \rho \mathbf{u}_h \cdot \nabla \mathbf{u}_h + \nabla p_h - \nabla \cdot (2\mu \boldsymbol{\epsilon}(\mathbf{u}_h)) - \mathbf{f}, \nabla q_h)_K = 0. \tag{3.32}$$

Compared to the standard Galerkin approach, this formulation features additional integrals evaluated element wise, that represent the effect of the sub-grid scales on the coarse scale, and allow to overcome the instability in convection-dominated regimes. Using the stabilization parameters and the residuals defined above yields ultimately:

$$\begin{aligned}
& (\rho \partial_t \mathbf{u}_h, \mathbf{w}_h) + (\rho \mathbf{u}_h \cdot \nabla \mathbf{u}_h, \mathbf{w}_h) + (2\mu \boldsymbol{\epsilon}(\mathbf{u}_h), \boldsymbol{\epsilon}(\mathbf{w}_h)) - (p_h, \nabla \cdot \mathbf{w}_h) \\
& + \sum_K (\tau_v \mathbf{R}_v, \rho \mathbf{u}_h \cdot \nabla \mathbf{w}_h)_K + \sum_K (\tau_p R_p, \nabla \cdot \mathbf{w}_h) = (\mathbf{f}, \mathbf{w}_h),
\end{aligned} \tag{3.33}$$

$$(\nabla \cdot \mathbf{u}_h, q_h) - \sum_K (\tau_v \mathbf{R}_v, \nabla q_h)_K = 0. \tag{3.34}$$

3.3.2.2 Temporal discretization

For simplicity of implementation, the time and convective derivatives in equations (3.33-3.34) are discretized in time by a semi-implicit scheme

$$\partial_t \mathbf{u}_h + \mathbf{u}_h \cdot \nabla \mathbf{u}_h \approx \frac{\mathbf{u}_h^{n+1} - \mathbf{u}_h^n}{\Delta t} + \mathbf{u}_h^n \cdot \nabla \mathbf{u}_h^{n+1}, \tag{3.35}$$

where the n superscript refers to the solution at time $t_n = n\Delta t$. The divergence term in the continuity equation, together with the convective, viscous and pressure terms in the momentum equations, are integrated implicitly using a backward Euler scheme. All other terms are integrated explicitly using a forward Euler scheme. The complete discretized formulation reads

$$\begin{aligned}
& \frac{1}{\Delta t} (\rho (\mathbf{u}_h^{n+1} - \mathbf{u}_h^n), \mathbf{w}_h) + (\rho \mathbf{u}_h^n \cdot \nabla \mathbf{u}_h^{n+1}, \mathbf{w}_h) + (2\mu \boldsymbol{\epsilon}(\mathbf{u}_h^{n+1}), \boldsymbol{\epsilon}(\mathbf{w}_h)) - (p_h^{n+1}, \nabla \cdot \mathbf{w}_h) \\
& + \sum_K (\tau_v^n \mathbf{R}_v^{n+1}, \rho \mathbf{u}_h^n \cdot \nabla \mathbf{w}_h)_K + \sum_K (\tau_p^n R_p^{n+1}, \nabla \cdot \mathbf{w}_h) = (\mathbf{f}^n, \mathbf{w}_h),
\end{aligned} \tag{3.36}$$

$$(\nabla \cdot \mathbf{u}_h^{n+1}, q_h) - \sum_K (\tau_v^n \mathbf{R}_v^{n+1}, \nabla q_h)_K = 0, \tag{3.37}$$

with stabilization parameters:

$$\tau_v^n = \left[\left(\frac{c_1 \mu}{\rho h_K^2} \right)^2 + \left(\frac{c_2 \|\mathbf{u}_h^n\|_K}{h_K} \right)^2 \right]^{-1/2}, \quad (3.38)$$

$$\tau_p^n = \left[\left(\frac{\mu}{\rho} \right)^2 + \left(\frac{c_2 \|\mathbf{u}_h^n\|_K h_K}{c_1} \right)^2 \right]^{-1/2}, \quad (3.39)$$

and residuals:

$$\mathbf{R}_v^{n+1} = \frac{\rho}{\Delta t} (\mathbf{u}_h^{n+1} - \mathbf{u}_h^n) + \rho \mathbf{u}_h^n \cdot \nabla \mathbf{u}_h^{n+1} + \nabla p_h^{n+1} - \mathbf{f}^n, \quad (3.40)$$

$$R_p^{n+1} = \nabla \cdot \mathbf{u}_h^{n+1}. \quad (3.41)$$

The resulting linear system is generally preconditioned with a block Jacobi method supplemented by an incomplete LU factorization, and solved with the GMRES algorithm.

3.4 Turbulence modeling

Turbulence has long been considered an intriguing scientific problem ubiquitous in many flows such as the smoke of a cigarettes or a chimney, the ripples of a stone in water, the blood flow in an artery, etc. Turbulence occurs when fluid particles move in a random and irregular way, leading to the formation of eddies of different sizes. The largest eddies are unstable and break up into smaller eddies, that in turn break up into even smaller eddies until the eddies reach the Kolmogorov scale, below which they dissipate due to viscous effects. Turbulence has been the subject of much experimental, theoretical, and numerical research in the past decades, with particular emphasis put on developing cheap, yet reliable turbulence models.

In this work, the focus is on the so-called Reynolds Averaged Navier–Stokes (RANS) approach, in which the traditional Navier–Stokes equations are averaged, which brings out an unknown Reynolds stress tensor determined from a turbulence model. More specifically, we are interested in the one-equation Spalart–Allmaras (SA) model, that offers good accuracy and ease of implementation, and has become prominent in many fields and especially for aerodynamics flows in engineering applications. This section now explains how the VMS method can be used to perform RANS-SA numerical simulations by solving an appropriate, stabilized Convection-Diffusion-Reaction equation. Additional derivation and implementation details can be found in [89, 90].

3.4.1 Reynolds Averaged Navier–Stokes (RANS)

RANS models are most often used for industrial applications, for which the cost of turbulent scale-resolving approaches is prohibitive. While turbulent fluctuations arise in the most general case at various scales and with varying degrees of coherence, RANS assumes that there exist a separation of scales, between the large-scale motion associated with the flow unsteadiness, and the small-scale motion developing in the shear layers and responsible for the production of turbulence by amplification of the background noise. This allows to decompose the dynamics into a stochastic part, and a coherent part assembled by phase-averaging. For instance the velocity is written as

$$\mathbf{u} = \langle \mathbf{u} \rangle + \tilde{\mathbf{u}}, \quad (3.42)$$

where the brackets is the average, and the tilde superscript denotes a zero-average fluctuation, so $\langle \langle \mathbf{u} \rangle \rangle = \langle \mathbf{u} \rangle$ and $\langle \tilde{\mathbf{u}} \rangle = 0$. Another key property of the averaging operator is thus as follows:

$$\langle \mathbf{u} \cdot \mathbf{v} \rangle = \langle \mathbf{u} \rangle \cdot \langle \mathbf{v} \rangle + \langle \tilde{\mathbf{u}} \cdot \tilde{\mathbf{v}} \rangle. \quad (3.43)$$

Applying the averaging procedure to the Navier–Stokes equations therefore yields

$$\rho(\partial_t \langle \mathbf{u} \rangle + \langle \mathbf{u} \rangle \cdot \nabla \langle \mathbf{u} \rangle) + \nabla \langle p \rangle - 2\mu \nabla \cdot \boldsymbol{\epsilon}(\langle \mathbf{u} \rangle) + \rho \langle \tilde{\mathbf{u}} \cdot \nabla \tilde{\mathbf{u}} \rangle = \langle \mathbf{f} \rangle \quad \text{in } \Omega \times [0, T], \quad (3.44)$$

$$\nabla \cdot \langle \mathbf{u} \rangle = 0 \quad \text{in } \Omega \times [0, T]. \quad (3.45)$$

Of particular interest in these equations is the Reynolds stress tensor $\langle \tilde{\mathbf{u}} \cdot \nabla \tilde{\mathbf{u}} \rangle$, that shows up due to the nonlinearity of the stochastic motion. In the literature, many method have been proposed to compute this term for the above system of equations to be closed, often using the Boussinesq approximation [91] to lump this term into a turbulent eddy viscosity μ_t . By doing so, one obtain the RANS equations

$$\rho(\partial_t \langle \mathbf{u} \rangle + \langle \mathbf{u} \rangle \cdot \nabla \langle \mathbf{u} \rangle) + \nabla \langle p \rangle - 2(\mu + \mu_t) \nabla \cdot \boldsymbol{\epsilon}(\langle \mathbf{u} \rangle) = \langle \mathbf{f} \rangle \quad \text{in } \Omega \times [0, T], \quad (3.46)$$

$$\nabla \cdot \langle \mathbf{u} \rangle = 0 \quad \text{in } \Omega \times [0, T]. \quad (3.47)$$

governing the coherent velocity and pressure, that turn to be formally identical to the Navier–Stokes equations (3.1-3.2), except that the total viscosity $\mu + \mu_t$ is featured instead of the sole molecular viscosity.

3.4.2 The Spalart-Allmaras SA turbulence model

RANS methods are classified according to the number of transport equations needed to determine the turbulent viscosity, usually zero-equation models such as Baldwin–Lomax [92] or Cebeci–Smith [93], one-equation models such as Prandtl [94] or

Baldwin–Bart [95], and two-equations models such as k - ϵ [96] or k - ω [97]. The Spalart–Allmaras (SA) model [98] used in this work is a one-equation model, that has been found to be simpler, less computationally expensive, and more robust than multi-equation models. It requires solving one single non-linear transport equation to represent the evolution of a so-called SA working variable $\tilde{\nu}$

$$\begin{aligned} \partial_t \tilde{\nu} + \mathbf{u} \cdot \nabla \tilde{\nu} - c_{b1}(1 - f_{t2})\tilde{S}\tilde{\nu} + \left(c_{w1}f_{w1} - \frac{c_{b1}}{\kappa^2}f_{t2} \right) \left(\frac{\tilde{\nu}^2}{d} \right) \\ - \frac{c_{b2}}{\sigma} \nabla \tilde{\nu} \cdot \nabla \tilde{\nu} - \frac{1}{\sigma} \nabla \cdot [(\nu + \tilde{\nu}) \nabla \tilde{\nu}] = 0, \end{aligned} \quad (3.48)$$

where $\kappa = 0.4$ is the Von Karman constant, d is the shortest distance to the wall, the other model coefficients are:

$$\begin{aligned} f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3}, \quad \chi = \frac{\tilde{\nu}}{\nu}, \quad f_{v2} = 1 - \frac{\chi}{1 - \chi f_{v1}}, \quad f_{t2} = c_{t3}e^{-c_{t4}\chi^2}, \quad f_w = g \left[\frac{1 + x_{w3}^6}{g^6 + \chi_{w3}^6} \right]^{\frac{1}{6}}, \\ g = r + c_{w2}(r^6 - r), \quad r = \frac{\tilde{\nu}}{\tilde{S}\kappa^2 d^2}, \quad \tilde{S} = S + \frac{\tilde{\nu}}{\kappa^2 d^2} f_{v2}, \quad S = \sqrt{2\epsilon(\mathbf{u}) : \epsilon(\mathbf{u})}, \end{aligned}$$

with

$$\begin{aligned} c_{b1} = 0.1355, \quad c_{b2} = 0.622, \quad \sigma = \frac{2}{3}, \quad c_{v1} = 7.1, \quad c_{v2} = 0.7, \quad c_{v3} = 0.9, \\ c_{w1} = \frac{c_{b1}}{\kappa^2} + \frac{1 + c_{b2}}{\sigma}, \quad c_{w2} = 0.3, \quad c_{w3} = 2, \quad c_{t3} = 1.2, \quad c_t = 0.5, \end{aligned}$$

and the eddy viscosity ultimately deduces as $\mu_t = \rho \tilde{\nu} f_{v1}$.

Here, we use the negative Spalart–Allmaras model [99] meant to avoid the generation of negative turbulent viscosity without clipping (as this is detrimental to the quality of the solution and to the convergence behavior), that consists in replacing (3.48) when $\tilde{\nu}$ is negative (in which case the eddy viscosity is forced to $\mu_t = 0$) by:

$$\begin{aligned} \partial_t \tilde{\nu} + \mathbf{u} \cdot \nabla \tilde{\nu} - c_{b1}(1 - c_{t3})\tilde{S}\tilde{\nu} - c_{w1} \left(\frac{\tilde{\nu}}{d} \right)^2 \\ - \frac{c_{b2}}{\sigma} \nabla \tilde{\nu} \cdot \nabla \tilde{\nu} - \frac{1}{\sigma} \nabla \cdot [(\nu + f_n \tilde{\nu}) \nabla \tilde{\nu}] = 0, \end{aligned} \quad (3.49)$$

with

$$f_n = \frac{c_{n1} + \chi^3}{c_{n1} - \chi^3}, \quad (3.50)$$

and $c_{n1} = 16$.

In practice, we use a semi-implicit discretization of all non-linear terms to recast equations (3.48) and (3.49) into linear Convection-Diffusion-Reaction form

$$\frac{\tilde{\nu}^{n+1} - \tilde{\nu}^n}{\Delta t} + \mathbf{C}^n \cdot \nabla \tilde{\nu}^{n+1} - \nabla \cdot (D^n \nabla \tilde{\nu}^{n+1}) - R^n \tilde{\nu}^{n+1} = 0, \quad (3.51)$$

where \mathbf{C}^n , D^n and R^n are convection, diffusion, and reaction terms respectively, that can be evaluated from knowledge of the SA working variable ν^n at the previous time step, and of the current velocity \mathbf{u}^{n+1} calculated before solving the SA equation at the same time step. Namely, we obtain

$$\mathbf{C}^n = \mathbf{u}^{n+1} - \frac{c_{b2}}{\sigma} \nabla \tilde{\nu}^n, \quad (3.52)$$

$$D^n = \frac{\nu + \tilde{\nu}^n}{\sigma}, \quad (3.53)$$

$$R^n = c_{b1}(1 - f_{t2}^n) \tilde{S}^n - (c_{w1} f_w^n - \frac{c_{b1}}{k^2} f_{t2}^n) \frac{\tilde{\nu}^n}{d^2}, \quad (3.54)$$

for (3.48) and

$$\mathbf{C}^n = \mathbf{u}^{n+1} - \frac{c_{b2}}{\sigma} \nabla \tilde{\nu}^n, \quad (3.55)$$

$$D^n = \frac{\nu + f_n^n \tilde{\nu}^n}{\sigma}, \quad (3.56)$$

$$R^n = c_{b1}(1 - c_{t3}) \tilde{S}^n + c_{w1} \left(\frac{\tilde{\nu}^n}{d^2} \right), \quad (3.57)$$

for (3.49). The SUPG method is then used to discretize the CDR form (3.51), with stabilized weak form for the coarse scale following from [100] as:

$$\begin{aligned} & \left(\frac{\tilde{\nu}_h^{n+1} - \tilde{\nu}_h^n}{\Delta t}, s_h \right)_\Omega + (\mathbf{C}_h^n \cdot \nabla \tilde{\nu}_h^{n+1}, s_h)_\Omega - (D_h^n \nabla \tilde{\nu}_h^{n+1}, \nabla s_h)_\Omega - (R_h^n \tilde{\nu}_h^{n+1}, s_h)_\Omega \\ & + \sum_K (\tau_{\tilde{\nu}}^n R_{\tilde{\nu}}^{n+1}, \mathbf{C}_h^n \cdot \nabla s_h)_K = 0 \quad \forall s_h \in W_h, \end{aligned} \quad (3.58)$$

where $R_{\tilde{\nu}}^{n+1}$ is the residual of (3.51):

$$R_{\tilde{\nu}}^{n+1} = \frac{\tilde{\nu}_h^{n+1} - \tilde{\nu}_h^n}{\Delta t} + \mathbf{C}_h^n \cdot \nabla \tilde{\nu}_h^{n+1} - R_h^n \tilde{\nu}_h^{n+1}, \quad (3.59)$$

and the stabilization parameter $\tau_{\tilde{\nu}}^n$ is computed as:

$$\tau_{\tilde{\nu}}^n = \left(\frac{c_2}{h_K} \|\mathbf{C}^n\|_K + \frac{c_1}{h_K^2} D^n + R^n \right)^{-1}, \quad (3.60)$$

where $\|\mathbf{C}^n\|_K$ is a normed measure of the convection term. The linear system arising from Equation (3.58) is solved using the same numerical method as for the Navier–Stokes equations as described in the previous section.

3.5 Immersed volume method

Many engineering scenarios involve the detailed study of the interaction between a solid body and the flow in which it is immersed. In fluid mechanics, this is generally done using a so-called body-fitted meshes whose boundaries coincide with the interface of the solid body. Nonetheless, in the case of complex 3D geometries, a body-fitted mesh is very difficult to achieve, especially if one is to consider solid bodies in motion. One solution is to use a monolithic approach, i.e. a single computational mesh that encompasses the whole problem and in particular the solid body, that ends up being literally immersed in the domain where the fluid flows. The solid body is therefore discretized by mesh elements and no longer by boundaries. The immersed volume method used in this work developed in [101] consists in identifying the interface between the fluid and the solid body using a signed distance function, in order to be able to solve a single system of equations with variable mechanical properties (density, viscosity), depending on whether a given node lies in the fluid or in the solid domain. The following sections review the main ingredients involved, namely a level-set interface capturing method to localize the fluid/solid interface, anisotropic mesh adaptation to achieve a high-fidelity description of said interface, and relevant mixing laws to distribute composite mechanical properties on either side of the interface. Such a numerical framework is especially relevant for DRL problems in which the mesh depends on the sampled action (for instance if a neural network is tasked with optimizing the position of a solid body, as will be done here on several occasions), as it allows ensuring that all actions sampled by a DRL agent are assessed with the same numerical accuracy.

3.5.1 The Level-Set Method

In the monolithic approach, a single mesh is used to describe both the fluid flow and the solid body. The interface between these two sub-domains is identified using a signed distance function, called levelset. In the case of simple geometries for which an analytical function is known, the latter is used to define the solid body. In the opposite and more general case, the geometry is first created using CAD (Computer Aided Design) software, then the surface mesh (STL for stereolithography) is exported and immersed into the monolithic domain. The levelset function is then calculated either easily from the analytical function (when it is known), or according to a more complex algorithm from the immersed geometry itself [102], in a way such that the minimum distance to the interface between the fluid and the solid body is known at each node and the interface between the fluid and the solid body is straightforwardly described by the zero iso-value of the levelset function. In order to discriminate between the interior and the exterior of the solid body, this distance is signed, as it takes (by convention) positive values inside the solid, and negative

values outside. For instance, if we let Ω be the whole domain split into fluid Ω_f and solid Ω_s domain, and if we denote by $\Gamma = \Omega_s \cap \Omega_f$ the fluid/solid interface, the level set is defined at each node x of Ω as:

$$\alpha(x) = \begin{cases} -dist(\Gamma, x) & \text{if } x \in \Omega_f, \\ 0 & \text{if } x \text{ on } \Gamma, \\ dist(\Gamma, x) & \text{if } x \in \Omega_s. \end{cases} \quad (3.61)$$

3.5.2 Anisotropic mesh adaptation

Because the level-set function is merely an implicit representation of the solid/fluid interface, the interface can intersect the elements of the mesh arbitrarily it is not in line with the element edges, in which case the high ratio in the material properties across the interface can yield spurious oscillations of the numerical solutions. Mesh adaptation allows considering a regularized interface instead, meaning that the mechanical properties are distributed smoothly over a certain thickness around the interface. Here, we use anisotropic mesh adaptation (driven by physics-based criteria) with highly stretched elements along the interface, as a mean to reduce to the minimum said interface thickness (as it bears no physical relevance), which in turn ensures maximum accuracy of the numerical solutions. As a result, we are able to:

- Produce very good accuracy properties for high Reynolds number flows,
- Ensure accurate and oscillation free numerical solutions,
- Reduce the computational cost of assessing a single DRL action,
- Reliably assess any DRL action with the same accuracy.

We use ratios of anisotropy up to 1000:1, which requires an appropriate definition of the stabilization parameters using the directional element diameter, as the element size is featured in all stabilization parameter.

In short, the procedure requires building a metric, that is, a symmetric positive defined tensor containing the information regarding the principal direction in space and the size with respect to each direction. As an example, a simple metric, isotropic far from the interface, with mesh size set equal to h_∞ in all directions, but anisotropic near the interface, with mesh size equal to h_\perp in the direction normal to the interface, and to h_∞ in the other directions, can be written for an intended thickness ϵ as

$$\mathbf{M} = K(\alpha)\mathbf{n} \otimes \mathbf{n} + \frac{1}{h_\infty^2}\mathbf{I} \quad \text{with} \quad K(\alpha) = \begin{cases} 0 & \text{if } |\alpha| \geq \epsilon/2, \\ \frac{1}{h_\perp^2} - \frac{1}{h_\infty^2} & \text{if } |\alpha| < \epsilon/2, \end{cases} \quad (3.62)$$

where $\mathbf{n} = \nabla\alpha/||\nabla\alpha||$ is the unit normal to the fluid/solid interface computed from the level set gradient.

Exhaustive details regarding the mathematical background and the method implementation in the Cimlib-CFD library can be found in [103]. Suffice it to say here that it is an automatic procedure that relies on the a posteriori definition of the metric driving the re-meshing procedure by minimizing the interpolation error of the numerical solution under the constraint of a fixed number of edges in the mesh. It is thus well suited for dynamic mesh adaptation (meaning that the mesh can be adapted on the fly during the simulation) based on several fields of interest. For instance the velocity norm or scalar components, the temperature field or its gradients, the levelset, or any combination of those, can be used as multi-component adaptation criteria, as the procedure allows building a unique metric without having to intersect several individual metrics (which can trigger a substantial computational cost).

3.5.3 Mixing Laws

The problem to be solved is then modeled using a single system of Navier–Stokes equations, whose mechanical properties vary according to the subdomain in which a given node is located. Such an approach uses the smooth Heaviside function defined in [104], that depends on the sign of the levelset function according to:

$$H(\alpha) = \begin{cases} 1 & \text{if } \alpha(x) > \epsilon, \\ \frac{1}{2} \left(1 + \frac{\alpha}{\epsilon} + \frac{1}{\pi} \sin \left(\frac{\pi\alpha}{\epsilon} \right) \right) & \text{if } |\alpha(x)| \leq \epsilon, \\ 0 & \text{if } \alpha < -\epsilon. \end{cases} \quad (3.63)$$

In (3.63) ϵ is the intended smooth interface thickness set to $\epsilon = 2h_{im}$ where h_{im} is the mesh size in the normal direction computed on element K as

$$h_{im} = \max_{j,l \in K} \nabla\alpha \cdot (\mathbf{x}^l - \mathbf{x}^j). \quad (3.64)$$

For aerodynamics applications, solving a unique set of Navier–Stokes equations on a single computational domain requires mixing the density ρ and the viscosity μ , which is done using arithmetic means as follows:

$$\rho = \rho_f H(\alpha) + \rho_s (1 - H(\alpha)), \quad (3.65)$$

$$\mu = \mu_f H(\alpha) + \mu_s (1 - H(\alpha)), \quad (3.66)$$

where the subscripts f and s refer to the fluid and the solid body, respectively. For thermal problems, solving the coupled Navier–Stokes and heat equations requires

mixing additionally the specific heat capacity C_p and the thermal conductivity λ . This is done using

$$\rho c_p = \rho_f c_{pf} H(\alpha) + \rho_s c_{ps} (1 - H(\alpha)), \quad (3.67)$$

$$\frac{1}{\lambda} = \frac{1}{\lambda_f} H(\alpha) + \frac{1}{\lambda_s} (1 - H(\alpha)), \quad (3.68)$$

i.e., we ensure continuity of the heat flux across the interface using the harmonic mean of the conductivity, as obtained from a steady, no source, one dimensional analysis of the heat flux when the conductivity varies stepwise from one medium to the next [105].

3.6 Conclusion

This chapter has presented the main ingredients of the Cimlib-CFD finite elements library used to solve the incompressible Navier–Stokes equations using the Variational Multiscale stabilization method. The extension to the Reynolds-Averaged Navier–Stokes approach with Spalart-Allmaras turbulence model as closure is also provided for which the stabilization proceeds from that of the Convection Diffusion Reaction equation. Finally, we present the monolithic Immersed Volume Method used to solve a single set of equations on a unique computational domain encompassing generic solid bodies immersed in a fluid domain, and the anisotropic mesh adaptation procedure used to achieve an accurate description of the solid/fluid interfaces. A numerical application of this numerical framework is provided in figure 3.1 where we present several snapshot of the flow past the fluidic pinball, an equilateral triangle arrangement of spanwise infinite cylinders immersed in a two-dimensional, uniform flow (further studied in chapter 4) together with the corresponding meshes adapted using several variables all at once. The left row is the result of an a-priori adaptation procedure using the level-set of the fluid/solid interface plus an additional level-set (corresponding to a fictitious inner interface) allowing to achieve progressive successive refinement of the background elements. The right row is the result of a dynamic adaptation procedure (*i.e.*, the mesh is adapted over the course of the simulation so as to keep the number of nodes of the a-priori mesh constant) using the level-set of the fluid/solid interface and the two components of the velocity vector. The obtained results stress that in adaptive mode, all boundary layers are sharply captured via extremely stretched elements, with the adaptation strategy yielding refined meshes along the boundary layers but more generally in all shear regions, while the elements in-between are coarser and essentially isotropic.

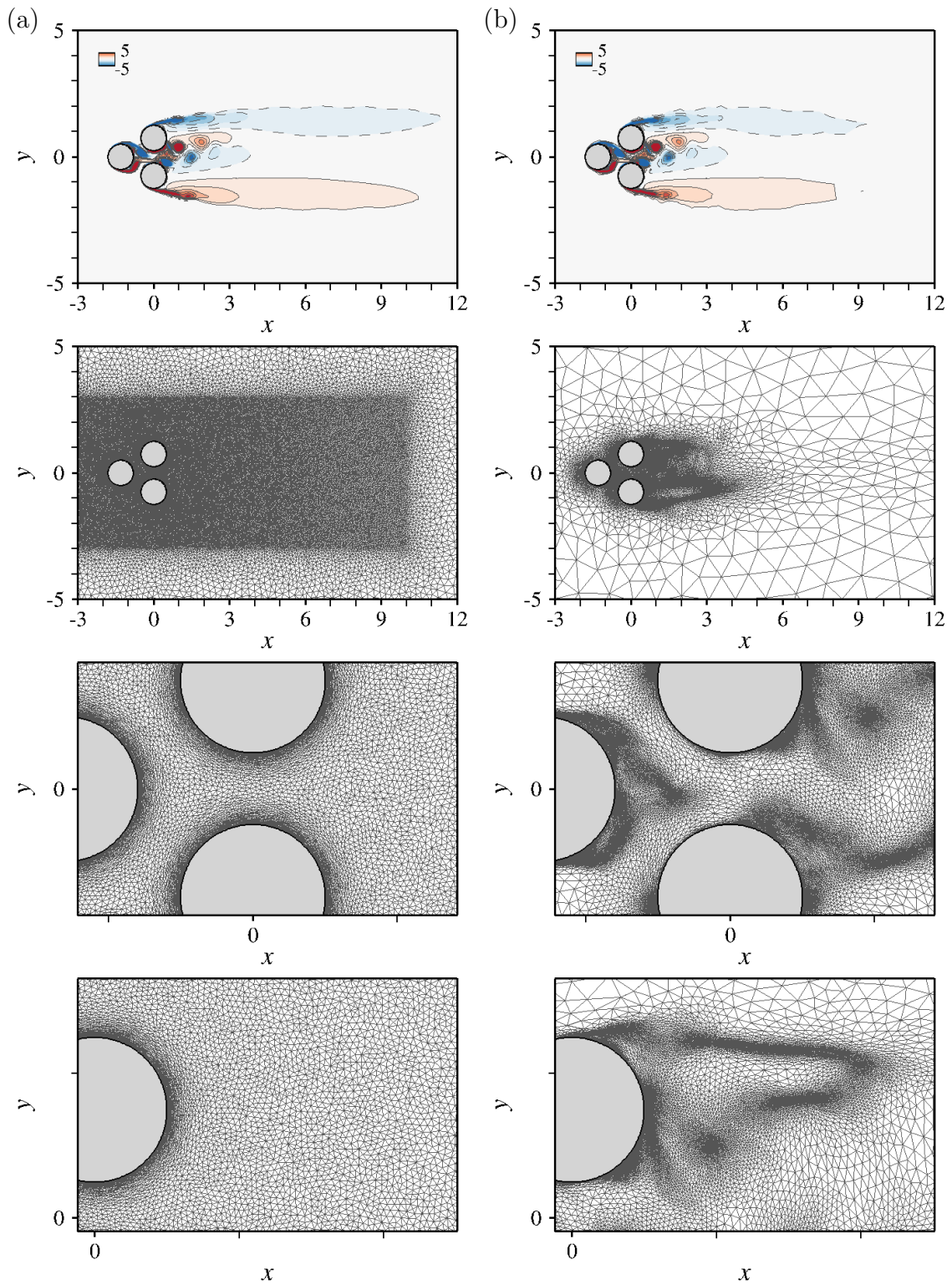


Figure 3.1: Results of the anisotropic mesh adaptation for the fluidic pinball test case. Results on the left column pertain to a mesh adapted prior to simulation, made up of 120000 elements. Results on the right column pertain to a mesh adapted over the course of simulation under the constraint of an identical number of nodes.

Chapter 4

Single-step DRL for open-loop control of laminar and turbulent flows

Contents

4.1	Introduction	56
4.2	Methodology	58
4.2.1	Deep reinforcement learning	58
4.2.2	Proximal policy optimization	58
4.2.3	Single-step PPO	60
4.2.4	Computational fluid dynamics environment	61
4.2.5	Numerical implementation	63
4.3	Application to flow optimization	65
4.3.1	Flow past a NACA 0012 airfoil	65
4.3.2	Flow past an arrangement of two side-by-side circular cylinders	67
4.4	Application to open-loop flow control	72
4.4.1	Optimal cylinder drag reduction using a smaller control cylinder	72
4.4.2	Optimal drag reduction of a triangular bluff-body using rotating cylinders	83
4.5	Discussion	90
4.5.1	Adjoint methods	90
4.5.2	Evolution strategies	92
4.6	Conclusion	93

This chapter is presented as a self-contained article published in 2021 in Physical Review Fluids [69], hence, some details (related to motivation, scope and/or methodology) are repeated from chapters 1-3 to preserve the consistency of the whole content.

Ce chapitre évalue la capacité des techniques d'apprentissage par renforcement profond (DRL) à aider à l'optimisation et au contrôle des systèmes de mécanique des fluides. Il s'appuie sur le PPO en une seule étape, une nouvelle version "dégénérée" de l'algorithme Proximal Policy Optimization (PPO), destinée aux situations où la politique optimale à apprendre par un réseau neuronal ne dépend pas de l'état, comme c'est notamment le cas dans les problèmes de contrôle en boucle ouverte. La récompense numérique fournie au réseau neuronal est calculée à l'aide d'un environnement interne d'éléments finis stabilisés mettant en œuvre la méthode variationnelle multi-échelle (VMS). Plusieurs écoulements en deux dimensions sont utilisés comme banc d'essai. La méthode est d'abord appliquée à deux cas d'optimisation relativement simples (maximisation de la portance moyenne d'un profil d'aile NACA 0012 et de la portance instantanée de deux cylindres circulaires côte à côte, tous deux en régime laminaire) pour évaluer la convergence et la précision en les comparant aux données DNS internes. Le potentiel de la PPO en une seule étape pour l'optimisation fiable en boîte noire de systèmes de dynamique des fluides numérique (CFD) est ensuite démontré en abordant plusieurs problèmes de contrôle en boucle ouverte avec des espaces de paramètres suffisamment grands pour écarter le DNS. L'approche s'avère pertinente pour cartographier les meilleures positions pour le placement d'un petit cylindre de contrôle dans le but de réduire la traînée dans des écoulements cylindriques laminaires et turbulents. Tous les résultats sont cohérents avec les données internes obtenues par la méthode adjointe, et la traînée d'un cylindre carré à des nombres de Reynolds de l'ordre de quelques milliers est réduite de 30%, ce qui correspond bien aux données expérimentales de référence disponibles dans la littérature. La méthode réduit également avec succès la traînée du fluidic pinball, un arrangement en triangle équilatéral de cylindres rotatifs immergés dans un écoulement turbulent. Conformément aux résultats de l'apprentissage automatique de référence tirés de la littérature, la traînée est réduite de près de 60% en utilisant un actionnement composé d'un cylindre amont tournant lentement et de deux cylindres en aval tournant dans des directions opposées de manière à réduire l'écart d'écoulement entre eux.

This chapter gauges the ability of deep reinforcement learning (DRL) techniques to assist the optimization and control of fluid mechanical systems. It relies on single-step PPO, a novel, "degenerate" version of the proximal policy optimization (PPO) algorithm, intended for situations where the optimal policy to be learnt by a neural network does not depend on state, as is notably the case in open-loop control problems. The numerical reward fed to the neural network is computed with an in-

house stabilized finite elements environment implementing the variational multiscale (VMS) method. Several prototypical separated flows in two dimensions are used as testbed. The method is applied first to two relatively simple optimization test cases (maximizing the mean lift of a NACA 0012 airfoil and the fluctuating lift of two side-by-side circular cylinders, both in laminar regimes) to assess convergence and accuracy by comparing to in-house DNS data. The potential of single-step PPO for reliable black-box optimization of computational fluid dynamics (CFD) systems is then showcased by tackling several problems of open-loop control with parameter spaces large enough to dismiss DNS. The approach proves relevant to map the best positions for placement of a small control cylinder in the attempt to reduce drag in laminar and turbulent cylinder flows. All results are consistent with in-house data obtained by the adjoint method, and the drag of a square cylinder at Reynolds numbers in the range of a few thousands is reduced by 30%, which matches well reference experimental data available from literature. The method also successfully reduces the drag of the fluidic pinball, an equilateral triangle arrangement of rotating cylinders immersed in a turbulent stream. Consistently with reference machine learning results from the literature, drag is reduced by almost 60% using a so-called boat tailing actuation made up of a slowly rotating front cylinder and two downstream cylinders rotating in opposite directions so as to reduce the gap flow in between them.

4.1 Introduction

Flow control, defined as the ability to finesse a flow into a more desired state, is a field of tremendous societal and economical importance. In applications such as ocean shipping or airline traffic, reducing the overall drag by just a few percent while maintaining lift can help reducing fossil fuel consumption and CO₂ emission while saving several billion dollars annually [23]. Many other scenario relevant to fluid mechanical systems call for similarly improved engineering design, e.g., the airline industry is greatly concerned with reducing the structural vibrations and the radiated noise that occur under unsteady flow conditions [106, 107], while microfluidics [108] and combustion [109] both benefit from enhanced mixing (which can be achieved by promoting unsteadiness in some appropriate manner). All such problems fall under the purview of this line of study.

Flow control is often benchmarked in the context of bluff body drag reduction. Numerous strategies have been implemented, either open-loop with passive appendices (e.g., end/splitter plates, small secondary cylinder, flexible tail), or open-loop with actuating devices (e.g., plasma actuation, base bleed, rotation) or closed-loop (e.g. transverse motion, blowing/suction, rotation, all relying on appropriate sensing of flow variables); see the comprehensive surveys of recent developments in [1, 2, 27, 110–115]. Nonetheless, most strategies are trial and error and rely on extensive, costly experimental or numerical campaigns, which has motivated the development of rigorous mathematical formalisms capable of achieving optimal design and control with minimal effort. The adjoint method is one family of such algorithms, that has proven efficient at accurately computing the objective gradient with respect to the control variables in large optimization spaces, and has gained prominence in many applications ranging from atmospheric sciences [29] to aerodynamic design [30, 116–118], by way of fresh developments meant to reshape the linear amplification of flow disturbances [119–124].

Another promising option for selecting optimal subsets of control parameters is to rely on machine learning algorithms running labeled data through several layers of artificial neural network while providing some form of corrective feedback. Neural networks are a family of versatile parametric tools that can learn how to hierarchically extract informative features from data, and have gained traction as effective and efficient computational processors for performing a variety of tasks, from exploratory data analysis to qualitative and quantitative predictive modeling. The increased affordability of high performance hardware (together with reduced costs for data acquisition and storage) has indeed allowed leveraging the ever-increasing volume of data generated for research and engineering purposes into novel insight and actionable information, which in turn has reshaped entire scientific disciplines such as image analysis [125] or robotics [64, 126]. Since neural networks have produced most remarkable results when applied to stiff large-scale nonlinear problems [127],

it is only natural to assume that they can successfully tackle the state-space models arising from the high-dimensional discretization of partial differential equation systems. Machine learning has thus been making rapid inroads in fluid mechanics, with consistent efforts aimed at solving the governing equations [128], predicting closure terms in turbulence models [129], building reduced-order models [130], controlling flows [131, 132], or performing flow measurements and visualization [133–135]; see also [136] for an overview of the current developments in this field.

The focus here is on deep reinforcement learning (DRL), an advanced branch of machine learning in which deep neural networks learn how to behave in an environment so as to maximize some notion of long-term reward (a task compounded by the fact that each action affects both immediate and future rewards). Several notable works using DRL in mastering games (e.g., Go, Poker) have stood out for attaining super-human level [48, 56], but the approach has also breakthrough potential for practical applications such as robotics [18, 57], computer vision [59], self-driving cars [60] or finance [58], to name a few. There is also great potential for applying DRL to fluid mechanics, for which efforts are ongoing but still at an early stage, with only a handful of pioneering studies providing insight into the performance improvements to be delivered in shape optimization [21, 137, 138] and flow control [139–141]. Nonetheless, sustained commitment from the machine learning community has allowed expanding the scope from computationally inexpensive, low-dimensional reductions of the underlying fluid dynamics [12–14] to complex Navier–Stokes systems [37, 38]. Proximal policy optimization (PPO [18]) has quickly gained momentum as one of the go-to algorithms for this purpose, as evidenced by several recent publications assessing relevance for open- and closed-loop drag reduction in cylinder flows at Reynolds numbers in the range of a few hundreds [39, 41, 43, 49].

This research draws on this foundation to further shape the capabilities of PPO (still a newcomer despite its data efficiency, simplicity of implementation and reliable performance) for flow control, and help narrow the gap between DRL and advanced numerical methods for multiscale, multi-physics computational fluid dynamics (CFD). The main novelty is the use of single-step PPO, a novel “degenerate” algorithm intended for open-loop control problems, as the optimal policy to be learnt is then state-independent, and it may be enough for the neural network to get only one attempt per episode at finding the optimal. The objective is twofold: first, to prove feasibility using several prototypical separated flows in two dimensions as testbed. Second, to assess convergence and relevance in the context of turbulent flows at moderately large Reynolds number (in the range of a few thousands). This is a topic whose surface is barely scratched by the available literature, as our literature review did not reveal any other study considering DRL-based control of turbulent flows besides [142], another research effort conducted in the same time frame as the present work. Single-step PPO has been speculated to hold a high po-

tential as a reliable black-box CFD optimizer [138], but we insist that it lies out of the scope of this chapter to provide exhaustive performance comparison data against state-of-the-art optimization techniques (e.g., evolution strategies or genetic algorithms). This would indeed require a tremendous amount of time and resources even though the efforts for developing the method remain at an early stage (to the best of our knowledge, no study in the literature has considered using DRL in a similar fashion) and new algorithms cannot be expected to reach right away the level of performance of their more established counterparts.

4.2 Methodology

4.2.1 Deep reinforcement learning

Reinforcement learning (RL) provides a consistent framework for modeling and solving decision-making problems through repeated interaction between an agent and an environment. We consider the standard formulation in which the agent takes an action a_t based on a partial observation of the current state s_t the environment is in. The environment transits to the next state s_{t+1} , and the agent is fed with a reward r_t that acts as the quality assessment of the actions recently taken. This repeats until some termination state is reached, the objective of the agent being to determine the succession of actions maximizing its cumulative reward over an episode (this is the reference unit for agent update, best understood as one instance of the scenario in which it takes actions). Deep reinforcement learning (DRL) combines RL and deep neural networks, i.e., collections of connected units or artificial neurons, that can be trained to arbitrarily well approximate the mapping function between input and output spaces. We consider here fully connected networks in which neurons are stacked in layers and information propagates forward from the input layer to the output layer via “hidden” layers. Each neuron performs a weighted sum of its inputs to assign significance with regard to the task the algorithm is trying to learn, adds a bias to figure out the part of the output independent of the input, and feeds an activation function that determines whether and to what extent the computed value should affect the outcome.

4.2.2 Proximal policy optimization

Proximal policy optimization (PPO) [18] is a model free, on-policy gradient, advantage actor-critic reinforcement algorithm. The related key concepts can be summarized as follows:

- **model free:** the agent interacts with the environment itself, not with a surrogate model of the environment (the corollary here being that it needs no assumptions

about the fluid dynamics underlying the control problems to be solved).

- **policy gradient:** the behavior of the agent is entirely defined by a probability distribution $\pi(s, a)$ over actions given states, optimized by gradient ascent. In DRL, the policy is represented by a neural network. The free parameters learnt from data are the network weights and biases, with respect to which the gradient is computed backwards from the output to the input layer according to the chain rule, one layer at the time, using the back-propagation algorithm [62].

- **on-policy:** the algorithm improves the policy used to generate the training data (in contrast to off-policy methods that also learn from data generated with other policies).

- **advantage:** the policy gradient is approximated by that of the policy loss

$$\mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^T \log(\pi(a_t | s_t)) \widehat{A}^\pi(s, a) \right], \quad (4.1)$$

where $\tau = (s_0, a_0, \dots, s_T, a_T)$ is a trajectory of state and actions with horizon T , A^π is the advantage function measuring the gain associated with taking action a in state s , compared to taking the average over all possible actions, and \widehat{A}^π is some biased estimator of the advantage, here its normalization to zero mean and unit variance.

- **actor-critic:** the learning performance is improved 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 estimates the advantage as

$$A^\pi(s_t, a_t) = r_t + \gamma V(s_{t+1}) - V(s_t), \quad (4.2)$$

where $V(s)$ is the expected value of the return of the policy in state s and $\gamma \in [0, 1]$ is a discount factor adjusting the trade-off between immediate and future rewards.

PPO uses conservative policy updates to alleviate the issue of performance collapse affecting standard policy gradient implementations¹. We use here PPO-clip²

¹Large policy updates can cause the agent to fall off the cliff and to restart from a poorly performing state with a locally bad policy, which is all the more harmful as the step size for policy updating cannot be tuned locally (an above average value can speed up learning in regions of the parameter space where the policy loss is relatively flat, but trigger exploding updates in sharper variation regions).

²As opposed to PPO-Penalty, a variant relying on a penalization on the average Kullback-Leibler divergence between the current and new policies, but that tends to perform less well in practice.

to optimize the surrogate loss

$$\mathbb{E}_{(s,a)\sim\pi} \left[\min \left(\frac{\pi(a|s)}{\pi_{old}(a|s)}, 1 + \epsilon \text{sgn}(\widehat{A}^\pi(s, a)) \right) \widehat{A}^\pi(s, a) \right], \quad (4.3)$$

where ϵ is the clipping range defining how far away the new policy is allowed to go from the old. 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 (4.3) 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. There exist more sophisticated PPO algorithms (e.g., Trust region PPO [143], that determines first a maximum step size relevant for exploration, then adaptively adjusts the clipping range to find the optimal within this trust region), but standard PPO has simple and effective heuristics. It is computationally inexpensive, easy to implement (as it involves only the first-order gradient of the policy log probability), and remains regarded as one of the most successful RL algorithms, achieving state-of-the-art performance across a wide range of challenging tasks, including flow control [39].

4.2.3 Single-step PPO

We now come to single-step PPO (hereafter denoted by PPO-1 to ease the reading), a “degenerate” version of PPO introduced in [138] and intended for situations where the optimal policy to be learnt by the neural network is state-independent, as is notably the case in 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 actions a_{opt} yielding the largest possible reward, single-step PPO seeks the optimal mapping $f_{\theta_{opt}}$ such that $a_{opt} = f_{\theta_{opt}}(s_0)$, where θ denotes the network free parameters and s_0 is some input state (usually a vector of zeros) consistently fed to the agent for the optimal policy to eventually embody the transformation from s_0 to a_{opt} . The agent initially implements a random state-action mapping f_{θ_0} from s_0 to an initial policy determined by the free parameters initialization 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 4.1 showing the agent draw a population of actions $a_t = f_{\theta_t}(s_0)$ from the current policy, and being returned incentives from the associated rewards to update the free parameters for the next population of actions $a_{t+1} = f_{\theta_{t+1}}(s_0)$ 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,

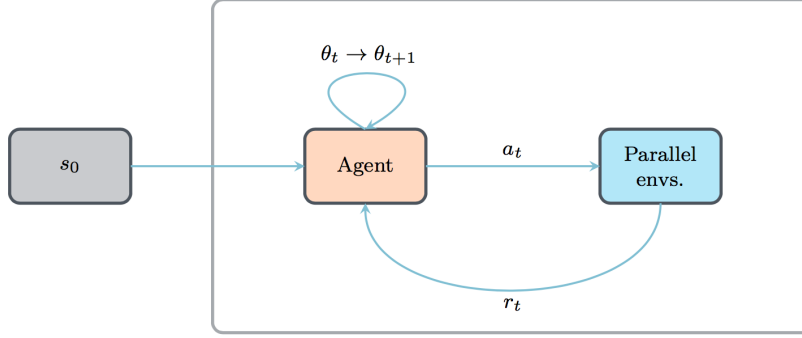


Figure 4.1: Action loop for single-step PPO. At each episode, the same input state s_0 is provided to the agent, which in turn provides n actions to n parallel environments. The latter return n rewards, that evaluate the quality of each action taken. Once all the rewards are collected, an update of the agent parameters is made using the PPO loss (4.3). The process is repeated until convergence.

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 4.2.2, only the surrogate loss reads

$$\mathbb{E}_{a \sim \pi} \left[\min \left(\frac{\pi(a)}{\pi_{old}(a)}, 1 + \epsilon \text{sgn}(\hat{A}^\pi(a)) \right) \hat{A}^\pi(a) \right], \quad (4.4)$$

and the advantage A^π reduces to the whitened reward r_t . This is because 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 two rightmost terms cancel each other out in (4.2), meaning that single-step PPO can do without the value-function evaluations of the critic network (and is thus not actually actor-critic).

4.2.4 Computational fluid dynamics environment

The CFD resolution framework relies on the in-house, parallel, finite element library CimLIB_CFD [103], whose main ingredients are as follows:

- the variational multiscale approach (VMS) is used to solve a stabilized weak form of the governing equations using linear approximations (P_1 elements) for all variables, which otherwise breaks the Babuska–Brezzi condition. The approach relies on an a priori decomposition of the solution into coarse and fine scale components [79, 144, 145]. Only the large scales are fully represented and resolved at the discrete level. The effect of the small scales is encompassed by consistently derived source terms

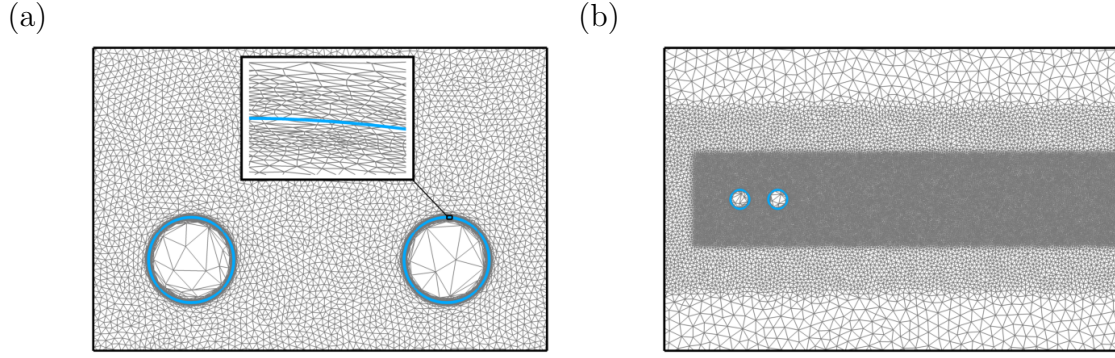


Figure 4.2: Details of (a) the boundary layer mesh and (b) successive refinement steps of the background mesh for the flow past a tandem arrangement of two circular cylinders. The blue line in (a) indicates the zero iso-contour of the level set function.

proportional to the residual of the resolved scale solution, hence ad-hoc stabilization parameters comparable to local coefficients of proportionality.

- in laminar regimes, velocity and pressure come as solutions to the Navier–Stokes equations. In turbulent regimes, the focus is on phase-averaged velocity and pressure modeled after the unsteady Reynolds averaged Navier–Stokes (uRANS) equations. In order to avoid transient negative turbulent viscosities, negative Spalart–Allmaras [99] is used as turbulence model, whose stabilization proceeds from that of the convection-diffusion-reaction equation [100, 146].

- the immersed volume method (IVM) is used to immerse and represent all geometries inside a unique mesh. The approach combines level-set functions to localize the solid/fluid interface, and anisotropic mesh adaptation to refine the mesh interface under the constraint of a fixed, number of edges. This ensures that the quality of all actions taken over the course of a PPO optimization is equally assessed, even though the interface can depend on the action.

Substantial evidence of the flexibility, accuracy and reliability of this numerical framework is documented in several papers to which the reader is referred for exhaustive details regarding the level-set and mesh adaptation algorithms [102, 103], the VMS formulations, stabilization parameters and discretization schemes used in laminar and turbulent regimes [89, 90, 147, 148], and the mathematical formulation of the IVM in the context of finite element VMS methods [149, 150].

4.2.5 Numerical implementation

In practice, actions are distributed to multiple environments running in parallel, each of which executes a self-contained MPI-parallel CFD simulation and feeds data to the DRL algorithm (hence, two levels of parallelism related to the environment and the computing architecture). Here, all CFD simulations are performed on 12 cores of a workstation of Intel Xeon E5-2640 processors. The algorithm waits for the simulations running in all parallel environments to be completed, then shuffles and splits the rewards data set collected from all environments into several buffers (or mini-batches) used sequentially to compute the loss and perform a network update. The process repeats for several epochs, i.e., several full passes of the training algorithm over the entire data set, which ultimately makes the algorithm slightly off-policy (since the policy network ends up being trained on samples generated by older policies, which is customary in standard PPO operation). This simple parallelization technique is key to use DRL in the context of CFD applications, as a sufficient number of actions drawn from the current policy must be evaluated to accurately estimate the policy gradient. This comes at the expense of computing the same amount of reward evaluations, and yields a substantial computational cost for high-dimensional fluid dynamics problems (typically from a few to several hundred CFD simulations for the cases considered herein). In the same vein, it should be noted that the common practice in DRL studies to gain insight into the performances of the selected algorithm by averaging results over multiple independent training runs with different random seeds is not tractable, as it would trigger a prohibitively large computational burden. The same random seeds have thus been deliberately used over the whole course of study to ensure a minimal level of performance comparison between cases. The remainder of the practical implementation details are as follows:

- the environment consists of CFD simulations of two-dimensional (2-D) flows described in a Cartesian coordinate system with drag positive in the $+x$ direction. All equations are discretized on rectangular grids whose side lengths documented in the coming sections have been checked to be large enough not to have a discernible influence on the results (with the exception of the square cylinder flow in section 4.4.1.3 and the fluidic pinball in section 4.4.2, for which we use respectively the values recommended in [151] and the same values as in [132]). Open flow conditions are used, that consist of a uniform inflow in the x direction, together with symmetric lateral, advective outflow and no-slip interface conditions. In turbulent regime, the ambient value of the Spalart–Allmaras variable is three times the molecular viscosity, as recommended to lead to immediate transition. Typical adapted meshes of the interface and wake regions are shown in figure 4.2, the latter also being accurately captured via successive refinement of the background elements.

Neural network	
2	Nb. hidden layers
4	Nb. neurons/layer
TBS	Nb. epochs
TBS	Nb. environments
TBS	Size of mini-batches
PPO	
5×10^{-3}	Learning rate
0.3	Clipping range
1	Discount factor

Table 4.1: Details of the network architecture and PPO hyper parameters. The number of epochs, environments and the size of the mini-batches are provided on a case-by-case basis in sections 4.3 and 4.4.

- the instant reward is (up to a plus/minus sign) either the time-averaged or the root mean square (rms) value of the force coefficient (drag or lift per unit span length), to consider either the mean or fluctuating force acting on the immersed body. Instantaneous values are computed with a variational approach featuring only volume integral terms, reportedly less sensitive to the approximation of the body interface than their surface counterparts [152, 153]. Time averages are performed over an interval $[t_i; t_f]$ with edges large enough to dismiss the initial transient and achieve convergence to statistical equilibrium. Moving average rewards and actions are also computed as the sliding average over the 50 latest values (or the whole sample if it has insufficient size).

- the agent is a fully connected network with 2 hidden layers, each of which holds 4 neurons with hyperbolic tangent activation functions. We use the default online PPO implementation of Stable Baselines, a toolset of reinforcement learning algorithms dedicated to the research community and industry [154], for which a custom OpenAI environment has been designed with the Gym library [155]. Unlike other RL algorithms, PPO does not generally require significant tuning of the hyper parameters (i.e., parameters that are not estimated from data). Nonetheless, all values used in this study are documented in table 4.1 to ease reproducibility, including the learning rate (the size of the step taken in the gradient direction for policy update), the PPO clipping range (set to the upper edge of the recommended range) and the discount factor (set to the default PPO-1 value).

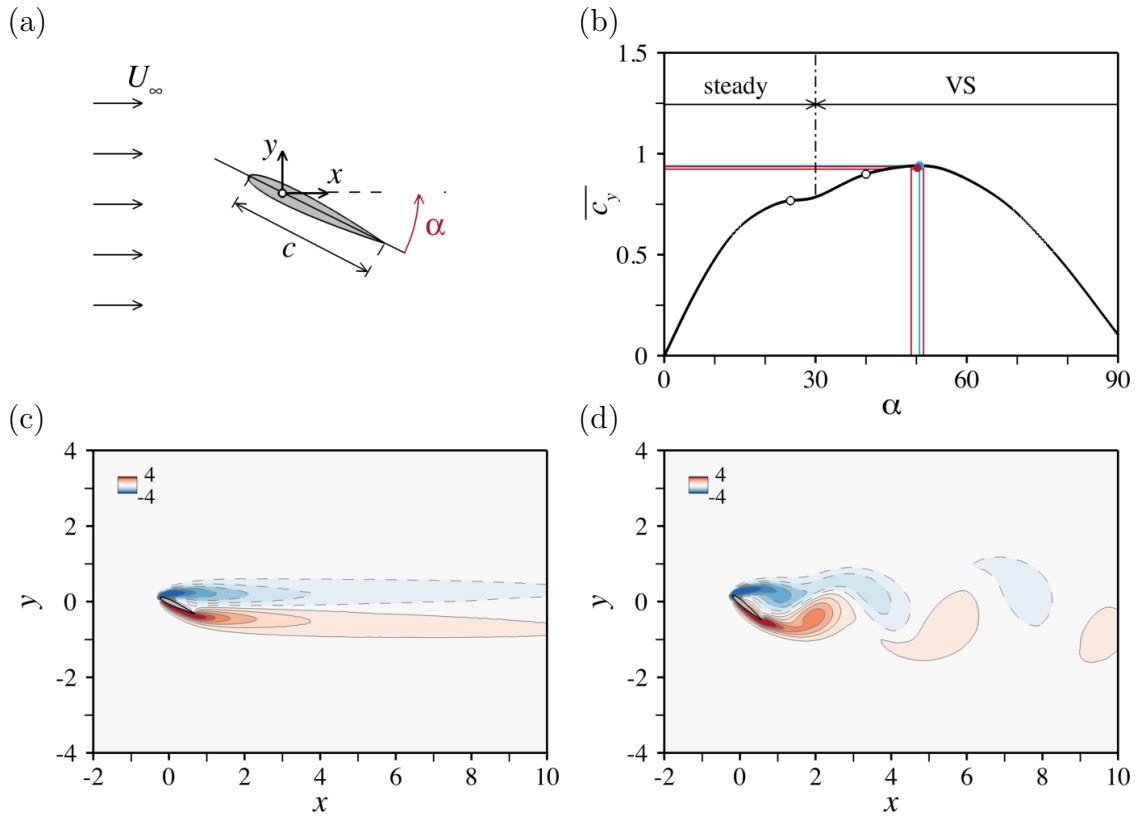


Figure 4.3: Flow past a NACA 0012 - (a) Schematic diagram of the configuration. (b) Mean lift against the angle of attack computed by DNS at $Re = 100$. The VS label indicates the angles for which the flow exhibits unsteadiness in the form of periodic vortex formation and shedding. The blue lines and symbols mark the optimal. The red symbol is the average over the 5 latest single-step PPO episodes, and the red lines delimit the corresponding variance intervals. (c-d) Instantaneous vorticity fields computed at $Re = 100$, for values marked by the circle symbols in (b), namely (c) $\alpha = 25$ and (d) $\alpha = 40$.

4.3 Application to flow optimization

4.3.1 Flow past a NACA 0012 airfoil

We consider first a NACA 0012 airfoil placed at incidence in a uniform stream, as depicted in figure 4.3(a). The origin of the coordinate system is at the airfoil pivot-point, set at quarter chord length from the leading edge. A laminar, time-dependent case at Reynolds number $Re = U_\infty c / \nu = 100$ is modeled after the Navier–Stokes equations, where U_∞ is the inflow velocity, c the straight chord distance and ν the kinematic viscosity. The objective is to maximize the mean lift $\overline{C_l}$, for which the sole control parameter is the angle of attack α measuring the incidence relative to

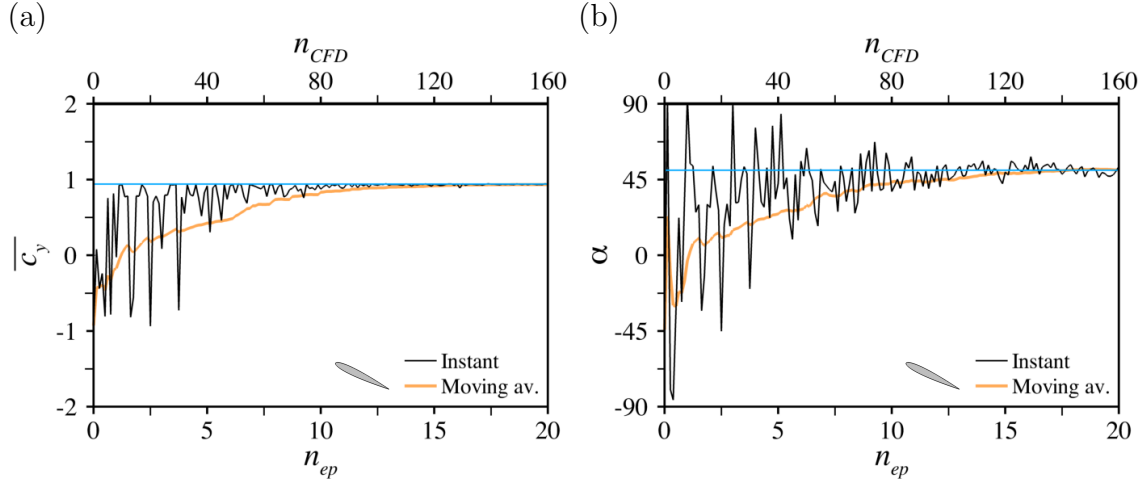


Figure 4.4: Flow past a NACA 0012 at $Re = 100$ - (a) Evolution per episode for the instant (black line) and moving average (over episodes, orange line) values of the mean lift (over time). The corresponding number of CFD simulations (obtained multiplying by the number of environments) is displayed on the secondary horizontal axis. (b) Same as (a) for the angle of attack. The blue lines mark the DNS optimal.

the chord (in degrees and with the convention that $\alpha > 0$ for the airfoil to generate positive lift. Also, we keep in mind that α is rather a state parameter than an adjustable control parameter in practical situations, but the methodology carries over to related optimization problems such as the design of multi-element high-lift systems). This is a problem simple enough to allow direct comparisons between PPO-1 and DNS (actually VMS, but the difference is clear from context), all the more so as lift varies smoothly with the incidence. This is evidenced in figure 4.3(b) showing reference data obtained from 15 DNS runs computing the mean lift to an accuracy of 3% with the simulation parameters documented in table 4.2. The distribution changes slope near $\alpha \sim 30^\circ$ (because the system bifurcates from a steady to a time-periodic vortex-shedding regime; see figure 4.3(c-d) showing instantaneous vorticity fields computed on either side of the threshold) but otherwise exhibits a well-defined, smooth maximum at $\alpha^* = 50.6$, associated with $\bar{c}_y^* = 0.94$.

For each PPO-1 learning episode, the network outputs a single value ξ in $[-1; 1]$ mapped into

$$\alpha = \xi \alpha_{\max}, \quad (4.5)$$

for the angle of attack to vary in $[-\alpha_{\max}; \alpha_{\max}]$ with $\alpha_{\max} = 90^\circ$. The reward $r = \bar{c}_y$ is then computed using the same simulation parameters, after which the network is updated for 32 epochs using 8 environments and 4 steps mini-batches. 20 episodes have been run for this case, which represents 160 simulations, each of which lasts

$\sim 25\text{mn}$ using 12 cores,³ hence $\sim 65\text{h}$ of total CPU cost (equivalently, $\sim 8\text{h}$ of resolution time). We show in figure 4.4(a) the evolution of the reward collected over the course of the optimization. The moving average increases almost monotonically and reaches a plateau after about 15 episodes, and the optimal lift computed as the average over the 5 latest episodes is $\bar{c}_y^* = 0.93 \pm 0.01$ (the variations are computed from the rms of the moving average over the same interval, which is a simple yet robust criterion to assess qualitatively convergence a posteriori). The associated angle $\alpha^* = 50.2^\circ \pm 1.2^\circ$ varies by a larger factor, which is because lift is relatively insensitive to the exact incidence in the vicinity of the optimal. This is perfectly in line with the DNS, as illustrated by the red lines in figure 4.3(b) showing the limits of the so-computed variance intervals. Nonetheless, PPO-1 turns to be rather inefficient at finding the optimal, because it must span continuous ranges of angles while the one-dimensionality of the control space and the smoothness of the optimal allow DNS to test only a few discrete values (hence it can converge within $\sim 1\text{h}$ using the same level of CFD parallelization).

4.3.2 Flow past an arrangement of two side-by-side circular cylinders

We examine now the side-by-side tandem arrangement of two identical circular cylinders in a uniform stream, whose configuration is sketched in figure 4.5(a). The origin of the coordinate system is at the center of the main cylinder, where we refer to the upstream and downstream cylinders as “main” and “surrounding”, respectively. A laminar, time-dependent case at $\text{Re} = U_\infty D/\nu = 300$ is modeled after the Navier–Stokes equations, where D is the diameter of either cylinder. The objective is to maximize the rms lift $c_{y,\text{rms}}$ of the two-cylinder system (for instance, to increase the amount of energy available for harnessing from fluid-structure interactions) for which the sole control parameter is the gap spacing G , i.e., the side-to-side distance between the two cylinders. On paper, this is another problem simple enough to allow direct comparisons between PPO-1 and DNS. In practice, the results are not so unequivocal, as evidenced in figure 4.5(b) showing reference data obtained from 30 DNS runs computing the rms lift to an accuracy of 5% with the simulation parameters documented in table 4.2. A steep global maximum lies at $G^* = 2.35$, associated with $c_{y,\text{rms}}^* = 1.99$, but there is a smoother local maximum at $G^{**} = 6.25$, associated with $c_{y,\text{rms}}^{**} = 1.36$, which reflects the high sensitivity of the pattern of flow unsteadiness to the center distance. Namely without going into too much detail (as

³This is the time needed to compute periodic vortex shedding solutions. It takes less than 10mn to march the solution to steady state, but this barely affects the total CPU cost, as the time needed to complete an episode is that of completing its longest simulation (so only the cost of those episodes exclusively computing steady state solutions is reduced by a few minutes).

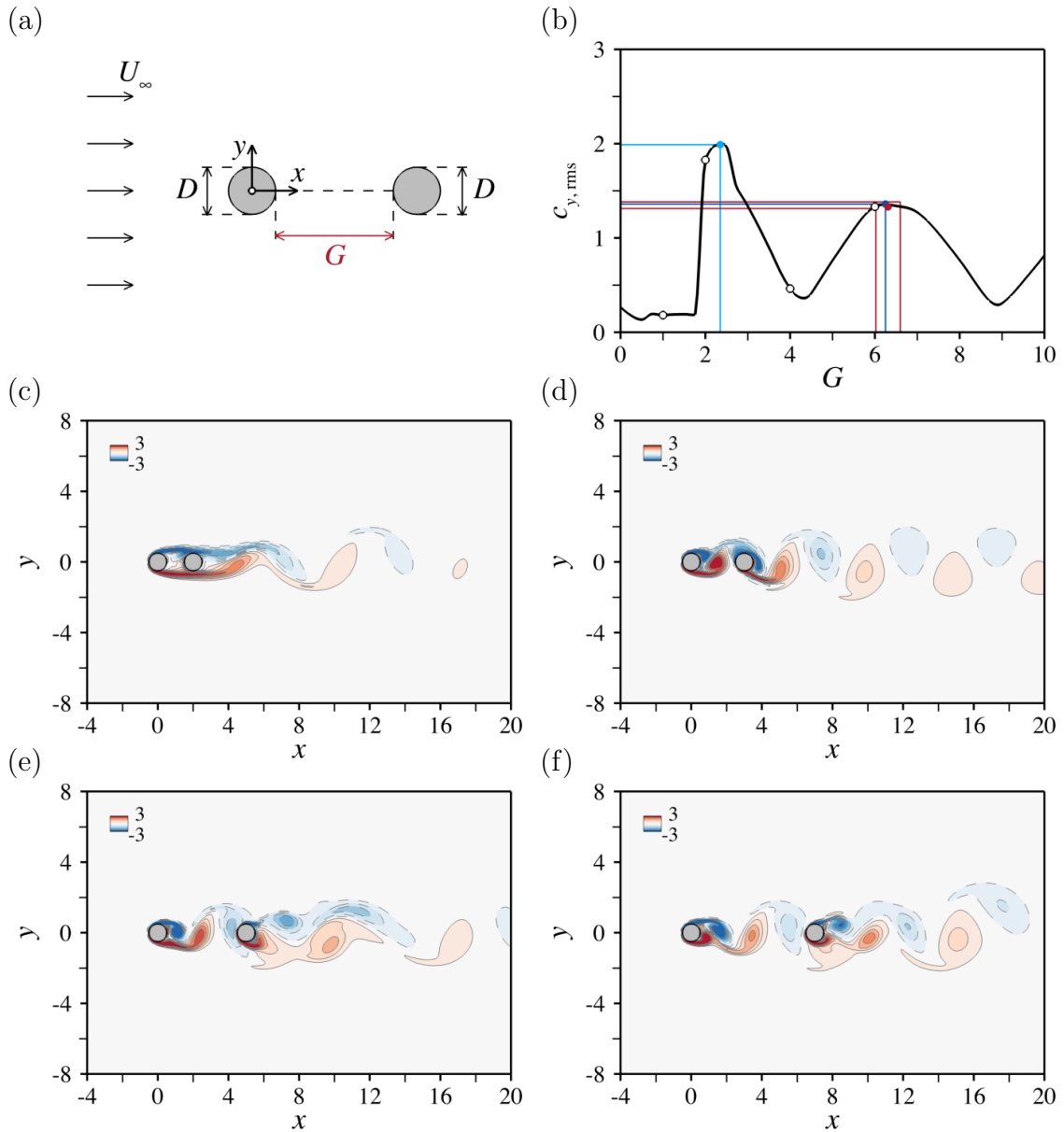


Figure 4.5: Flow past the tandem arrangement of two circular cylinders - (a) Schematic diagram of the configuration. (b) Fluctuating (rms) lift against the gap spacing computed by DNS at $Re = 300$. The red symbol is the average over the 5 latest single-step PPO episodes, and the red lines delimit the corresponding variance intervals. (c-f) Instantaneous vorticity fields computed at $Re = 300$, for values marked by the circle symbols in (b), namely (c) $G = 1$, (d) $G = 2$, (d) $G = 4$, and (e) $G = 6$.

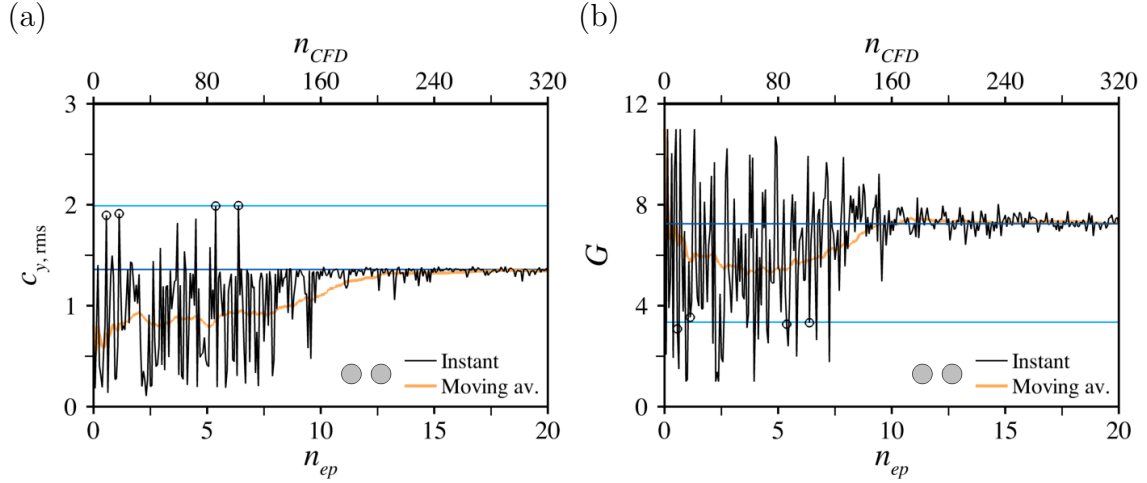


Figure 4.6: Flow past the tandem arrangement of two circular cylinders at $Re = 300$ - (a) Evolution per episode for the instant (black line) and moving average (over episodes, orange line) values of the rms lift. (b) Same as (a) for the gap spacing. The light (resp. dark) blue lines mark the DNS global (resp. local) maximum. The circles are high reward parameters close to the DNS global maximum.

this has been extensively discussed in the literature [156–159]), the instantaneous vorticity field computed for $G = 1$ in figure 4.5(c) shows that the gap flow between the two cylinders is initially steady, while the shear layers separating from the main cylinder engulf those of the surrounding cylinder and trigger vortex shedding in the far wake. For $G = 2$ (close to the global maximum), the gap flow is unsteady, but the gap vortices are not fully developed by the time they impinge on the surrounding cylinder, hence a single vortex street in the far wake; see figure 4.5(d). For $G = 4$, one pair of gap vortices fully develops, then impinges on the surrounding cylinder, which triggers a complex interaction in the near wake before a vortex street eventually forms further downstream; see figure 4.5(e). Finally for $G = 6$ (close to the local maximum) the wake of the surrounding cylinder is unsteady again, and both cylinders shed synchronized vortices close to anti-phase; see figure 4.5(f).

For each PPO-1 learning episode, the network outputs a single value ξ in $[-1; 1]$ mapped into

$$G = \frac{1 + \xi}{2} G_{\max}, \quad (4.6)$$

for the gap to vary in $[0; G_{\max}]$ with $G_{\max} = 10$. This enables contact between the two cylinders and keeps the computational cost affordable, as pushing the surrounding cylinder further downstream would require extending the computational domain and increasing the numbers of grid points accordingly (all the more so as we do not

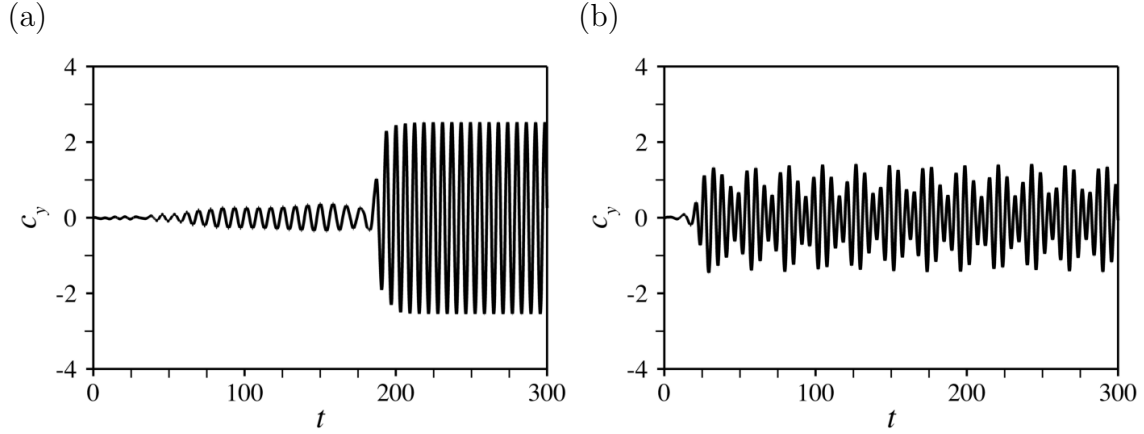


Figure 4.7: Flow past the tandem arrangement of two circular cylinders at $Re = 300$ - Time history of lift computed by DNS for gap spacings (a) $G = 2$ (close to the DNS global maximum) and (b) $G = 8$.

anticipate such large distances to be relevant from the standpoint of optimization because the interaction between both cylinders will weaken increasingly at some point, although it can take up to several tens of diameters to do so). The reward $r = c_{y,\text{rms}}$ is then computed using the same simulation parameters, after which the network is updated for 32 epochs using 16 environments and 4 steps mini-batches. Another 20 episodes have been run for this case. This represents 320 simulations, each of which lasts $\sim 60\text{mn}$ on 12 cores (much longer than in the NACA case due to the increased simulation time), hence $\sim 320\text{h}$ of total CPU cost (equivalently, $\sim 20\text{h}$ of resolution time), still much more than by DNS because DRL keeps spanning continuous ranges of distances while DNS can settle for only a few discrete values despite the sharpness of the global maximum (hence it can converge within $\sim 3\text{h}$ using the same level of CFD parallelization). Figure 4.6(a) shows a plateau in the moving average reward after about 15 episodes. The optimal lift computed as the average over the 5 latest episodes is $c_{y,\text{rms}}^* = 1.34 \pm 0.02$, associated with $G^* = 6.31 \pm 0.04$, meaning that the agent misses the global maximum, but converges to a value close to the local maximum; see the red lines in figure 4.5(b) indicating the limits of the computed variance intervals.

This half-failure can be explained by the steepness of the reward gradients with respect to the control variable in the vicinity of the global maximum. This is due to the existence of a secondary instability mechanism at play in a narrow range of center distances, as illustrated in figure 4.7(a) showing that for $G \sim 2$, the flow settles to a first time-periodic solution, then bifurcates to a second time-periodic solution associated with increased lift oscillations (hence the large values of t_i used for this case). Actually, DRL does identify high reward positions close to $G = 2$ (circle symbols in figure 4.6), whose value $c_{y,\text{rms}} \sim 2$ is consistent with the global


	$\overline{c_y}$	α		$c_{y,rms}$	G		
	0.93	50.2°	PPO-1	1.34	6.31	PPO-1	Optimal
	0.94	50.6°	DNS	1.99 1.36	2.35 6.25	DNS	
CFD							
	100			300			Reynolds number
	0.125			>			Time-step
	[50; 150]			[200; 300]			Averaging time span
$[-15; 40] \times [-15; 15]$				>			Mesh dimensions
	115000			125000			Nb. mesh elements
	0.001			>			Interface \perp mesh size
	12			>			Nb. Cores
PPO-1							
	20			>			Nb. DRL episodes
	8			16			Nb. Environments
	32			>			Nb. Epochs
	4			>			Size of mini-batches
	60h			320h			CPU time
	7.5h			20h			Resolution time

Table 4.2: Simulation parameters and convergence data for the flow past a NACA 0012 at $Re = 100$ and the flow past the tandem arrangement of two circular cylinders at $Re = 300$. NACA 0012: the interface mesh size yields ~ 20 grid points in the boundary-layer at mid-chord, under zero incidence, and the averaging time-span represents $\sim 15 - 20$ shedding cycles, depending on the incidence. Tandem arrangement of two circular cylinders: the interface mesh size yields ~ 20 grid points in the boundary-layer of the main cylinder, just prior to separation, and the averaging time-span represents ~ 20 shedding cycles.

maximum, but there are very few times where the global maximum is met during the exploration phase (compared to its local counterpart, again because of the topology of the reward function). Because PPO voluntarily dismisses large policy updates to avoid performance collapse, the clipped policy updates only lead to limited exploration and trap the optimization process into a local maximum. Low to moderate Reynolds numbers are likely required for such instability cascade scenario to occur, so such results do not cast doubt on the applicability of single-step PPO to practically meaningful high Reynolds flows. They do stress, however, that the method

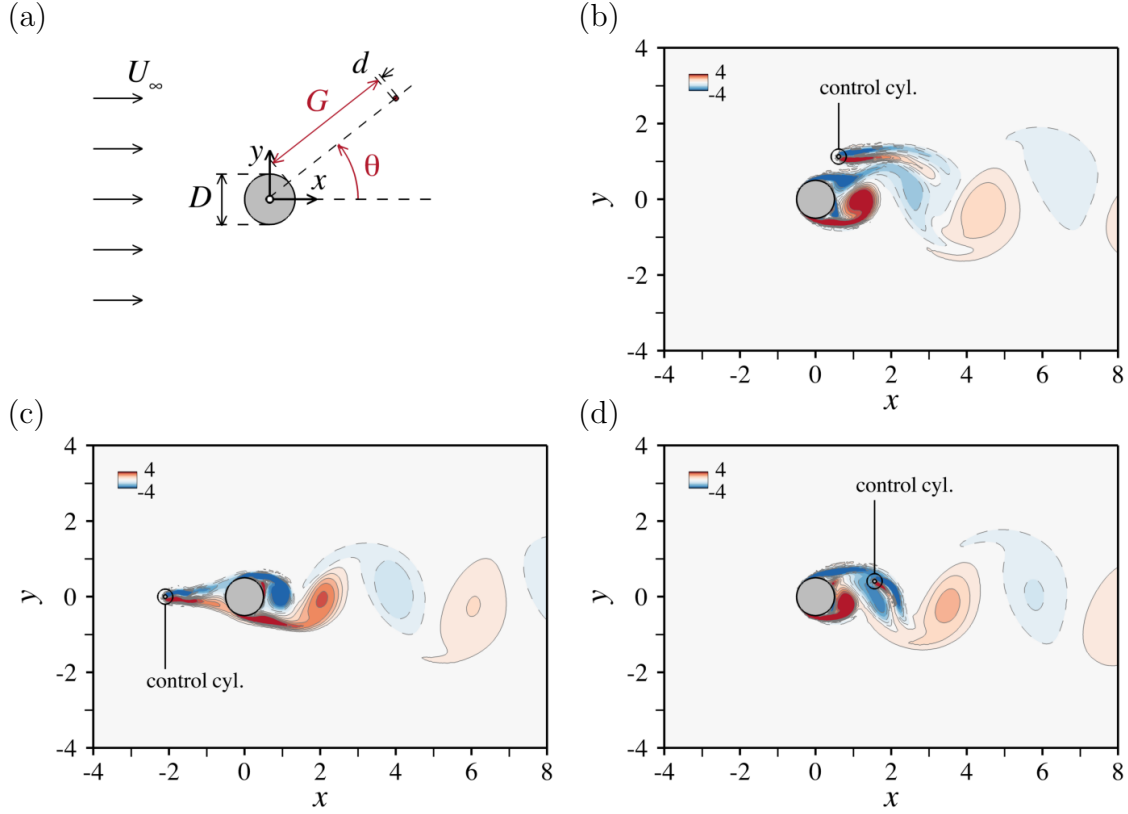


Figure 4.8: Open-loop control of the circular cylinder flow by a small control cylinder of diameter $d = 0.1$ - (a) Schematic diagram of the configuration. (b-d) Isocontours of the vorticity field computed at $Re = 3900$ for representative positions (x_c, y_c) of the control cylinder, namely (b) $(0.61, 1.13)$, (c) $(-2.10, 0.00)$ and (d) $(1.56, 0.41)$.

can benefit from carefully tuning the trade-off between exploration and exploitation, which will be addressed in future work.

4.4 Application to open-loop flow control

4.4.1 Optimal cylinder drag reduction using a smaller control cylinder

The relevance of single-step PPO is now showcased by tackling various open-loop control problems. The first one is that of a cylinder in a uniform stream, controlled open-loop by a much smaller circular cylinder. Figure 4.8(a) presents a sketch of the configuration pertaining to a circular geometry of the main cylinder, where we refer to the large and small cylinders as “main” and “control”, respectively,

but section 4.4.1.4 also considers a square geometry. The origin of the coordinate system is at the center of the main cylinder. The objective is to minimize the mean drag \bar{c}_x of the two-cylinder system, which requires reducing the drag of the main cylinder sufficiently to compensate for the fact that the control cylinder itself is a source of drag. Several laminar and turbulent Reynolds numbers $\text{Re} = U_\infty D/\nu$ are considered, where D is the diameter of the main cylinder. The diameter of the control cylinder is set to $d = 0.1$, therefore the sole control parameter is the 2-D position of the control cylinder center, measured by the gap distance G between the two cylinders and the azimuthal position θ with respect to the rear stagnation point. This may not seem overly complicated on paper, but the parameter space is actually large enough to dismiss mapping the best positions for placement of the control cylinder by DNS, as tens of thousands of runs are required to cover merely a few diameters around the main cylinder. In the following, single-step PPO is thus compared to theoretical predictions obtained by the adjoint method. The latter has proven fruitful to gain insight into the most efficient region from the linear sensitivity of the uncontrolled flow (i.e., the flow past the main cylinder), without ever calculating the controlled states, using instead a simple model of the force exerted by the control cylinder on the flow. We shall not go into the technicalities of how to derive the related adjoint equations, as the line of thought here is to take the output sensitivity as a given to assess relevance of PPO-1. Suffice it to say here that we rely on various levels of adjoint modeling whose key assumptions are reviewed in appendix 4.6. The reader interested in more details is directed to the original literature on this topic [124, 160, 161], where in-depth technical and mathematical information, together with extensive discussions regarding the validity of the approximations are available. From the numerical standpoint, all calculations are performed with the mixed finite elements adjoint solver presented and validated in [124].

On the CFD side, one of the challenges lies in the fact that the control cylinder acts as a small local disturbance redistributing the vorticity in the separated shear layers; see figures 4.8(b-d) showing instantaneous vorticity fields computed for representative positions of the control cylinder. Accurate numerical methods are thus mandatory to capture the small drag variations induced by the control. Several values of the Reynolds number are investigated : a laminar, steady case at $\text{Re} = 40$, for which the flow remains steady-state regardless of the position of the control cylinder, a laminar, time-dependent case at $\text{Re} = 100$, for which vortex shedding consistently develops from the main cylinder but the flow past the control cylinder remains steady, and two turbulent cases at $\text{Re} = 3900$ and at $\text{Re} = 22000$ (hence modeled after the uRANS equations with negative Spalart–Allmaras as turbulence model), for which vortex shedding develops from both cylinders. This is because the Reynolds number in the wake of the control cylinder must be scaled by the ratio

of the cylinder diameters, which yields values below (resp. above) the instability threshold at $\text{Re} = 100$ (resp. $\text{Re} = 3900$ and $\text{Re} = 22000$).

For each PPO-1 episode, the network outputs two values $\xi_{1,2}$ in $[-1; 1]^2$ mapped into

$$G = \frac{1 + \xi_1}{2} G_{\max}, \quad \theta = \frac{1 + \xi_2}{2} \theta_{\max}, \quad (4.7)$$

for the gap to vary in $[0; G_{\max}]$ with $G_{\max} = 3$, and the azimuthal position to vary in $[0; \theta_{\max}]$ with $\theta_{\max} = 180^\circ$. This enables contact between the two cylinders, and allows taking advantage of the problem symmetry, as it amounts to moving the control cylinder in the upper half of a torus bounded by the surface of the main cylinder and the user-defined exterior radius G_{\max} . In the following, the center position is conveniently presented in terms of the Cartesian coordinates $x_c = \rho \cos \theta$ and $y_c = \rho \sin \theta$, where we note $\rho = G + (1+d)/2$. Since the aim is to minimize drag, the reward $r = -\bar{D}$ is then computed using the simulation parameters documented in table 4.3, after which the network is updated for 32 epochs using 8 environments and 2 steps mini-batches (note the zero averaging span in table 4.3 for $\text{Re} = 40$, as this is a steady case for which the steady asymptotic value of total drag can be evaluated at the final time t_f , provided it is large enough for the solution to relax to steady-state).

4.4.1.1 Laminar steady regime and circular geometry at $\text{Re}=40$

For this first case, 100 episodes have been run, which represents 800 simulations, each of which lasts $\sim 35\text{mn}$ on 12 cores, hence $\sim 480\text{h}$ of total CPU cost (equivalently, $\sim 60\text{h}$ of resolution time). The moving average value of drag reaches a plateau after about 60 episodes in figure 4.9(a), with the optimal value $\bar{c}_x^* = 1.53 \pm 0.01$ computed as the average over the 5 latest episodes representing a reduction by roughly 2% with respect to the uncontrolled value 1.56 (in good agreement with the reference 1.54 from the literature [162, 163]). Meanwhile, the instant value of drag actually keeps oscillating over the next 40 episodes with small but finite amplitude, which is further evidenced in figure 4.9(b-c) showing the instant and moving average center positions of the control cylinder. On the one hand, y_c^* quickly settles to zero, i.e., the control cylinder converges to the horizontal centerline. On the other hand, x_c keeps exchanging positions between two regions distributed almost symmetrically on either side of the main cylinder, an upstream region associated with $\bar{c}_x \sim 1.51$ and a slightly less efficient downstream region associated with $\bar{c}_x \sim 1.54$, which suggests that the drag functional has global and local minima located in valleys of comparable depth. Confirmation comes from the theoretical drag variations computed (in steady mode) from the baseline adjoint method described in appendix 4.6, whose negative iso-values (associated to drag reduction) are mapped in figure 4.9(d). The latter

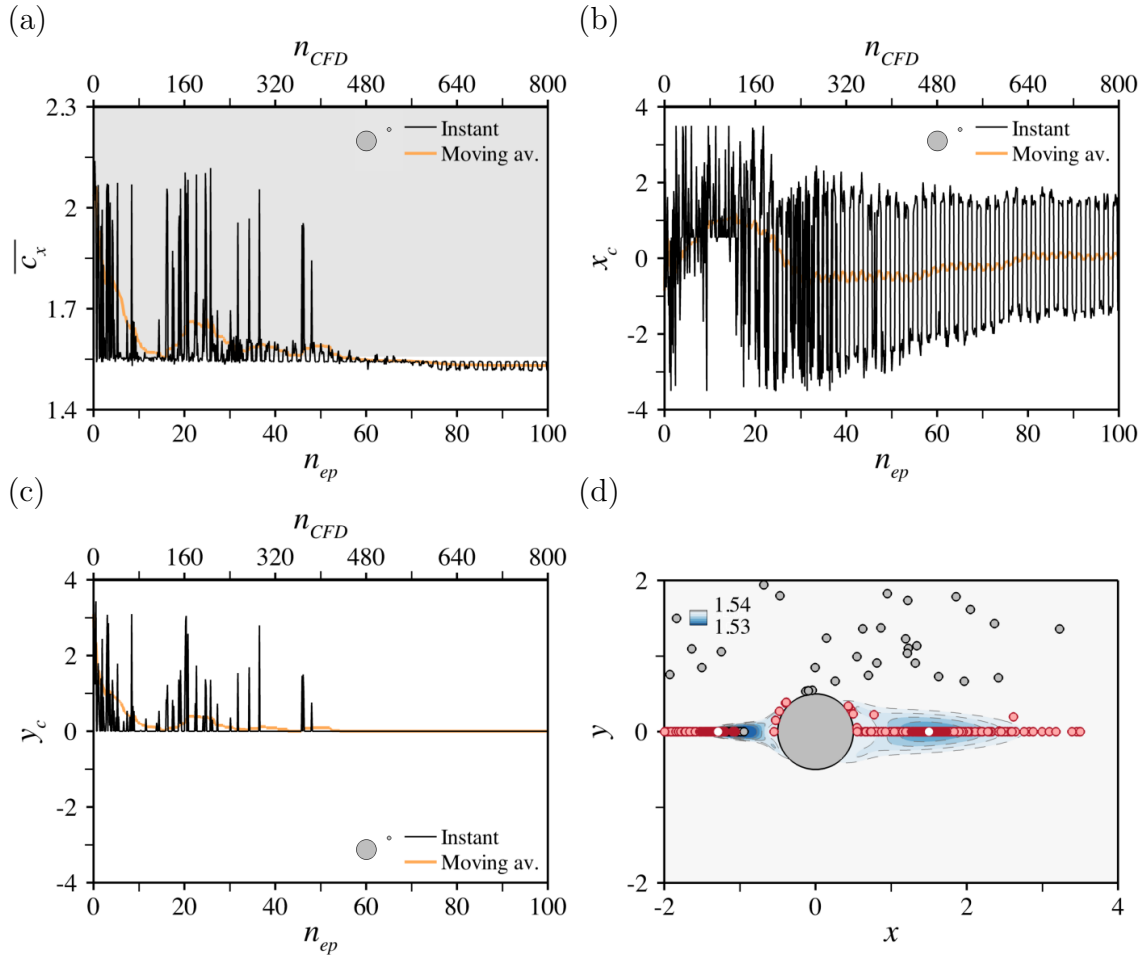


Figure 4.9: Open-loop control of the circular cylinder flow by a small control cylinder of diameter $d = 0.1$ at $Re = 40$ - (a) Evolution per episode for the instant (black line) and moving average (over episodes, orange line) values of the mean drag (over time). The uncontrolled drag is at the bottom of the grey shaded area. (b-c) Same as (a) for the x_c and (c) y_c positions of the control cylinder center. (d) Theoretical mean drag variation computed by a steady adjoint method modeling the presence of the control cylinder by a pointwise reacting force localized at the same location where the control cylinder is placed (only the negative iso-contours are reported for clarity). The grey circles are the positions investigated by the DRL. The light red circles are high reward positions spanned over the course of optimization. The dark red circles are those high reward positions spanned over the last 5 episodes. The white circles are the median values reported in the summarizing table 4.3.

unveil two regions nestled against either side of the main cylinder and achieving similar drag reduction by $\sim 2\%$, a first one extending upstream over approximately

1 diameter, and a second one, slightly less efficient and extending downstream and along the outer boundary of the recirculation over 3 diameters. DRL manages to find high-reward positions in both, which is best seen from the various symbols in figure 4.9(d) showing the complete set of PPO-1 positions investigated over the course of optimization (grey circles) together with those positions achieving optimal drag reduction within 5% (light red circles), including a few non-centerline positions along the edge of both drag reduction regions. Nonetheless, the algorithm ultimately converges to almost symmetrical core positions, as evidenced by the dark red circles in figure 4.9(d) showing the positions spanned over the 5 latest episodes. Despite limited discrepancies regarding the exact position of the upstream region (slightly shifted upstream in the present approach), this is consistent with the adjoint-based results and clearly assesses the ability of single-step PPO to identify both regions of interest and to accurately predict the drag reduction achieved in these regions.

4.4.1.2 Laminar time-dependent regime and circular geometry at $\text{Re}=100$

For this case, 40 episodes have been run, which represents 320 simulations, each of which lasts $\sim 1\text{h}$ on 12 cores, hence $\sim 320\text{h}$ of total CPU cost (equivalently, $\sim 40\text{h}$ of resolution time). The moving average reward plateaus after about 25 episodes in figure 4.10(a), with the optimal drag $\bar{c}_x^* = 1.30 \pm 0.01$ computed as the average over the 5 latest episodes representing a reduction by roughly 5% with respect to the uncontrolled value 1.37 (close to the reference 1.35 from the literature [163]). Unlike the previous steady case at $\text{Re} = 40$, the center position of the control cylinder exhibits a similarly converging behavior in figure 4.10(b-c) with $x_c^* = 1.76 \pm 0.03$ and $y_c^* = 0$, which suggests that the drag functional now has a well-defined global minimum. Confirmation comes from the theoretical drag variations computed (in unsteady mode) from the baseline adjoint method, whose negative iso-values mapped in figure 4.10(d) are reproduced from [164]. The latter unveil again two regions nestled against either side of the main cylinder, a first one extending upstream over approximately 2 diameter (more than at $\text{Re} = 40$), and a second one extending downstream and along the outer boundary of the mean recirculation over 2 diameters (less than at $\text{Re} = 40$). Drag is reduced by roughly 2% upstream, but almost 8% downstream, meaning that the drag functional has global and local minima in valleys of different depth, in line with the DRL results. Again, DRL finds high-reward positions in both regions, as evidenced in figure 4.10(d) by the complete set of PPO-1 positions investigated over the course of optimization (small grey circles) and the positions achieving optimal drag reduction within 5% (light red circles), including a few centerline upstream positions. The algorithm however quickly settles for the most efficient downstream region, as the positions spanned over the 5 latest episodes (dark red circles) all lie in the core of the mean recirculation region, in striking agreement with the adjoint-based results.

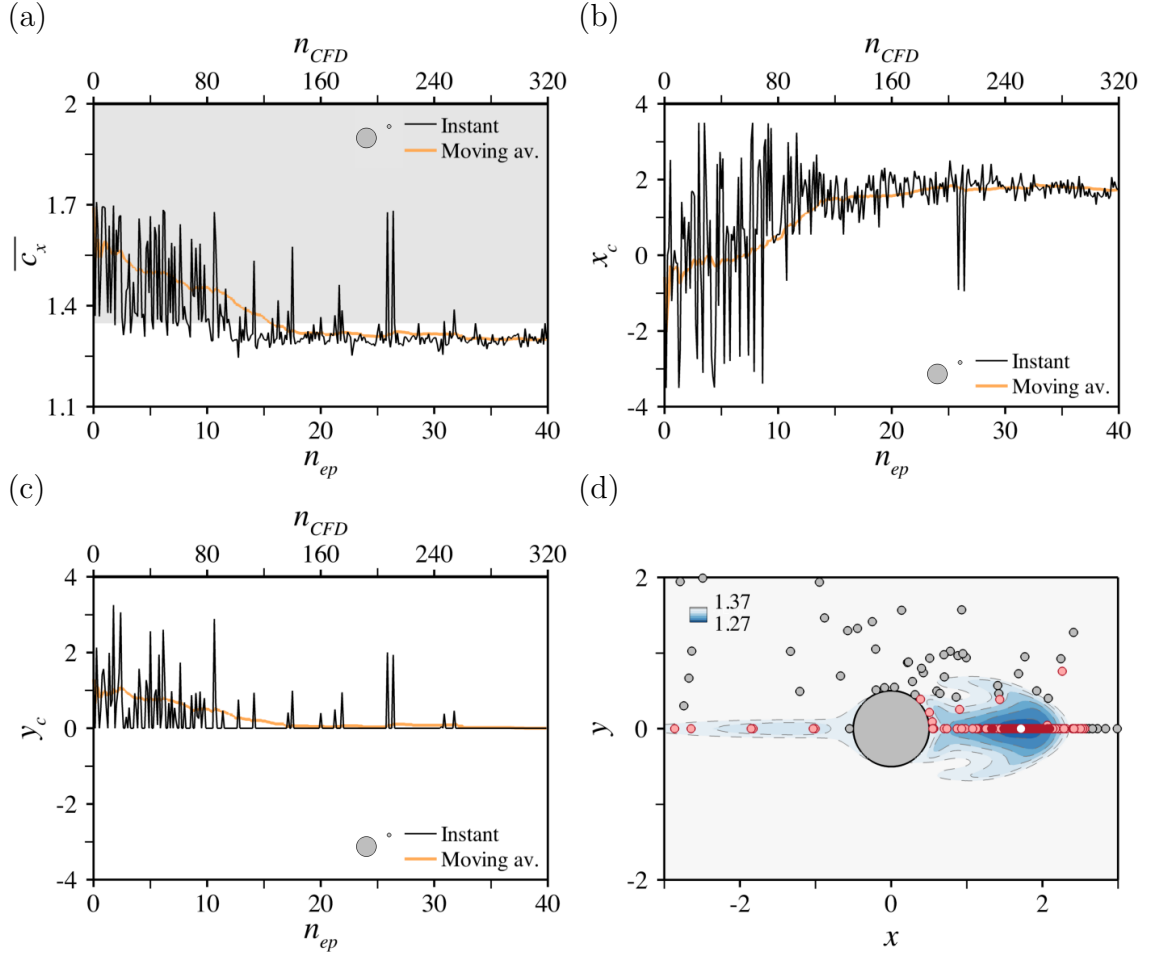


Figure 4.10: Open-loop control of the circular cylinder flow by a small control cylinder of diameter $d = 0.1$ at $Re = 100$ - Same as figure 4.9, only the theoretical variations in (d) have been computed by the time-varying adjoint method presented in [124].

4.4.1.3 Turbulent regime and circular geometry at $Re=3900$

Another 40 episodes have been run for this case, which represents 320 simulations, each of which lasts $\sim 2\text{h}30$ on 12 cores (much longer than at $Re = 100$ due to the halved time step), hence $\sim 800\text{h}$ of total CPU cost (equivalently, $\sim 100\text{h}$ of resolution time). After about 20 episodes, the moving average reward in figure 4.11(a) converges to $\bar{c}_x^* = 1.50 \pm 0.01$, which represents a reduction of drag by 9% with respect to the uncontrolled value 1.65 (in good agreement with reference 2-D RANS data from the literature [166]). The center position of the control cylinder however keeps oscillating over the next 15 episodes in figure 4.11(b-c), as y_c^* goes to zero but x_c exchanges positions between two regions located on either side of the main cylinder,

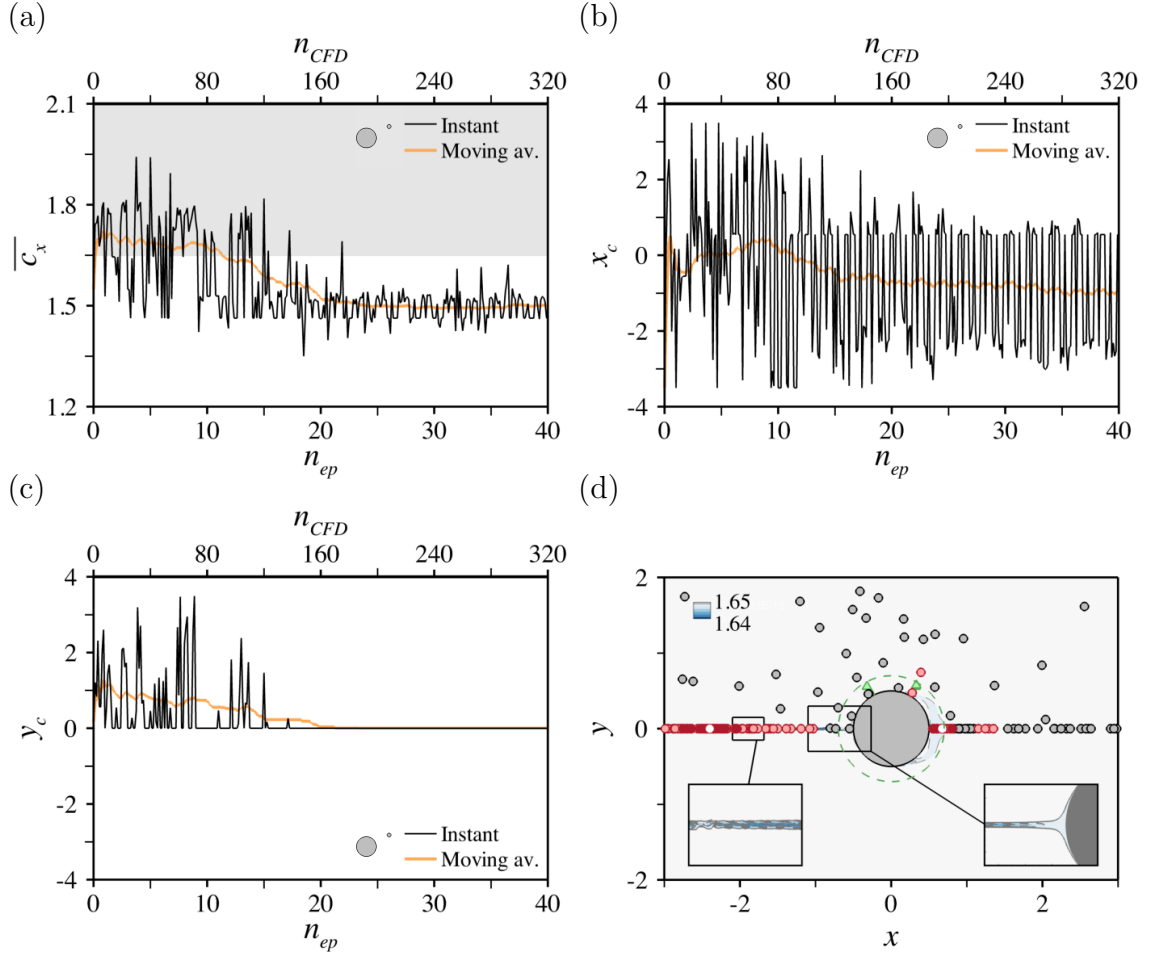


Figure 4.11: Open-loop control of the circular cylinder flow by a small control cylinder of diameter $d = 0.1$ at $Re = 3900$ - Same as figure 4.9, only the theoretical variations in (d) have been computed by the (steady) simplified mean-flow adjoint method presented in [124]. The green dashed circle in (d) indicates the range of center positions spanned experimentally in [165], with the green triangles marking the sets of positions found to optimally reduce the drag of the main cylinder only.

an upstream region associated with $\bar{c}_x \sim 1.52$ and a downstream region associated with $\bar{c}_x \sim 1.46$. This suggests that the drag functional has global and local minima in valleys of comparable depth, which is reminiscent of the steady case at $Re = 40$, only the deepest valley is now downstream, not upstream. Interestingly, Ref. [165] determines experimentally different optimal positions $(G, \theta) = (0.14 - 0.16, 60^\circ)$ and $(0.06 - 0.14, 115^\circ)$, shown as the green triangles in figure 4.11(d). Additional DNS runs have thus been carried out to confirm sub-optimality for our case, although the algorithm does identify a couple of high-reward positions in the vicinity of the

downstream experimental region. This probably stems from the noticeable differences between both studies, as the Reynolds number in [165] is larger by one order of magnitude ($Re = 65000$), the control cylinder is almost twice as small ($d = 0.06$), and the experiments focus on the drag of the main cylinder (not the total drag) while spanning a much smaller range of center positions (indicated by the green dashed circle in figure 4.11(d)).

The DRL results are conversely qualitatively in line with the negative iso-values of the adjoint-based drag variations shown in figure 4.11(d). Those indicate that drag is reduced in two distinct regions nestled against either side of the main cylinder, a first narrow one extending upstream along the centerline over approximately 2 diameters, and a second one extending downstream over a half-diameter and in the vicinity of the mean separation points. Nonetheless, the agreement is not quantitative, as the theoretical variations are by a mere 1% upstream (and even lower downstream). This is most likely because all theoretical variations have been modeled after a simplified adjoint method intended to guide near-optimal design with marginal computational effort (as it requires knowledge of the sole mean uncontrolled solution, as explained in appendix 4.6), that ends up miscalculating the effect of the control cylinder because of an insufficient level of sophistication. On the one hand, the marginal size of the downstream region (as well as the marginal drag reduction predicted in this region) is ascribed to the fact that the approach has been shown to possibly miss out on sensitivity regions involving strong interactions of the mean and fluctuating solution components via the formation of Reynolds stresses [164]: the mean recirculation is one such region where reducing the drag of the main cylinder, even by a small amount, suffices to reduce the total drag because the x velocity is negative and the control cylinder is thus a source of thrust, not drag. On the other hand, the outcome in the upstream region is sensitive to the force model used to mimic the effect of the control cylinder, as it turns out its drag balances almost exactly the amount by which it reduces the drag of the main cylinder. The weak upstream control efficiency may thus be due to the fact that the simplified adjoint method considers only the mean component of the force acting on the control cylinder, but overlooks the potential for additional drag reduction via the fluctuating component. Moreover, this is a region where the control cylinder likely induces strong mean flow modifications because the local inhomogeneity length scale becomes smaller than the diameter of the control cylinder, which in turn may invalidate the linear assumption inherent to the adjoint method (the retained diameter $d = 0.1$ is a compromise between smallness and cost control, as implementing a smaller control cylinder would require increasing the number of grid points and decreasing the time-step to capture properly the wake of the control cylinder).

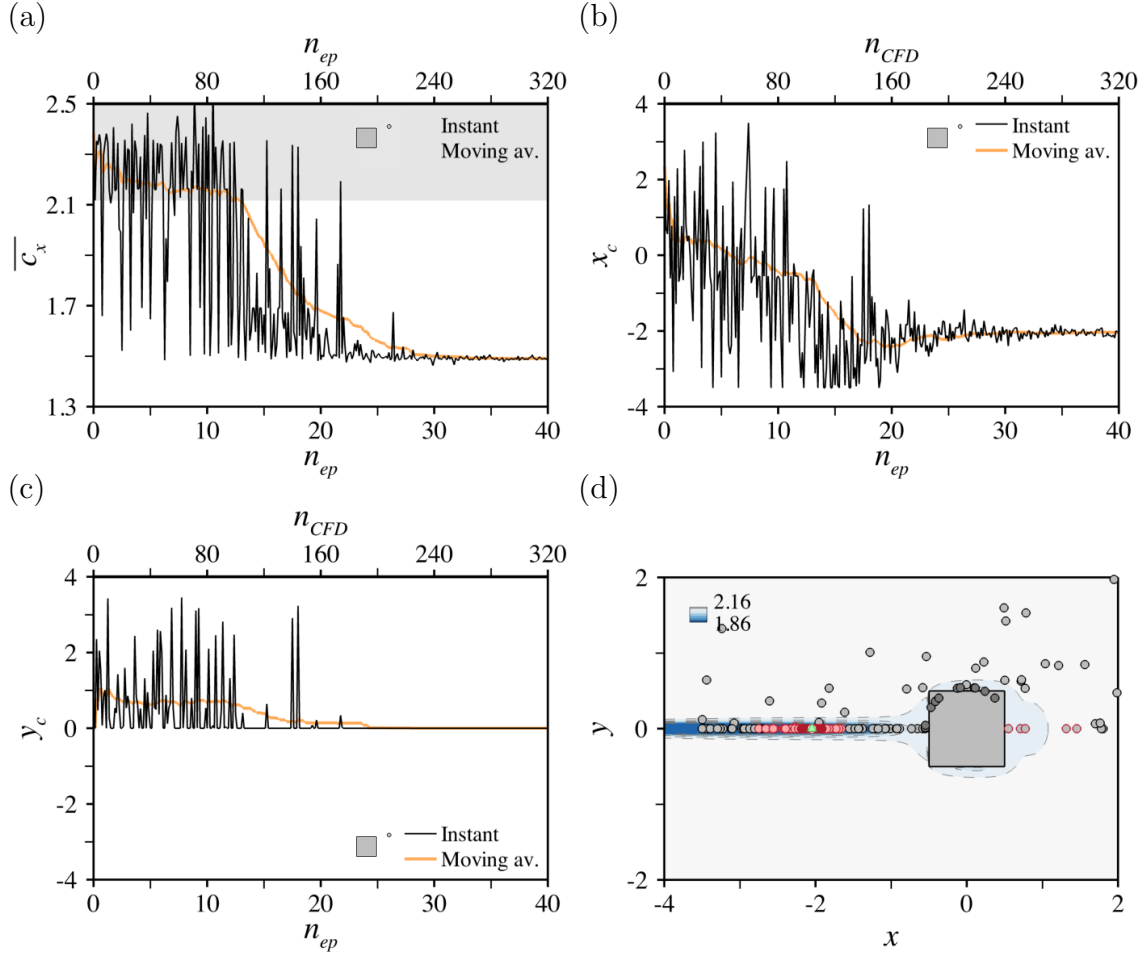


Figure 4.12: Same as figure 4.11 for the open-loop control of the square cylinder flow by a small control cylinder of diameter $d = 0.1$ at $Re = 22000$. In (d), the grey symbols circled in red are downstream sub-optimal position. The dark circles are dismissed positions for which the control and main cylinders intersect, and the green triangle marks the experimental position found in [26] to optimally reduce the drag of the two-cylinder system at $Re = 32000$.



4.4.1.4 Turbulent regime and square geometry at $Re=22000$

In order to push the comparison further, additional calculations have been undertaken for a square geometry of the main cylinder, whose larger upstream sensitivity yields more clear-cut control efficiency, as can be inferred from the results in [124, 164]. This is because the blunt square geometry strengthens the upstream pressure gradient (compared to its bluff circular shape). In return, the gap flow velocity between the two cylinders decreases and so does the drag of the control cylinder, hence a boost in efficiency that helps mitigate the issue of sensitivity to

the force model.

Another 40 episodes have been run for this case, which represents 320 simulations, each of which lasts $\sim 3\text{h}20$ on 12 cores, hence $\sim 1020\text{h}$ of total CPU cost (equivalently, $\sim 130\text{h}$ of resolution time). One difficulty for this case is that the main and control cylinders can intersect each other under mapping (4.7), in which case it has been found relevant to simply discard the CFD and force the reward to its uncontrolled value. The moving average reward plateaus after about 30 episodes in figure 4.12(a), with the optimal drag $\bar{c}_x^* = 1.49 \pm 0.01$ computed as the average over the 5 latest episodes representing a reduction by 30% with respect to the uncontrolled value 2.16 (close to the reference 2.1 – 2.2 from the literature [167, 168]). The center position of the control cylinder exhibits a similarly converging behavior in figures 4.12(b-c) with $x_c^* = -2.04 \pm 0.02$, and $y_c^* = 0$, which suggests that the drag functional has a well-defined global minimum. This is in excellent agreement with [26] reporting experimental reduction of the total drag by 30% inserting control cylinders of comparable sizes upstream of the main cylinder at a slightly different Reynolds number $\text{Re} = 32000$ (the optimal reported position for $d = 0.1$ being $x_c^* \sim -2.0$). This is also in line with the theoretical drag variations computed from the same simplified adjoint method as in section 4.4.1.3, whose negative iso-values mapped in figure 4.12(d) are reproduced from [124]. The latter unveil a main region of interest, that extends upstream over approximately 4 diameters, and in which drag is reduced by almost 20%, which represents a satisfactory qualitative and quantitative compliance with the present PPO-1 results. Drag is also reduced in a second region originating from the separation points (pinned here at the front edges), that extends downstream and along the outer boundary of the mean recirculation over 1 diameter (similar to what has been found using a circular geometry of the main cylinder). It is worth noticing that the algorithm does identify sub-optimal positions in this region (shown in figure 4.12(d) as the grey symbols circled in red). Also, a couple of other low-efficiency PPO-1 positions lie further downstream, which is consistent with the idea that the simplified adjoint method may miss on additional drag reduction occurring via the formation of Reynolds stresses (this is not true of the upstream drag reduction region, whose flow is essentially steady, except for low-amplitude oscillations in the gap flow between the two cylinders).

4.4. Application to open-loop flow control

		\bar{C}_x	x_c^a	y_c	\bar{C}_x	x_c^a	y_c	\bar{C}_x	x_c^a	y_c		\bar{C}_x	x_c^a	y_c	
CFD		1.51	-1.29	0	1.30	1.72	0	1.46	0.68	0		1.49	-2.0	0	Optimal
		1.54	1.50	0				1.52	-2.40	0					
PPO-1				40			100			3900			22000		Reynolds number
				0.1			\gg			0.05			\gg		Time step
				$[150; 150]$			$[100; 200]$			\gg			\gg		Averaging time span
				$[-15; 40] \times [-15; 15]$			\gg			\gg			$[-6; 15] \times [-7; 7]$		Mesh dimensions
				150000			\gg			\gg			190000		Nb. mesh elements
				0.001			\gg			\gg			\gg		Interface \perp mesh size
				12			\gg			\gg			\gg		Nb. Cores
				100			40			\gg			\gg		Nb. episodes
				8			\gg			\gg			\gg		Nb. environments
				32			\gg			\gg			\gg		Nb. epochs
				1			\gg			2			\gg		Size of mini-batches
				480h			320h			800h			1020h		CPU time
			60h			40h			100h			130h		Resolution time	

^a Only the median value of the optimal interval is reported to ease the presentation.

Table 4.3: Open-loop control of circular and square cylinder flows by a small control cylinder of diameter $d = 0.1$ - Simulation parameters and convergence data. The interface mesh size yields $\sim 25 - 40$ grid points in the boundary-layer of the control cylinder, just prior to separation, and the averaging time-span in unsteady flow regimes represents $\sim 15 - 25$ shedding cycles, depending on the geometry of the main cylinder, the position of the control cylinder and the Reynolds number.

4.4.2 Optimal drag reduction of a triangular bluff-body using rotating cylinders

The second control problem presented in figure 4.13(a) is the fluidic pinball [169], an equilateral triangle arrangement of three identical circular cylinders oriented against a uniform stream (i.e., the leftmost triangle vertex points upstream, and the rightmost side is orthogonal to the on-coming flow), controlled open-loop via user-defined angular velocities. The origin of the coordinate system is between the top and bottom cylinders, where we refer to the upstream and downstream cylinders as “front”, “top”, and “bottom”, respectively (also labeled 1, 2 and 3 to ease the notation). The gap spacing $G = 1.5$ between cylinders yields a master cross-section of 2.5. A turbulent case at $\text{Re} = U_\infty D/\nu = 2200$ is modeled after the negative Spalart–Allmaras uRANS equations, where D is the diameter of either cylinder. The objective is to minimize the mean drag \bar{D} of the three-cylinder system, using the cylinders individual angular velocities Ω_{1-3} as control parameters (with the convention that $\Omega_k < 0$ for clockwise rotation). This is a versatile experiment well suited to challenge the single-step approach, as the requirement to span large ranges of control parameters emulating a variety of steady and unsteady actuation (e.g., base bleed, suction) under turbulent conditions makes it especially challenging to rely on the adjoint method (as further discussed in section 4.5), not to mention DNS.

4.4.2.1 Steady actuation

First, constant angular velocities are applied to each cylinder to alter the vorticity flux fed to the wake, as evidenced in figure 4.13(b-d) showing instantaneous vorticity fields computed under several control configurations. Drag is optimized by minimizing the compound reward function

$$r = -\bar{D} - \beta \sum_{k=1}^3 |\Omega_k|^3, \quad (4.8)$$

where the leftmost term is the power of the drag force and is thus associated to performance, the rightmost term estimates the power to be supplied to the rotating cylinders and is thus associated to cost, and β is a weighting coefficient set empirically to $\beta = 0.025$ (a value found to be large enough for cost considerations to impact the optimization procedure, but not so large as to dominate the reward signal, in which case actuating is meaningless). For each PPO-1 learning episode, the network outputs three values ξ_{1-3} in $[-1; 1]^3$ mapped into

$$\Omega_k = \xi_k \Omega_{\max}, \quad (4.9)$$

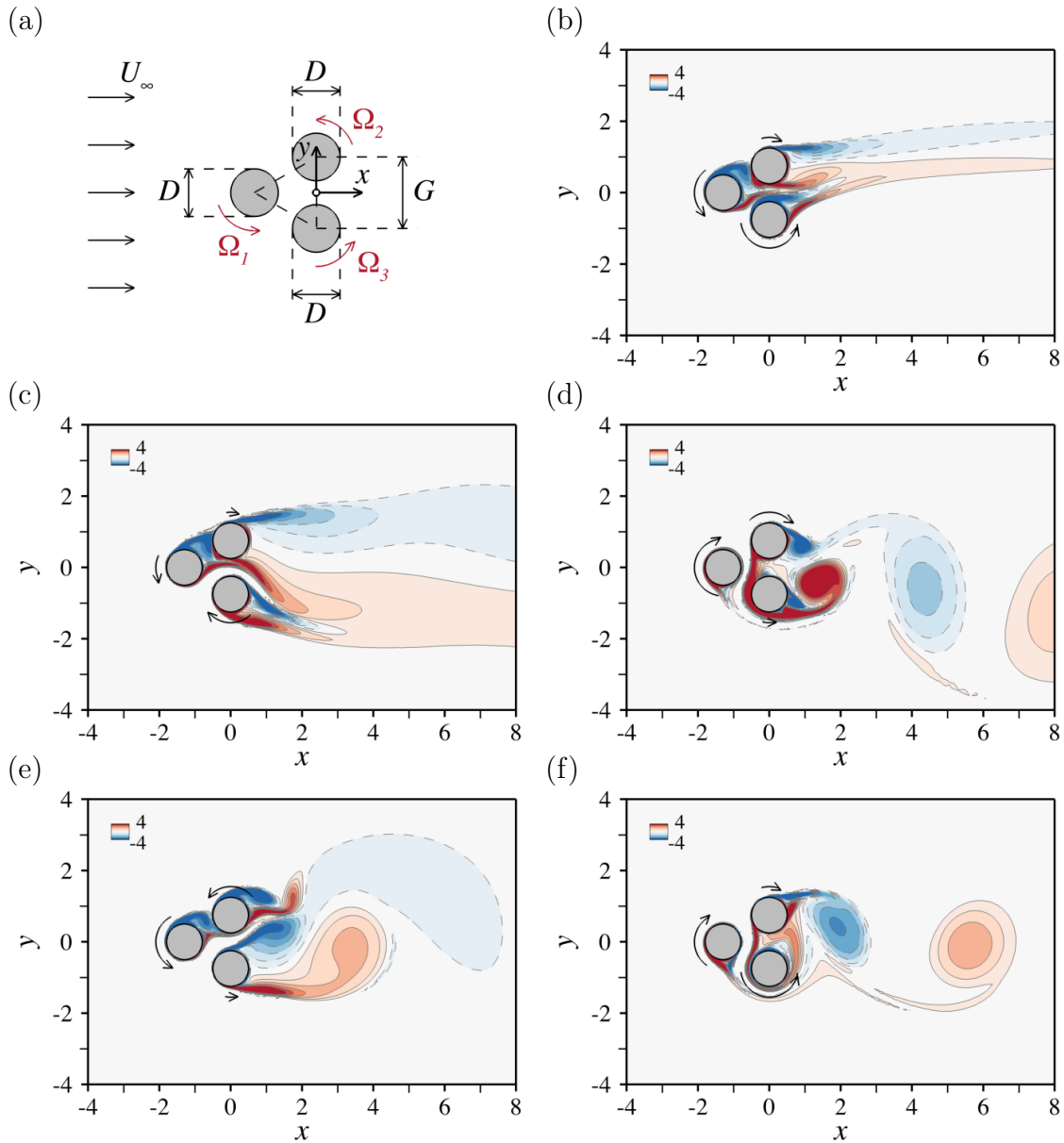


Figure 4.13: Open-loop control of a fluid pinball - (a) Schematic diagram of the configuration. (b-f) Iso-contours of the vorticity field computed at $Re = 2200$ for steady angular velocities $(\Omega_1, \Omega_2, \Omega_3)$ of the individual cylinders, namely (b) $(3.09, -1.05, 5.00)$, (c) $(1.46, -0.62, -2.75)$, (d) $(-5.00, -2.74, 0.81)$, (e) $(3.70, 3.00, 0.61)$ and (f) $(-3.48, -1.04, 5.00)$. The rotation directions are marked by the various arrows whose length is proportional to the angular velocity.

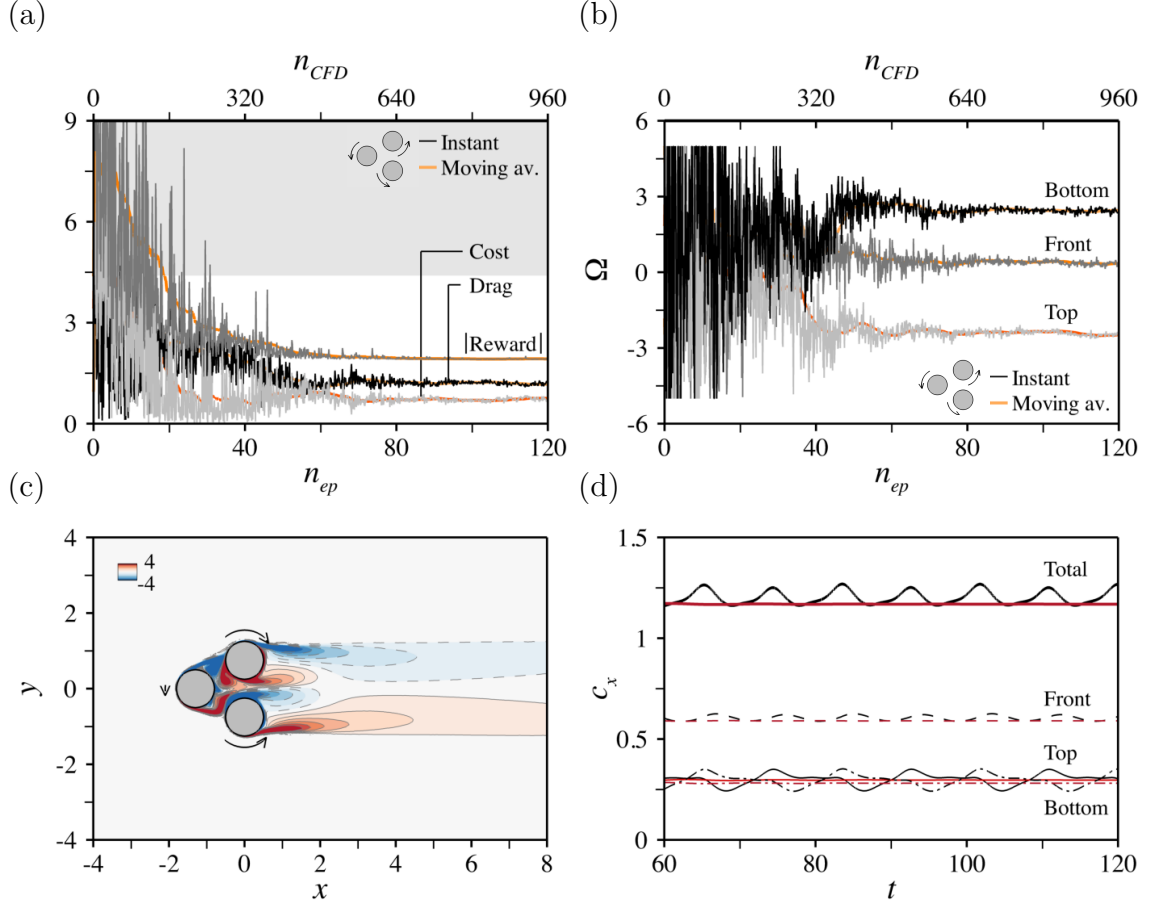


Figure 4.14: Open-loop control of a fluid pinball at $Re = 2200$ - (a) Evolution per episode for the instant (black line) and moving average (over episodes, light orange line) values of the mean drag (over time), together with related cost (light grey/dark orange) and reward (dark grey/orange) information computed under steady actuation for $\beta = 0.025$. The uncontrolled drag is at the bottom of the grey shaded area. (b) Same as (a) for the angular velocities of the front (black/light orange), top (dark grey/orange) and bottom cylinders (light grey/dark orange). (c) Iso-contours of the vorticity field computed under the optimal velocities $(\Omega_1^*, \Omega_2^*, \Omega_3^*) = (0.34, -2.49, 2.44)$. (d) Time history of drag computed under the sub-optimal velocities $(\Omega_1, \Omega_2, \Omega_3) = (0, -2.47, 2.47)$ (black lines), whose cost is identical to that of the optimal (red lines). The thick lines denote the drag of the three-cylinder system. The thin lines pertain to the front (dashed lines), top (solid lines) and bottom cylinders (dash-dotted lines).

for the non-dimensional angular velocities to vary in $[-\Omega_{\max}; \Omega_{\max}]$ with $\Omega_{\max} = 5$. The reward defined in (4.8) is computed using the simulation parameters docu-

mented in table 4.4, after which the network is updated for 32 epochs using 8 environments and 2 steps mini-batches. Note, rotation is actually ramped up over a time-span $[t_{\Omega_i}; t_{\Omega_f}]$ to smooth out the transient, using effective rates

$$\tilde{\Omega}_k(t) = \frac{\min(\max(t, t_{\Omega_i}), t_{\Omega_f}) - t_{\Omega_i}}{t_{\Omega_f} - t_{\Omega_i}} \Omega_k, \quad (4.10)$$

forced to zero on $[0, t_{\Omega_i}]$, to Ω_k on $[t_{\Omega_f}; t_f]$, and linearly increasing in between.

For this case, 120 episodes have been run, which represents 960 simulations, each of which lasts $\sim 3\text{h}20$ on 12 cores, hence $\sim 3200\text{h}$ of total CPU cost (equivalently, $\sim 400\text{h}$ of resolution time). The moving average reward reaches a plateau after about 80 episodes in figure 4.14(a), where the relevance of the weighing coefficient value $\beta = 0.025$ shows through the fact that the performance and cost components of the reward are of the same order of magnitude. The optimal value of drag $\bar{c}_x^* = 1.17 \pm 0.01$ computed as the average over the 5 latest episodes represents a tremendous reduction by almost 60% with respect to the uncontrolled value 2.91. The associated angular velocities whose evolution is depicted in figure 4.14(b) correspond to a boat tail-like arrangement, i.e., the top cylinder rotates clockwise ($\Omega_2^* = -2.49 \pm 0.01$), the bottom cylinder rotates counter-clockwise and almost symmetrically ($\Omega_3^* = 2.44 \pm 0.01$), and the front cylinder rotates more slowly and also counter-clockwise ($\Omega_1^* = 0.34 \pm 0.01$). The net rotation is thus in the same direction as the front cylinder, and we show in figure 4.14(c) that the tilting of the shear layers to the centerline alleviates the secondary flow from the gap between the two downstream cylinders, which is found to eventually suppress vortex shedding. Interestingly, an experimentally implemented machine learning approach using genetic algorithms yields similar optimal arrangements in [132]. For two different values of the weighing parameter, the authors therein report optimal angular velocities (0.68, -2.26 , 2.56) and (1.40, -1.70 , 2.04) and optimal drag reductions by 78% and 49%, respectively, but it is uneasy to push further the comparison because the latter study uses a different reward function in which drag is approximated from a small, discrete number of sensors distributed in the wake.

For the purpose of reducing drag, the above asymmetrical boat tailing actuation turns to be more efficient than its pure, symmetrical counterpart emulated by $(\Omega_1, \Omega_2, \Omega_3) = (0, -|\Omega|, |\Omega|)$.⁴ This is illustrated in figure 4.14(d) comparing the optimal drag to its symmetrical value computed with $|\Omega| = 2.47$ (to maintain the same cost efficiency, the associated drag reduction being by $\sim 58\%$). Pure boat tailing is insufficient to inhibit vortex shedding, as the symmetrical drag of all three individual cylinders is seen to exhibit small but finite-amplitude oscillations. Moreover, the drag of the downstream cylinders turns to be roughly identical on average.

⁴At least if β is large enough for cost to matter in the optimization procedure, otherwise the algorithm has been found to converge to the symmetrical boat tailing configuration $(0, -\Omega_{\max}, \Omega_{\max})$, and the reverse flow is completely suppressed (not shown here).

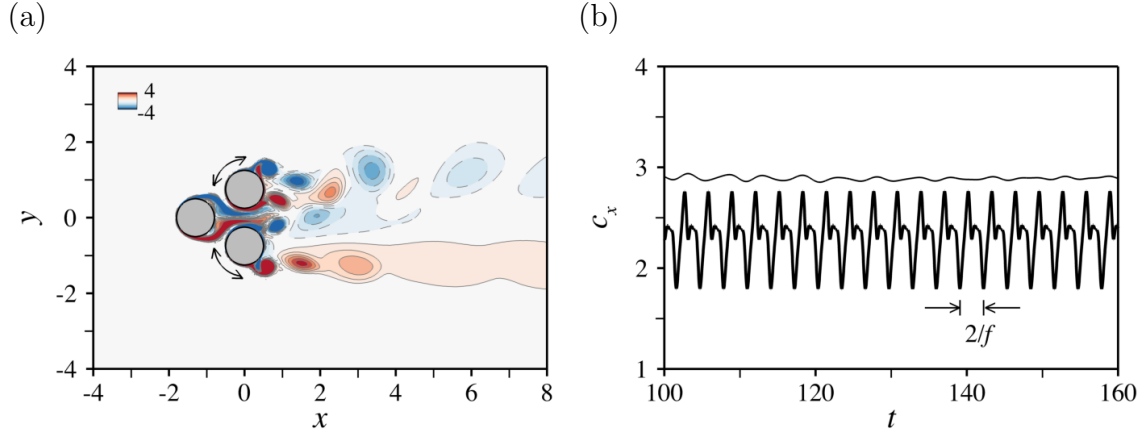


Figure 4.15: Open-loop control of a fluid pinball at $Re = 2200$ - (a) Iso-contours of the vorticity field and (b) time history of drag computed under periodic actuation (4.11) with angular velocity $\Omega = 2.47$ and frequency $f = 4f_0$. The thick and fine lines in (b) denote the controlled and uncontrolled values, respectively.

This suggests that the edge of asymmetrical over symmetrical boat tailing lies in its ability to reduce the drag of the front cylinder, an effect similar to that of suppressing vortex development and reducing drag by creating circulation around a single rotating bluff body [170]. Asymmetrical boat tailing is also more efficient than base bleed, another method widely used to reduce drag by blowing fluid directly into the wake, and that can be emulated by $(\Omega_1, \Omega_2, \Omega_3) = (0, |\Omega|, -|\Omega|)$ for the reverse rotation of the downstream cylinders to conversely enhance the gap flow in between them (not shown here).

4.4.2.2 Periodic actuation

Periodic actuation at frequency f has also been considered using a simplified configuration

$$\Omega_1 = 0, \quad \Omega_2 = -\Omega_3 = \Omega \sin(2\pi ft), \quad (4.11)$$

whose front cylinder is fixed, and whose downstream cylinders are periodically and symmetrically driven with maximum angular velocity Ω . Such a control oscillates between symmetrical boat tailing (found to be nearly-optimal under steady actuation) and base-bleed, and we assess the extent to which an additional degree of freedom (the oscillation frequency) creates room to improve the performance. The optimization relies on the compound reward

$$r = -\bar{D} - 2\beta|\Omega|^3, \quad (4.12)$$

computed using the same weighing parameter $\beta = 0.025$ as before. For each PPO-1 learning episode, the network outputs two values $\xi_{1,2}$ in $[-1; 1]^2$ mapped into

$$\Omega = \frac{1 + \xi_1}{2} \Omega_{\max}, \quad \frac{f}{f_0} = \frac{1 - \xi_2}{2} \lambda_{\min} + \frac{1 + \xi_2}{2} \lambda_{\max}, \quad (4.13)$$

where $f_0 = 0.16$ is the dominant frequency of vortex shedding computed in the absence of control. The angular velocity therefore varies in $[0; \Omega_{\max}]$ with $\Omega_{\max} = 5$ (the case $\Omega < 0$ is covered by periodicity) and the frequency ratio varies in $[\lambda_{\min}; \lambda_{\max}]$ with $\lambda_{\min} = 0.5$ and $\lambda_{\max} = 4$. This is a compromise between size of the parameter space and cost control, as investigating smaller frequencies would require to increase the averaging time-span, and resolving accurately larger frequencies would require to decrease the time-step. We shall not go into the details of the obtained results, because the frequency ratio ends up oscillating randomly in $[\lambda_{\min}; \lambda_{\max}]$, while the angular velocity converges to $\Omega^* = 0$. It is definitively possible to reduce drag under the considered periodic actuation, as we show for instance in figure 4.15 that a velocity $\Omega = 2.47$ (identical to that used previously to compare asymmetrical and symmetrical boat tailing) and a frequency ratio $\lambda = 4$ reduce drag by 20%, but the cost of doing so is too large, as the associated reward actually increases by 5% (note the period doubling bifurcation phenomenon in figure 4.15(b): drag is found to exhibit sub-harmonic oscillations at half the forcing frequency, which is a classical dynamical responses of harmonically forced nonlinear oscillators). These are only preliminary results intended to compare the efficiency of steady and periodic strategies using identical reward functions. We therefore defer to future work the computation of non-trivial periodic optimal distributions, for which it may be necessary to modify the reward function and/or to reduce the cost (by adequately decreasing the weighing parameter).



	r	\bar{c}_x	Ω_1	Ω_2	Ω_3		r	\bar{c}_x	Ω	f	
	-1.93	1.17	0.34	-2.47	2.44		-2.91	2.91	0	N/D	Optimal
											CFD
					2200						Reynolds number
				Steady							Actuation
				0.05					Periodic		Time-step
				[5; 10]					0.025		Rotation ramp-up time span
				[300; 400]							Averaging time span
				[-6; 20] × [-6; 6]							Mesh dimensions
				110000							Nb. mesh elements
				0.001							Interface ⊥ mesh size
				12							Nb. Cores
											PPO-1
					120						Nb. episodes
					8				40		Nb. environments
					32						Nb. epochs
					2						Size of mini-batches
					3200h						CPU time
					400h						Resolution time

Table 4.4: Simulation parameters and convergence data for open-loop control of a fluid pinball at $Re = 2200$. The interface mesh size yields ~ 20 grid points in the boundary-layer of the non-rotating front, top and bottom cylinder, just prior to separation, and the averaging time-span represents $\sim 15 - 20$ shedding cycles, depending on the angular velocities. For the periodic case, the time step yields ~ 60 data points over the smallest actuation period.

4.5 Discussion

This section is intended to provide insight into the efficiency of the single-step PPO algorithm compared to that of other well-established methods. We skip voluntarily DNS, as systematical optimization procedures are useless if a problem is simple enough that a small number of numerical simulations suffices to find the optimal. This is true of the optimization cases documented in section 4.3, although the results remain valuable to assess accuracy and highlight the limit of applying conservative policy updates to optimize sharp reward functions (that are common occurrence in low to moderate-Reynolds-number-fluid mechanical systems sustaining linear instabilities).

4.5.1 Adjoint methods

We begin with the adjoint method used in section 4.4.1 for systematic validation purposes. As explained in appendix 4.6, this is an approach intended to compute the drag of a control-induced disturbance modeled after the linearized governing equations forced by small-amplitude momentum source δf and wall velocity $\delta \mathbf{u}_w$, without ever computing the disturbance itself. The main assumptions and limitations at various levels of sophistication are reviewed in the appendix, so the line of thought is to describe only the specifics of the control problems considered herein. The general picture is that the baseline adjoint method is accurate and fairly efficient in terms of CPU cost, but demanding in terms of storage and increasingly difficult to apply rigorously when turbulence sets in (this is discussed in appendix 4.6). On the other hand, the frozen Reynolds stresses approximation has marginal CPU and storage costs, it carries over to any turbulence modeling under the so-called frozen viscosity assumption, but accuracy must be assessed on a case-by-case basis (see appendix 4.6).

4.5.1.1 Open-loop control by a small control cylinder

Open-loop control by a small control cylinder is a favorable case in the sense that only the center position of the control cylinder (not its shape, nor its size) is optimized, hence the adjoint problem needs be solved only once. Nonetheless, it comes with a substantial modeling component, as the source term δf used in the adjoint calculations must adequately represent the effect of a true control cylinder. We use here the pointwise reacting force proposed in [124], equal and opposite to the force felt by a control cylinder of same diameter in a uniform flow at the local, mean velocity. The latter is carefully crafted to reference data, but there are inherent approximations associated with overlooking the lift component of the force induced by the local velocity gradient (since the control cylinder, albeit small, has finite size)

and inertia (for the model force at each time instant to be the force that would act if the upstream flow at the same instant was a steady one). This can hurt accuracy and undermine the results in flow regions where the control cylinder drag is close to balancing the decrease in the drag of the main cylinder, all the more so in turbulent regimes where additional simplifications are needed to allow implementing the adjoint method itself (e.g., frozen eddy viscosity and/or Reynolds stresses).

In terms of pure performance, the baseline adjoint method is beyond compare for the laminar, steady case at $Re = 40$, because it merely requires solving a couple of steady solutions (one nonlinear, one linear), and PPO-1 would need converge in less than two episodes to approach that cost. Regarding the laminar, time-dependent case at $Re = 100$, the results reported herein rely on a naive implementation of the adjoint method: all time steps of the uncontrolled solution are written to disk, the adjoint equations are solved over the same time interval and with the same time step, and meaningful time averages of the adjoint-based integrands are computing after discarding the early and late time steps (corresponding to transients of the uncontrolled and adjoint solutions). In practice, this takes 45 Gb of storage. The cost of tackling similarly a three-dimensional (3-D) case with 40 points distributed in the span-wise direction would thus be about 2 Tb (as estimated by simple cross-multiplication), which is close to intractable without sophisticated integration, interpolation and/or checkpointing schemes. Meanwhile, the storage cost of PPO-1 is barely a few hundred Mb overall, and is expected to jump to a few ten Gb in 3-D without any additional development. As for CPU cost, the adjoint method amounts to roughly 7-8 episodes, which is about thrice as less as the number of episodes needed to achieve convergence with PPO-1 (this is an estimation for two numerical simulations oversized by the repeated IO calls, although an exact comparison is difficult because our DRL and adjoint results have been obtained using a different finite element codes on different hardware resources). Finally, for the turbulent cases at $Re = 3900$ and $Re = 22000$, the cost of the adjoint method is again marginal, as we relied on the frozen Reynolds stresses formulation for which it suffices to compute a nonlinear uncontrolled mean flow and a linear steady adjoint solution. PPO-1 would need to converge in one single episode to match the cost, but we believe the case at $Re = 3900$ to provide clear evidence that the simplifying assumptions can make it intricate to compare both qualitatively and quantitatively.

4.5.1.2 Open-loop control of a fluidic pinball

The adjoint modeling of the fluidic pinball is straightforward, since the wall velocity $\delta \mathbf{u}_w$ is simply the cylinder linear velocity. The challenge for this case rather lies in the large value of the optimal angular velocities (found to induce velocities close to the ambient velocity in the vicinity of the downstream cylinders), that suffice to invalidate the linearity assumption inherent to the adjoint method. On paper, this

problem can still be tackled with a nonlinear steepest descent algorithm recursively solving an adjoint problem and modifying the control parameters in the direction of the negative gradient. While it usually takes about ten iterations for fluid mechanical systems to converge (provided relevant update strategy and descent step are used), we did not attempt to do so, as it would magnify the limitations of the adjoint method underlined in the appendix. Namely, the storage cost would increase (even a simple conjugate gradient algorithm would require availability of multiple time histories of adjoint solutions) and convergence could be weakened or even sapped if the simplifications made in turbulent regimes yield inaccurate gradient evaluations.

4.5.2 Evolution strategies

Evolution strategies (ES) are another popular family of division of population-based algorithms performing black-box optimization in continuous search spaces without computing directly the gradient of the target function. ES imitate principles of organic evolution processes as rules for optimum seeking procedures, using repeated interplay of variation (via recombination and mutation) and selection in populations of candidate solutions. They rely on a stochastic description of the variables to optimize, i.e., they consider probability density functions instead of deterministic variables. At each generation (or iteration) new candidate solutions are sampled isotropically by variation of the current parental individuals according to a multivariate normal distribution. After applying recombination and mutation transformations (respectively amounting to selecting a new mean for the distribution, and to adding a random perturbation with zero mean), the individuals with the highest cost function are then selected to become the parents in the next generation. Improved variants include the covariance matrix adaptation evolution strategy (CMA-ES), that also updates its full covariance matrix to accelerate convergence toward the optimum (which amounts to learning a second-order model of the underlying objective function).

As has been said for introductory purposes, it lies out of the scope of this chapter to provide exhaustive performance comparison data against state-of-the-art evolution algorithms. The efforts for developing single-step PPO remain at an early stage, so we do not expect the method to be able to compete right away. Nonetheless, we do not expect it to be utterly outmatched either, as genetic algorithms⁵ have been shown capable to learn optimal open- and closed-loop control strategies within a few hundreds to a few thousands test runs (see [171] and the references therein), and

⁵Another class of evolutionary algorithms with slightly different implementation details. Namely, most parameters in genetic algorithms (GA) are exogenous, i.e., set by the practitioner, while ES features endogenous parameters associated with individuals, that evolve together with them. Also, only the fittest individuals are selected to become parents in GA, while parents are selected randomly in ES and the fittest offsprings are selected and inserted in the next generation.

it takes a few hundred (resp. less than one thousand) simulations for single-step PPO to learn the optimal open-loop strategy for control by a small cylinder (resp. for control of the fluidic pinball). In present form, the method can be thought as an evolutionary-like algorithm with simpler heuristics (i.e., without an evolutionary update strategy, since the optimal model parameters are learnt via gradient ascent). Its performance should thus be comparable to that of standard ES methods with isotropic covariance matrix, meaning that further characterization and fine-tuning, as well as pre-trained deep learning models (as is done in transfer learning) are likely required to outperform more advanced methods.

4.6 Conclusion

Open-loop control of laminar and turbulent flow past bluff bodies is achieved here training a fully connected network with a novel single-step PPO 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 finite elements CFD environment solving stabilized weak forms of the governing equations (Navier–Stokes, otherwise uRANS with negative Spalart–Allmaras as turbulence model) with a combination of variational multiscale approach, immersed volume method and anisotropic mesh adaptation.

Convergence and accuracy are assessed from two optimization cases (maximizing the mean lift of a NACA 0012 airfoil and the fluctuating lift of two side-by-side circular cylinders, both in laminar regimes). Those are simple enough to allow comparison to in-house DNS data, yet they stress that the occurrence of instability yields sharp reward functions for which the conservative policy updates specific to PPO can trap the optimization process into local optima. The method is also applied to two open-loop control problems whose parameter spaces are large enough to dismiss DNS. Single-step PPO is found to successfully reduce the drag of laminar and turbulent cylinder flows by mapping the best positions for placement of a small control cylinder in good agreement with reference data obtained by the adjoint method. The achieved reduction ranges from 2% using a circular geometry of the main cylinder at $Re = 40$, up to 30% using a square geometry at $Re = 22000$. Second, the method proves fruitful to reduce the drag of the fluidic pinball, an arrangement of three identical, rotating circular cylinders immersed in a turbulent stream. An optimal reduction by almost 60% (consistent with that recently obtained using genetic algorithms) is reported using a boat tailing actuation made up of a slowly rotating front cylinder and two downstream cylinders rotating in opposite directions so as to reduce the gap flow in between them. For both cases, convergence is reached after a few ten episodes, which represents a few hundreds CFD runs. Exhaustive computational efficiency data are reported with the hope to foster future

comparisons, but it is worth emphasizing that we did not seek to optimize said efficiency, neither by optimizing the hyper parameters, nor by using pre-trained deep learning models.

Fluid dynamicists have just begun to gauge the relevance of deep reinforcement learning techniques to assist the design of optimal flow control strategies. This research weighs in on this issue and shows that the proposed single-step PPO holds a high potential as a reliable, go-to black-box optimizer for complex CFD problems. The one advantages here are scope and applicability, as the storage cost of an episode is simply that of a CFD run (times the number of environments), and there is no prerequisite beyond the ability to compute accurate numerical solutions (which behaves the CFD solver, not the RL algorithm). Consequently, we would not anticipate any additional numerical developments before tackling a 3-D turbulent flow with the same CFD environment, even with a more sophisticated turbulence modeling (since the built-in small-scale component of the VMS solution also acts as an implicit LES). Despite these achievements, further development, characterization and fine-tuning are needed to consolidate the acquired knowledge, whether it be via an improved balance between exploration and exploitation to deal with steep global maxima (for instance using Trust Region-Guided PPO, as it effectively encourages the policy to explore more on the potential valuable actions, no matter whether they were preferred by the previous policies or not), via non-normal probability density functions to deal with multiple global maxima, or via coupling with a surrogate model trained on-the-fly.

Appendix: A quick survey of adjoint-based optimization

We briefly review here the various adjoint frameworks used in section 4.4.1 for systematic validation purposes of the PPO-1 results. The starting point is a so-called uncontrolled solution (\mathbf{u}, p) to the non-linear equations of motion (Navier–Stokes, unless specified otherwise) forced by a momentum source f and a velocity \mathbf{u}_w distributed over all solid surfaces Γ_w in the computational domain (although it is possible to restrict to a subset).

A. Baseline adjoint method

The adjoint method computes the change in drag induced by small variations $(\delta f, \delta \mathbf{u}_w)$ of these control parameters as

$$\delta \bar{c}_x = \int_{\Omega} \overline{\mathbf{u}^\dagger \cdot \delta f} \, ds + \int_{\Gamma_w} \overline{(\boldsymbol{\sigma}^\dagger(-p^\dagger, \mathbf{u}^\dagger) \cdot \mathbf{n}) \cdot \delta \mathbf{u}_w} \, dl, \quad (4.14)$$

where \mathbf{n} is the unit outward normal to Γ_w and we note $\boldsymbol{\sigma}^\dagger(-p^\dagger, \mathbf{u}^\dagger) = p^\dagger \mathbf{I} + \frac{1}{\text{Re}} \nabla \mathbf{u}^\dagger$. Finally, $(\mathbf{u}^\dagger, p^\dagger)$ are adjoint velocity and pressure fields solution to

$$\nabla \cdot \mathbf{u}^\dagger = 0, \quad -(\partial_t \mathbf{u}^\dagger + \nabla \mathbf{u}^\dagger \cdot \mathbf{u}) + \nabla \mathbf{u}^T \cdot \mathbf{u}^\dagger + \nabla \cdot \boldsymbol{\sigma}^\dagger(-p^\dagger, \mathbf{u}^\dagger) = \mathbf{0}, \quad (4.15)$$

forced at Γ_w by a velocity equal to twice the ambient velocity (the factor of 2 stems from the definition of dynamic pressure), as obtained multiplying \mathbf{u}^\dagger and p^\dagger onto the linearized momentum and continuity equations, using the divergence theorem to integrate by parts over the computational domain, and integrating in time over the span of the simulation. In essence, this amounts to computing the drag of the control induced disturbance modeled after the forced, linearized Navier–Stokes equations, without ever computing the disturbance itself.

A typical implementation consists of two sequential numerical simulations (for the uncontrolled and adjoint solutions, respectively) plus a series of vector dot products, to give the drag variation at each grid point. This is simple on paper, but the method has some limitations :

- the adjoint equations are problem-specific and must be derived and implemented manually on a case-by-case basis.
- the cost is marginal in steady flow regimes, because the time-independence of the uncontrolled solution makes the adjoint problem purely linear. Otherwise, the entire time history of uncontrolled solutions must be available at every adjoint time step because of the reversal of space-time directionality; see the minus sign ahead of the material derivative term in eqs. (4.15). This is very demanding in terms of storage (the repeated IO also increases the computational burden compared to a classical CFD run with identical simulation parameters) but these issues can be mitigated using checkpointing [172] and high-order time-integration and interpolation schemes [173].
- not all cost functions are admissible due to the need for consistent adjoint boundary conditions, although this can be overcome with augmented Lagrangian methods based on auxiliary boundary equations [174].
- applicability to high-fidelity turbulence modeling is uncertain because the noise-induced sensitivity to initial conditions (the “butterfly effect”) is expected to yield exponentially diverging solutions if the length of the adjoint simulation exceeds the predictability time scale. Possible solutions include averaging over a large number of ensemble calculations [175] (which increases significantly the computational cost and decreases the attractiveness of the method) or invoking sophisticated shadowing and space-split techniques sampling on selected flow trajectories [176, 177] (which comes

at the cost of ease of implementation). Moreover, the literature somehow oddly reports several cases of turbulent adjoint solution blowing up in 2-D [178, 179] and 3-D [180], but also several instances in 3-D where no blow-up is observed [181–183].

- applicability to RANS simulations is conversely generally acknowledged. However, discarding the linearization and adjointization of even the simplest turbulence models (using the so-called frozen eddy-viscosity approximation) to avoid massive debugging and validation efforts has somehow become standard lore, even though completeness and exactness are required to ensure numerical accuracy and avoid diverging adjoint solutions due to error propagation and amplification.

B. Frozen Reynolds stresses approximation

A simple adjoint formalism has been proposed in [124] to provide insight into the reliability of adjoint-based predictions in practical situations where no complete history of time and space-accurate solutions is available. The approach is closely related to existing studies considering the mean flow an admissible solution for linear stability analysis, as it simply dismisses the way the control-induced modification to the fluctuating uncontrolled solution feeds back onto the mean (hence the frozen Reynolds stress moniker to echo the above frozen eddy viscosity). In doing so, (4.14) can be shown to reduce to

$$\delta \overline{\overline{c_x}} = \int_{\Omega} \overline{\overline{\mathbf{u}^\dagger}} \cdot \overline{\delta f} \, ds + \int_{\Gamma_s} (\boldsymbol{\sigma}^\dagger(\overline{\overline{p^\dagger}}, \overline{\overline{\mathbf{u}^\dagger}}) \cdot \mathbf{n}) \cdot \overline{\delta \mathbf{u}_w} \, dl, \quad (4.16)$$

where the double overline denotes approximations to the true time-averaged quantities, and the adjoint velocity and pressure fields are solution to

$$\nabla \cdot \overline{\overline{\mathbf{u}^\dagger}} = 0, \quad -\nabla \overline{\overline{\mathbf{u}^\dagger}} \cdot \overline{\mathbf{u}} + \nabla \overline{\mathbf{u}^T} \cdot \overline{\overline{\mathbf{u}^\dagger}} + \nabla \cdot \boldsymbol{\sigma}(-\overline{\overline{p^\dagger}}, \overline{\overline{\mathbf{u}^\dagger}}) = \mathbf{0}, \quad (4.17)$$

forced at Γ_w by the same velocity equal to twice the ambient velocity. The strength of the approach lies in the fact that once the mean uncontrolled solution is known, computing the approximated adjoint solution merely requires solving a single linear problem. Accuracy must be assessed on a case-by-case basis, but the computational and storage costs of doing so are marginal, and the approach carries over to any turbulence modeling method under the frozen viscosity assumption.

Chapter 5

DRL for the control of conjugate heat transfer

Contents

5.1	Introduction	101
5.2	Computational fluid dynamics	103
5.2.1	The immersed volume method	104
5.2.2	Variational multi-scale approach (VMS)	106
5.3	Deep reinforcement learning and proximal policy optimization	108
5.3.1	Neural networks	108
5.3.2	Deep reinforcement learning	108
5.3.3	From policy methods to Proximal policy optimization	110
5.3.4	Single-step PPO	113
5.3.5	Numerical implementation	115
5.4	Control of natural convection in 2-D closed cavity	115
5.4.1	Case description	115
5.4.2	Control	118
5.4.3	Results	119
5.5	Control of forced convection in 2-D open cavity	122
5.5.1	Case description	122
5.5.2	Control	123
5.5.3	Results	124
5.5.4	Discussion	133
5.6	Extension to 3-D forced convection	135

5.6.1	Case description	135
5.6.2	Control strategy	136
5.6.3	Results	138
5.7	Conclusion	140

This chapter is presented as a self-contained article published in 2021 in the Journal of Computational Physics [184], hence, some details (related to motivation, scope and/or methodology) are repeated from chapters 1-3 to preserve the consistency of the whole content.

Ce chapitre évalue la capacité des techniques d'apprentissage par renforcement profond (DRL) à aider au contrôle des systèmes de transfert de chaleur conjugués régis par les équations couplées de Navier–Stokes et de la chaleur. Il utilise une nouvelle version "dégénérée" de l'algorithme Proximal Policy Optimization (PPO), destinée aux situations où la politique optimale à apprendre par un réseau neuronal ne dépend pas de l'état, comme c'est notamment le cas dans les problèmes d'optimisation et de contrôle en boucle ouverte. La récompense fournie au réseau neuronal est calculée à l'aide d'un environnement interne d'éléments finis stabilisés combinant une modélisation multi-échelle variationnelle (VMS) des équations gouvernantes, une méthode de volume immergé et une adaptation de maillage anisotrope multi-composant. Plusieurs cas d'essai de convection naturelle et forcée en deux et trois dimensions sont utilisés comme banc d'essai pour développer la méthodologie. L'approche atténuée avec succès l'augmentation du transfert de chaleur induite par la convection naturelle dans une cavité carrée bidimensionnelle à chauffage différentiel contrôlée par des fluctuations constantes par morceaux de la température de la paroi latérale. Elle s'avère également capable d'améliorer l'homogénéité de la température à la surface de pièces chaudes bidimensionnelles et tridimensionnelles sous refroidissement par impact. Divers cas sont abordés, dans lesquels la position de plusieurs injecteurs d'air froid est optimisée par rapport à une position fixe de la pièce. La flexibilité du cadre numérique permet de résoudre également le problème inverse, c'est-à-dire d'optimiser la position de la pièce par rapport à une distribution fixe des injecteurs. Les résultats obtenus montrent le potentiel de la méthode pour l'optimisation en boîte noire de systèmes de transfert de chaleur conjugués en dynamique des fluides numérique (CFD). De manière plus significative, ils soulignent comment la méthode DRL peut révéler des solutions ou des relations de paramètres non anticipées (comme la position optimale de la pièce sous actionnement symétrique qui s'avère être décalée de l'axe de symétrie), en plus d'être un outil pour optimiser les recherches dans de grands espaces de paramètres.

This chapter gauges the ability of deep reinforcement learning (DRL) techniques to assist the control of conjugate heat transfer systems governed by the coupled Navier–Stokes and heat equations. It uses a novel, "degenerate" version of the proximal policy optimization (PPO) algorithm, intended for situations where the optimal policy to be learnt by a neural network does not depend on state, as is notably the case in optimization and open-loop control problems. The numerical reward fed to the neural network is computed with an in-house stabilized finite elements environment combining variational multi-scale (VMS) modeling of the governing equations,

immerse volume method, and multi-component anisotropic mesh adaptation. Several test cases of natural and forced convection in two and three dimensions are used as testbed for developing the methodology. The approach successfully alleviates the natural convection induced enhancement of heat transfer in a two-dimensional, differentially heated square cavity controlled by piece-wise constant fluctuations of the sidewall temperature. It also proves capable of improving the homogeneity of temperature across the surface of two and three-dimensional hot workpieces under impingement cooling. Various cases are tackled, in which the position of multiple cold air injectors is optimized relative to a fixed workpiece position. The flexibility of the numerical framework makes it tractable to solve also the inverse problem, i.e., to optimize the workpiece position relative to a fixed injector distribution. The obtained results showcase the potential of the method for black-box optimization of practically meaningful computational fluid dynamics (CFD) conjugate heat transfer systems. More significantly, they stress how DRL can reveal unanticipated solutions or parameter relations (as the optimal workpiece position under symmetrical actuation turns to be offset from the symmetry axis), in addition to being a tool for optimizing searches in large parameter spaces.

5.1 Introduction

Thermal control, defined as the ability to finesse the thermal properties of a volume of fluid (and of the solid objects inside) into a certain desired state, is a field of tremendous societal and economical importance. For instance, heat/cool exchangers are used in a broad range of industrial applications to regulate process temperatures by heat or cool transfer between fluid media, which in turn ensures that machinery, chemicals, water, gas, and other substances remain within safe operating conditions. Green building engineering is another field whose focus is on regulating indoor thermal conditions (temperature, humidity) under substantial variations of the ambient conditions to provide high-quality living and working environments. In many manufacturing processes, thermal conditioning is also intended to improve the final mechanical (e.g., hardness, toughness, resistance), electrical, or optical properties of the product, the general picture being that high temperature gradients are useful to speed up the process but generally harm the quality of the outcome because of heat transfer inhomogeneities caused by the increased convection by the fluid particles. All such problems fall under the purview of this line of study.

Numerous strategies have been implemented to control fluid mechanical systems (including conjugate heat transfer systems combining thermal conduction in the solid and convective transfer in the fluid), either open-loop with passive appendices (e.g., end plate, splitter plate, small secondary cylinder, or flexible tail), or open-loop with actuating devices (e.g., plasma actuation, boundary temperatures, steady or unsteady base bleeding, rotation) or closed-loop (e.g. via transverse motion, perturbations of the thermal boundary layer, blowing/suction, rotation, all relying on an appropriate sensing of flow variables). Nonetheless, many of the proposed strategies are trial and error, and therefore require extensive and costly experimental or numerical campaigns. This has motivated the development of analytical methods and numerical algorithms for the optimal control of Navier–Stokes systems [116, 118, 185], and the maturing of mathematical methods in flow control and discrete concepts for PDE constrained optimization. Applications to the heat equation [186] and the coupled Navier–Stokes and heat equations [187–190] have also been considered, including fresh developments meant to alter the linear amplification of flow disturbances [191], but the general picture remains that the optimal control of conducting-convecting (possibly radiating) fluids has not been extensively studied.

The premise of this research is that the related task of selecting an optimal subset of control parameters can alternatively be assisted by machine learning algorithms. Indeed, the introduction of the back-propagation algorithm [62] has progressively turned Artificial Neural Networks (ANN) into a family of versatile parametric tools that can be trained to hierarchically extract informative features from data and to provide qualitative and quantitative modeling predictions. Together with the in-

creased affordability of high-performance computational hardware, this has allowed leveraging the ever-increasing volume of data generated for research and engineering purposes into novel insight and actionable information, which in turn has entirely transformed scientific disciplines, such as robotics [64, 126] or image analysis [125]. Owing to the ability of neural networks to handle stiff, large-scale nonlinear problems [127], machine learning algorithms have also been making rapid inroads in fluid mechanics, as a mean to solve the Navier–Stokes equations [128] or to predict closure terms in turbulence models [129]; see also Ref. [136] for an overview of the current developments and opportunities.

Neural networks can also be used to solve decision-making problems, which is the purpose of Deep Reinforcement Learning (DRL, where the *deep* terminology generally weighs on the sizable depth of the network), an advanced branch of machine learning. Simply put, a neural network trains in finding out which actions or succession of actions maximize a numerical reward signal, with the possibility for a given action to affect not only the immediate but also the future rewards. Successful applications of DRL range from AlphaGo, the well-known ANN that defeated the top-level human player at the game of Go [48] to the real-world deployment of legged robots [57], to breakthroughs in computer vision (e.g., filtering, or extracting image features) [59] and optimal control problems [18, 61]. There is also great potential for applying DRL to fluid mechanics, for which efforts are ongoing but still at an early stage. Sustained commitment from the machine learning community has allowed expanding the scope from computationally inexpensive, low-dimensional reductions of the underlying fluid dynamics [12–14] to complex Navier–Stokes systems [37, 38], with a handful of pioneering studies providing insight into the performance improvements to be delivered in shape optimization [21, 138, 192] and flow control [39, 41, 43, 49, 139–141], including recent advances assessing experimentally the effectiveness of reinforcement learning control strategies [15]. The literature on thermal control is even more scarce, as our literature review did not reveal any other study considering DRL-based control of conjugate heat transfer aside from [17], another research effort conducted in the same time frame as the present work that will be discussed further on, plus a few other publications relying on dealing with energy efficiency in civil engineering from low-dimensional thermodynamic models basically unrelated to the equations of fluid dynamics [193, 194].

This research assesses the feasibility of using proximal policy optimization (PPO [18]) for control and optimization purposes of conjugate heat transfer systems, as governed by the coupled Navier–Stokes and heat equations. The objective here is to keep shaping the capabilities of the method (PPO is still a relatively newcomer that has quickly emerged as the go-to DRL algorithm due to its data efficiency, simplicity of implementation and reliable performance) and to narrow the gap between DRL and advanced numerical methods for multi-scale, multi-physics computational fluid

dynamics (CFD). We investigate more specifically the “degenerate” single-step PPO algorithm introduced in [138] for optimization and open-loop control problems, as the optimal policy to be learnt is then state-independent, and it may be enough for the neural network to get only one attempt per episode at finding the optimal. Several problems of conjugate heat transfer in two and three dimensions are used as testbed to push forward the development of this novel approach, whose potential for reliable black-box optimization of computational fluid dynamics (CFD) systems has been recently assessed for open-loop drag reduction in cylinder flows at Reynolds numbers ranging from a few hundreds to a few ten thousands [69]. To the best of the authors knowledge, this constitutes the first attempt to achieve DRL-based control of conjugate *forced* convection heat transfer, while [17] is the first attempt to achieve DRL control of conjugate *natural* convection heat transfer.

The organization is as follows: section 5.2 outlines the main features of the finite element CFD environment used to compute the numerical reward fed to the neural network, that combines variational multi-scale (VMS) modeling of the governing equations, immerse volume method, and multi-component anisotropic mesh adaptation. The baseline principles and assumptions of DRL and PPO are presented in section 5.3, together with the specifics of the single-step PPO algorithm. Section 5.4 revisits the natural convection case of [17] for the purpose of validation and assessment part of the method capabilities. In section 5.5, DRL is used to control conjugate heat transfer in a model setup of two-dimensional workpiece cooling by impingement of a fluid. An extension to three-dimensional workpieces is proposed in section 5.6.

5.2 Computational fluid dynamics

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

$$\nabla \cdot \mathbf{u} = 0, \quad (5.1)$$

$$\rho(\partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u}) = \nabla \cdot (-p\mathbf{I} + 2\mu\boldsymbol{\epsilon}(\mathbf{u})) + \boldsymbol{\psi}, \quad (5.2)$$

$$\rho c_p(\partial_t T + \mathbf{u} \cdot \nabla T) = \nabla \cdot (\lambda \nabla T) + \chi, \quad (5.3)$$

where \mathbf{u} is the velocity field, p is the pressure, T is the temperature, $\boldsymbol{\epsilon}(\mathbf{u}) = (\nabla \mathbf{u} + \nabla \mathbf{u}^T)/2$ is the rate of deformation tensor, $\boldsymbol{\psi}$ and χ are source terms (modeling, e.g., buoyancy or radiative heat transfer), and we assume here constant fluid density ρ , dynamic viscosity μ , thermal conductivity λ , and specific heat c_p .

5.2.1 The immersed volume method

The numerical modeling of conjugate heat transfer mostly depends upon a heat transfer coefficient to ensure that the proper amount of heat is exchanged at the fluid/solid interface via thermal boundary conditions. Computing said coefficient is no small task (as it requires solving an inverse problem to assimilate relevant experimental data, which in turn requires such data to be available), and is generally acknowledged to be a limiting issue for practical applications where one must vary, e.g., the shape, number and position of the solid, or the fluid and/or solid material properties. We thus rather use here the immersed volume method (IVM) to combine both the fluid and solid phases into a single fluid with variable material properties. Simply put, we solve equations formally identical to (5.1)-(5.3) on a unique computational domain Ω , but with variable density, dynamic viscosity, conductivity, and specific heat, which removes the need for a heat transfer coefficient since the amount of heat exchanged at the interface then proceeds solely from the individual material properties on either side of it. In order to ensure numerical accuracy, such an approach must combine three key ingredients, that are briefly reviewed in the next paragraphs: an interface capturing method, anisotropic mesh adaptation to achieve a high-fidelity description of said interface, and relevant mixing laws to describe the properties of the composite fluid. One point worth mentioning is that the interface here is static, although the same numerical framework can be used to dynamically track moving interfaces, and thus to encompass the effect of solid displacements. This is because the solid is fixed once an action has been taken by the PPO agent, although not fixed over the course of optimization, as the solid position can very well be the quantity subjected to optimization, as illustrated in section 5.5.3.4.

- **Level set method** The level set approach is used to localize the fluid/solid interface by the zero iso-value of a smooth function. In practice, a signed distance function ϕ is used to localize the interface and initialize the material properties on both either side of it, with the convention that $\phi > 0$ (resp. $\phi < 0$) in the fluid (resp. the solid).

- **Anisotropic mesh adaptation** The interface may intersect arbitrarily the mesh elements if it is not aligned with the element edges, in which case discontinuous material properties across the interface can yield oscillations of the numerical solutions. We thus use the anisotropic mesh adaptation technique presented in [103] to ensure that the material properties are distributed as accurately and smoothly as possible over the smallest possible thickness around the interface. This is done computing modified distances from a symmetric positive defined tensor (the metric) whose

eigenvectors define preferential directions along which mesh sizes can be prescribed from the related eigenvalues. The metric used here is isotropic far from the interface, with mesh size set equal to h_∞ in all directions, but anisotropic near the interface, with mesh size equal to h_\perp in the direction normal to the interface, and to h_∞ in the other directions. This is written for an intended thickness δ as

$$\mathbf{M} = K(\phi)\mathbf{n} \otimes \mathbf{n} + \frac{1}{h_\infty^2}\mathbf{I} \quad \text{with} \quad K(\phi) = \begin{cases} 0 & \text{if } |\phi| \geq \delta/2, \\ \frac{1}{h_\perp^2} - \frac{1}{h_\infty^2} & \text{if } |\phi| < \delta/2, \end{cases} \quad (5.4)$$

where $\mathbf{n} = \nabla\phi/|\nabla\phi|$ is the unit normal to the fluid/solid interface computed from the level set gradient. A posteriori anisotropic error estimator is then used to minimize the interpolation error under the constraint of a fixed number of edges in the mesh. A unique metric can be built from multi-component error vectors [103], which is especially relevant for conjugate heat transfer optimization, as it allows each learning episode to use an equally accurate mesh adapted from the velocity vector and magnitude, the temperature field, and the level set.

- **Mixing laws** The composite density, dynamic viscosity and specific heat featured in equations (5.1)-(5.3) are computed as the arithmetic means of the fluid and solid values, for instance the composite density is

$$\rho = \rho_f H_\epsilon(\phi) + \rho_s(1 - H_\epsilon(\phi)), \quad (5.5)$$

where H_ϵ is the smoothed Heaviside function defined as

$$H_\epsilon(\phi) = \begin{cases} 0 & \text{if } \phi < -\epsilon, \\ \frac{1}{2}\left(1 + \frac{\phi}{\epsilon} + \frac{1}{\pi}\sin\left(\pi\frac{\phi}{\epsilon}\right)\right) & \text{if } |\phi| \leq \epsilon, \\ 1 & \text{if } \phi > \epsilon, \end{cases} \quad (5.6)$$

and ϵ is a regularization parameter proportional to the mesh size in the normal direction to the interface, set here to $\epsilon = 2h_\perp$. In order to ensure continuity of the heat flux across the interface, the thermal conductivity is computed as the harmonic mean

$$\frac{1}{\lambda} = \frac{1}{\lambda_f} H_\epsilon(\phi) + \frac{1}{\lambda_s} (1 - H_\epsilon(\phi)), \quad (5.7)$$

as obtained from a steady, no source, one dimensional analysis of the heat flux when the conductivity varies stepwise from one medium to the next; see [105] for detailed derivation and analysis, and [195] for proof of the gain in numerical accuracy (with respect to the arithmetic mean model) by comparison with analytical solutions.

5.2.2 Variational multi-scale approach (VMS)

In the context of finite element methods (that remain widely used to simulate engineering CFD systems due to their ability to handle complex geometries), direct numerical simulation (DNS) solves the weak form of (5.1)-(5.3), obtained by integrating by parts the pressure, viscous and conductive terms, to give

$$(\rho(\partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u}), \mathbf{w}) + (2\mu \boldsymbol{\epsilon}(\mathbf{u}), \boldsymbol{\epsilon}(\mathbf{w})) - (p, \nabla \cdot \mathbf{w}) + (\nabla \cdot \mathbf{u}, q) = (\boldsymbol{\psi}, \mathbf{w}), \quad (5.8)$$

$$(\rho c_p(\partial_t T + \mathbf{u} \cdot \nabla T), s) + (\lambda \nabla T, \nabla s) = (\chi, s), \quad (5.9)$$

where (\cdot, \cdot) is the L^2 inner product on the computational domain, \mathbf{w} , q and s are relevant test functions for the velocity, pressure and temperature variables, and all fluid properties are those mixed with the smoothed Heaviside function (5.6).

We use here the variational multi-scale (VMS) approach [79, 144, 145] to solve a stabilized formulation of (5.8)-(5.9), which allows circumventing the Babuska—Brezzi condition (that otherwise imposes that different interpolation orders be used to discretize the velocity and pressure variables, while we use here simple continuous piecewise linear P_1 elements for all variables) and prevents numerical instabilities in convection regimes at high Reynolds numbers. We shall not go into the extensive details about the derivation of the stabilized formulations, for which the reader is referred to [101, 147]. Suffice it to say here that the flow quantities are split into coarse and fine scale components, that correspond to different levels of resolution. The fine scales are solved in an approximate manner to allow modeling their effect into the large-scale equations. This gives rise to additional terms in the right-hand side of (5.8)-(5.9), and yields the following weak forms for the large scale

$$\begin{aligned} &(\rho(\partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u}), \mathbf{w}) + (2\mu \boldsymbol{\epsilon}(\mathbf{u}), \boldsymbol{\epsilon}(\mathbf{w})) - (p, \nabla \cdot \mathbf{w}) + (\nabla \cdot \mathbf{u}, q) = (\boldsymbol{\psi}, \mathbf{w}) \\ &+ \sum_{K \in \mathcal{T}_h} [(\tau_1 \mathcal{R}_M, \mathbf{u} \cdot \nabla \mathbf{w})_K + (\tau_1 \mathcal{R}_M, \nabla q)_K + (\tau_2 \mathcal{R}_C, \nabla \cdot \mathbf{w})_K], \end{aligned} \quad (5.10)$$

$$\begin{aligned} &(\rho c_p(\partial_t T + \mathbf{u} \cdot \nabla T), s) + (\lambda \nabla T, \nabla s) = (\chi, s) \\ &+ \sum_{K \in \mathcal{T}_h} [(\tau_3 \mathcal{R}_T, \mathbf{u} \cdot \nabla s)_K + (\tau_4 \mathcal{R}_T, \zeta \nabla T \cdot \nabla s)_K], \end{aligned} \quad (5.11)$$

where $(\cdot, \cdot)_K$ is the inner product on element K , we denote by $\zeta = \mathbf{u} \cdot \nabla T / \|\nabla T\|^2$ the (normalized) velocity projected along the direction of the temperature gradient, and the \mathcal{R} terms are the governing equations residuals

$$\begin{aligned} -\mathcal{R}_C &= \nabla \cdot \mathbf{u}, & -\mathcal{R}_M &= \rho(\partial_t \mathbf{u} + \mathbf{u} \cdot \nabla \mathbf{u}) + \nabla p - \boldsymbol{\psi} \\ & & -\mathcal{R}_T &= \rho c_p(\partial_t T + \mathbf{u} \cdot \nabla T) - \chi, \end{aligned} \quad (5.12)$$

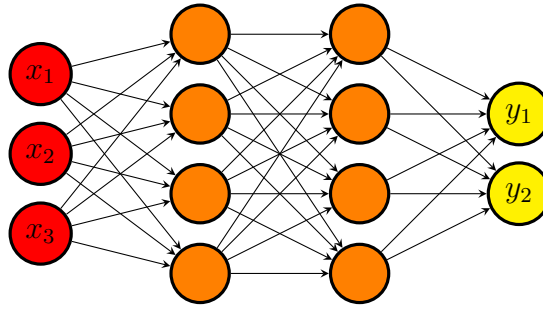


Figure 5.1: Fully connected neural network with two hidden layers, modeling a mapping from \mathbb{R}^3 to \mathbb{R}^2 .

whose second derivatives vanish since we use linear interpolation functions. In (5.10), $\tau_{1,2}$ are ad-hoc mesh-dependent stabilization parameters defined in [150, 196]. Conversely, in (5.11), $\tau_{3,4}$ are mesh-independent stabilization parameters acting both in the direction of the solution and of its gradient, that proceed from the stabilization of the ubiquitous convection-diffusion-reaction equation [100, 146], whose definition is given in [197, 198].

We solve the above equations sequentially (*i.e.*, we solve first (5.10), then use the resulting fluid velocity to solve (5.11)) with an in-house VMS solver whose flexibility, accuracy and reliability is assessed in a series of previous papers to which the reader is referred for further information, see in particular [149, 150] for the detailed mathematical formulation of the IVM in the context of finite element VMS methods. The ability of the IVM to handle the abrupt conductivity change across the fluid/solid interface is documented in [101, 149, 199]. Excellent agreement with reference solutions available from the literature and in-house data obtained enforcing proper thermal conditions at the boundary of body-fitted meshes is reported for several time-dependent conjugate heat transfer test cases (e.g., mixed convection in a plane channel flow, combined convection in square enclosures and conduction/radiation heat transfer, all in two dimensions). Ref. [149] also reports favorable agreement between the IVM and in-house experimental data pertaining to a three-dimensional test case representative of an industrial cooling system, which provides strong evidence of relevance for the intended application.

5.3 Deep reinforcement learning and proximal policy optimization

5.3.1 Neural networks

A neural network (NN) is a collection of artificial neurons, i.e., connected computational units that can be trained to arbitrarily well approximate the mapping function between input and output spaces. Each connection provides the output of a neuron as an input to another neuron. Each neuron performs a weighted sum of its inputs, to assign significance to the inputs with regard to the task the algorithm is trying to learn. It then adds a bias to better represent the part of the output that is actually independent of the input. Finally, it feeds an activation function that determines whether and to what extent the computed value should affect the outcome. As sketched in figure 5.1, a fully connected network is generally organized into layers, with the neurons of one layer being connected solely to those of the immediately preceding and following layers. The layer that receives the external data is the input layer, the layer that produces the outcome is the output layer, and in between them are zero or more hidden layers.

The design of an efficient neural network requires a proper optimization of the weights and biases, together with a relevant nonlinear activation function. The abundant literature available on this topic points to a relevant network architecture (e.g., type of network, depth, width of each layer), finely tuned hyper parameters (i.e., parameters whose value cannot be estimated from data, e.g., optimizer, learning rate, batch size) and a sufficiently large amount of data to learn from as being the key ingredients for success; see, e.g., Ref. [63] and the references therein.

5.3.2 Deep reinforcement learning

Deep reinforcement learning (DRL) is an advanced branch of machine learning in which deep neural networks train in solving sequential decision-making problems. It is a natural extension of reinforcement learning (RL), in which an agent (the neural network) is taught how to behave in an environment by taking actions and by receiving feedback from it under the form of a reward (to measure how good or bad the action was) and information (to gauge how the action has affected the environment). This can be formulated as a Markov Decision Process, for which a typical execution goes as follows (see also figure 5.2):

- assume the environment is in state $s_t \in \mathcal{S}$ at iteration t , where \mathcal{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 $a_t \in \mathcal{A}$, where \mathcal{A} is a set of actions,

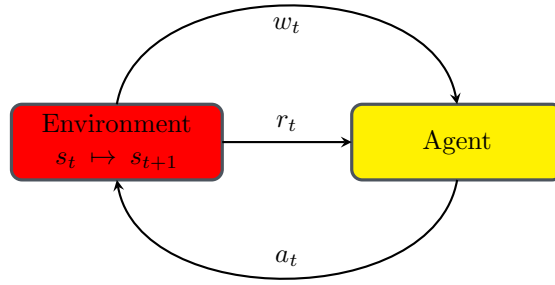


Figure 5.2: RL agent and its interactions with its environment.

- the environment reacts to the action and transitions from s_t to state $s_{t+1} \in \mathcal{S}$,
- the agent is fed with a reward $r_t \in \mathcal{R}$, where \mathcal{R} is a set of rewards, and a new observation w_{t+1} ,

This repeats until some termination state is reached, the succession of states and actions defining a trajectory $\tau = (s_0, a_0, s_1, a_1, \dots)$. In any given state, the objective of the agent is to determine the action maximizing its cumulative reward over an episode, i.e., over one instance of the scenario in which the agent takes actions. Most often, the quantity of interest is the discounted cumulative reward along a trajectory defined as

$$R(\tau) = \sum_{t=0}^T \gamma^t r_t, \quad (5.13)$$

where T is the horizon of the trajectory, and $\gamma \in [0, 1]$ is a discount factor that weighs the relative importance of present and future rewards (the agent being short-sighted in the limit where $\gamma \rightarrow 0$, since it then cares solely about the first reward, and far-sighted in the limit where $\gamma \rightarrow 1$, since it then cares equally about all rewards).

There exist two main types of RL algorithms, namely model-based methods, in which the agent tries to build a model of how the environment works to make predictions about what the next state and reward will be before taking any action, and model-free methods, in which the agent conversely interacts with the environment without trying to understand it, and are prominent in the DRL community. Another important distinction to be made within model-free algorithms is that between value-based methods, in which the agent learns to predict the future reward of taking an action when provided a given state, then selects the maximum action based on these estimates, and policy-based methods, in which it optimizes the expected reward of a decision policy mapping states to actions. Many of the most successful algorithms in DRL (including proximal policy optimization, whose assessment for

flow control and optimization purposes is the primary motivation for this research) proceed from policy gradient methods, in which gradient ascent is used to optimize a parameterized policy with respect to the expected return, as further explained in the next section. The reader interested in a more thorough introduction to the zoology of RL methods (together with their respective pros and cons) is referred to Ref. [50].

5.3.3 From policy methods to Proximal policy optimization

This section intended for the non-specialist reader briefly reviews the basic principles and assumptions of policy gradient methods, together with the various steps taken for improvement.

- Policy methods A policy method maximizes the expected discounted cumulative reward of a decision policy mapping states to actions. It resorts not to a value function, but to a probability distribution over actions given states, that fully defines the behavior of the agent. Since policies are most 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 return of the policy after taking action a in state s (also termed state-action value function or Q-function)

$$Q^\pi(s, a) = \mathbb{E}_\pi [R(\tau)|s, a], \quad (5.14)$$

where we use \mathbb{E}_π for the expected value \mathbb{E} under policy π .

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

$$V^\pi(s) = \mathbb{E}_\pi [R(\tau)|s]. \quad (5.15)$$

The V and Q functions are therefore such that

$$V^\pi(s) = \sum_a \pi(s, a) Q^\pi(s, a), \quad (5.16)$$

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

- **Policy gradient methods** A policy gradient method aims at optimizing a parametrized policy π_θ , where θ denotes the free parameters whose value 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

$$J(\theta) = \mathbb{E}_{\pi_\theta} [R(\tau)], \quad (5.17)$$

and seeks the parameterization θ^* maximizing $J(\theta)$, hence such that

$$\theta^* = \arg \max_{\theta} \mathbb{E}_{\pi_\theta} [R(\tau)], \quad (5.18)$$

which can be done on paper by plugging an estimator of the policy gradient $\nabla_\theta J(\theta)$ into a gradient ascent algorithm. This is no small task as one is looking for the gradient with respect to the policy parameters, in a context where the effects of policy changes on the state distribution are unknown (since modifying the policy will most likely modify the set of visited states, which will in turn affect performance in some indefinite manner). One commonly used estimator, derived in [50] using the log-probability trick, reads

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta} \left[\sum_{t=0}^T \nabla_\theta \log(\pi_\theta(s_t, a_t)) R(\tau) \right] \sim \mathbb{E}_{\pi_\theta} \left[\sum_{t=0}^T \nabla_\theta \log(\pi_\theta(s_t, a_t)) \widehat{A}^\pi(s_t, a_t) \right], \quad (5.19)$$

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

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s), \quad (5.20)$$

that measures the improvement (if $A^\pi > 0$, otherwise the lack thereof) associated with taking action a in state s compared to taking the average over all possible actions. This is because the value function does not depend on θ , so taking it off changes neither the expected value, nor the gradient, but it does reduce the variance, and speeds up the training. Furthermore, when the policy π_θ is represented by a neural network (in which case θ simply denotes the network weights and biases to be optimized), the focus is rather on the policy loss defined as

$$L(\theta) = \mathbb{E}_{\pi_\theta} \left[\sum_{t=0}^T \log(\pi_\theta(a_t | s_t)) \widehat{A}(s_t, a_t) \right], \quad (5.21)$$

whose gradient is equal to the (approximated) policy gradient (5.19) (since the gradient operator acts only on the log-policy term, not on the advantage) and is computed with respect to each weight and bias by the chain rule, one layer at the time, using the back-propagation algorithm [62].

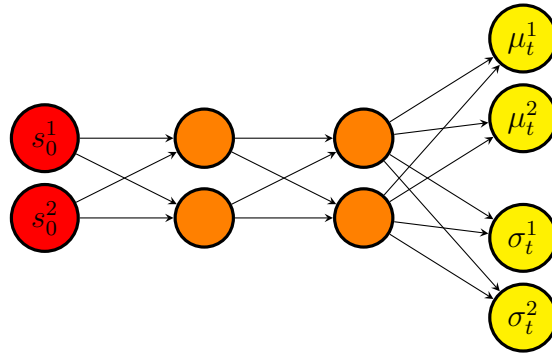


Figure 5.3: Agent network example used to map states to policy. The input state \mathbf{s}_0 , here of size 2, is mapped to a mean $\boldsymbol{\mu}$ and a standard deviation $\boldsymbol{\sigma}$ vectors, each of size 2. All activation functions are ReLu, except for that of the last layer, which are linear for the μ output, and softplus for the σ output. Orthogonal weights initialization is used throughout the network.

- **Trust regions** The performance of policy gradient methods is hurt by the high sensitivity to the learning rate, i.e., the size of the step to be taken in the gradient direction. Indeed, small learning rates are detrimental to learning, but large learning rates can lead to a performance collapse if the agent falls off the cliff and restarts from a poorly performing state with a locally bad policy. This is all the more harmful as the learning rate cannot be tuned locally, meaning that an above average learning rate will speed up learning in some regions of the parameter space where the policy loss is relatively flat, but will possibly trigger an exploding policy update in other regions exhibiting sharper variations. One way to ensure continuous improvement is by imposing a trust region constraint to limit the difference between the current and updated policies, which can be done by determining first a maximum step size relevant for exploration, then by locating the optimal point within this trust region. 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 [200]), or trust region policy optimization (TRPO [68]). Suffice it to say that they use the minorize-maximization 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 another approach with simple and effective heuristics, that uses a probability ratio between the two policies to maximize improvement without the risk of performance

collapse [18]. The focus here is on the PPO-clip algorithm¹, that optimizes the surrogate loss

$$L(\theta) = \mathbb{E}_{\pi_{\theta}} \left[\min \left(\frac{\pi_{\theta}(a|s)}{\pi_{\theta_{old}}(a|s)}, g(\epsilon, \hat{A}^{\pi}(s, a)) \right) \hat{A}^{\pi}(s, a) \right], \quad (5.22)$$

where

$$g(\epsilon, A) = \begin{cases} 1 + \epsilon & A \geq 0, \\ 1 - \epsilon & A < 0, \end{cases} \quad (5.23)$$

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. 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 (5.22) 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.

There exist more sophisticated PPO algorithms (e.g., Trust region PPO [143], that determines first a maximum step size relevant for exploration, then adaptively adjusts the clipping range to find the optimal within this trust region), but standard PPO has simple and effective heuristics. Namely, it is computationally inexpensive, easy to implement (as only the first-order gradient of the policy log probability is needed to calculate the clipped surrogate), and remains regarded as one of the most successful RL algorithms, achieving state-of-the-art performance across a wide range of challenging tasks.

5.3.4 Single-step PPO

We now come to single-step PPO, a “degenerate” version of PPO introduced in [138] and intended for situations where the optimal policy to be learnt by the neural network is state-independent, 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 actions a^* yielding the largest possible reward, single-step PPO seeks the optimal mapping f_{θ^*} such that $a^* = f_{\theta^*}(s_0)$, where θ denotes the network free parameters and s_0 is some input state (usually a vector of zeros) consistently fed to the agent for the optimal policy to eventually embody the transformation from

¹There 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.

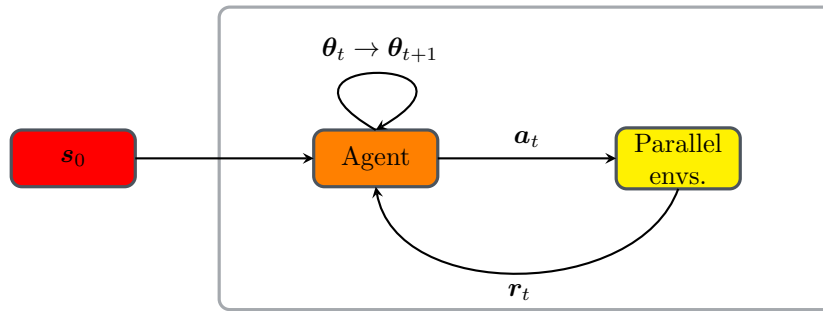


Figure 5.4: Action loop for single-step PPO. At each episode, the input state s_0 is provided to the agent, which in turn provides n actions to n parallel environments. The latter return n rewards, that evaluate the quality of each action taken. Once all the rewards are collected, an update of the agent parameters is made using the PPO loss (5.22).

s_0 to a^* . The agent initially implements a random state-action mapping f_{θ_0} from s_0 to an initial policy determined by the free parameters initialization 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 5.4 showing the agent draw a population of actions $a_t = f_{\theta_t}(s_0)$ from the current policy, and being returned incentives from the associated rewards to update the free parameters for the next population of actions $a_{t+1} = f_{\theta_{t+1}}(s_0)$ 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 5.3.3, 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

$$A(s_t, a_t) = r_t + \gamma V(s_{t+1}) - V(s_t). \quad (5.24)$$

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 (5.24). This means that the approach can do without the value-function evaluations of the critic network, i.e., it is not actually actor-critic.

5.3.5 Numerical implementation

The present workflow relies on the online PPO implementation of Stable Baselines, a toolset of reinforcement learning algorithms dedicated to the research community and industry [154], for which a custom OpenAI environment has been designed using the Gym library [155]. Hyperbolic tangent is used as default activation function. The instant reward r_t used to train the neural network is simply the quantity subjected to optimization (modulo a plus or minus sign to tackle both maximization and minimization problems). A moving average reward is also computed on the fly as the sliding average over the 100 latest values of r_t (or the whole sample if it has insufficient size). All other relevant hyper parameters are documented in the next sections, with the exception of the discount factor (set to $\gamma = 1$).

In practice, actions are distributed to multiple environments running in parallel, each of which executes a self-contained MPI-parallel CFD simulation and feeds data to the DRL algorithm (hence, two levels of parallelism related to the environment and the computing architecture). The algorithm waits for the simulations running in all parallel environments to be completed, then shuffles and splits the rewards data set collected from all environments into several buffers (or mini-batches) used sequentially to compute the loss and perform a network update. The process repeats for several epochs, i.e., several full passes of the training algorithm over the entire data set (so the policy network ends up being trained on samples generated by older policies, which is customary in standard PPO operation). This simple parallelization technique is key to use DRL in the context of CFD applications, as a sufficient number of actions drawn from the current policy must be evaluated to accurately estimate the policy gradient. This comes at the expense of computing the same amount of reward evaluations, and yields a substantial computational cost for high-dimensional fluid dynamics problems (typically from a few tens to several thousand hours for the steady-state optimization problems considered herein). In the same vein, it should be noted that the common practice in DRL studies to gain insight into the performances of the selected algorithm by averaging results over multiple independent training runs with different random seeds is not tractable, as it would trigger a prohibitively large computational burden. The same random seeds have thus been deliberately used over the whole course of study to ensure a minimal level of performance comparison between cases.

5.4 Control of natural convection in 2-D closed cavity

5.4.1 Case description

We address first the control of natural convection in the two-dimensional differentially heated square cavity schematically illustrated in figure 5.5(a). This is a

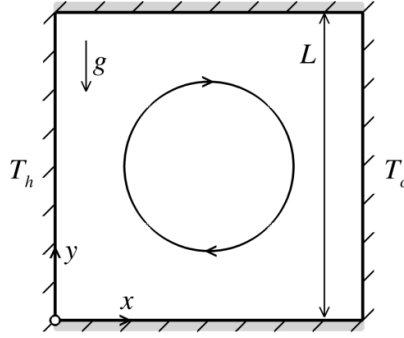


Figure 5.5: Schematic of the two-dimensional Rayleigh–Bénard set-up.

widely studied benchmark system for thermally-driven flows, relevant in nature and technical applications (e.g., ocean and atmospheric convection, materials processing, metallurgy), that is thus suitable to validate and compare numerical solution algorithms while enriching the knowledge base for future projects in this field. A Cartesian coordinate system is used with origin at the lower-left edge, horizontal x -axis, and vertical y -axis. The cavity has side L , its top and bottom horizontal walls are perfectly insulated from the outside, and the vertical sidewalls are isothermal. Namely, the right sidewall is kept at a constant, homogeneous “cold” temperature T_c , and the left sidewall is entirely controllable via a constant in time, varying in space “hot” distribution $T_h(y)$ such that

$$\langle T_h \rangle > T_c, \quad (5.25)$$

where the brackets denote the average over space (here over the vertical position along the sidewall).

In the following, we neglect radiative heat transfer ($\chi = 0$) and consider a Boussinesq system driven by buoyancy, hence

$$\boldsymbol{\psi} = \rho_0 \beta (T - T_c) g \mathbf{e}_y, \quad (5.26)$$

where \mathbf{g} is the gravitational acceleration parallel to the sidewalls, β is the thermal expansion coefficient, and we use the cold sidewall temperature as Boussinesq reference temperature. By doing so, the pressure featured in the momentum equation (5.2) and related weak forms must be understood as the pressure correction representing the deviation from hydrostatic equilibrium. The governing equations are solved with no-slip conditions $\mathbf{u} = \mathbf{0}$ on $\partial\Omega$ and temperature boundary conditions

$$\partial_y T(x, 0, t) = \partial_y T(x, L, t) = 0, \quad T(0, y, t) = \langle T_h \rangle + \tilde{T}_h(y), \quad T(L, y, t) = T_c, \quad (5.27)$$

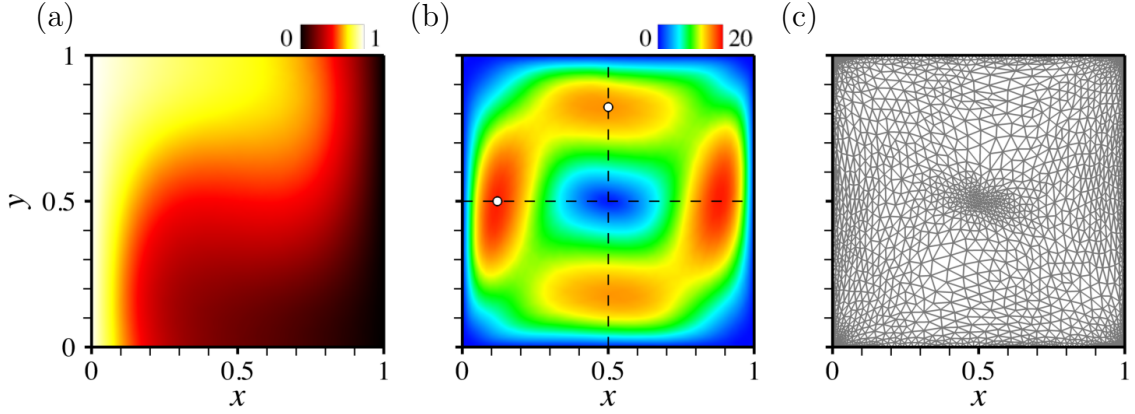


Figure 5.6: Iso-contours of the uncontrolled steady state (a) temperature and (b) velocity magnitude. (c) Adapted mesh. The circle symbols in (b) mark the positions of the maximum horizontal and vertical velocity along the centerlines reported in table 5.1.

where \tilde{T}_h is a zero-mean (in the sense of the average over space) distribution of hot temperature fluctuations subjected to optimization, whose magnitude is bounded by some constant ΔT_{max} according to

$$|\tilde{T}_h(y)| \leq \Delta T_{max}, \quad (5.28)$$

to avoid extreme and nonphysical temperature gradients. All results are made non-dimensional using the cavity side, the heat conductivity time, and the well-defined, constant in time difference between the averaged sidewall temperatures. The retained fluid properties yield values of the Rayleigh and Prandtl numbers

$$\text{Ra} = \frac{g\beta(\langle T_h \rangle - T_c)L^3}{\nu\alpha} = 10^4, \quad \text{Pr} = \frac{\nu}{\alpha} = 0.71, \quad (5.29)$$

where $\alpha = \lambda/(\rho c_p)$ is the thermal diffusivity.

In order to assess the accuracy of the numerical framework, the uncontrolled solution has been computed by performing 60 iterations with time step $\Delta t = 0.5$ to march the initial solution (consisting of zero velocity and uniform temperature, except at the hot sidewall) to steady state. At each time step, an initially isotropic mesh is adapted under the constraint of a fixed number of elements $n_{el} = 4000$ using a multiple-component criterion featuring velocity and temperature, but no level-set. This is because the case is heat transfer but not conjugate heat transfer, as the solid is solely at the boundary $\partial\Omega$ of the computational domain, where either the temperature is known, or the heat flux is zero. It is thus implemented without the IVM and without a level set (although accurate IVM numerical solutions have been obtained in [101] using thick sidewalls with high thermal conductivity). The

		Present	Ref. [201]	Ref. [202]	Ref. [203]	Ref. [204]	Ref. [205]
Ra = 10 ⁴	Nu	2.267	2.238	2.245	2.201	2.245	2.245
	max $u(0.5, y)$	16.048	16.178	16.179	–	16.262	16.178
	y_{max}	0.823	0.823	0.824	0.832	0.818	0.827
	max $v(x, 0.5)$	19.067	19.617	19.619	–	19.717	19.633
	x_{max}	0.120	0.119	0.121	0.113	0.119	0.123

Table 5.1: Comparison of the present numerical results in the absence of control with reference benchmark solutions from the literature.

solution shown in figure 5.6(a,b) features a centered roll confined by the cavity walls, consistently with the fact that Ra exceeds the critical value $Ra_c \sim 920$ for the onset of convection (as extrapolated from the near-critical benchmark data in [201]) by one order of magnitude, and heat transfer is thus driven by both conduction and convection. This shows in the Nusselt number, i.e., the non-dimensional temperature gradient averaged over the hot sidewall

$$Nu = -\langle \partial_x T \rangle, \quad (5.30)$$

whose present value $Nu = 2.27$ (as computed from 68 points uniformly distributed along the sidewall) exceeds that $Nu = 1$ of the purely conductive solution, and exhibits excellent agreement with benchmark results from the literature. This is evidenced in table 5.1 where we also report the magnitude and position of the maximum horizontal velocity u (resp. the vertical velocity v) along the vertical centerline (resp. the horizontal centerline). The corresponding adapted mesh shown in figure 5.6(c) stresses that all boundary layers are sharply captured via extremely stretched elements, and that the adaptation strategy yields refined meshes near high temperature gradients and close to the side walls. Note however, the mesh refinement is not only along the boundary layers but also close to the recirculation regions near the cavity center, while the elements in-between are coarse and essentially isotropic.

5.4.2 Control

The question now being raised is whether DRL can be used to find a distribution of temperature fluctuations \tilde{T}_h capable of alleviating convective heat transfer. To do so, we follow [17] and train a DRL agent in selecting piece-wise constant temperature distributions over n_s identical segments, each of which allows only two pre-determined states referred to as hot or cold. This is intended to reduce the complexity and the computational resources, as large/continuous action spaces are known to be challenging for the convergence of RL methods [192, 206]. Simply put,

the network action output consists of n_s values $\hat{T}_{hk \in \{1 \dots n_s\}} = \pm \Delta T_{max}$, mapped into the actual fluctuations according to

$$\tilde{T}_{hk} = \frac{\hat{T}_{hk} - \langle \hat{T}_{hk} \rangle}{\max_l \left\{ 1, \frac{|\hat{T}_{hl} - \langle \hat{T}_{hl} \rangle|}{\Delta T_{max}} \right\}}, \quad (5.31)$$

to fulfill the zero-mean and upper bound constraints.² Ultimately, the agent receives the reward $r_t = -\text{Nu}$ to minimize the space averaged heat flux at the hot sidewall.

All results reported herein are for $\Delta T_{max} = 0.75$ (so the hot temperature varies in the range $[0.25; 1.75]$) and $n_s = 10$ segments, as [17] report that $n_s = 20$ was computationally too demanding for their case, and that $n_s = 5$ yielded poor control efficiency. 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 5×10^{-3} , and PPO loss clipping range $\epsilon = 0.2$.

5.4.3 Results

For this case, 120 episodes have been run, each of which follows the exact same procedure as above and performs 60 iterations with time step $\Delta t = 0.5$ to march the zero-initial condition to steady state. This represents 960 simulations, each of which is performed on 4 cores and lasts 20s, hence 5h of total CPU cost. We present in figure 5.7 representative iso-contours of the steady-state temperature and velocity magnitude computed over the course of the optimization. The latter exhibit strong temperature gradients at the hot sidewall, together with a robust steady roll-shaped pattern accompanied by a small corner eddy at the upper-left edge of the cavity, whose size and position depends on the specifics of the temperature distribution. The corresponding meshes are displayed in figure 5.7(c) to stress the ability of the adaptation procedure to handle well the anisotropy of the solution caused by the intrinsic flow dynamics and the discontinuous boundary conditions.

We show in figure 5.8 the evolution of the controlled averaged Nusselt number, whose moving average decreases monotonically and reaches a plateau after about 90 episodes, although we notice that sub-optimal distributions keep being explored occasionally. The optimal computed by averaging over the 10 latest episodes (hence

²Another possible approach would have been to penalize the reward passed to the DRL for those temperature distributions deemed non-admissible (either because the average temperature is non-zero or the temperature magnitude is beyond the threshold). However, this would have made returning admissible solutions part of the tasks the network is trained on (not to mention that non-zero average temperatures amount to a change in the Rayleigh number), which would likely have slowed down learning substantially.

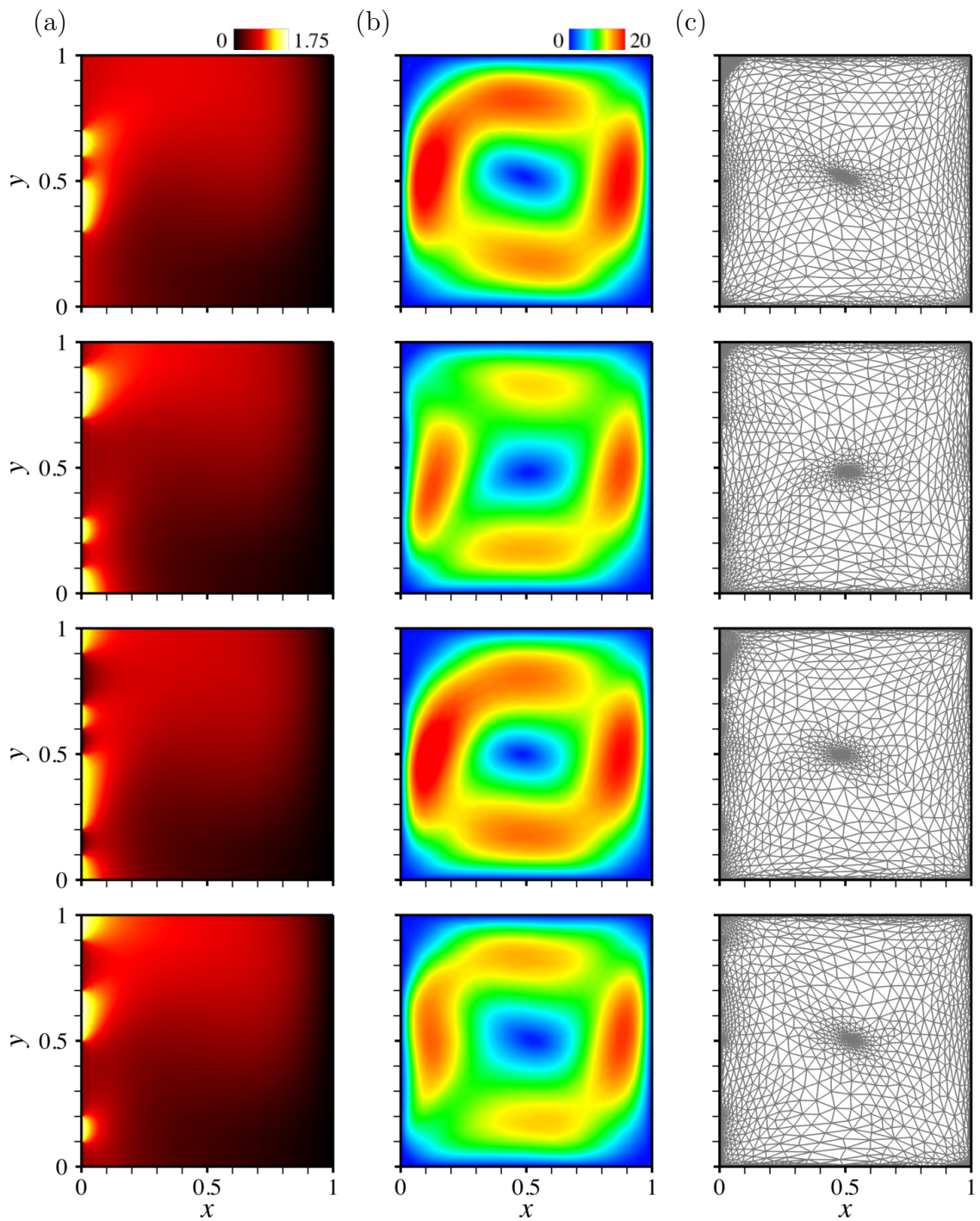


Figure 5.7: (a,b) Steady-state (a) temperature and (b) velocity magnitude against zero-mean temperature distributed at the left sidewall. (c) Adapted meshes.

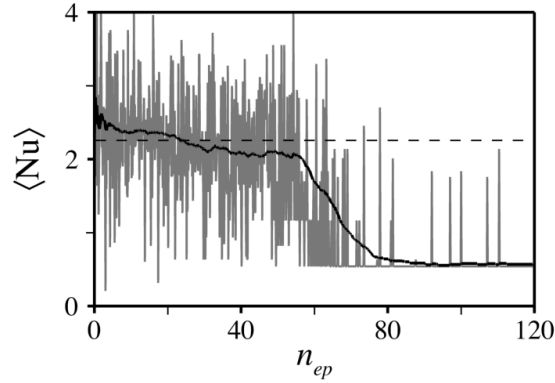


Figure 5.8: Evolution per learning episode of the instant (in grey) and moving average (in black) Nusselt number. The horizontal dashed line marks the uncontrolled value.

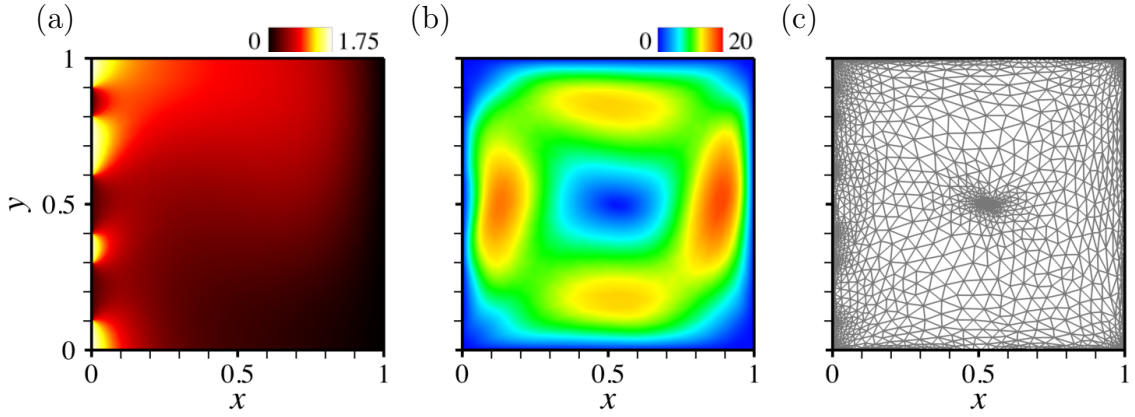


Figure 5.9: (a,b) Steady-state (a) temperature and (b) velocity magnitude for the optimal zero-mean temperature distribution. (c) Adapted mesh.

the 800 latest instant values) is $\langle \text{Nu} \rangle^* \sim 0.57$, with variations ± 0.01 computed from the root-mean-square of the moving average over the same interval (which is a simple yet robust criterion to assess qualitative convergence a posteriori). Interestingly, the optimized Nusselt number is almost twice as small as the purely conductive value ($\text{Nu} = 1$), meaning that the approach successfully alleviates the heat transfer enhancement generated by the onset of convection, although it does not alleviate convection itself, as evidenced by the consistent roll-shaped pattern in figure 5.9. Similar results are reported in [17], for a different set-up in which the horizontal cavity walls are isothermal and control is applied at the bottom at the cavity (hence a different physics because of buoyancy), albeit with lower numerical and control efficiency since the authors report an optimal Nusselt number $\text{Nu} \sim 1$ using up to 512 DRL environments with learning rate of 2.5×10^{-4} . The reason

for such discrepancies probably lies in different ways of achieving and assessing control, as we use single-step PPO to optimize the steady-state Nusselt number via a time-independent control, which requires choosing a sidewall temperature, marching the controlled solution to steady state, then computing the reward. The problem considered in [17] is more intricate, as classical PPO is used to optimize the reward accumulated over time via a time-dependent control temperature updated with a certain period scaling with the convection time in the cavity (the so-determined optimal control being ultimately time-independent for the considered value of Ra , but truly time-dependent for Rayleigh numbers above $\sim 10^5$).

5.5 Control of forced convection in 2-D open cavity

5.5.1 Case description

This second test case addresses the control of actual conjugate heat transfer in a model setup for the cooling of a hot solid by impingement of a fluid; see figure 5.10(a). A Cartesian coordinate system is used with origin at the center of mass of the solid, horizontal x -axis, and vertical y -axis. The solid has rectangular shape with height h and aspect ratio 2:1, and is initially at the hot temperature T_h . It is fixed at the center of a rectangular cavity with height H and aspect ratio 4:1, whose walls are isothermal and kept at temperature T_w . The top cavity side is flush with n_j identical holes of width e_i whose distribution is subjected to optimization, each of which models the exit plane of an injector blowing cold air at velocity V_i and temperature T_c , and is identified by the horizontal position of its center $x_{k \in \{1 \dots n_j\}}$. Hot air is released through the cavity sidewalls, blown with two identical exhaust areas of height e_o , and identified by the vertical position of their center $(e_o - H)/2$.

For this case, both buoyancy and radiative heat transfer are neglected ($\psi = \mathbf{0}$ and $\chi = 0$), meaning that temperature evolves as a passive scalar, similar to the mass fraction of a reactant in a chemical reaction. All relevant parameters are provided in Table 5.2, including the material properties used to model the composite fluid, that yield fluid values of the Reynolds and Prandtl numbers

$$\text{Re} = \frac{\rho V_i e}{\mu} = 200, \quad \text{Pr} = 2. \quad (5.32)$$

Note the very high value of the solid to fluid viscosity ratio, that ensures that the velocity is zero in the solid domain and that the no-slip interface condition is satisfied. By doing so, the convective terms drop out in the energy equation, that reduces to the pure conduction equation for 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 injection exit planes ($\mathbf{u} = -V_i \mathbf{e}_y$, $T = T_c$), and at the exhaust areas, where a zero-pressure outflow condition is imposed ($p = \partial_x u = \partial_x T = 0$). No thermal condition is

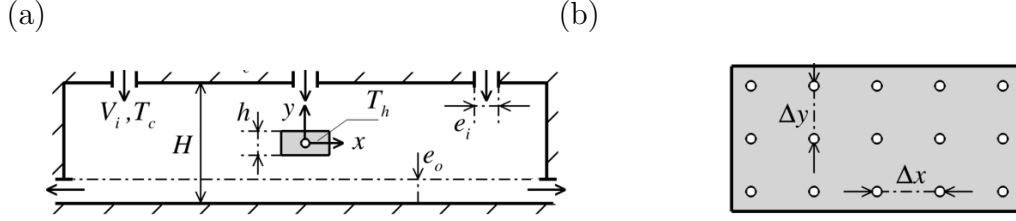


Figure 5.10: (a) Schematic of the 2-D forced convection set-up. (b) Sensors positions in the solid domain.

H	h	e_i	e_o	V_i	T_w	T_c	T_h	μ	ρ	λ	c_p	
1	0.2	0.2	0.2	1	10	10	150	0.001	1	0.5	1000	Fluid
								1000	100	15	300	Solid

Table 5.2: Numerical parameters used in the 2-D forced convection problem. All values in SI units, with the exception of temperatures given in Celsius.

imposed at the interface, where heat exchange is implicitly driven by the difference in the individual material properties.

5.5.2 Control

The quantity being optimized is the distribution of the injectors center positions $x_{k \in \{1 \dots n_j\}}$. Several control strategies are assessed in the following, whose ability to manage increasing design complexity translates into less constrained operation when it comes to optimizing a practically meaningful device. In practice, each injector is forced to sit in an interval $[x_k^-; x_k^+]$ whose edge values are determined beforehand and recomputed on the fly (depending on the control strategy), and bounded according to

$$|x_k^\pm| \leq x_m, \quad (5.33)$$

where we set $x_m = 2H - 0.75e_i$ to avoid numerical issues at the upper cavity edges. The network action output therefore consists of n_j values $\hat{x} \in [-1; 1]$, mapped into the actual positions according to

$$x_k = \frac{x_k^+(\hat{x}_k + 1) - x_k^-(\hat{x}_k - 1)}{2}. \quad (5.34)$$

In order to compute the reward passed to the DRL, we distribute uniformly 15 probes in the solid domain, into $n_x = 5$ columns and $n_y = 3$ rows with resolutions

$\Delta x = 0.09$ and $\Delta y = 0.075$, respectively; see figure 5.10(b). Selected tests have been carried out to check that the outcome of the learning process does not change using $n_y = 5$ rows of $n_x = 5$ probes (not shown here). The magnitude of the tangential heat flux is estimated by averaging the norm of the temperature gradient over all columns and rows, i.e., i -th column (resp. the j -th row) as

$$\langle \|\nabla_{\parallel} T\| \rangle_i = \frac{2}{n_y - 1} \left| \sum_{j \neq 0} \text{sgn}(j) \|\nabla T\|_{ij} \right|, \quad \langle \|\nabla_{\parallel} T\| \rangle_j = \frac{2}{n_x - 1} \left| \sum_{i \neq 0} \text{sgn}(i) \|\nabla T\|_{ij} \right|, \quad (5.35)$$

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 numerical reward $r_t = -\langle \|\nabla_{\parallel} T\| \rangle$ fed to the DRL agent deduces ultimately by averaging over all rows and columns, to give

$$\langle \|\nabla_{\parallel} T\| \rangle = \frac{1}{n_x + n_y} \sum_{i,j} \langle \|\nabla_{\parallel} T\| \rangle_i + \langle \|\nabla_{\parallel} T\| \rangle_j, \quad (5.36)$$

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

All results reported in the following are for $n_j = 3$ injectors. 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 PPO loss clipping range to $\epsilon = 0.3$.

5.5.3 Results

5.5.3.1 Fixed domain decomposition strategy

We consider first the so-called fixed domain decomposition strategy S_1 in which the top cavity wall is split into n_j equal subdomains, and each injector is forced to sit in a different subdomain (a somehow heavily constrained optimization problem if n_j is not too small, relevant for cases where the design is rigid and the practitioner has limited freedom to act). The edge values for the position x_k of the k -th injector read

$$x_k^- = -x_m + (k-1) \frac{2x_m + e_i}{n_j}, \quad x_k^+ = x_k^- + \frac{2x_m - (n_j - 1)e_i}{n_j}. \quad (5.37)$$

It can be checked that $x_k^- = x_{k-1}^+ + e_i$, so, it is possible to end up with two side-by-side injectors, which is numerically equivalent to having $n_j - 1$ injectors, $n_j - 2$ of width e_i plus one of width $2e_i$. For this case, 60 episodes have been run, each of which performs 1500 iterations with time step $\Delta t = 0.1$ to march the same initial

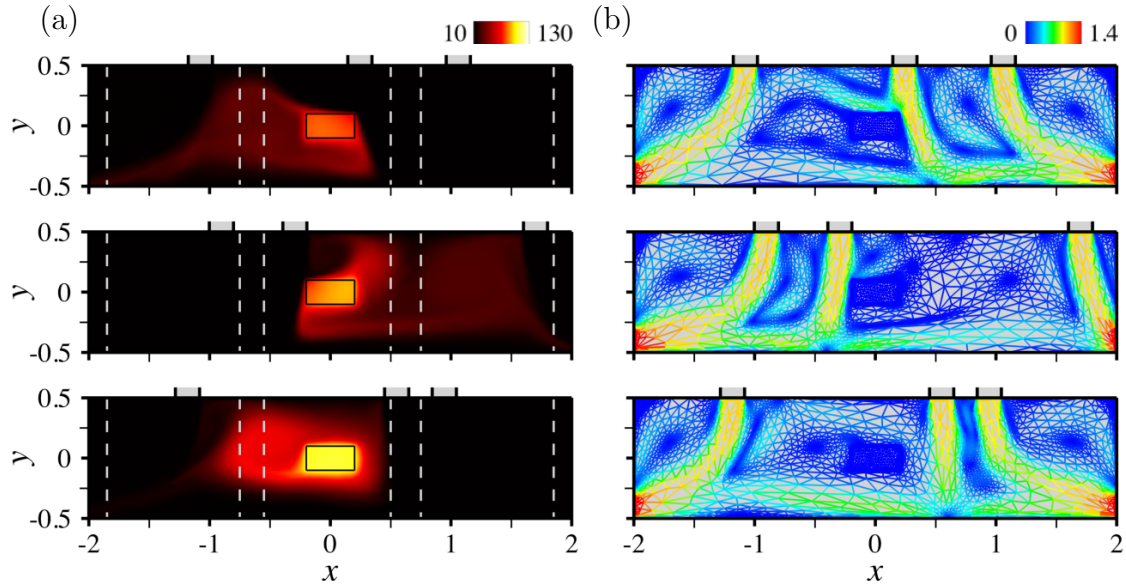


Figure 5.11: (a) Steady-state temperature against arrangements of 3 injectors, with admissible values under the fixed domain decomposition strategy S_1 delimited by the dashed lines. (b) Adapted meshes colored by the magnitude of velocity.

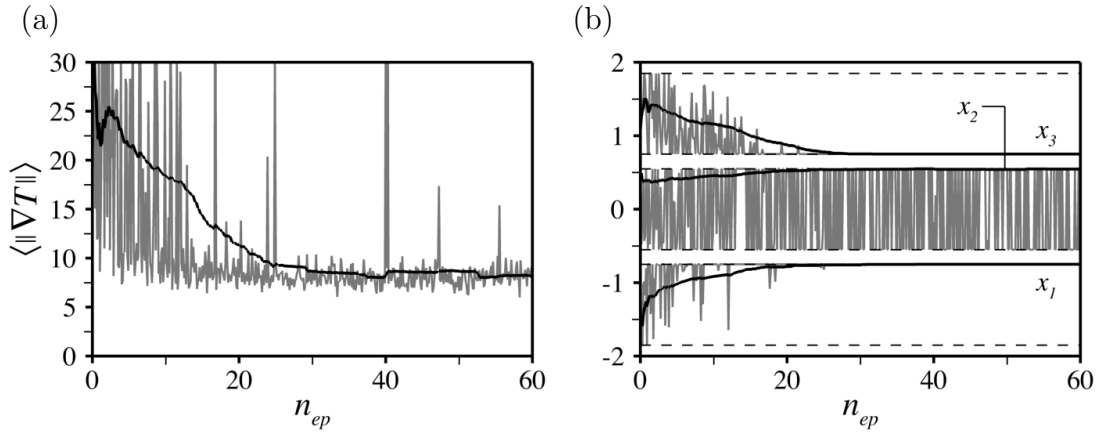


Figure 5.12: (a) Evolution per learning episode of the instant (in grey) and moving average (in black) rewards under the fixed domain decomposition strategy S_1 . (b) Same as (a) for the injectors center positions, with admissible values delimited by the dashed lines.

condition (consisting of zero velocity and uniform temperature, except in the solid domain) to steady state, 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 480 simulations, each of which is performed on 8 cores and lasts 10mn, hence 80h of total CPU cost.

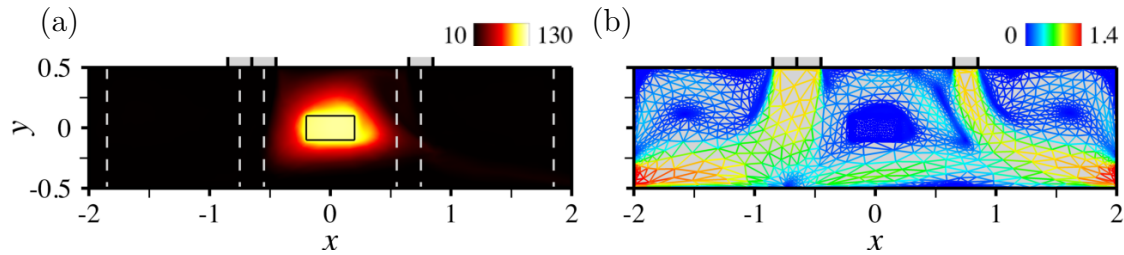


Figure 5.13: Same as figure 5.11 for the optimal arrangement of 3 injectors under the fixed domain decomposition strategy S_1 .

It is out of the scope of this work to analyze in details the many flow patterns that develop when the blown fluid travels through the cavity. Suffice it to say that the outcome depends dramatically on the injectors arrangement, and features complex rebound phenomena (either fluid/solid, when a jet impinges on the cavity walls or on the workpiece itself, or fluid/fluid, when a deflected jet meets the crossflow of another jet), leading to the formation of multiple recirculation varying in number, position and size. Several such cases are illustrated in figure 5.11 via iso-contours of the steady-state temperature distributions, together with the corresponding adapted meshes colored by the magnitude of velocity to illustrate the ability of the numerical framework to capture accurately all boundary layers and shear regions via extremely stretched elements.

One point worth mentioning is that the individual position signals are best suited to draw robust quantitative conclusion, as there is noise in the reward signal shown in figure 5.12(a). We believe the issue to be twofold: on the one hand, the reward is approximated from point-wise temperature data (similar to experimental measurements) that are more sensitive to small numerical errors (e.g., the interpolation error at the probes position) than an integral quantity. On the other hand, the mesh adaptation procedure is not a deterministic process, as the outcome depends on the processors and number of processors used, and any initial difference propagates over the course of the simulation because the meshes keep being adapted dynamically. In return, two exact same control parameters can thus yield different rewards on behalf of different interpolation errors at the probes position. This likely slows down learning and convergence, but we show in figure 5.12(b) that the moving average distribution does converge to an optimal arrangement after roughly 25 episodes. The latter consists of an injector at the right-end of the left subdomain ($x_1^* = -0.75$) and two side-by-side injectors sitting astride the center and right subdomains ($x_2^* = 0.55$ and $x_3^* = 0.75$), that enclose the workpiece in a double-cell recirculation; see figure 5.13. These values have been computed by averaging the instant positions of each injector over the 10 latest episodes, with variations ± 0.002 computed from the root-mean-square of the moving average over the same interval, a procedure that will be used consistently to assess convergence for all cases

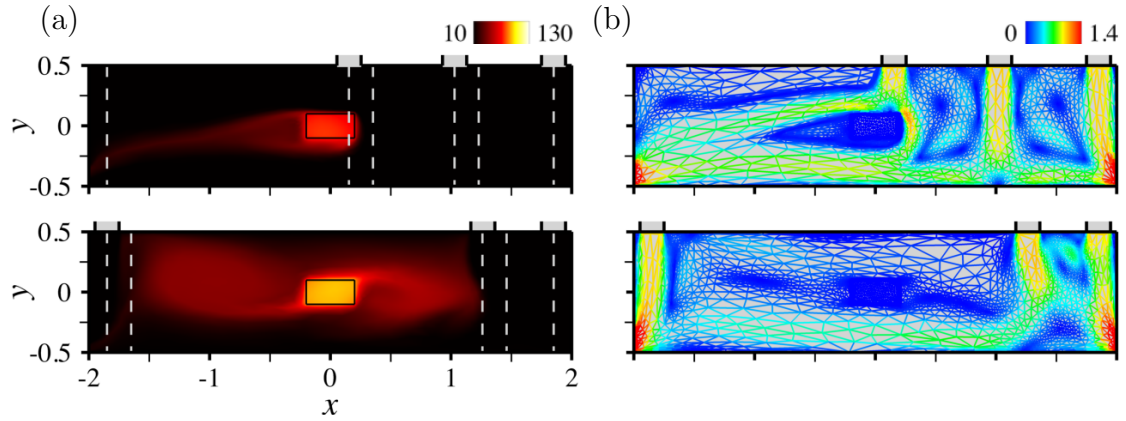


Figure 5.14: (a) Steady-state temperature against arrangements of 3 injectors, with admissible values under the follow-up strategy S_2 delimited by the dashed lines. (b) Adapted meshes colored by the magnitude of velocity.

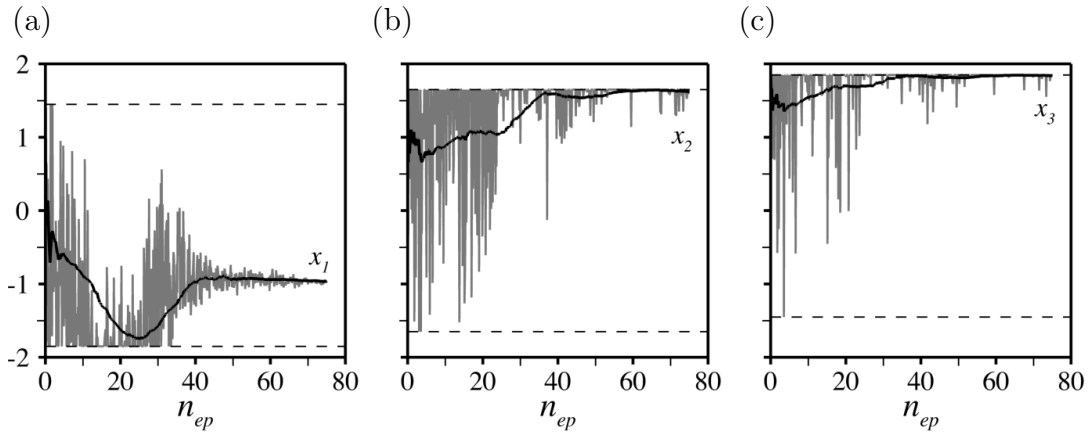


Figure 5.15: Evolution per learning episode of the instant (in grey) and moving average (in black) injectors center positions under the follow-up strategy S_2 , with admissible values delimited by the dashed lines.

reported in the following. The efficiency of the control itself is estimated by computing the magnitude of tangential heat flux averaged over the same interval, found to be $\langle ||\nabla_{\parallel} T|| \rangle^* \sim 8.3$. Note, the position x_2^* is actually obtained by averaging the absolute value of the instant position x_2 (although the true, signed value is depicted in the figure), as the center injector keeps oscillating between two end positions ± 0.55 on behalf of reflectional symmetry with respect to the vertical centerline.

5.5.3.2 Follow-up strategy

A less constrained problem is considered here using the so-called follow-up strategy S_2 , in which all injectors are distributed sequentially the ones with respect to the

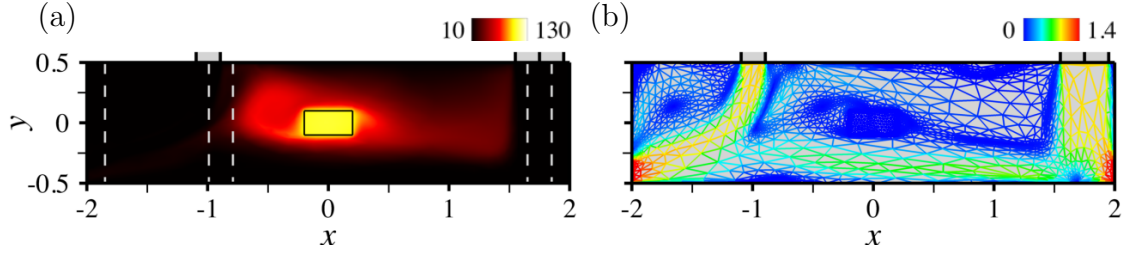


Figure 5.16: Same as figure 5.14 for the optimal arrangement of 3 injectors under the follow-up strategy S_2 .

others. The corresponding edge values

$$x_1^- = -x_m, \quad x_1^+ = x_m - (n_j - 1)e_i, \quad (5.38)$$

$$x_k^- = x_{k-1}^+ + e_i, \quad x_k^+ = x_m - (n_j - k)e_i, \quad (5.39)$$

readily express that the k -th injector is forced to sit between the $k-1$ -th one and the upper-right cavity edge while leaving enough space to distribute the remaining $n_j - k$ injectors, which increases the size of the control parameter space while again leaving the possibility for side-by-side injectors (since $x_k^- = x_{k-1}^+ + e_i$ by construction). 75 episodes have been run for this case following the exact same procedure as above, i.e., marching the zero-initial condition in time up to $t = 150$ with $\Delta t = 0.1$, hence 600 simulations, each of which is performed on 8 cores and lasts 10mn, hence 100h of total CPU cost.

The computed flow patterns closely resemble those obtained under the previous fixed domain decomposition strategy, although figure 5.14 exhibits increased dissymmetry when two or more injectors move simultaneously to the same side of the cavity. We show in figure 5.15 that the moving average distribution converges after roughly 60 episodes, with the optimal arrangement consisting of one injector roughly midway between the left cavity sidewall and the workpiece ($x_1^* = -0.96$), and two side-by-side injectors at the right end of the cavity ($x_2^* = 1.65$ and $x_3^* = 1.85$). The variations over the same interval are by ± 0.006 ; see also figure 5.16 for the corresponding flow pattern. Convergence here is much slower than under S_1 , as the search for an optimal is complicated by the fact that all injector positions are interdependent the ones on the others and it is up to the network to figure out exactly how. Another contingent matter is that the agent initially spans a fraction of the control parameter space because the large values of x_1 considered limit the space available to distribute the other two injectors. This is all the more so as such configurations turn to be far from optimality, for instance the magnitude of tangential heat flux is $\langle \|\nabla_{\parallel} T\| \rangle \sim 41.3$ for $x_1 = 1.45$, $x_2 = 1.65$ and $x_3 = 1.85$, but $\langle \|\nabla_{\parallel} T\| \rangle^* \sim 6.3$ at optimality. The latter value is smaller than the optimal achieved under S_1 , consistently with the fact that all positions spanned under S_1 are admissible under S_2 , hence the S_1 optimal is expected to be a S_2 sub-optimal.

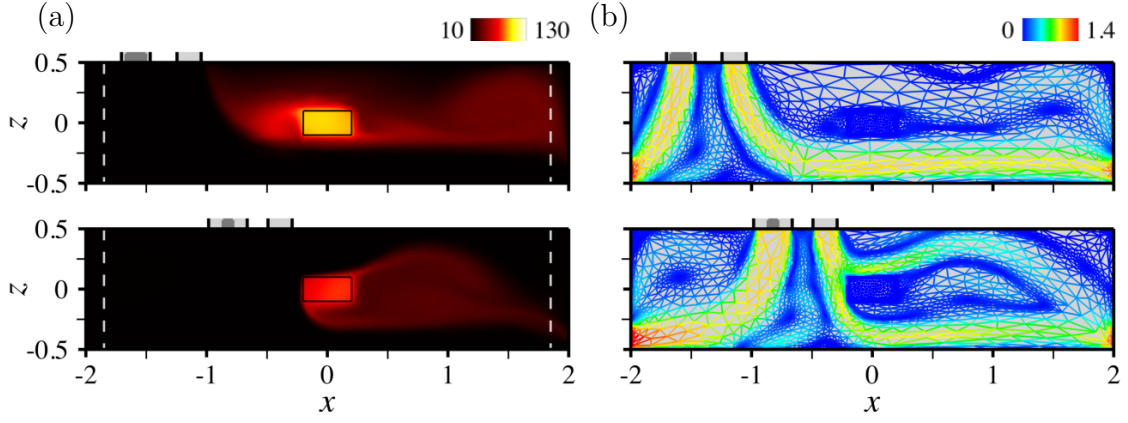


Figure 5.17: (a) Steady-state temperature against arrangements of 3 injectors, with admissible values under the free strategy S_3 delimited by the dashed lines and overlaps marked by the dark grey shade. (b) Adapted meshes colored by the magnitude of velocity.

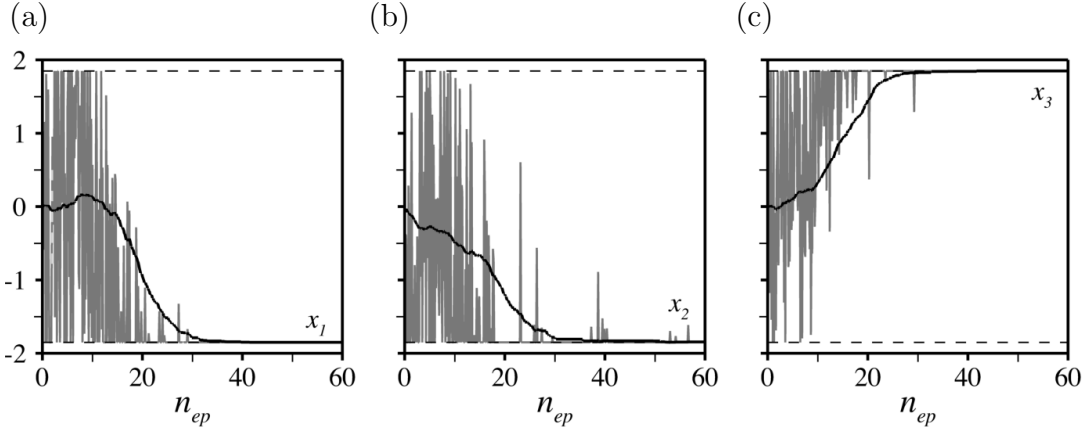


Figure 5.18: Evolution per learning episode of the instant (in grey) and moving average (in black) injectors center positions under the free strategy S_3 , with admissible values delimited by the dashed lines.

5.5.3.3 Free strategy

We examine now a third strategy S_3 referred to as the free strategy, in which all injectors are independent and free to move along the top cavity wall (a mildly constrained optimization problem, relevant for cases where the design is flexible and the practitioner has great freedom to act). The edge values for the position x_k of the k -th injector read

$$x_k^- = -x_m, \quad x_k^+ = x_m, \quad (5.40)$$

so two injectors can end up side-by-side and even overlapping one another if $|x_l - x_m| < e_i$. If so, we implement a single injector of width $e_i + |x_l - x_m|$ and maintain

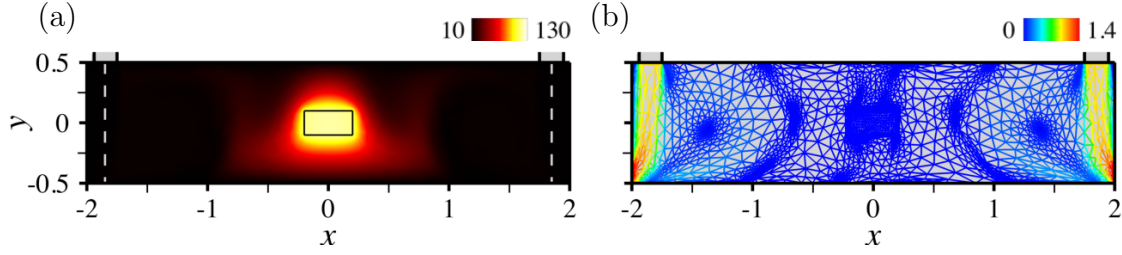


Figure 5.19: Same as figure 5.17 for the optimal arrangement of 3 injectors under the free strategy S_3 .

the blowing velocity (not the flow rate) for the purpose of automating the set-up design process, meaning that having n_j injectors, two of which overlap exactly (i.e., $|x_l - x_m| = 0$) is rigorously equivalent to having $n_j - 1$ injectors. 60 episodes have been run for this case following the exact same procedure as above.

All flow patterns are reminiscent of those obtained under the previous fix decomposition S_1 and follow-up S_2 strategies, even when two injectors overlap; see figure 5.17. Other than that, we show in figures 5.18 that the moving average distribution converges to an optimal consisting of two injectors almost perfectly overlapping one another at the left end of the cavity ($x_1^* = -1.85$ and $x_2^* = -1.82$), and a third injector at the right end of the cavity ($x_3^* = 1.85$). The variations over the same interval are by ± 0.007 , and the associated flow pattern shown in figure 5.19 is symmetrical and features two large recirculation regions on either side of the workpiece. Convergence occurs after roughly 40 episodes, i.e., faster than under S_2 (consistently with the fact that there is no need to learn anymore about how the network outputs depend the ones on the others) but slower than under S_1 (consistently with the fact that the size of the control parameter space has increased substantially). It is worth noticing that the system is invariant by permutations of the network outputs, meaning that there exist $2^{n_j} - 2$ distributions (hence 6 for $n_j = 3$) associated with the same reward. Nonetheless, a single optimal is selected, which is essentially fortuitous since the agent does not learn about symmetries under the optimization process (otherwise S_1 would have similarly selected a single optimal). The magnitude of tangential heat flux is $\langle \|\nabla_{\parallel} T\| \rangle^* \sim 11.2$ at optimality, i.e., larger than that achieved under S_2 . This can seem surprising at first, because all positions spanned under S_2 are admissible under S_3 , and the S_2 optimal is thus expected to be a S_3 sub-optimal. However, the argument does not hold here because the overlap in the S_3 optimal reduces the flow rate to that of a two-injectors set-up, so the comparison should be with the S_2 optimal with $n_j = 2$.

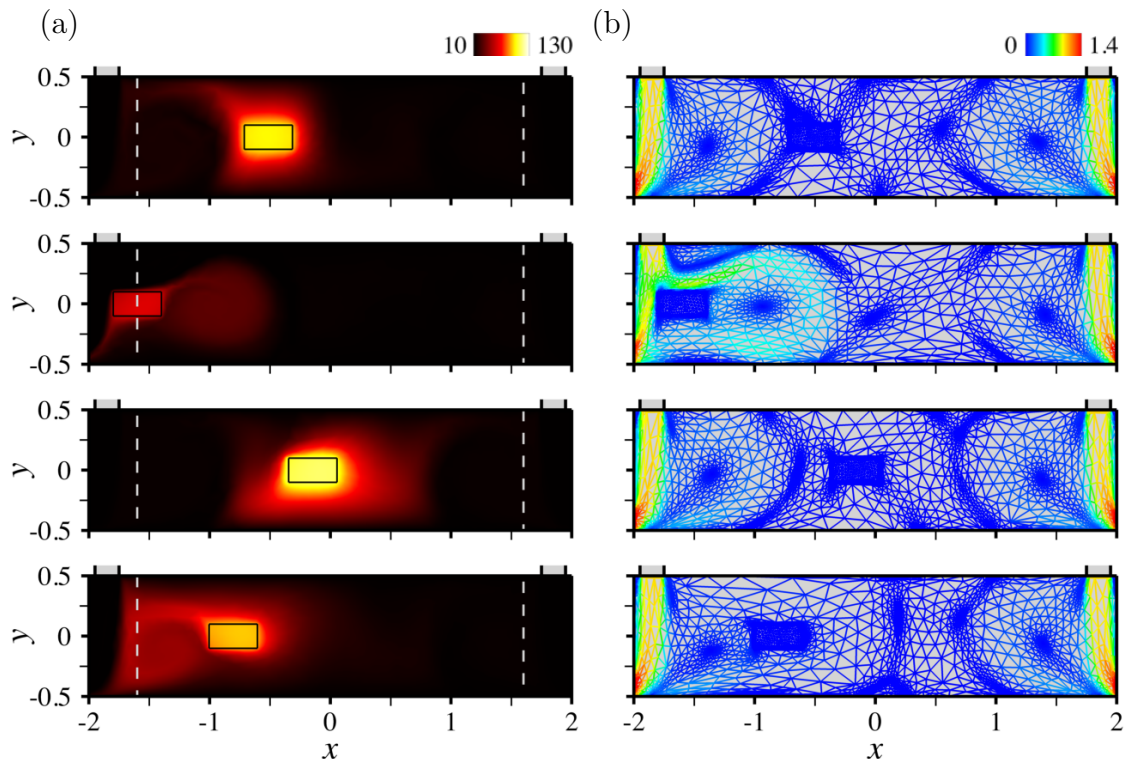


Figure 5.20: (a) Steady-state temperature against solid center of mass position, with admissible domains under the inverse strategy S_4 marked by the dashed lines. (b) Adapted meshes colored by the norm of velocity.

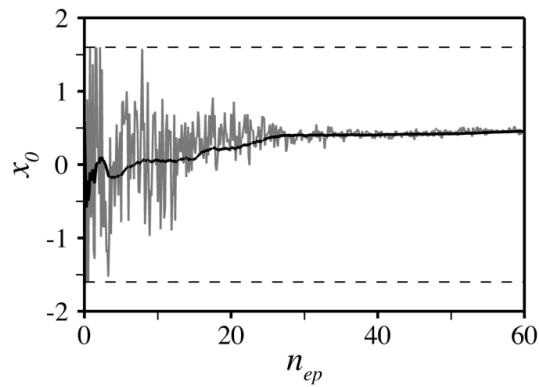


Figure 5.21: Evolution per learning episode of the instant (in grey) and moving average (in black) center of mass positions under the inverse strategy S_4 , with admissible values delimited by the dashed lines.

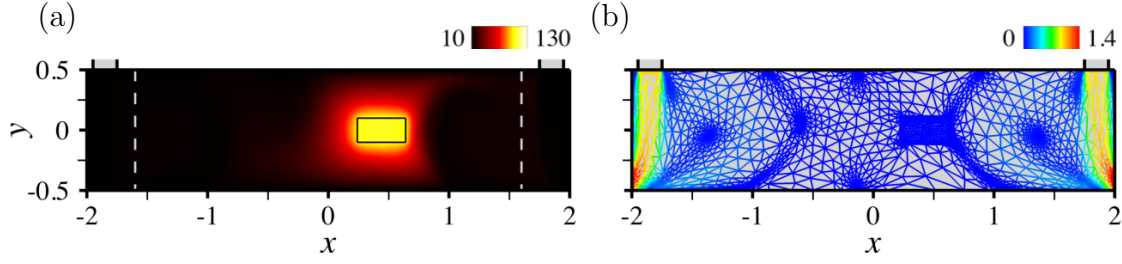


Figure 5.22: Same as figure 5.20 for the optimal center of mass position under the inverse strategy S_4 .

5.5.3.4 Inverse strategy

Finally, we propose here to make the most of the numerical framework flexibility to solve a different optimization problem consisting in selecting first an injector distribution, then in finding the position x_0 of the solid center of mass minimizing the magnitude of tangential heat flux (which is relevant for cases where the practitioner simply cannot act on the design). The so-called inverse strategy S_4 considered herein features two injectors at each end of the cavity ($x_1 = -1.85$ and $x_2 = 1.85$), identical to the optimal arrangement of 3 injectors under the free strategy S_3 . The center of mass can take any value in $[-x_{0m}; x_{0m}]$ where we set $x_{0m} = 2(H - h)$ to avoid numerical issues at the sidewalls. The same coordinate system as above is used, but with reference frame attached to the cavity, not the moving solid (hence all results obtained under the previous strategies pertain to $x_0 = 0$ in the new system).

A total of 60 episodes have been run for this case using the exact same DRL agent, the only difference being in the network action output, now made up of a single value $\hat{x}_0 \in [-1; 1]$, mapped into the actual position using

$$x_0 = x_{0m} \hat{x}_0. \quad (5.41)$$

A large variety of flow patterns is obtained by doing so, that closely resemble those computed under the previous strategies, only the outcome is now also altered by the width of the gap between the cavity sidewalls and the workpiece, as illustrated in figure 5.20. We show in figure 5.21 that the position of the solid center of mass converges to an optimal $x_0^* = 0.42$ (the variations over the same interval being by ± 0.005), the associated magnitude of tangential heat flux $\langle \|\nabla_{\parallel} T\| \rangle^* \sim 4.1$, being smaller than that achieved under S_3 using a centered workpiece. The fact that the optimal position is offset from the vertical centerline is a little surprising at first, because intuition suggests that the simplest way to achieve homogeneous heat transfer is by having symmetrically distributed injectors. Nonetheless, examining carefully the norm of the temperature gradient in the solid domain shows that $x_0 = 0$ achieves close to perfect horizontal symmetry but vertical asymmetry, owing to the formation of two large-scale, small velocity end vortices entraining heat laterally

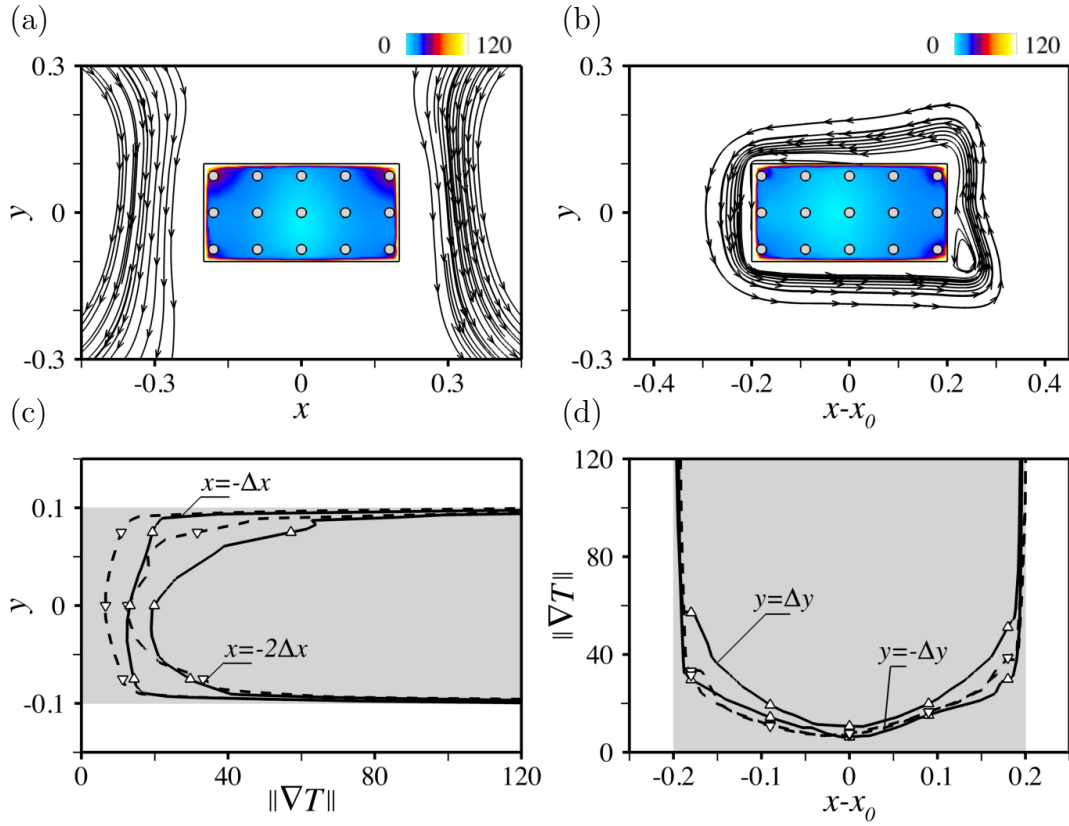


Figure 5.23: (a,b) Norm of the temperature gradient in the solid domain with superimposed streamlines of the underlying velocity field, as computed for (a) $x_0 = 0$, and (b) $x_0^* = 0.45$, i.e., the optimal position selected under the inverse strategy S_4 . (c) Cuts along the two leftmost columns of probes. The solid and dashed lines refer to $x_0 = 0$ and $x_0^* = 0.42$, respectively, and the symbols mark the probe values. (d) Same as (c) for cuts along the lower and upper rows of columns.

downwards; see figure 5.23(a). Conversely, for $x \sim x_0^*$, the workpiece is almost at the core of the closest recirculation region, hence the surrounding fluid particles have small velocities and wrap almost perfectly around its surface, as illustrated in figure 5.23(b). This restores excellent vertical symmetry, as evidenced by relevant cuts along the two leftmost columns of probes in figure 5.23(c), and along the lower and upper rows in figure 5.23(d), which explains the improved reward.

5.5.4 Discussion

Figure 5.24 reproduces the optimal temperature distributions computed under the various strategies considered above. For benchmarking purposes, we also provide

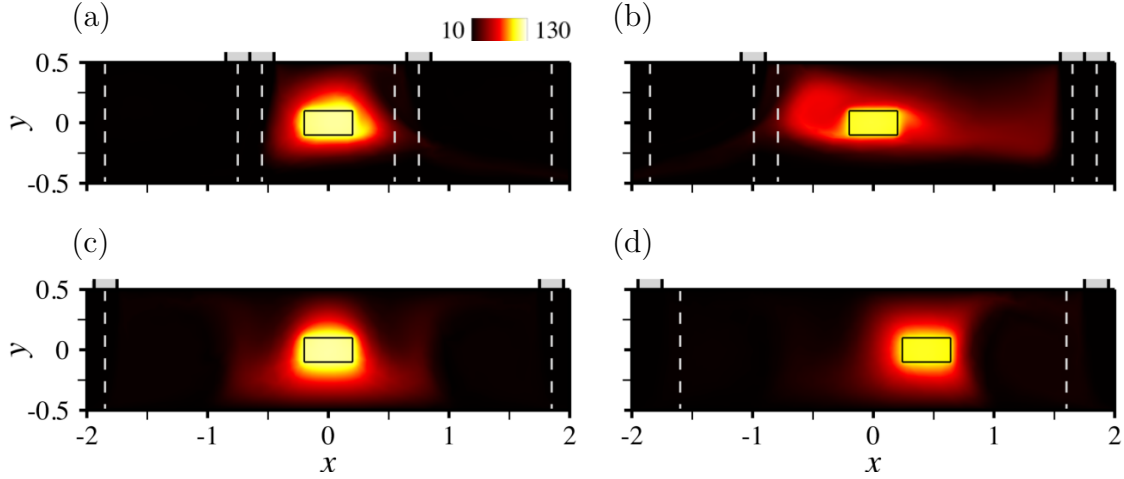


Figure 5.24: (a-c) Optimal arrangements of 3 injectors under the (a) fixed decomposition domain strategy S_1 , (b) follow-up strategy S_2 and (c) free strategy S_3 . (d) Optimal position of the workpiece under the inverse strategy S_4 .

	n_j	n_{ep}	x_0	x_1	x_2	x_3	$\langle \ \nabla_{\parallel} T\ \rangle$
S_1	3	60	0	-0.75	± 0.55	0.75	8.3
S_2	3	75	0	-0.96	1.65	1.85	6.3
S_3	3	60	0	-1.85	-1.82	1.85	11.2
S_4	2	60	0.42	-1.85	1.85	–	4.1

Table 5.3: Numerical data for the optimal arrangements computed under strategies S_{1-4} . All values computed by averaging the instant signal over the 10 latest learning episodes.

in table 5.3 relevant convergence data computed over the 10 latest episodes. To recap, the most homogeneous cooling is achieved under the follow-up strategy S_2 , but the DRL agent seems more easily trained under the fixed decomposition domain strategy S_1 and the free strategy S_3 . Another interesting point is the extent to which the workpiece is actually cooled, for which S_2 seems more relevant, on behalf of the dissymmetry in the left and right flow rates that creates order one velocities at the bottom of the cavity. This stresses S_2 as a possible compromise to achieve efficient *and* homogeneous cooling, although a true optimal with this regard can be computed rigorously by applying the same approach to compound functionals weighing, e.g., the magnitude of the tangential heat flux and the solid center temperature (which we defer to future work).

These results provide a basis for future self-assessment of the method and identifies potential for improvement regarding the convergence efficiency. The approach

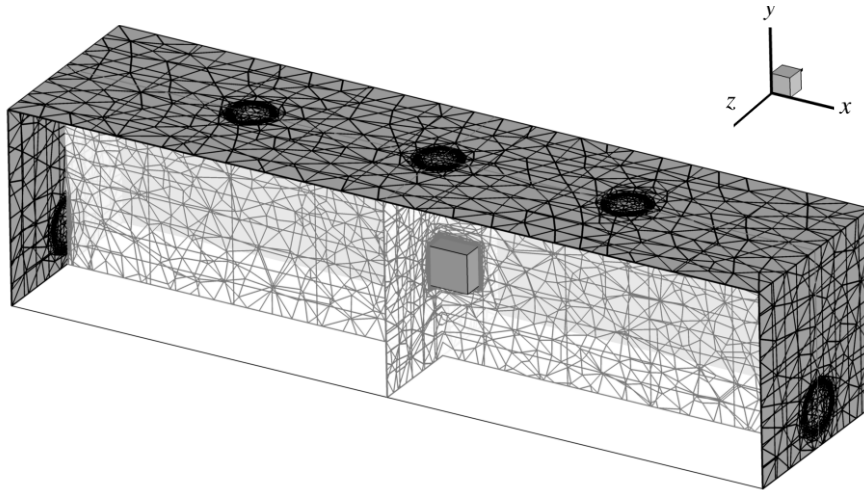


Figure 5.25: Schematic of the 3-D forced convection set-up.

can certainly benefit from a fine tuning of the reward computation, as having sufficient spatial resolution on the relevant state of the system is an obvious requirement to allow a successful control. Adjusting the trade-off between exploration and exploitation is also worth consideration to better handle the existence of multiple global optima (whether they stem from symmetries or from the topology of the reward itself) which could be done using non-normal probability density functions.

5.6 Extension to 3-D forced convection

5.6.1 Case description

The model cooling set up considered in section 5.5 is extended here to 3-D to assess the extent to which the approach carries over to three-dimensional conjugate heat transfer. The main differences between 2-D and 3-D are as follows: a Cartesian coordinate system is used with origin at the center of mass of the solid, horizontal x -axis, vertical y -axis, and the z -axis completes the direct triad; see figure 5.25. The solid is a rectangular prism with aspect ratio 2:1:1, and is fixed at the center of a rectangular cavity with height H and aspect ratio 4:1:1. We consider n_j circular-shaped injectors with diameter d_i , whose exit planes are forced to be symmetrical with respect to $z = 0$, hence each injector is identified by the horizontal position of its center $x_{k \in \{1 \dots n_j\}}$. We also use circular-shaped exhaust areas with diameter d_o , offset by a distance δ_o from the bottom of the cavity, and whose exit planes are also symmetrical with respect to $z = 0$, hence each exhaust area is identified by the vertical position of its center $(d_o + \delta_o - H)/2$. The governing equations are solved with the exact same boundary conditions as in section 5.5. All parameters are

H	h	d_i	d_0	δ_0	V_i	T_w	T_c	T_h	μ	ρ	λ	c_p	
1	0.2	0.2	0.24	0.16	1	10	10	150	0.01	1	0.5	1000	Fluid
									1000	100	15	300	Solid

Table 5.4: Numerical parameters used in the 3-D forced convection problem. All values in SI units, with the exception of temperatures given in Celsius.

provided in Table 5.4, including the material properties used to model the composite fluid, that yield fluid values of the Reynolds and Prandtl numbers

$$\mathbf{Re} = \frac{\rho V_i d_i}{\mu} = 20, \quad \mathbf{Pr} = 20. \quad (5.42)$$

5.6.2 Control strategy

We keep here the same control objective and compute the reward fed to the DRL from 45 probes arranged symmetrically into $n_z = 3$ transverse layers with resolution $\Delta z = 0.075$, each of which distributes uniformly 15 probes into $n_x = 5$ columns and $n_y = 3$ rows with resolutions $\Delta x = 0.09$ and $\Delta y = 0.075$. In practice, the 3-D reward is simply the average over z of the 2-D reward defined in section 5.5, hence $r_t = -\langle \|\nabla_{\parallel} T\| \rangle$ with

$$\langle \|\nabla_{\parallel} T\| \rangle = \frac{1}{(n_x + n_y)n_z} \sum_{i,j,k} \langle \|\nabla_{\parallel} T\| \rangle_{ik} + \langle \|\nabla_{\parallel} T\| \rangle_{jk}, \quad (5.43)$$

with

$$\langle \|\nabla_{\parallel} T\| \rangle_{ik} = \frac{2}{n_y - 1} \left| \sum_{j \neq 0} \text{sgn}(j) \|\nabla T\|_{ijk} \right|, \quad \langle \|\nabla_{\parallel} T\| \rangle_{jk} = \frac{2}{n_x - 1} \left| \sum_{i \neq 0} \text{sgn}(i) \|\nabla T\|_{ijk} \right|, \quad (5.44)$$

and the subscripts ik , jk and ijk denote quantities evaluated at $(x, z) = (i\Delta x, k\Delta z)$, $(y, z) = (j\Delta y, k\Delta z)$ and $(x, y, z) = (i\Delta x, j\Delta y, k\Delta z)$, respectively.

All results reported in the following are for $n_j = 3$ injectors. The edge values needed to map the network action output into the actual injectors positions deduce straightforwardly from (5.37)-(5.40) substituting the diameter d_i of the 3-D injectors for the length e_i of the 2-D injectors. The same DRL agent is used, that consists of two hidden layers, each holding 2 neurons, and the resolution process uses 8 environments and 2 steps mini-batches to update the network for 32 epochs. Each environment performs 1250 iterations with time step $\Delta t = 0.1$ to march the same

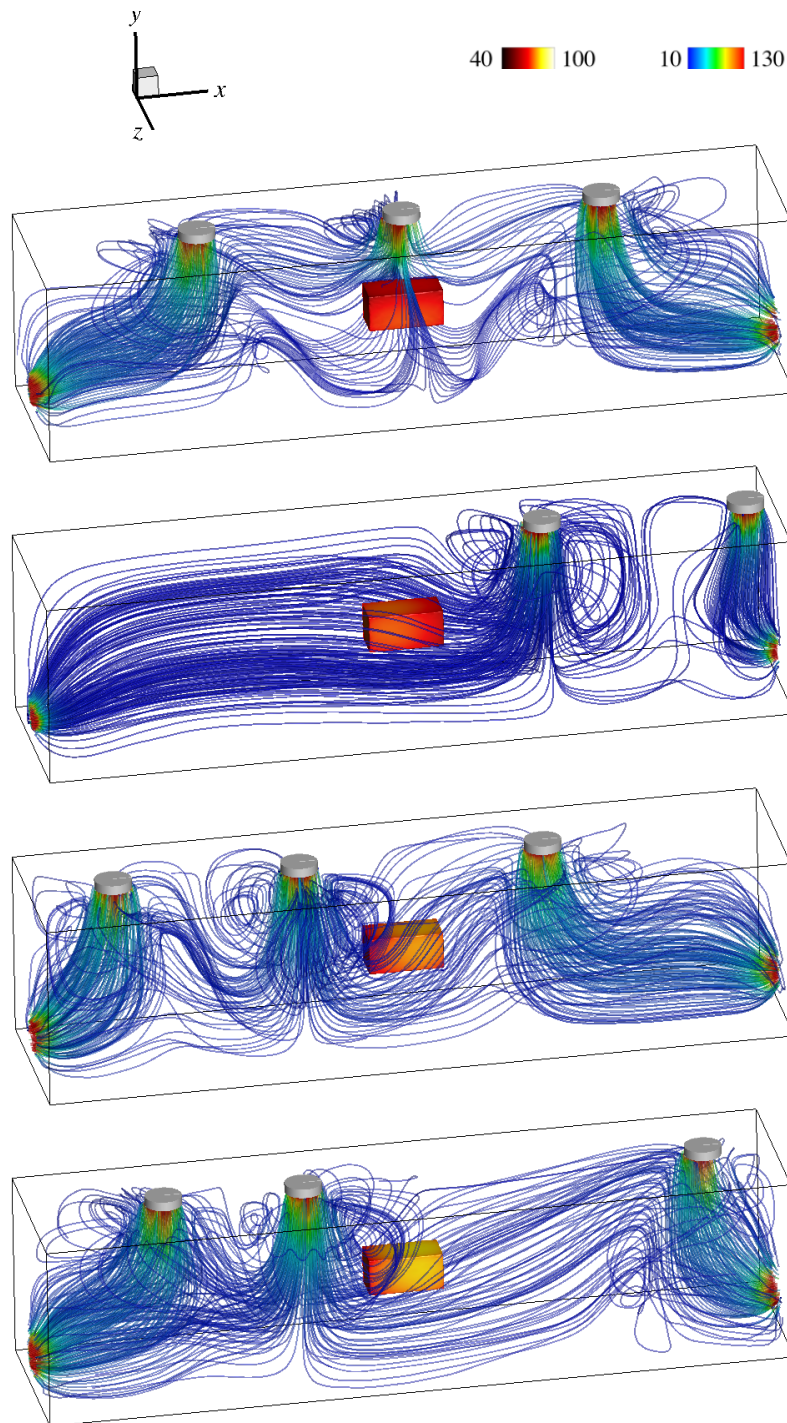


Figure 5.26: Representative steady-state temperature distributions at the solid/fluid interface together with 3-D streamlines colored by the magnitude of velocity.

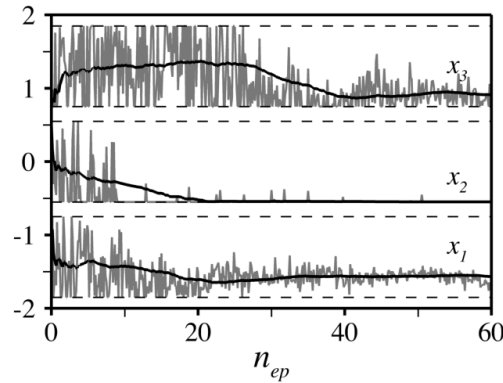


Figure 5.27: Evolution per learning episode of the instant (in grey) and moving average (in black) injector center positions under the three-dimensional fixed domain decomposition strategy S_1 , with admissible values delimited by the dashed lines.

initial condition (consisting of zero velocity and uniform temperature, except in the solid domain) to steady state, using the level set, velocity and temperature as multiple-component criterion to adapt the mesh (initially pre-adapted using the sole level set) every 10 time steps under the constraint of a fixed number of elements $n_{el} = 120000$. This is likely insufficient to claim true numerical accuracy, but given the numerical cost (320 3-D simulations per strategy, each of which is performed on 8 cores and lasts 2h30, hence 800h of total CPU cost), we believe this is a reasonable compromise to assess feasibility while producing qualitative results to build on.

5.6.3 Results

Only the fixed domain decomposition S_1 strategy (in which the top cavity wall is split into n_j equal subdomains and each injector is forced to sit in a different subdomain) and the free S_3 strategy (in which the injectors are entirely independent and free to move along the top cavity wall) are considered here to save computational resources, as learning has been seen to be slower in 2-D under the follow-up S_2 strategy.

A total of 60 episodes have been run under the fixed domain decomposition strategy S_1 . Several representative flow patterns computed over the course of optimization are shown in figure 5.26 via iso-contours of the steady-state temperature at the fluid-solid interface and 3-D streamlines colored by the magnitude of velocity, to put special emphasis on transverse inhomogeneities and display the increased degree of complexity due to the formation of large-scale horseshoe vortices wrapped around the nozzle jets. We show in figure 5.27 that the distribution slowly converges to an optimal arrangement consisting of one injector at the left end of the left subdomain ($x_1^* = -1.63$), another one at the left end of the center subdomain

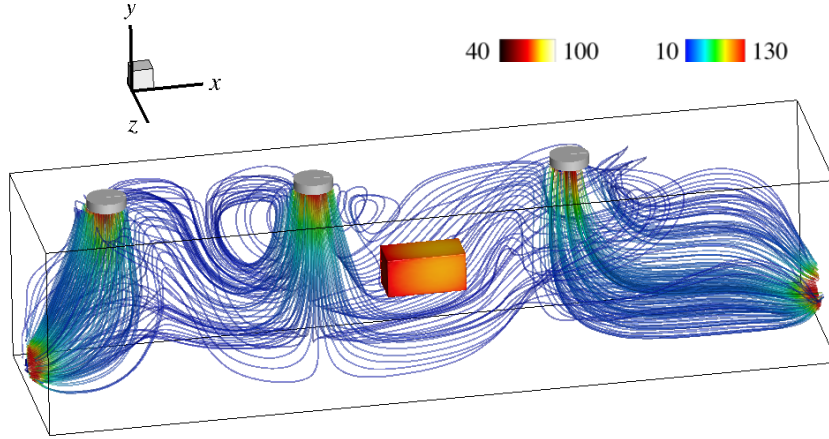


Figure 5.28: Optimal 3 injector arrangement under the three-dimensional fixed decomposition domain strategy S_1 .

($x_2^* = -0.55$), and a third one at the left end of the right subdomain ($x_3^* = 0.87$), as has been determined by averaging the instant positions of each injector over the latest 10 learning episodes, with variations by roughly ± 0.04 computed from the root-mean-square of the moving average over the same interval. This is larger by one order of magnitude than the variations reported in 2-D, as the agent keeps exploring slightly sub-optimal positions of the lateral injectors, which likely simply reflects the challenging nature of performing three-dimensional optimal control. The 3-D S_1 optimal somehow resemble its 2-D counterpart, namely the center injector is at the exact same position, while the lateral injectors (especially the leftmost one) have been pushed towards the cavity sidewalls. The associated flow pattern is reported in figure 5.28. The associated optimal reward computed over the same interval is $\langle \|\nabla_{\parallel} T\| \rangle^* \sim 19.5$, i.e. twice as large than in 2-D, although it is difficult to compare further because of the difference in the Reynolds and Prandtl number.

Another 40 episodes have been run under the free strategy S_3 , for which the results are almost identical to their 2-D counterparts, as the distribution converges in figure 5.29 to an optimal arrangement consisting of two overlapping injectors at the left end of the cavity ($x_1^* = -1.83$ and $x_2^* = -1.82$), and a third injector at the right end ($x_3^* = 1.83$), with variations by with ± 0.01 for the lateral injectors, but ± 0.03 for the center injector, for which the agent keeps occasionally exploring sub-optimal positions. The corresponding flow pattern shown in figure 5.30 is thus again symmetrical with two large, 3-D recirculation regions on either side of the workpiece. The associated optimal reward computed over the same interval is $\langle \|\nabla_{\parallel} T\| \rangle^* \sim 4.7$ substantially smaller than that achieved under the 3-D S_1 strategy, which again demonstrates the feasibility to improve performances by allowing overlaps. All relevant numerical data are reported in table 5.5 for the sake of completeness.

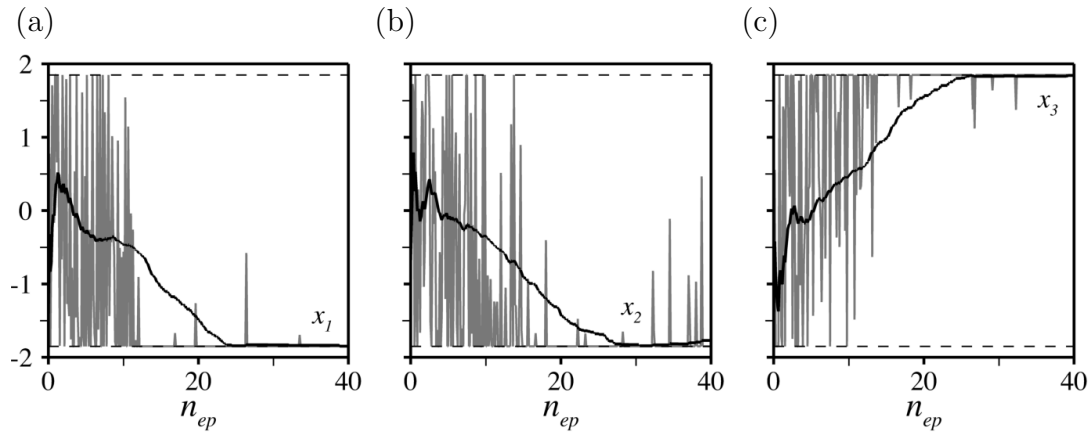


Figure 5.29: Evolution per learning episode of the instant (in grey) and moving average (in black) injectors center positions under the three-dimensional free strategy S_3 , with admissible values delimited by the dashed lines.

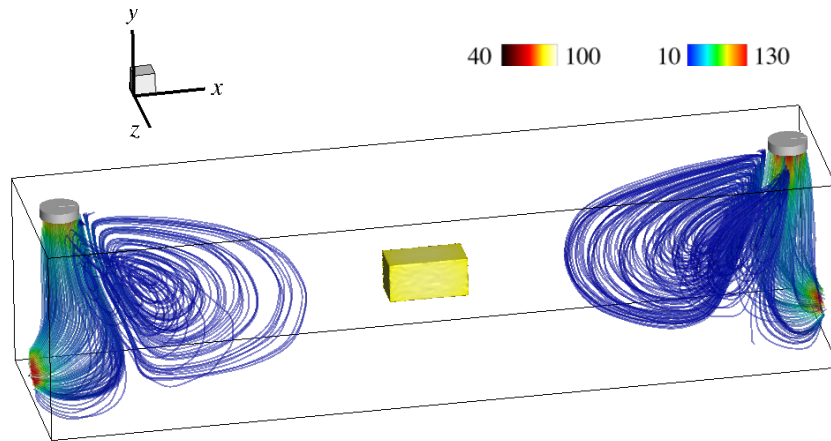


Figure 5.30: Optimal 3 injector arrangement under the three-dimensional free strategy S_3 .

	n_j	n_{ep}	x_0	x_1	x_2	x_3	$\langle \ \nabla_{\parallel} T\ \rangle$
S_1	3	60	0	-1.63	-0.55	0.87	19.5
S_3	3	40	0	-1.83	-1.82	1.83	4.7

Table 5.5: Numerical data for the optimal arrangements computed in three-dimensions under strategies S_1 and S_3 . All values computed by averaging the instant signal over the 10 latest learning episodes.

5.7 Conclusion

Optimization of conjugate natural and forced heat transfer systems is achieved here

training a fully connected network with a novel single-step PPO 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 finite elements CFD environment solving stabilized weak forms of the coupled Navier–Stokes and heat equations with a combination of variational multi-scale modeling, immerse volume method, and multi-component anisotropic mesh adaptation.

Convergence is assessed by alleviating the natural convection induced enhancement of heat transfer in a two-dimensional, differentially heated square cavity controlled by piece-wise constant fluctuations of the sidewall temperature. The approach is also relevant to forced convection problems, as single-step PPO shows capable of improving the homogeneity of temperature across the surface of two and three-dimensional hot workpieces under impingement cooling. Several control strategies are considered, in which the position of multiple cold air injectors is optimized relative to a fixed workpiece position, each of which mimics a different levels of design constraint. The flexibility of the numerical framework also allows solving the inverse problem, i.e., optimizing the workpiece position relative to a fixed injector distribution, which is relevant in situations where the design cannot be changed. The approach is beneficial in two important respects: first, it is efficient, even though the parameter spaces are large and it may be costly to identify optimal control parameters from simple parametric searches. Second, and more significantly, it is capable of determining additional optimal configurations, as the results of the inverse problem under symmetrical actuation indicate that the workpiece is best positioned offset from the symmetry axis, which had not been anticipated. Such results clearly stress that single-step PPO (and DRL in general) can be effective to explore and discover new solutions from unforeseen parameter combinations.

Fluid dynamicists have just begun to gauge the ability of DRL to design optimal control strategies. The efforts for developing single-step PPO are ongoing and remain at an early stage, so we do not expect the approach to compete right away with more established methods, for instance Evolution strategies (ES), a popular class of algorithms imitating principles of organic evolution processes as rules for black-box optimum seeking. ES rely on a stochastic description of the variables to optimize, i.e., they consider probability density functions, not deterministic variables. Simply put, at each generation (or iteration) new candidate solutions are sampled isotropically by variation of the current parental individuals according to a multivariate normal distribution. Recombination and mutation transformations are applied (that amount respectively to changing the mean and adding a random, zero-mean perturbation), after which the individuals with the highest cost function are selected to become the parents in the next generation. Improved variants include the covariance matrix adaptation evolution strategy (CMA-ES), that speeds up convergence by updating its full covariance matrix (which amounts to learning

a second-order model of the objective function). In present form, single-step PPO can be thought as an evolutionary-like algorithm with simpler heuristics (i.e., without an evolutionary update strategy, as the optimal model parameters are learnt via gradient ascent), so it is our guess that the performance should be comparable to that of standard ES algorithms with isotropic covariance matrix. Besides consolidating the acquired knowledge, future research should thus aim at improving efficiency (by fine-tuning the hyper parameters, or using pre-trained deep learning models) and convergence (by coupling with a surrogate model trained on-the-fly, using non-normal probability density functions, or modifying the balance between exploration and exploitation, as PPO prevents large updates of the policy to avoid the issue of performance collapse). For complex configurations representative of industrial applications, the implementation of properly designed numerical rewards (under partial state information) and noise reduction techniques is another issue that deserves consideration, as pointed out in [15].

Scope is another key ingredient to keep pushing forward the state of the art. The next step is to tackle more complex test cases exhibiting flow unsteadiness and turbulence, which the CFD environment is perfectly suited to do via a combination of Reynolds-averaged Navier–Stokes modeling [89, 90] and second-order, semi-implicit time discretization [207]. We believe that this will highlight even more clearly the relevance of the methodology, as [69] speculates that DRL should be able to handle chaotic systems without suffering from the shortcomings and limitations of the adjoint method, and it is shown in [17] to outperform a canonical linear proportional-derivative controller in controlling turbulent natural convection. The long-term objective would be to enrich the description of the test cases using multi-physics modeling (e.g., radiative heat transfer, phase transformation) in order to pave the way toward flexible, ready-to-use control of industrially relevant applications, such as thermal comfort for building design or manufacturing processes.

Chapter 6

Single-step DRL for two- and three-dimensional optimal shape design

Contents

6.1	Introduction	146
6.2	Methodology	149
6.2.1	Deep reinforcement learning	149
6.2.2	Single-step deep reinforcement learning	149
6.2.3	Policy based optimization	150
6.2.4	Computational fluid dynamics environment	151
6.2.5	Numerical implementation	153
6.3	Validation	156
6.3.1	Test case description	156
6.3.2	Results	159
6.3.3	Discussion	162
6.4	Application to optimal aerodynamic design	164
6.4.1	Test case description	164
6.4.2	Laminar regime at $Re = 250$	170
6.4.3	Turbulent (transitional) regime at $Re = 5000$	171
6.5	Extension to 3-D shape optimization.	172
6.6	Conclusion	173

This chapter is presented as a self-contained article submitted in 2022 to Physics of Fluids, hence, some details (related to motivation, scope and/or methodology) are repeated from chapters 1-3 to preserve the consistency of the whole content.

Ce chapitre évalue les capacités des techniques d'apprentissage par renforcement profond (DRL) pour la conception optimale directe de formes dans les systèmes de dynamique des fluides numérique (CFD). Il utilise Policy Based Optimization (PBO), un algorithme d'apprentissage par renforcement profond en une seule étape destiné aux situations où la politique optimale à apprendre par un réseau neuronal ne dépend pas de l'état. La récompense fournie au réseau neuronal est calculée à l'aide d'un environnement interne d'éléments finis stabilisés combinant une modélisation variationnelle multi-échelle (VMS) des équations gouvernantes, une méthode de volume immergé et une adaptation de maillage anisotrope multi-composant. Plusieurs cas sont abordés en deux et trois dimensions, pour lesquels des formes avec une ligne de cambrure, un angle d'attaque et une section transversale fixes sont générées en faisant varier une longueur de corde et une distribution symétrique de l'épaisseur (et éventuellement en extrudant dans la direction hors du corps). À incidence nulle, le cadre DRL-CFD proposé réduit avec succès la traînée du cylindre équivalent (c'est-à-dire le cylindre de même surface de section transversale) de 48% à un nombre de Reynolds de l'ordre de quelques centaines. Pour une incidence de 30°, il augmente le rapport portance/traînée de l'ellipse équivalente de 13% en deux dimensions et de 5% en trois dimensions à un nombre de Reynolds de l'ordre de quelques milliers. Bien que le faible nombre de degrés de liberté limite inévitablement l'éventail des formes réalisables, on constate systématiquement que la forme optimale est aussi performante qu'un profil aérodynamique conventionnel, bien que le DRL parte de zéro et n'ait aucune connaissance préalable des concepts aérodynamiques. De tels résultats démontrent le potentiel de la méthode pour l'optimisation de forme en boîte noire de systèmes CFD significatifs en pratique. Comme le processus de résolution est agnostique aux détails de la dynamique des fluides sous-jacente, il ouvre également la voie à une évolution générale des stratégies d'optimisation des formes de référence pour la mécanique des fluides et tout autre domaine où une fonction de récompense pertinente peut être définie.

This chapter gauges the capabilities of deep reinforcement learning (DRL) techniques for direct optimal shape design in computational fluid dynamics (CFD) systems. It uses Policy Based Optimization, a single-step DRL algorithm intended for situations where the optimal policy to be learnt by a neural network does not depend on state. The numerical reward fed to the neural network is computed with an in-house stabilized finite elements environment combining variational multi-scale (VMS) modeling of the governing equations, immersed volume method, and multi-component anisotropic mesh adaptation. Several cases are tackled in two and three dimensions, for which shapes with fixed camber line, angle of attack and cross-

sectional area are generated by varying a chord length and a symmetric thickness distribution (and possibly extruding in the off-body direction). At zero incidence, the proposed DRL-CFD framework successfully reduces the drag of the equivalent cylinder (*i.e.* the cylinder of same cross-sectional area) by 48% at a Reynolds numbers in the range of a few hundreds. At an incidence of 30° , it increases the lift to drag ratio of the equivalent ellipse by 13% in two dimensions and 5% in three dimensions at a chord Reynolds numbers in the range of a few thousands. Although the low number of degrees of freedom inevitably constrains the range of attainable shapes, the optimal is systematically found to perform just as well as a conventional airfoil, despite DRL starting from the ground up and having no priori knowledge of aerodynamic concepts. Such results showcase the potential of the method for black-box shape optimization of practically meaningful CFD systems. Since the resolution process is agnostic to details of the underlying fluid dynamics, they also pave the way for a general evolution of reference shape optimization strategies for fluid mechanics and any other domain where a relevant reward function can be defined.

6.1 Introduction

Shape optimization is ubiquitous in engineering applications ranging from magnetostatics [208], acoustics [209], image restoration and segmentation [210], composite material identification [211] to nano-optics [212], just to name a few. Shape optimization in fluid mechanics dates back to the pioneering work of Pironneau on the minimization of energy loss in Stokes and Navier–Stokes flows [213, 214]. Since then, it has become an increasingly important research topic in the attempt to enhance drag reduction capabilities, which is due to the ever growing concerns on aerodynamic energy efficiency (to give a taste, reducing the overall drag by just a few percent while maintaining lift can help reducing fossil fuel consumption and CO₂ emission while saving several billion dollars annually in ocean shipping or airline traffic [23]). In the following, the focus is essentially on airfoil shape optimization, a key component of aircraft flight mechanics that has come into prominence in a variety of other applications such as acoustic noise reduction [215] or energy harvesting [216]. One of the major challenges in the field is that the majority of flows of engineering interest are time-dependent and even turbulent (*e.g.* fluttering, buffeting, dynamic stall), and therefore require sophisticated unsteady methods and optimization techniques, thus drastically increasing the computational cost.

Shape optimization has historically been tackled by two main classes of approaches, namely gradient-based and gradient-free methods. Gradient-based methods rely on the evaluation of the gradient of the objective function with respect to the design parameters. They have proven effective in large optimization spaces when said gradient is computed by the adjoint method [29, 117, 118], whose cost is comparable to that of solving the governing equation (unlike more computationally expensive alternatives such as variance-based and regression-based methods, in which the governing equations need to be solved repeatedly, up to a hundred times). Nonetheless, gradient-based algorithms are easily trapped in local optima, meaning that the solution optimality can be very sensitive to the initial guess, all the more so when applied to stiff nonlinear problems [32]. Gradient-free methods are better equipped in this regard, but can be more complex to implement and to use. Among the available methods, genetic algorithms [217], particle swarm optimization [218] or metropolis algorithms [219] feature good global optimization capabilities, but they can be highly sensitive to heuristically chosen meta-parameters, plus their cost is usually higher and can easily exceed the available computational budget, thus limiting the number of design parameters [36]. It should be noted that both classes of methods can make use of cheap-to-evaluate surrogate models to approximate expensive objective and constraint functions without resorting systematically to numerical simulations [220]. Several approaches exist for building such surrogate models, *e.g.* polynomial response surfaces, radial basis functions, kriging, or supervised artificial neural networks [19], for which geometric parametrization plays a determinant role,

in terms of both the attainable geometries and the tractability of the optimization process [221].

The premise of this research is that the related task of selecting an optimal subset of design parameters can alternatively be assisted using deep reinforcement learning (DRL). DRL is the advanced branch of machine learning that couples deep neural networks (DNNs, a family of versatile parametric tools that can learn how to hierarchically extract informative features from data, and have gained traction as efficient computational processors for performing a variety of tasks, from exploratory data analysis to qualitative and quantitative predictive modeling) and reinforcement learning, a class of decision-making algorithms that can autonomously learn effective policies for sequential decision problems. In practice, DRL involves DNNs learning how to behave in an environment so as to maximize some notion of long-term reward, a task compounded by the fact that each action taken affects both immediate and future rewards. The feature extraction capabilities of DNNs, as well as their ability to handle quasi-arbitrary nonlinear input/output mappings, have lifted several major obstacles that hindered classical reinforcement learning and has led unprecedented efficiency in the context of nonlinear optimal control problems with high-dimensional state spaces. Several notable works using DRL in mastering games (e.g., Go, Poker) have stood out for attaining super-human level [48, 56], but the approach has also breakthrough potential for practical applications such as robotics [18, 57], computer vision [59], finance [58], autonomous cars [10, 222], or data center cooling [11].

The efforts for applying DRL to fluid mechanics are ongoing but still at an early stage, as recently reviewed in [223]. Nonetheless, the domain has undergone a large inflow of contributions with clear focus on drag reduction problems [39–45, 49, 69, 141, 224–226]. This enthusiasm is likely due to the increasing number of open-source initiatives [12, 39, 138], that has led to an accelerated diffusion of the methods in the community, and to the sustained commitment from the machine learning community, that has allowed concurrently expanding the scope from computationally inexpensive, low-dimensional reductions of the underlying fluid dynamics to complex Navier–Stokes systems [37, 38], all the way to experimental set-ups [15]. A handful of studies have recently provided insight into the performance improvements to be delivered in shape optimization, but it is worth emphasizing that figuring out a fixed shape that best meets a set of required criteria (*e.g.* high lift-to-drag ratio, low pressure loss) requires optimizing state-independent parameters, which is not *per se* the original purpose of DRL. Nonetheless, two main classes of methods have emerged in the community, namely the direct and incremental approaches. The incremental approach uses the state-to-action mapping as a way to incrementally modify an initial shape into an optimal one [21, 46, 47, 227], which exploits the capabilities of the DRL paradigm (in which network updates are performed after

multi-step episodes) in performing active flow control. The direct approach [138] conversely relies on single-step DRL, a subset of DRL in which network updates are performed after one-step episodes (hence the *stateless* moniker), and builds on recent efforts to assess the relevance of DRL in the context of open-loop control [69, 184].

This research is a follow-up on to our contribution showcasing the first application of DRL to direct shape optimization [138]. It uses Policy Based Optimization (PBO [70]), a novel single-step algorithm developed in-house, that improves the convergence rate of the previously used single-step Proximal Policy Optimization (PPO [18]) algorithm by adopting several key heuristics from the covariance matrix adaptation evolution strategy (CMA-ES). In short, PBO learns the mean, variance and correlation parameters of a multivariate normal search distribution from three separate neural networks, while single-step PPO updates the mean and variance (the same for all variables) from a single network, which can prematurely shrink the exploration variance. The objective is twofold: first, to further shape the capabilities of PBO for fluid mechanics applications (as it has so far been limited to textbook problems of analytic functions minimization), to help narrow the gap between DRL and advanced numerical methods for multi-scale, multi-physics computational fluid dynamics (CFD). Second, to gauge the feasibility of learning optimal designs from a low, yet suitable number of design parameters, for which Bézier curves, B-splines and NURBS are good candidates. We believe this is chief to mitigate the computational burden without deteriorating the geometric accuracy, since the parametrization in the direct approach provides a complete description of the shape itself, not that of a perturbation to a reference shape. The PBO agent is trained on high-fidelity CFD simulations, in contrast to most aforementioned studies about incremental shape optimization, in which a pre-trained surrogate or a simplified model is used for full agent training, or to perform an initial learning phase before re-training on a CFD environment using transfer learning. This is because the uncertainty of surrogate models cannot be quantified during optimization, which may misguide policy updating. We insist that it lies out of the scope of this paper to provide exhaustive performance comparison data against state-of-the art optimization techniques (e.g., evolution strategies or genetic algorithms). This would indeed require a tremendous amount of time and resources even though the efforts for developing the method remain at an early stage. Nonetheless, it is worth mentioning that PBO is shown in [70] to compare well against standard CMA-ES and to significantly outperform our previous PPO-based single-step algorithm, even though new algorithms cannot be expected to reach right away the level of performance of their more established counterparts.

The organization is as follows: section 6.2 introduces PBO (together with the baseline principles of DRL and single-step DRL), and outlines the main features of the finite element CFD environment used to compute the numerical reward fed to

the neural networks. Section 6.3 revisits the classical problem of finding the two-dimensional shapes minimizing drag in a uniform flow for the purpose of validation and assessment part of the method capabilities. In section 6.4, PBO is applied to more meaningful aerodynamic optimization problems consisting of finding the two-dimensional shapes maximizing the lift to drag ratio in the context of turbulent flows at moderately large Reynolds number (in the range of a few thousands). Finally, an extension to three-dimensional shapes is proposed in section 6.5.

6.2 Methodology

6.2.1 Deep reinforcement learning

Reinforcement learning (RL) is a process by which an agent learns to earn rewards through trial-and-error interaction with its environment. At each turn, the agent observes the state s_t of the environment and takes an action a_t , that prompts both the transition to the next state s_{t+1} and the reward received r_t . This repeats until some termination state is reached, the core objective of the agent being to learn the succession of actions maximizing its cumulative reward over an episode (this is the reference unit for agent update, best understood as one instance of the scenario in which it takes actions). In a deep reinforcement learning context (deep RL or DRL), the agent is a deep neural network (DNN) patterned after the neural circuits formed by neurons in human brains. The most general form of neural network architecture is the fully connected DNN, in which the processing units (the artificial neurons) are stacked in layers and information propagates forward from the input layer to the output layer via “hidden” layers. Each neuron performs a weighted sum of its inputs to assign significance with regard to the task the algorithm is trying to learn, adds a bias to figure out the part of the output independent of the input, and feeds an activation function that determines whether and to what extent the computed value should affect the outcome. The neural network learns to represent the relation between input (action) and output (reward) data by repeatedly adjusting the weights and biases by back-propagation, from the output layer back through the hidden layers to the input layer (a process known as training).

6.2.2 Single-step deep reinforcement learning

Single-step DRL is a subset of DRL that has recently emerged from the premise that tweaked versions of regular DRL algorithms can be used as black-box optimizers. The underlying idea is that it may be enough for the agent to interact only once per episode with its environment (hence, single-step episodes, and by extension, single-step DRL) if the optimal behavior to be learnt is independent of state, as is notably the case in optimization and open-loop control problems. The novelty

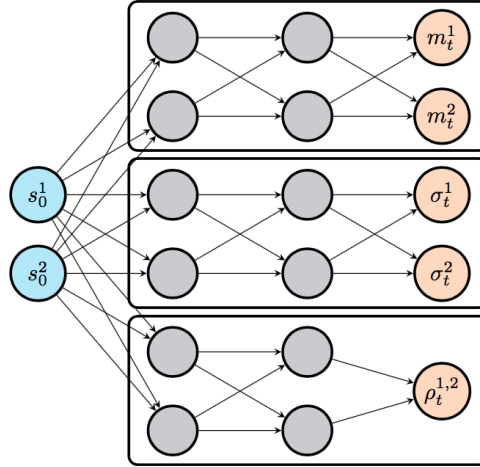


Figure 6.1: Policy networks used in PBO to map states to policy. Three networks trained separately are used for the prediction of mean, standard deviation, and correlation parameters. Orthogonal weights initialization is used throughout the networks, with a unit gain for all layers except the output layers, for which the gain is set to 10^{-2} .

of the approach can be summed up as follows: in DRL, a DNN learns the optimal set of observation-based actions a^* yielding the largest possible reward. In single-step DRL, it learns instead the optimal mapping f_{θ^*} such that $a^* = f_{\theta^*}(s_0)$, where s_0 is some input state (usually a constant vector) repeatedly fed to the agent for the optimal policy to eventually embody the transformation from s_0 to a^* . A direct consequence is that single-step DRL algorithms can use much smaller networks (compared to the usual agent architecture used in other DRL contributions), because the agent is not required to learn a complex state-action relation, but only a transformation from a constant input state to a given action.

6.2.3 Policy based optimization

The present research relies on policy-based optimization (PBO) a single-step, model free, off-policy gradient RL algorithm whose key features are summarized as follows:

- the agent interacts with the environment itself, not a surrogate model of the environment (model free, hence no assumptions about the fluid dynamics of the problems to be solved),
- its behavior is modeled after a parametrized probability distribution of actions $\pi_{\theta}(a)$, optimized by gradient ascent (policy gradient),

- the agent is not required to sample the training data with the current policy (off-policy),

PBO draws actions from a d -dimensional multivariate normal distribution (with d the dimension of the action required by the environment). A full co-variance matrix is used to improve the balance between exploration and exploitation (the single-step PPO algorithm used in [138] conversely assumes all variables to have the same variance and to be uncorrelated, which can prematurely shrink the exploration variance). The co-variance matrix also accelerates convergence to the optimum by aligning the contour of the sampling distribution with the contour lines of the objective function and thereby the direction of steepest ascent.

As shown in figure 6.1, three independent neural networks output the necessary mean, standard deviation, and correlation information, using hypersphere decomposition [72, 228] to generate valid symmetric, positive semidefinite covariance matrices. Different meta-parameters and architectures can be used for each network, which is shown in [70] to substantially impact the convergence rate. Actions drawn in $[-1; 1]^d$ are then mapped into relevant physical ranges, a step deferred to the environment as being problem-specific. Finally, the Adam algorithm [229] runs stochastic gradient ascent by computing adaptive learning rates (*i.e.* the step sizes to be taken in the gradient direction) for each policy parameter, using the gradient of the loss function

$$L(\theta) = \mathbb{E}_{a \sim \pi_\theta} \left[\max(\tilde{r}, 0) \log \pi_\theta(a) \right]. \quad (6.1)$$

In (6.1), \tilde{r} is the whitened reward normalized to zero mean and unit variance, considered a suitable advantage estimator. The rationale for this choice is as follows: as is customary in DRL, the discounted cumulative reward is approximated by the advantage function, that measures the improvement (if positive, otherwise the lack thereof) associated with taking action a in state s compared to taking the average over all possible actions. Because a single-step trajectory consists of a unique state-action pair, the discount factor adjusting the trade-off between immediate and future rewards can be set to unity, in which case the advantage reduces to the reward; see [69]. Substituting the whitened reward for r introduces bias but reduces variance, and thus the number of actions needed to estimate the expected value. Finally, the max allows discarding negative-advantage actions, that may destabilize learning when performing multiple mini-batch gradient steps using the same data (as each step drives the policy further away from the sampled actions).

6.2.4 Computational fluid dynamics environment

At the core of the CFD resolution framework is the in-house, CimLIB_CFD parallel finite element library [103], whose main ingredients are as follows:

- the variational multiscale approach (VMS) is used to solve a stabilized weak form of the governing equations using linear approximations (\mathbb{P}_1 elements) for all variables, which otherwise breaks the Babuska–Brezzi condition. The approach relies on an a priori decomposition of the solution into coarse and fine scale components [79, 144, 145]. Only the large scales are fully represented and resolved at the discrete level. The effect of the small scales is encompassed by consistently derived source terms proportional to the residual of the resolved scale solution, hence ad-hoc stabilization parameters comparable to local coefficients of proportionality.

- in laminar regimes, velocity and pressure come as solutions to the Navier–Stokes equations. In turbulent regimes, the focus is on phase-averaged velocity and pressure modeled after the unsteady Reynolds averaged Navier–Stokes (uRANS) equations. In order to avoid transient negative turbulent viscosities, negative Spalart–Allmaras [99] is used as turbulence model, whose stabilization proceeds from that of the convection-diffusion-reaction equation [100, 146].

- two-dimensional airfoil sections with fixed camber line are generated by varying a chord length and a thickness distribution. The chord direction is constant, just as the angle of attack measuring the incidence relative to the oncoming flow. The upper (suction/leeward) and lower (pressure/windward) sides are discretized into n_p control points equally spaced in the camber line direction. All shapes are closed and symmetrical with respect to the chord line, as achieved forcing zero thickness at the edges and identical (half)-thicknesses at each forward and rearward facing points. Consecutive points are connected by a cubic Bézier curve using local position and curvature information. The final step consists of sampling all Bézier curves and in exporting a closed loop, to be either used as an immersed mesh in a two-dimensional (2-D) environment, or extruded in the off-body direction to serve as an immersed mesh in a three-dimensional (3-D) environment.

- the immersed volume method (IVM) is used to immerse and represent all geometries inside a unique mesh. The approach combines level-set functions, using the zero-iso value of a signed distance function to localize the solid/fluid interface, and anisotropic mesh adaptation, to align the mesh element edges with the interface and refine the mesh interface under the constraint of a fixed, number of edges. This ensures that the quality of all actions taken over the course of a DRL optimization is equally assessed, even though the interface is action-dependent.

Substantial evidence of the flexibility, accuracy and reliability of the numerical framework for the intended application is documented in several papers to which the reader is referred for exhaustive details regarding the shape generation using Bézier curves [138, 230], the level-set and mesh adaptation algorithms [102, 103],

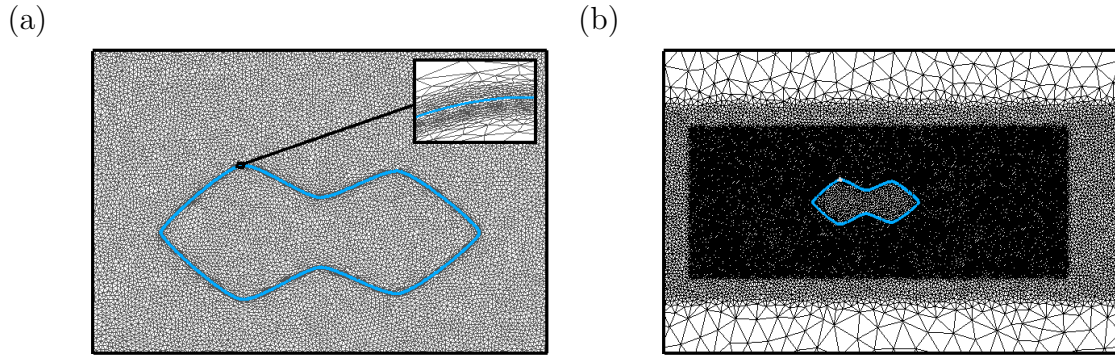


Figure 6.2: Details of (a) an anisotropic adapted mesh and (b) successive refinement steps of the background mesh. The blue line in (a) indicates the zero iso-contour of the level set function.

the VMS formulations, stabilization parameters and discretization schemes used in laminar and turbulent regimes [89, 90, 147, 148], and the mathematical formulation of the IVM in the context of finite element VMS methods [149, 150].

6.2.5 Numerical implementation

At each episode, actions drawn from the current policy are distributed to n_{env} environments running in parallel, each of which executes a self-contained MPI-parallel numerical simulation (here, all simulations are performed on a few tens of cores on a workstation of Intel Xeon E5-2640 processors) and feeds the reward associated to its input action to the DRL algorithm. There are thus two levels of parallelism related to the environment and the computing architecture. This simple parallelization technique is key to use DRL in the context of CFD applications, as a large number of actions drawn from the current policy must be evaluated to accurately compute the expected value of the policy loss (6.1). Even though, the high CPU cost of performing massive, unsteady numerical simulations involving hundreds of thousands (even millions) of degrees of freedom caps the number of environments that can efficiently run in parallel, and thus the number of state-action-reward triplets that can be sampled from the current policy (which also makes intractable the common practice in DRL studies to gain insight into the performances of the selected algorithm by averaging results over multiple independent training runs with different random seeds, as it would trigger a prohibitively large computational burden. The same random seeds are thus used for all computations to ensure a minimal level of performance comparison between cases.) PBO therefore improves the reliability of the loss evaluation by incorporating the reward data available from several previous episodes, using an empirical decay parameter that exponentially decreases the advantage history (to give recent episodes more weight) while retaining a longer

Mean	Variance	Correlation	Neural network
5×10^{-3}	\gg	10^{-3}	Learning rate
128	8	\gg	Nb. epochs
1	8	16	Nb. learning episodes
1	4	8	Nb. mini-batches
[2,2,2]	\gg	\gg	Architecture

Table 6.1: Details of the PBO meta-parameters and network architectures. For the architectures, only the sizes of the hidden layers are provided.

memory of the previous episodes as the problem dimensionality increases (in accordance with the idea that more state-action-triplets are then needed to build a coherent covariance matrix). The remainder of the practical implementation details are as follows:

- the environment consists of CFD simulations of incompressible flows described in a Cartesian coordinate system with drag (resp. lift) positive in the $+x$ (resp. $+y$) direction. All equations are discretized on 2-D and 3-D rectangular grids whose side lengths documented in the coming sections have been checked to be large enough not to have a discernible influence on the results (with the exception of the 3-D case in section 6.5, for which we favor computing all numerical solutions at affordable CPU cost using a limited transverse dimension). Open flow conditions are used, that consist of a uniform inflow in the x direction, together with symmetric lateral, advective outflow and no-slip interface conditions. In turbulent regime, the ambient value of the Spalart–Allmaras variable is three times the molecular viscosity, as recommended to lead to immediate transition. Typical adapted meshes of the interface and wake regions are shown in figure 6.2, the latter also being accurately captured via successive refinement of the background elements.

- optimal surface shapes subject to a target cross-sectional area S_{ref} are determined by maximizing a compound reward function

$$r = \overline{J} - \beta |S - S_{ref}|, \quad (6.2)$$

where J is the objective function associated to performance, S is the cross-sectional area (also abbreviated as CSA in the following) of the shape, the overline indicates time-averaging, and β is a weighting coefficient that increasingly penalizes the shape when its area strays away from the target value. In practice, the cross-sectional area

is computed as

$$S = \frac{1}{L} \int_{\Omega} H_{\epsilon}(\phi) \, d\Omega \quad (6.3)$$

where H_{ϵ} is the smoothed Heaviside function introduced in [231], and L is the extrusion length in the off-body direction (hence equal to unity in 2-D). Moving average rewards and actions are also computed as the sliding average over the 50 latest values (or the whole sample if it has insufficient size). Time averages are performed over an interval $[t_i; t_f]$ with edges large enough to dismiss the initial transient and achieve convergence to statistical equilibrium. In the following, we take J to be a function of the drag and lift coefficients per unit span in the transverse direction, denoted by D and L , respectively, whose instantaneous values are computed with a variational approach featuring only volume integral terms, reportedly less sensitive to the approximation of the body interface than their surface counterparts [152, 153].

- the agent consists of three identical fully connected networks with 3 hidden layers, each of which holds 2 neurons (this is by design, as we recall that the PBO networks can theoretically use different architectures). The only difference lies in the activation function applied to the output layer, namely the first network uses hyperbolic tangent to output the mean of the d -dimensional multivariate normal distribution in $[-1; 1]^d$, the second network uses sigmoid to output the standard deviations in $[0; 1]^d$, and the third network also uses sigmoid to output a set of coefficients in $[0, 1]^d$, eventually assembled into a full correlation matrix by hypersphere decomposition [72, 228]. As to the meta-parameters, the number of parallel environments used to collect rewards before performing the network updates is set from the well-established heuristics of CMA-ES (that similarly relies on full co-variance matrices and uses an evolution path to add information about correlations across consecutive generations [232] to

$$n_{env} = \lfloor 4 + 3 \ln d \rfloor, \quad (6.4)$$

where $\lfloor \cdot \rfloor$ denotes the floor function. Each network is updated for n_e epochs (the number of full passes of the algorithm over the entire data set) using a learning rate λ (the size of the step taken in the gradient direction for policy update) and a history of n_{ep} episodes, shuffled and organized in n_b mini-batches (whose sizes are in multiples of n_{env}). The values used in this study are documented in table 6.1 to ease reproducibility.

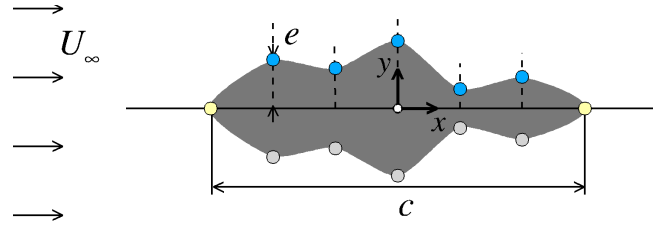


Figure 6.3: Schematic diagram of the minimum drag test case. The DRL agent optimizes the chord length, the curvature radius at the edge control points marked in yellow, and the thickness at the inner control points marked in blue. The thickness at the inner control points marked in grey deduces by symmetry.

6.3 Validation

6.3.1 Test case description

We assess first the relevance of the proposed numerical framework by revisiting the classical problem of finding the 2-D shape minimizing the drag force induced by a surrounding uniform flow at zero incidence. A sketch of the configuration is provided in figure 6.3. The origin of the coordinate system is at the half chord length. Several laminar cases at Reynolds number $\text{Re} = U_\infty \sqrt{S_{ref}} / \nu$ are modeled after the Navier–Stokes equations, where U_∞ is the inflow velocity, ν the kinematic viscosity, and we have used the square root of the target cross-sectional area (set to $S_{ref} = 1$ in our implementation) as reference length. The objective function is simply

$$J = -D, \quad (6.5)$$

and the weighing coefficient is set empirically to $\beta = 8$. All CFD environments use the simulation parameters documented in table 6.2, found to offer a good compromise between numerical accuracy and computational effort since numerical tests carried out at two other grid resolutions and spatial extents yield limited variations within 2% – 3%.

The control points used to parametrize the shape are labeled clockwise from 0 at the leading edge. All inner (*i.e.* non-end) curvature radii are set to 0.4 to provide sufficient smoothness (as this is a tad below the value 0.5 required for maximal smoothness). This leaves $n_p + 1$ independent design variables, the chord length c , two end curvature radii $\rho_{j \in \{0, n_p - 1\}}$ and $n_p - 2$ inner thicknesses $e_{k \in \{1, \dots, n_p - 2\}}$. The network action output consists accordingly of values $(\hat{c}, \hat{\rho}_j, \hat{e}_k)$ in $[-1; 1]^{n_p + 1}$, mapped

into the actual physical quantities using

$$c = \frac{1 - \hat{c}}{2}c_{min} + \frac{1 + \hat{c}}{2}c_{max}, \quad \rho_j = \frac{1 - \hat{\rho}_j}{2}\rho_{min} + \frac{1 + \hat{\rho}_j}{2}\rho_{max} \quad e_k = e_{k,max} - \frac{1 - \hat{e}}{2}\delta e, \quad (6.6)$$

for the chord to vary in $[c_{min}; c_{max}]$ with $c_{min} = 1$ and $c_{max} = 4$, the curvature radii to vary in $[\rho_{min}; \rho_{max}]$ with $\rho_{min} = 0.1$ and $\rho_{max} = 0.4$, and the thickness to vary in $[e_{k,max} - \delta e; e_{k,max}]$ with $\delta e = 0.4$ and $e_{k,max}$ a maximum value tuned locally for each problem. At each episode, the position of the inner points is adjusted to the current chord length to maintain equal spacing. Unless specified otherwise, all results documented hereafter are for $n_p = 5$, for which DRL evolves six design parameters, the chord length, two end curvature radii and three inner thicknesses.

					Case setup
	1	20	50	100	Reynolds number
	5	»	»	»	Nb. points
	6	»	»	»	Nb. design variables
					CFD
	2	»	»	»	Dimensionality
	0.2	»	0.125	»	Time-step
	[50;50]	»	»	»	Averaging time span
	8	»	»	»	Penalty coeff.
	$[-10; 20] \times [-10; 10]$	»	»	»	Mesh dimensions
	100000	»	110000	»	Nb. mesh elements
	0.0005	»	»	»	Interface \perp mesh size
					PBO
	100	120	115	»	Nb. episodes
	10	12	11	»	Nb. environments
	3mn	3mn	5mn	10mn	CPU time [†] ‡
	5h	6h	10h	18h	Resolution time [‡]
					Parameter ranges
	[1;4]	»	»	»	Chord length
	[0.1;0.4]	»	»	»	LE curv. radius
	[0.072;0.472]	»	»	»	↓ Thickness
	[0.152;0.552]	»	»	»	
	[0.072;0.472]	»	»	»	↓
	[0.1;0.4]	»	»	»	
					Optimal
	1.95	2.41	2.64	2.46	Chord length
	0.309	0.300	0.186	0.344	LE curv. radius
	0.297	0.246	0.223	0.290	↓ Thickness
	0.362	0.273	0.270	0.267	
	0.299	0.227	0.199	0.166	↓
	0.115	0.366	0.258	0.395	
	1.000	1.000	1.000	1.001	Ratio of actual to target CSA
	13.1	1.83	1.10	0.71	Drag (present)
	12.10	1.81	1.10	0.76	Drag [233]

Table 6.2: Case setup, simulation parameters and convergence data for the drag minimization problem, computed by averaging over the 10 latest learning episodes. Leading-edge (front end) and trailing edge (rear end) data are labeled LE and TE, respectively. † CPU times provided per episode and per environment. ‡ Values obtained averaging over 5 independent runs using 12 cores.

6.3.2 Results

Several Reynolds numbers have been considered up to $Re = 100$, for which random shapes collected over the course of the optimization, are presented in figures 6.4-6.7, together with their respective iso-contours of vorticity. Because the aspect ratio (as defined from the ratio of the maximum thickness to the chord length) barely exceeds unity, all solutions at $Re \lesssim 50$ relax to steady state regardless of the DRL action (hence we do not report a proper averaging span for these cases in table 6.2, as we simply evaluate reward at a final time chosen large enough to flush out the transient behavior). Meanwhile, a small number of shapes with aspect ratio close to unity have been found to exhibit vortex shedding at $Re = 100$, for which we pay attention to performing the necessary time averages. Figures 6.4-6.7 also provide exhaustive convergence history for the reward, the objective function, the ratio of actual to target CSA and the design parameters. The moving average reward especially decreases almost monotonically and reaches a plateau after a few ten episodes. At this point, the optimal CSA exhibits near-perfect agreement with the target value, hence evidencing the relevance of the reward penalty approach.

At $Re = 1$, the minimum drag body in figure 6.4 is that of a perfectly front-rear symmetric rugby ball, with chord length $1.95 \pm 0.6\%$ and aspect ratio $0.369 \pm 1.1\%$. These values have been obtained by averaging over the 10 latest episodes (with associated variance interval computed from the root-mean-square over the same interval, a simple yet robust criterion that will be used systematically to assess convergence for all cases reported in the following). They are close to the creeping flow optimal, whose chord length (relative to a unit target surface) and aspect ratio derived analytically in [234] are 1.88 and 0.40, respectively. The only noticeable difference lies in the fact that the DRL optimal has a pointed rear end with wedge angle about 90° , and a slightly more rounded front end with wedge angle $\sim 120^\circ$, while the creeping flow optimal has two pointed ends with wedge angle about 100° . As the Reynolds number increases, the optimal chord length increases but the thickness decreases, hence the aspect ratio of the optimal body decreases (likely because the increasing adverse pressure gradient at the front needs to be counterbalance to avoid flow separation). At $Re = 20$, the optimal shape in figure 6.5 has chord length $2.40 \pm 0.8\%$ and aspect ratio $0.228 \pm 1.5\%$, and remains almost front-rear symmetric, although the rear section is slightly more streamlined. Similar results are obtained at $Re = 50$ (figure 6.6), with chord length $2.65 \pm 0.7\%$ and aspect ratio $0.204 \pm 1.0\%$. At $Re = 100$, the front-rear symmetry is lost as we obtain a streamlined shape with chord length $2.46 \pm 0.4\%$ and aspect ratio $0.235 \pm 0.8\%$; see figure 6.7. At $Re = 1$, the optimal drag ($13.10 \pm 0.02\%$) cuts down that of the equivalent cylinder (*i.e.* the cylinder of diameter $2\sqrt{S_{ref}/\pi}$, for the area to be equal to S_{ref}) by 6%, which is small but simply reflects that the ratio of drags on any two bodies tends to 1 in the limit where the Reynolds number tends to 0. In comparison, the achieved reduction

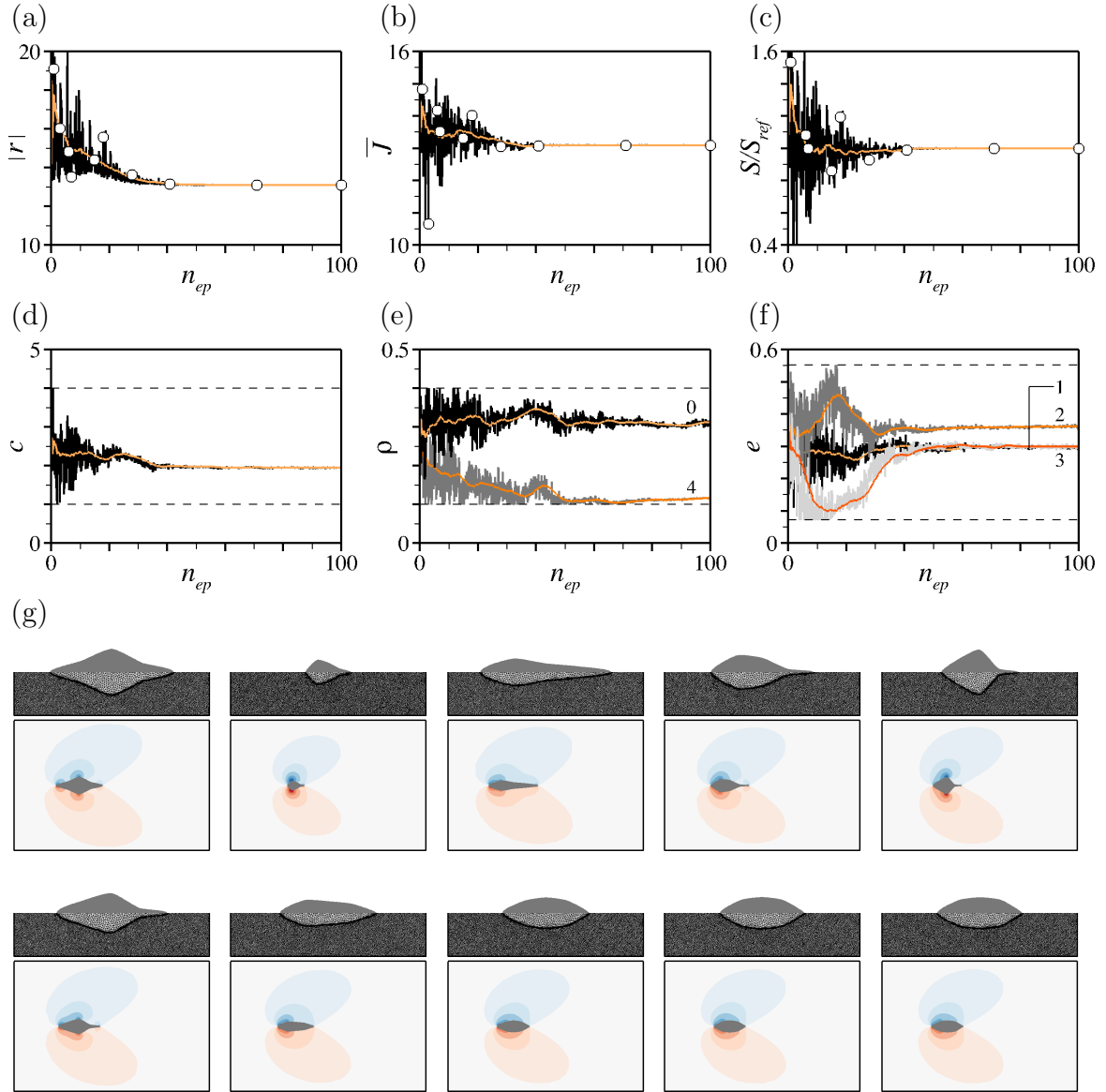


Figure 6.4: Maximum lift to drag ratio test case at $Re = 1$ under constant area constraint $S_{ref} = 1$. (a) Evolution per episode of the instant (black line) and moving average (over episodes, light orange line) reward (in absolute value). (b-f) Same as (a) for the (b) averaged (over time) drag, (c) ratio of the actual to target cross-sectional areas, (d) chord, (e) edge curvature radii and (f) inner thicknesses. All labels in (e-f) are ordered clockwise from the leading edge. The horizontal dashed lines in (d-f) mark the admissible values. (g) Shapes generated over the course of optimization for random episodes marked by the circle symbols in (a-c), together with corresponding iso-contours of vorticity. The last three shapes pertain to episodes 40, 70 and 100, respectively.

is by 22% at $Re = 20$ (optimal drag $1.83 \pm 0.04\%$), 24% at $Re = 50$ ($1.10 \pm 0.05\%$), and 48% at $Re = 100$ ($0.71 \pm 4\%$).

Other than that, it is difficult to accurately validate the results because, although the search for optimal profiles of minimum drag in Navier–Stokes flows having received substantial interest in the literature, there is a wide variability in the problem formulation, especially in terms of design constraints (some authors specify a target surface, others impose only a lower bound, plus the values can vary from one reference to another), and the exact geometrical properties of the optimal (*e.g.* length, aspect ratio) are rarely documented. The closest study to our work is from Kondoh *et al.* [233], who tackle similar drag minimization problems via topology optimization, using a body force to model the effect of classical no-slip boundary conditions at the fluid/solid interface. It has not been possible to assess the convergence rate of DRL in the absence of any reference information in this regard, and it is not entirely clear whether the exact same optimization problem is solved due to inconsistent statements in the study regarding the nature of the design constraint, but even so, the reported optimal shapes and drags turn to be in good agreement with the present DRL results. One minor difference is that the shapes look pointed in [233] (but the blending of the interface makes it difficult to see in the original images), while the present ones are generally rounded at both ends, with little to no effect on the reward. On this point, we note that the end radii can vary substantially even after the reward has converged (as is the case for instance in figure 6.6(d) at $Re = 50$), which evidences a general lack of sensitivity to these specific design parameters). At $Re = 1$, the optimal drags differ by approximately 7%, which may seem large at first sight but is actually fair given the high sensitivity of drag to small changes in the Reynolds number in this regime. The drags and chord lengths are nearly identical at $Re = 20$ and 50, as we find the ratio of the chord length at the current Reynolds number to its $Re = 1$ counterpart to be 1.24 at $Re = 20$ and 1.35 at $Re = 50$ using DRL, while extracting data from the reference figures using a graph digitizer software yields values of 1.26 at $Re = 20$ and 1.33 at $Re = 50$ (it has not been possible to similarly extract the aspect ratio due to blurred and/or mixed pixels). At $Re = 100$, the shapes somewhat differ as the optimal in [233] is more elongated and less streamlined in the rear section. Meanwhile the optimal drags differ by only 6%, which raises the possibility that the objective function has either a unique flat minimum, or several nearly equivalent minima. Figure 6.7 constitutes a favorable presumption in this regard, as the objective function exhibits surprisingly low variations over the course of optimization, and most shapes in figure 6.7(b) actually are within the 6% variance interval marked by the grey shade.

6.3.3 Discussion

We believe the above results assess the relevance of the proposed DRL-CFD framework for optimal shape design. Relying on low-dimensional parametrization of the body shape is one key parameter in this regard, as it improves the tractability of the optimization process and avoids the oscillations between points that have been found to occur when using a larger (about 10) number of control points. Nonetheless, we believe important to discuss the impact on robustness, and the extent to which decreasing the number of control points exaggerates (or not) the sensitivity to the curvature radii. This is because using different curvature radii to connect the same set of control points can yield two slightly different cross-sectional areas, that in turn can earn two substantially different reward via the penalization term (this is not on the Bézier parametrization itself, though, only on the need to smoothly connect a discrete set of control points. For instance, one must also specify tangency at both endpoints of a spline).

As a first insight into this issue, we report here results obtained at $Re = 1$ using three alternative parametrizations:

- a case with $n_p = 7$ control points evolving the chord length, five inner thicknesses and two end curvature radii (which amounts to replicating the above reference case, but with two additional inner thicknesses, hence 8 independent design parameters),
- a case with $n_p = 5$ points evolving the chord length, three inner thicknesses, two end curvature radii, plus an additional radius common to all inner control points (which amounts to replicating the reference case, but with one additional inner curvature radius, hence 7 independent design parameters),
- a case with $n_p = 7$ points whose thickness distribution is frozen, as obtained interpolating from the reference $n_p = 5$ optimal (for which it suffices to sample the connecting Bezier curves at the relevant positions), after which a dedicated DRL agent restores the proper cross-sectional area by evolving two end curvature radii, plus an additional radius common to all inner control points (hence, 3 independent design parameters) with reward

$$r = -|S - S_{ref}|, \quad (6.7)$$

formally identical to (6.2) with $J = 0$ and $\beta = 1$.

The results reported in table 6.3 exhibit limited discrepancy with respect to the reference (reproduced from table 6.2 in the first column), as the maximum deviation on the chord and the inner thicknesses is by 4%. All runs converge to similar curvature radii at the front. The value at the rear is noticeably different, but with

Case setup				
5	5	7	7	Nb. points
×	✓	×	✓	Inner curv. radius
6	7	8	3	Dimensionality
Optimal				
1.95	1.96	1.87	1.95	Chord length
0.4	0.332	0.4	0.398	Inner curv. radius
0.309	0.359	0.310	0.394	LE curv. radius
0.297	0.284	0.233	0.206	<div style="display: flex; align-items: center;"> <div style="border-left: 1px solid black; border-right: 1px solid black; width: 10px; height: 20px; margin-right: 5px;"></div> Thickness </div>
0.362	0.367	0.340	0.324	
0.299	0.303	0.369	0.359	
-	-	0.341	0.331	<div style="display: flex; align-items: center;"> <div style="border-left: 1px solid black; border-right: 1px solid black; width: 10px; height: 20px; margin-right: 5px;"></div> ↓ </div>
-	-	0.258	0.227	
0.115	0.159	0.392	0.389	TE curv. radius
1.00	1.00	1.00	1.00	Ratio of actual to target CSA
13.10	13.09	13.09	13.09	Drag

Table 6.3: Sensitivity of the drag minimization problem at $Re = 1$ to the discretization parameters. Leading-edge (front end) and trailing edge (rear end) data are labeled LE and TE, respectively. The first column is reproduced from table 6.2.

little to no effect on the reward and the objective function, which simply reflects the smallness of the reward gradients with respect to the control variables in the vicinity of the optimal. Although the impact needs to be assessed on a case to case basis, this suggests that the method ability to provide robust optima may not be strained by the use of low-end geometrical parametrizations.

6.4 Application to optimal aerodynamic design

6.4.1 Test case description

We apply now the method to more meaningful aerodynamic shape optimization problems, as we seek the shape maximizing the lift to drag ratio (used as an indicator of the aerodynamic efficiency) induced by a surrounding uniform flow at angle of attack of $\alpha = 30^\circ$. A sketch of the configuration is provided in figure 6.9. A Cartesian coordinate system is used with origin at quarter chord length from the leading edge. The target cross-sectional area is set to $S_{ref} = 0.0822$, which corresponds to the CSA of a NACA (National Advisory Committee for Aeronautics) 0012 airfoil. The objective function is

$$J = \frac{L}{D}, \quad (6.8)$$

and the weighing coefficient is set to $\beta = 100$. Two time-dependent flow regimes are modeled after either the Navier–Stokes or the uRANS equations, for which all CFD environments use the numerical simulation parameters provided in table 6.4.

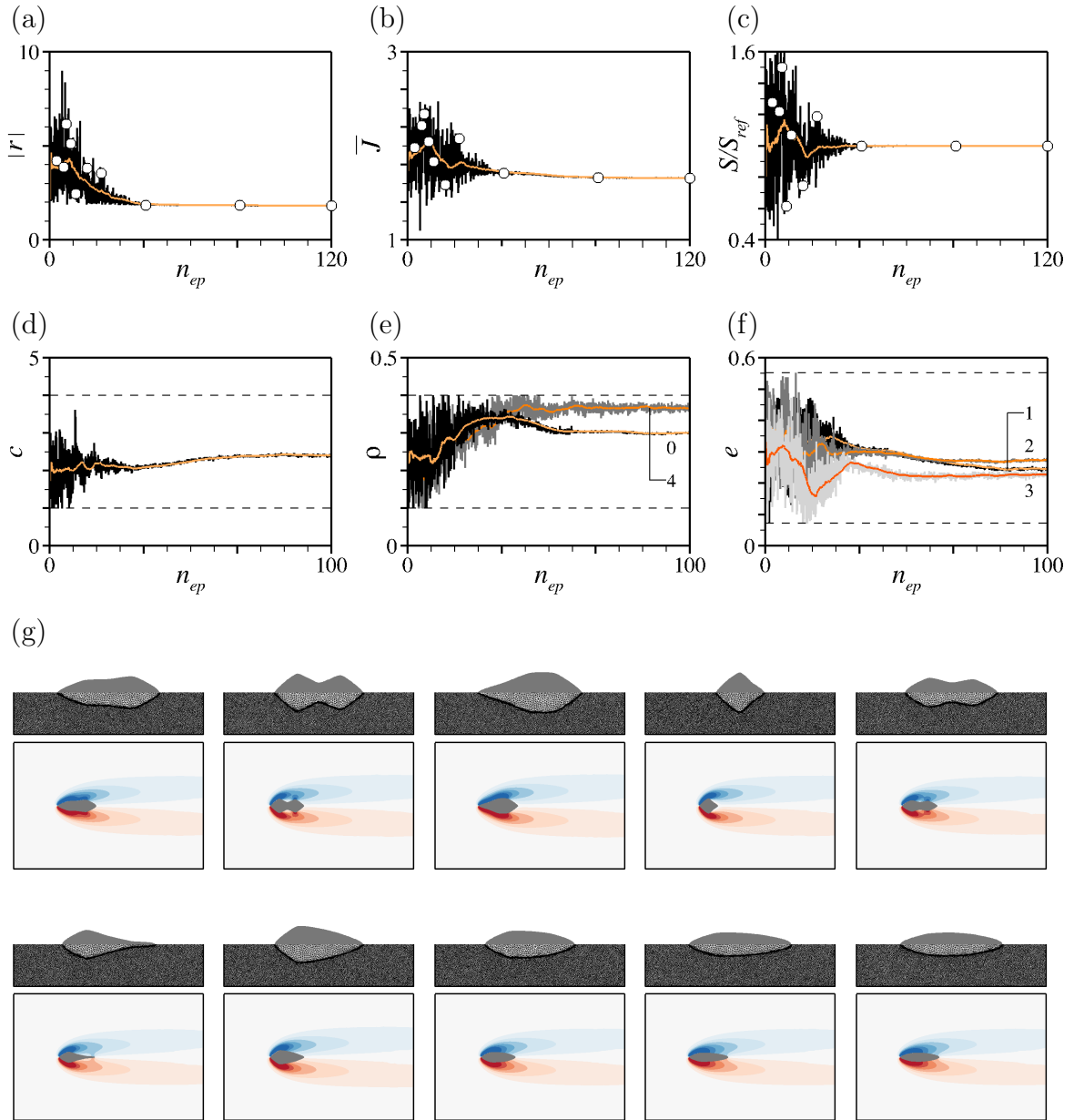


Figure 6.5: Maximum lift to drag ratio test case at $Re = 20$ under constant area constraint $S_{ref} = 1$. (a) Evolution per episode of the instant (black line) and moving average (over episodes, light orange line) reward (in absolute value). (b-f) Same as (a) for the (b) averaged (over time) drag, (c) ratio of the actual to target cross-sectional areas, (d) chord, (e) edge curvature radii and (f) inner thicknesses. All labels in (e-f) are ordered clockwise from the leading edge. The horizontal dashed lines in (d-f) mark the admissible values. (g) Shapes generated over the course of optimization for random episodes marked by the circle symbols in (a-c), together with corresponding iso-contours of vorticity. The last three shapes pertain to episodes 40, 80 and 120, respectively.

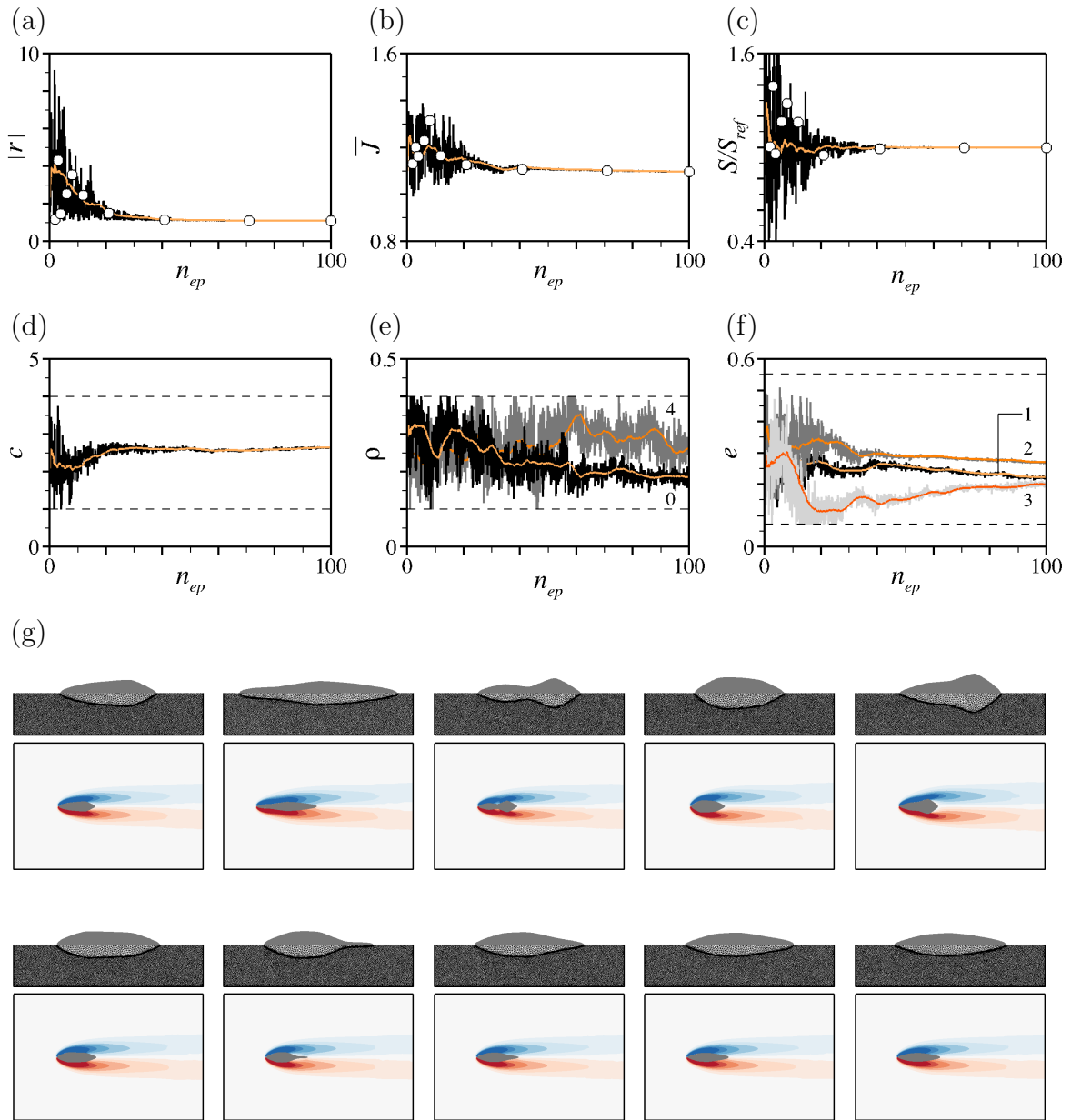


Figure 6.6: Maximum lift to drag ratio test case at $Re = 50$ under constant area constraint $S_{ref} = 1$. (a) Evolution per episode of the instant (black line) and moving average (over episodes, light orange line) reward (in absolute value). (b-f) Same as (a) for the (b) averaged (over time) drag, (c) ratio of the actual to target cross-sectional areas, (d) chord, (e) edge curvature radii and (f) inner thicknesses. All labels in (e-f) are ordered clockwise from the leading edge. The horizontal dashed lines in (d-f) mark the admissible values. (g) Shapes generated over the course of optimization for random episodes marked by the circle symbols in (a-c), together with corresponding iso-contours of vorticity. The last three shapes pertain to episodes 40, 70 and 120, respectively.

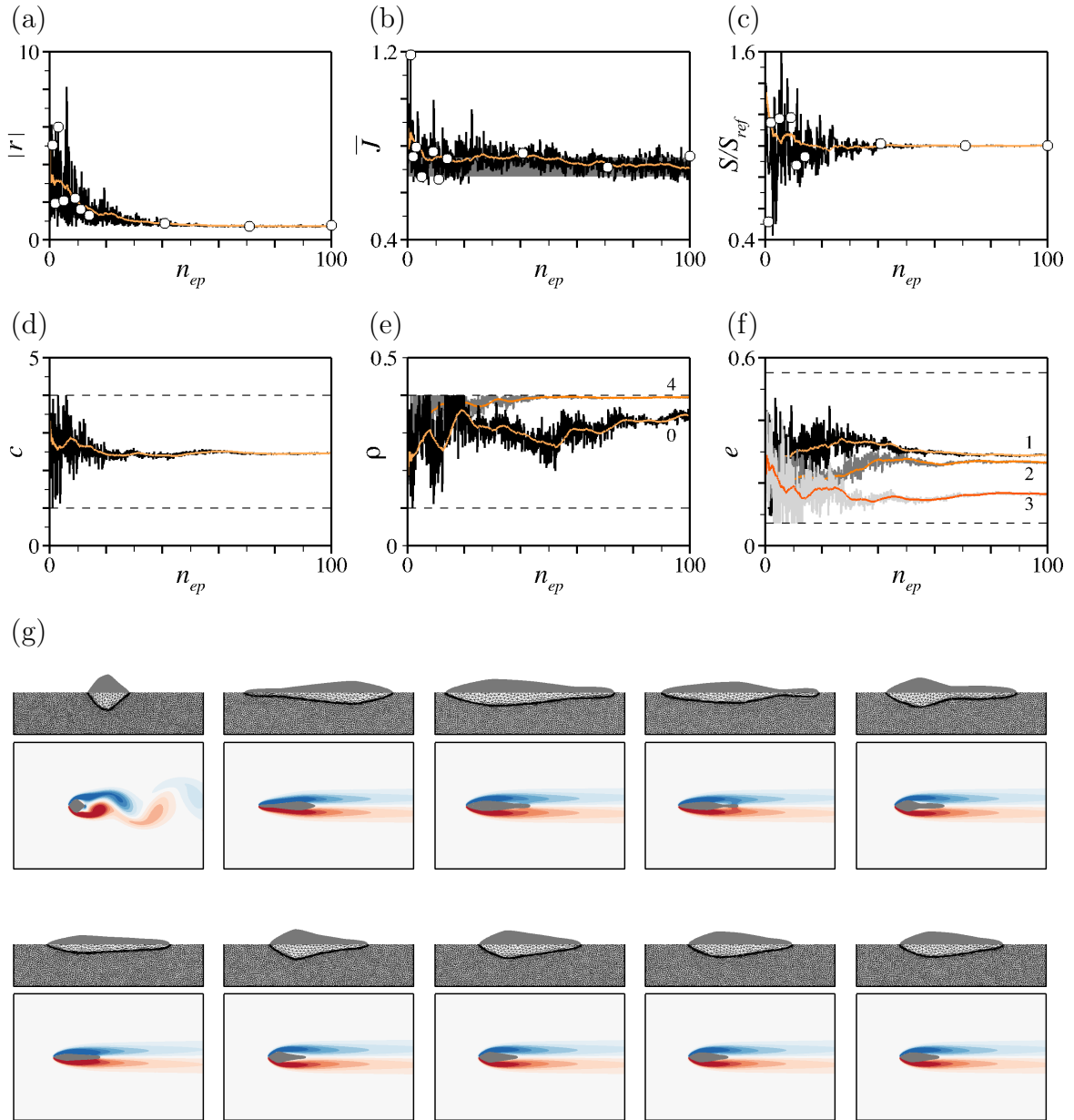


Figure 6.7: Maximum lift to drag ratio test case at $Re = 100$ under constant area constraint $S_{ref} = 1$. (a) Evolution per episode of the instant (black line) and moving average (over episodes, light orange line) reward (in absolute value). (b-f) Same as (a) for the (b) averaged (over time) drag, (c) ratio of the actual to target cross-sectional areas, (d) chord, (e) edge curvature radii and (f) inner thicknesses. The grey shade in (b) marks the 6% variance interval with respect to the average over the 10 latest learning episodes. All labels in (e-f) are ordered clockwise from the leading edge. The horizontal dashed lines in (d-f) mark the admissible values. (g) Shapes generated over the course of optimization for random episodes marked by the circle symbols in (a-c), together with corresponding iso-contours of vorticity. The last three shapes pertain to episodes 40, 70 and 120, respectively.



Figure 6.8: (a) Reference optimal shape for the minimum drag test case at $Re = 1$ with $n_p = 5$ control points and fixed inner curvature radius. (b) Same as (a) with $n_p = 7$ control points and fixed inner curvature radius. (c) Same as (a) with $n_p = 5$ control points and variable inner curvature radius. (d) Reference optimal shape discretized with $n_p = 7$ control points, after DRL has adjusted the end and inner curvature radii to restore the proper cross-sectional area.

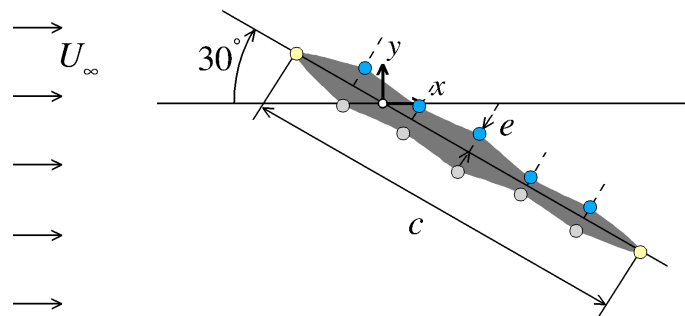


Figure 6.9: Schematic diagram of the maximum lift to drag ratio test case.

				Case setup
250	5000	»		Reynolds number
5	»	»		Nb. points
5	»	3		Nb. design variables
				CFD
2	»	3		Dimensionality
-	RANS	»		Turb. model
0.125	0.05	»		Time-step
[100;150]	[150;200]	[100;150]		Averaging time span
100	»	90		Penalty coeff.
$[-10; 20] \times [-10; 10]$	$[-6; 15] \times [-7; 7]$	$[-5; 10] \times [-5; 5] \times [0; 5]$		Mesh dimensions
100000	120000	500000		Nb. mesh elements
0.0005	»	0.001		Interface \perp mesh size
				PBO
100	100	80		Nb. episodes
14	14	12		Nb. environments
20mn	2h45mn	9h30mn		CPU time [†]
35h	275h	760h		Resolution time [†]
				Parameter ranges
-	-	-		Chord length
[0.1;0.4]	»	-		LE curv. radius
[0.024;0.084]	»	»		↓ Thickness
[0.03;0.09]	»	»		
[0.024;0.084]	»	»		↓ TE curv. radius
[0.1;0.4]	»	-		
				Optimal
1	1	1		Chord length
0.394	0.398	0.3		LE curv. radius
0.0638	0.0549	0.0420		↓ Thickness
0.0514	0.0627	0.0536		
0.0253	0.0252	0.0454		↓ TE curv. radius
0.156	0.104	0.1		
1.00	1.00	0.996		Ratio of actual to target CSA
1.24	1.54	1.34		Lift to drag ratio

Table 6.4: Case setup, simulation parameters and convergence data for the lift to drag ratio maximization problem, as computed by averaging over the 10 latest learning episodes. † All CPU times provided per episode and per environment. ‡ All values obtained averaging over 5 independent runs using 12 cores.

As has been done for the drag minimization test case, we simplify the parametrization by setting all inner curvature radii to 0.4. Additionally, we fix the chord length to $c = 1$ for the chord Reynolds number $\text{Re} = U_\infty c / \nu$ to remain constant over the course of optimization (which we believe is necessary to meaningfully compare the performances). This leaves n_p independent design variables, two end curvature radii $\rho_{j \in \{0, n_p - 1\}}$ and $n_p - 2$ inner thicknesses $e_{k \in \{1, \dots, n_p - 2\}}$. The network action output consists accordingly of values $(\hat{\rho}_j, \hat{e}_k)$ in $[-1; 1]^{n_p}$, converted into the actual physical quantities using the same mapping (6.6), only we set $\delta e = 0.03$ to account for the smaller target CSA. All results in the following are for $n_p = 5$, for which DRL evolves five design parameters, two end curvature radii and three inner thicknesses.

6.4.2 Laminar regime at $\text{Re} = 250$

We consider first a 2-D laminar case at $\text{Re} = 250$ modeled after the Navier–Stokes equations, for which the dimensions of the computational domain provided in table 6.4 yield a blockage ratio of 2.5%. All mesh adaptations are performed under the constraint of a fixed total number of elements $n_{el} = 100000$. A total of 100 episodes has been run for this case, that yield the variety of shapes illustrated in figure 6.10, together with their respective iso-contours of (instantaneous) vorticity. The general picture is that all shapes exhibit an oscillating pattern of leading- and trailing-edge vortex shedding following the shedding of the initial leading-edge vortex. This stems from the interaction between the (lower) negative vorticity sheet, that separates at the leading edge and then rolls up into a large clockwise vortex, and the (upper) positive vorticity sheet, that remains attached to the windward side and rolls up counter-clockwise from the trailing edge (in average, this yields a massive separation originating at the leading edge and extending on the leeward side, all the way to the trailing edge; not shown here). The Strouhal number for vortex shedding frequency built from the windward width is $St = fc \sin \alpha / u_\infty \sim 0.13$ (regardless of the shape), which is identical to experimental measurements performed on a high-aspect ratio NACA 0012 airfoil under the same incidence at $\text{Re} = 100$ [235].

The moving reward in figure 6.10(a) increases almost monotonically and reaches a plateau after about 40 episodes. The optimal lift to drag ratio computed as the average over the 10 latest episodes is $1.24 \pm 1.0\%$, at which point the cross-sectional area is equal to its target value down to the fifth decimal place. We note that 40 episodes is actually the number of episodes needed for the end radii to converge, as the thickness distribution exhibits excellent convergence after as little as 20 episodes. Interestingly, the agent has generated a wing-like optimal shape representative of a high-lift configuration without any priori knowledge of aerodynamic concepts: the optimal features a rounded leading edge to help maintain a smooth airflow (with curvature radius $0.394 \pm 0.01\%$ close to maximum) and a sharp trailing edge to generate lift (with curvature radius 0.156 ± 0.02 close to minimum). The optimal

lift to drag ratio exceeds that of the equivalent ellipse (*i.e.* of major diameter c and minor diameter $2S_{ref}/\pi c$, for the area to be equal to S_{ref}) by 6% and that of a NACA 0012 airfoil by 1%, as has been estimated from dedicated in-house calculations. This is small but consistent with the overall lack of sensitivity, as the objective function in figure 6.10(c) actually remains within 3% of the optimal over the course of optimization, as indicated by the grey shade delimiting the related variance interval.

6.4.3 Turbulent (transitional) regime at $\text{Re} = 5000$

We consider now a case at $\text{Re} = 5000$ corresponding to the ultra-low Reynolds number regime, that has assumed greater significance in the last few decades due to relevance for micro air vehicles and micro-turbines [236, 237]. We believe this constitutes a valuable first step towards applying the method to more prototypal aerodynamic applications in which airfoils operate at Reynolds numbers of $\sim 10^6$ and exhibit some degree of stochastic dynamics (as they carry turbulent energy distributed over a wide range of scales with varying degrees of spatial and temporal coherence), which might lead to high variance gradient estimates and hamper learning. Here, at the high value of angle of attack considered, the flow is expected to be transitional, for instance, transition in the wake of a NACA 0012 has been shown to occur in the separated shear-layer, shortly after the leading edge, at a location strongly dependent on the level of external noise [238]. This has been confirmed vetting preliminary Navier–Stokes simulations for which the built-in small-scale component of the VMS solution acts as an implicit large eddy simulation. While the solutions (not reported here for the sake of conciseness) are dominated by the large-scale component, with small-scale turbulence noticeably absent downstream, intermittent small-scale fluctuations develop on the leeward side, that prompt asymmetric vortex street (at least is the trailing edge is not too sharp for the separation point to be free to move) with vortices convected downstream along an axis inclined upward with respect to the streamwise direction, similar to the behavior observed in 2-D LES simulations of the transitional flow past a circular cylinder [239].

Accordingly, the case is modeled here after the uRANS equations, using negative Spalart–Allmaras as turbulence model. Such an approach is not without shortcomings (namely RANS is inherently designed to damp out the small-scales, and Spalart–Allmaras assumes fully turbulent behavior), but given the cost of accurately resolving the complex, unsteady vortex interaction described above, we believe the deficiencies are more than offset by the tremendous gain in computational efficiency derived from the relatively coarse meshes necessary to predict the most important large scale features of the flow. In practice, a scaled-down computational domain is used, whose dimensions reported in table 6.4 yield a blockage ratio of 3.5%. All mesh adaptations are performed under the constraint of a fixed total number of

elements $n_{el} = 120000$. A total of 100 episodes has been run, for which the selected iso-contours of vorticity documented in figure 6.11 are reminiscent of their laminar counterparts, with in-line vortex shedding (since the effect of the intermittent small-scale fluctuations has been lumped into the eddy viscosity model) and robust shedding frequency $S_t = 0.15$.

The moving average reward in figure 6.11(a) is seen to converge within about 50 episodes but the thickness distribution again converges faster (within roughly 40 episodes). As was already the case at $Re = 250$, the optimal resembles the airfoil of an airplane wing, with a rounded leading edge and a sharp trailing edge. The end radii are nearly identical to their laminar counterparts, but the shape is streamlined differently, namely it is a tad thinner in the front ($0.0549 \pm 0.2\%$ at $Re = 5000$ vs. $0.0638 \pm 0.2\%$ at $Re = 250$) but slightly thicker in the center ($0.0627 \pm 0.4\%$ at $Re = 5000$ vs. $0.0514 \pm 0.25\%$ at $Re = 250$). The optimal lift to drag ratio ($1.54 \pm 0.3\%$) exceeds that of the equivalent ellipse by 13% but is ultimately identical to that of a NACA 0012, despite the objective function exhibiting substantial variations in figure 6.11(b). The inability to outperform a conventional airfoil should not be interpreted as failure of the method, though, as aerodynamic shape design classically requires fine-tuning of the local geometry for a gain that often adds up to a few percent. This is not manageable here because the low number of degrees of freedom inevitably constrains the underlying space of shapes, and the expected gain is comparable to the typical convergence threshold of a DRL run. We believe the results should rather be considered proof that DRL can start from the ground up and generate shapes that perform just as well as a conventional airfoil. Actually, there is ample room for improvement if the optimization is to be tailored to airfoil shape optimization problems (which it is not here for the sake of generality), one may seek for instance to locally refine the DRL optimal by repeating the same analysis, but clustering the control points in specific regions of interest (*e.g.* the leading-edge, or the rear-end of the leeward side), or to rely on alternative parametrizations better suited to airfoils, such as CST [240].

6.5 Extension to 3-D shape optimization.

The ultra low-Reynolds number case at $Re = 5000$ is extended here to 3-D to assess the extent to which the approach carries over to three-dimensional shape optimization. All shapes generated over the course of optimization are unswept, rectangular wings, whose cross-section is set up from the DRL outputs following the exact same process as in sections 6.3 and 6.4. The span aspect ratio (relative to the chord length) is set to 3 in our implementation. A Cartesian coordinate system is used with origin in the mid-span plane, at quarter chord length from the leading edge. The number of control points remains set to $n_p = 5$, but we force the

leading and trailing edge curvature radii to 0.3 (round edge) and 0.1 (sharp edge) to keep the computational cost manageable, which leaves $n_p - 2 = 3$ independent design variables corresponding to the inner thicknesses. In practice, only a half-span wing body is simulated with symmetry boundary condition prescribed at the mid-span. The computational domain shown in figure 6.12 is a rectangular prism, whose dimensions reported in table 6.4 yield a blockage ratio of 5%. All mesh adaptations are performed under the constraint of a fixed total number of elements $n_{el} = 500000$. This is likely insufficient to claim true numerical accuracy, but given the numerical cost (960 3-D simulations total, each of which is performed on 12 cores and lasts about 10h, hence 9600h of total CPU cost), we believe this is a reasonable compromise to assess feasibility while producing qualitative results to build on.

A total of 80 episodes has been run for this case, using a slightly lower weighing coefficient $\beta = 90$ (to take into account that the coarser mesh yields a small loss in accuracy in the computation of the cross-sectional area). Several representative flow patterns computed over the course of optimization are illustrated in figure 6.13 to display the increased degree of complexity due to transverse inhomogeneities. All solutions exhibit vortex shedding, which is because the span aspect ratio is large enough for the tip vortex to remain relatively steady. Conversely, preliminary simulations carried out at lower aspect ratios of order 1 systematically relaxed to steady-state, due to the strong tip-vortex induced downwash over the entire span (the same behavior has been reported in laminar flows at Reynolds numbers of about in the range of a few hundreds [241], and is ascribed here to the RANS damping of the small-scale transverse motion, that should otherwise strengthen the unsteadiness). The moving averager reward in figure 6.13 plateaus after about 35 episodes. The 3-D distribution is almost front-rear symmetric but the shape itself surprisingly slightly thinner in the front than in the rear, although the rear is ultimately more streamlined due to the smaller trailing edge curvature radius. Compared to its 2-D counterpart, the 3-D optimal is thinner in the front and in the center, but much thicker in the rear. The optimal lift to drag ratio ($1.34 \pm 0.5\%$) exceeds that of the equivalent ellipse by 5% and is identical to that of a NACA 0012. This is consistent with the above findings, in the sense that the DRL optimal performs at the level of a conventional airfoil, and that the limited improvement with respect to the equivalent ellipse should not be taken as an indictment of the method, just a consequence of the flow regime considered (precisely because a similar improvement is achieved using a NACA 0012).

6.6 Conclusion

Shape optimization in computational fluid dynamics systems is achieved here training fully connected networks with PBO, a recently introduced deep reinforcement al-

gorithm at the crossroad of policy gradient methods and evolution strategies. PBO is single-step, meaning that the DRL agent gets only one attempt per learning episode at finding the optimal. The numerical reward fed to the PBO agent is computed with a finite elements CFD environment solving stabilized weak forms of the governing equations (Navier–Stokes, otherwise uRANS with negative Spalart–Allmaras as turbulence model) with a combination of variational multiscale approach, immersed volume method and anisotropic mesh adaptation.

Several cases are documented, for which shapes with fixed camber line, angle of attack and cross-sectional area are generated by varying a chord length and a symmetric thickness distribution (and possibly extruding in the off-body direction), connecting consecutive points by a cubic Bézier curve using local position and curvature information. The classical problem of finding the 2-D shape of minimum drag in a uniform flow is revisited first to validate and assess the method capabilities. The method is also applied to the more practically meaningful problem of finding the shape of maximum lift to drag ratio (in 2-D or 3-D) at an incidence of 30° and under constant chord Reynolds number. The DRL optimal increases the performance the equivalent ellipse (*i.e.* the ellipse of same cross-sectional area) by 13% in 2-D and 5% in 3-D. It is systematically found to perform just as well as a conventional airfoil, despite DRL starting from the ground up and having no priori knowledge of aerodynamic concepts. Exhaustive convergence and efficiency data are reported here with the hope to foster future comparisons, but it is worth emphasizing that we did not seek to optimize said efficiency, neither by optimizing the PBO meta-parameters, nor by using pre-trained deep learning models (as is done in transfer learning).

Fluid dynamicists have just begun to gauge the relevance of DRL and its application to optimal shape design. This research weighs in on this issue and shows that the proposed single-step method holds a high potential as a reliable, go-to black-box optimizer for complex CFD problems. Moreover, the optimization process is entirely domain-agnostic, meaning that the proposed framework allows for easy application to any domain in which shape optimization may be beneficial. We believe further work should now focus on the challenges specific to fluid mechanics that still prevent DRL capabilities from meeting the requirements for practical deployment, *e.g.* stochasticity, sampling efficiency (CFD environments are resource expensive as they routinely involve numerical simulations with tens or hundreds of millions of degrees of freedom, while classical RL methods have low sample efficiency, *i.e.* many trials are required for the agent to learn a purposive behavior), the need to leverage experience from multiple agents learning concurrently (multi-agent DRL) or to train an agent in reasoning about several weighted objectives (multi-objective reward).

Appendix

Shape generation using Bézier curves

This section describes the process followed to generate shapes from a set of n_p control points. Once the position has been reconstructed from the agent outputs, the angles between consecutive points are computed. An average angle is then computed around each point (see Fig. 6.14(b)) as

$$\theta_i^* = r\theta_{i-1,i} + (1-r)\theta_{i,i+1}, \quad (6.9)$$

where $r \in [0; 1]$ is the curvature radius that control the local sharpness of the curve. Then, each pair of points is joined using a cubic Bézier curve, defined by four points: the first and last points, p_i and p_{i+1} belong to the curve, while the second and third ones, p_i^* and p_{i+1}^* , are supplemental control points that define the tangent of the curve at p_i and p_{i+1} . The tangents at p_i and p_{i+1} are respectively controlled by θ_i and θ_{i+1} (Fig. 6.14(b)). A final sampling of the successive Bézier curves leads to a boundary description of the shape (Fig. 6.14(c)). Using this method, a wide variety of shapes can be attained.

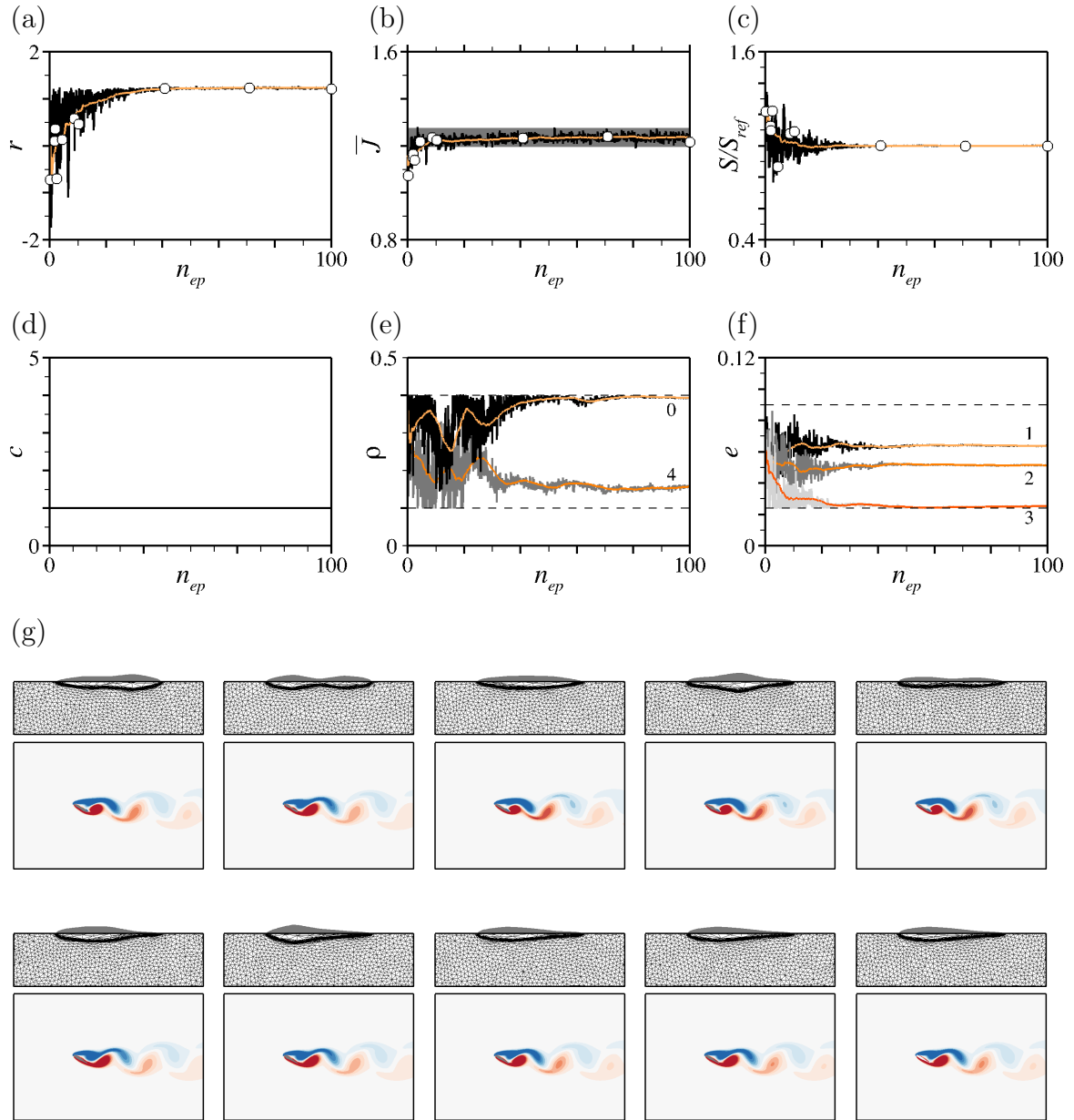


Figure 6.10: Maximum lift to drag ratio test case at $Re = 250$ under constant area constraint $S_{ref} = 0.0822$. (a) Evolution per episode of the instant (black line) and moving average (over episodes, light orange line) reward. (b-f) Same as (a) for the (b) averaged (over time) lift to drag ratio, (c) ratio of the actual to target cross-sectional areas, (d) chord (fixed), (e) edge curvature radii and (f) inner thicknesses. The grey shade in (b) marks the 3% variance interval with respect to the average over the 10 latest learning episodes. All labels in (e-f) are ordered clockwise from the leading edge. The horizontal dashed lines in (e-f) mark the admissible values. (g) Shapes generated over the course of optimization for random episodes marked by the circle symbols in (a-c), together with corresponding iso-contours of vorticity. The last three shapes pertain respectively to episodes 40, 70 and 100.

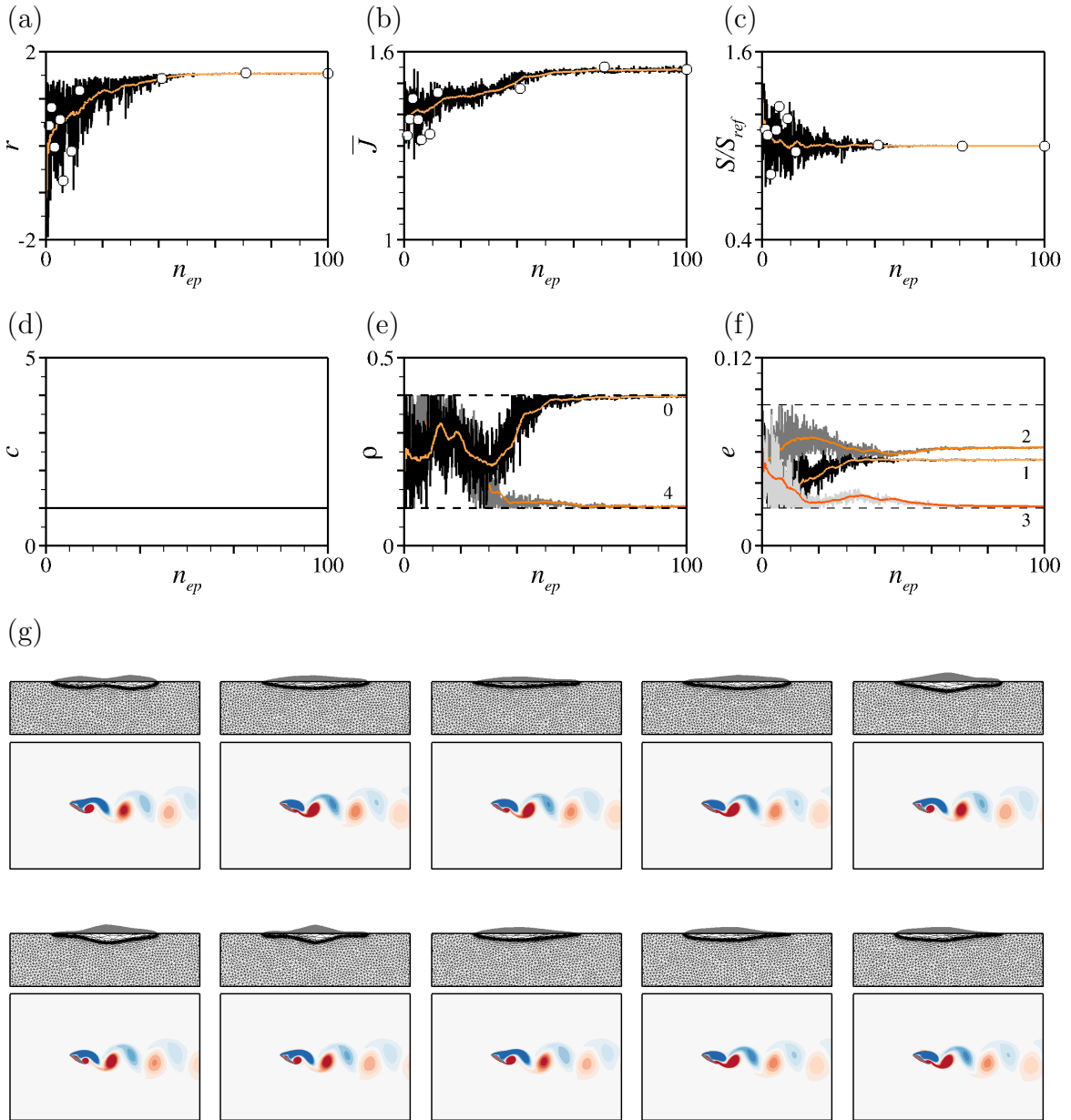


Figure 6.11: Maximum lift to drag ratio test case at $Re = 5000$ with negative Spalart–Allmaras turbulence model, under constant area constraint $S_{ref} = 0.0822$. (a) Evolution per episode of the instant (black line) and moving average (over episodes, light orange line) reward. (b–f) Same as (a) for the (b) averaged (over time) lift to drag ratio, (c) ratio of the actual to target cross-sectional areas, (d) chord (fixed), (e) edge curvature radii and (f) inner thicknesses. All labels in (e–f) are ordered clockwise from the leading edge. The horizontal dashed lines in (d–f) mark the admissible values. (g) Shapes generated over the course of optimization for random episodes marked by the circle symbols in (a–c), together with corresponding iso-contours of vorticity. The last three shapes pertain respectively to episodes 40, 70 and 100.

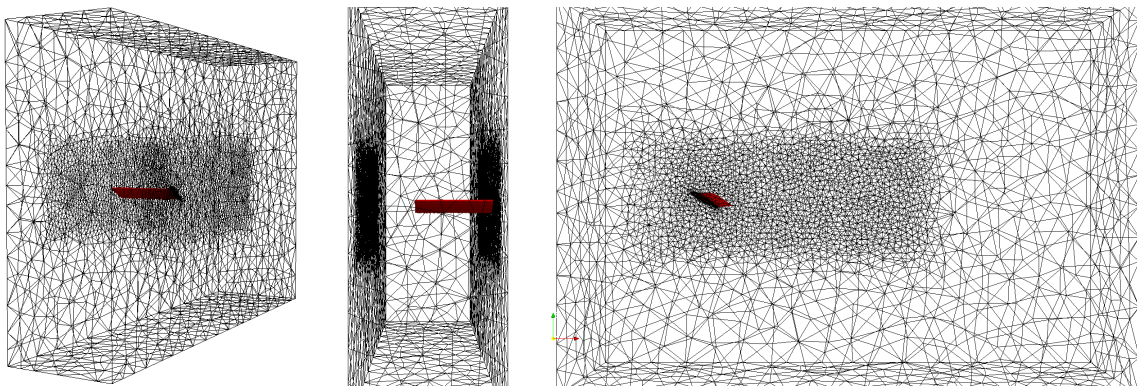


Figure 6.12: Anisotropic adapted mesh around an immersed three-dimensional unswept, rectangular wing.

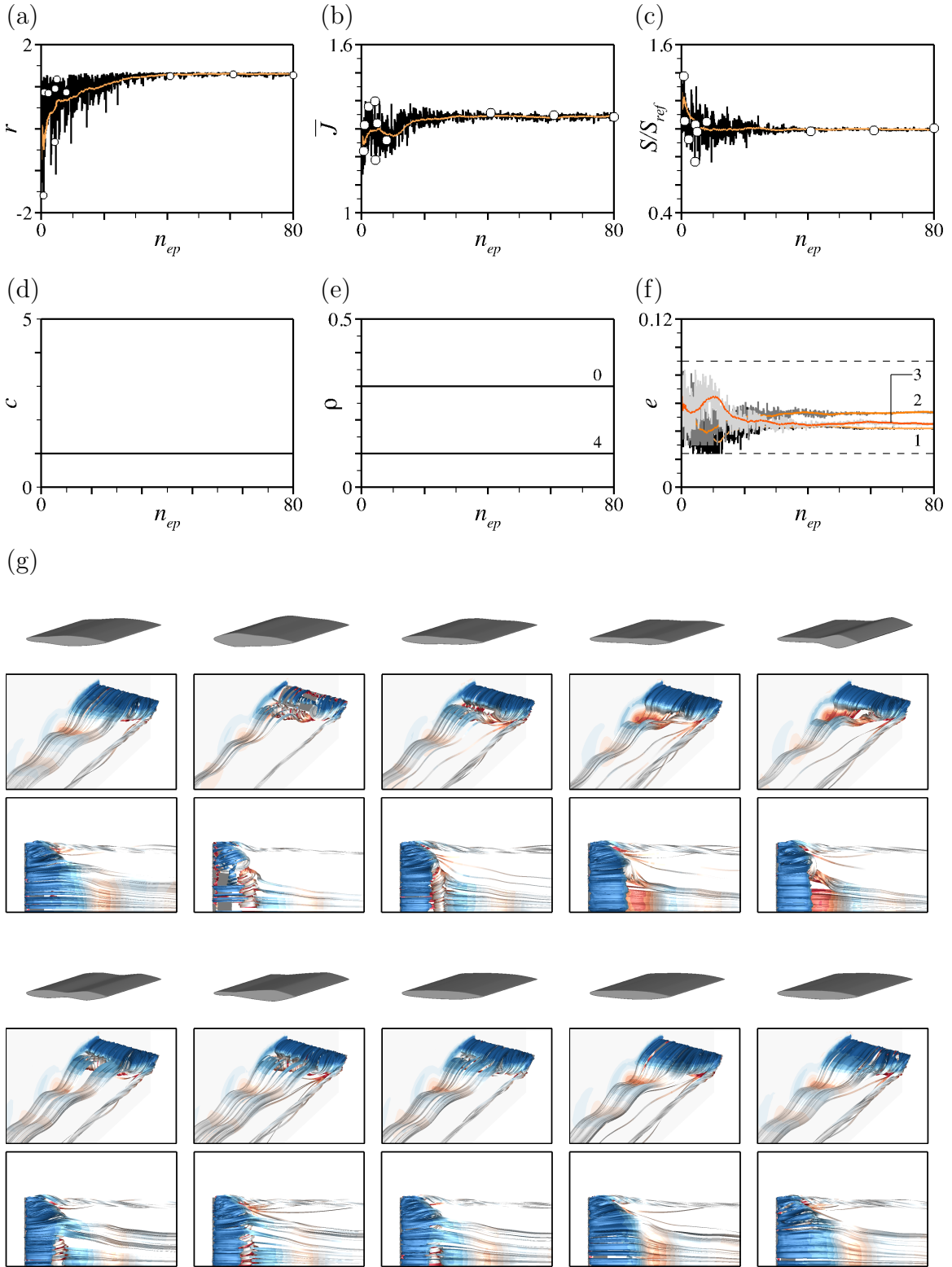


Figure 6.13: Maximum lift to drag ratio test case in 3-D at $Re = 5000$ with negative Spalart–Allmaras turbulence model, under constant area constraint $S_{ref} = 0.0822$. (a) Evolution per episode of the instant (black line) and moving average (over episodes, light orange line) reward. (b-f) Same as (a) for the (b) averaged (over time) lift to drag ratio, (c) ratio of the actual to target cross-sectional areas, (d) chord (fixed over the course of optimization), (e) edge curvature radii (also fixed) and (f) inner thicknesses. (g) Shapes generated over the course of optimization for random episodes marked by the circle symbols in (a-c), together with corresponding iso-contours of vorticity. The last three shapes pertain respectively to episodes 40, 70 and 100.

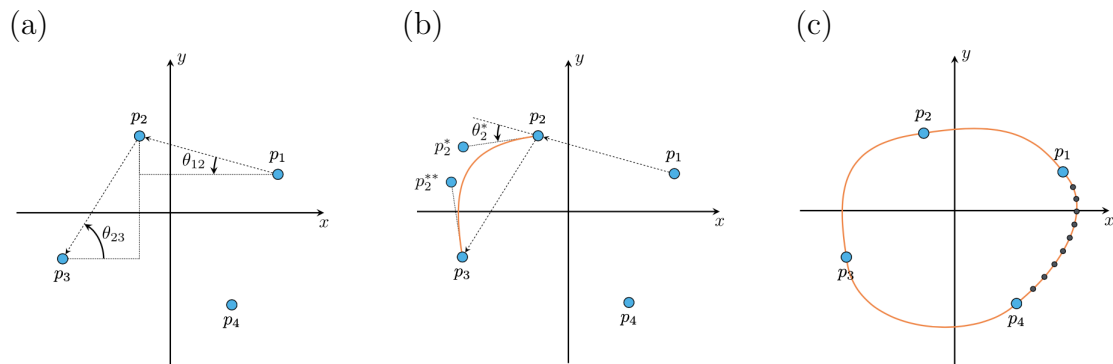


Figure 6.14: Shape generation using cubic Bézier curves. Each subfigure illustrates one of the consecutive steps used in the process. (a) Compute angles between points and compute an average angle θ_i^* around each point. (b) Compute supplemental control points coordinates from averaged angles and generate cubic Bézier curve. (d) Sample all Bézier lines and export for mesh immersion.

Chapter 7

Multi-step Deep Reinforcement Learning

Contents

7.1	Introduction	183
7.2	Conjugate heat transfer control for homogeneous cooling	184
7.2.1	Uncontrolled flow	187
7.2.2	Open-loop control using single-step DRL	188
7.2.3	Closed-loop control using multistep-step DRL	190
7.3	Drag reduction in the flow past a circular cylinder	194
7.3.1	Uncontrolled flow	195
7.3.2	Open-loop control using single-step DRL	195
7.3.3	Closed-loop control using multi-step DRL	198
7.4	Conclusion	202

Ce chapitre évalue la capacité des méthodes classiques de DRL multi-étapes à réaliser des problèmes de contrôle en boucle fermée, où l'agent optimise une relation état-action complète pour laquelle l'entrée de contrôle dépend de certaines mesures de champs d'écoulement. La récompense est calculée à l'aide d'un environnement interne d'éléments finis stabilisés combinant la modélisation variationnelle multi-échelle (VMS) des équations gouvernantes, la méthode des volumes immergés et l'adaptation du maillage anisotrope multi-composant. Deux problèmes ont été abordés en utilisant l'algorithme proximal policy optimization PPO, un premier problème de transfert de chaleur conjugué en régime permanent dont l'optimum est en boucle ouverte, et un second problème dépendant du temps dont l'optimum est en boucle fermée. Des résultats prometteurs ont également été obtenus sur le second problème de réduction active de la traînée autour d'un cylindre circulaire où une réduction de la traînée de 8% a été obtenue.

This chapter assesses the capability of classical multi-step DRL methods to perform closed-loop control problems, where the agent optimizes a complete state-action relation for which the control input depends on some flow fields measurements. The numerical reward is computed using an in-house stabilized finite elements environment combining variational multi-scale (VMS) modeling of the governing equations, immersed volume method, and multi-component anisotropic mesh adaptation. Two problems have been tackled using the proximal policy optimization PPO algorithm, a first steady-state conjugate heat transfer problem whose optimal is open-loop, and a second time-dependent problem whose optimal is closed-loop. Promising results have been also obtained on the second problem of active drag reduction around a circular cylinder where a drag reduction of 8% has been achieved.

7.1 Introduction

So far, the presented applications have focused on optimization and open-loop control problems, for which the policy to be learnt by the DRL agent does not depend on state. Such problems have been tackled using in-house, tweaked DRL algorithms with single-step learning episodes (i.e., the agent gets one attempt per episode at finding the optimal), which in turn has allowed the use of extremely small networks (basically, 2 hidden layers with 2 up to 4 neurons per layer), as the agent is then not required to learn a complex state-action relation, but only a transformation from a constant input state to a given action. In this chapter, we conversely assess the feasibility to use DRL to perform closed-loop control problems, for which the control input depends on some appropriate measure of the flow. This amounts to say that the policy to be learnt by the DRL agent does depend on state, which adds several layers of complexity to the problem:

- In fluid flows, classical states are easily prohibitively large for policy learning, as tens or hundreds of millions of degrees of freedom are routinely involved in numerical simulations. This is not troublesome in single-step DRL where state is prescribed, meaning that the agent can operate with perfect state knowledge as a result of a single numerical simulation. Otherwise, the agent must learn under partially observable environments, meaning that the performance of a multi-step DRL algorithm can highly depend on the quality and relevance of the data available for observation (in practice, the number and position of velocity and/or pressure sensors positioned in the flow to provide feedback observations). The very same issue is also encountered in data-driven model reduction techniques for large scale dynamical systems, that usually require using measures of observability as an information quality metric [242].
- A multi-step learning episode consists of a series of actions drawn by the DRL agent under the current policy. If we denote by Δt_{act} the action time step between the sampling of two consecutive actions (so the agent provides new actions to the environment with a frequency $1/\Delta t_{act}$), by Δt the time step used in each environment to perform the numerical simulations, and by Δt_{phys} a characteristic time scale of the physical process to be controlled (for instance a vortex shedding period in the context of wakes), then it can be inferred that $\Delta t < \Delta t_{act} < \Delta t_{phys}$ for the agent to be able to observe the effects of its actions on the environment ($\Delta t < \Delta t_{act}$), and for the actions to be able to significantly alter the flow dynamics ($\Delta t_{act} < \Delta t_{phys}$). The user has entire control to adjust the action time step within these two bounds, but because a sufficient number of actions must be sampled over a given episode, one must either perform very long learning episodes, or put a cap on the simulation time step regardless

of numerical stability considerations, which in both cases can considerably increase the CPU cost (at least with respect to the single-step approach). In most contributions of the literature, the ratio of the physical time scale to the action time step is about a few tens (so a few tens of actions are sampled over a characteristic time interval) but the ratio of the action to the simulation time-steps varies considerably from a few tens to a few hundreds [42, 49, 224].

- One must take into account the existence of delays that can hinder the accuracy of the reward estimation and prevent learning. This is because each action generates a perturbation, and there is always a certain time between the moment an action is applied, and the moment it efficiently reaches the current state (depending on the ability of the said perturbation to grow and saturate).
- Computing relevant estimates of the policy loss gradient (for relevant network update) is no small task because the effects of policy changes on the state distribution are unknown (since modifying the policy will most likely modify the set of visited states, which will in turn affect the performance in some indefinite manner).

The following sections present two test cases in two dimensions, used as testbed for implementing and validating the multi-step approach. In order to echo the scientific approach followed in the previous chapters, the first one is a steady-state conjugate heat transfer problem whose optimal is time-independent, for which we validate the implementation by comparison with single-step DRL. The second one is the drag reduction problem proposed by Rabault *et al.* [39], and is intended to serve as demonstrator of the high potential of the approach in the context of active-control of time-dependent flow control. Both problems are tackled with the go-to Proximal Policy Optimization (PPO) algorithm, using the native Stable Baselines implementation and a custom OpenAI environment designed using the Gym library.

7.2 Conjugate heat transfer control for homogeneous cooling

The first test case is that of a two-dimensional hot workpiece under impingement cooling, designed after those considered in Chapter 5. Without anticipating on the results, this is a steady-state conjugate forced convection problem, whose solutions (both uncontrolled and controlled), ultimately settle down to a time-independent regime. It is not per se in the scope of multi-step DRL (as the optimal control also is expected to be time-independent, that is, open-loop, while multi-step DRL seeks optimal time-dependent state-to-action relations, and is thus best applied to control

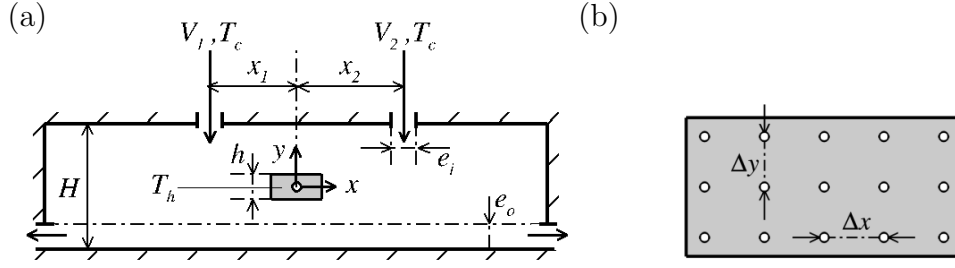


Figure 7.1: (a) Schematic of the two-dimensional forced convection set-up. (b) Probe array in the solid domain.

H	h	e_i	T_w	T_c	T_h	μ	ρ	λ	c_p	
1	0.2	0.2	10	10	150	0.001	1	0.5	1000	Fluid
						1000	100	15	300	Solid

Table 7.1: Simulation parameters. All values in SI units, with the exception of temperatures given in Celsius.

time-dependent conjugate heat transfer problems [17]), but it does carry value for validation purposes as it allows comparing to dedicated single-step results.

The set-up sketched in figure 7.1 features a similar rectangular solid with height h and aspect ratio of 2 : 1, initially at the (hot) temperature T_h . A Cartesian coordinate system is used with origin at the center of mass of the solid. The latter is fixed and at the center of a rectangular cavity of height H and aspect ratio of 4 : 1, with isothermal walls at temperature T_w . The top cavity side is flush with two identical holes of width e for two injectors to blow cold air at possibly different velocities V_1 and V_2 , but same (cold) temperature T_c . The horizontal position of the injector centers is set to $x_1 = -1.1$ and $x_2 = 0.9$, respectively, slightly asymmetrical with respect to the centerline of the domain in the x direction. Hot air is released through the cavity sidewalls by two identical exhaust areas of height e_0 , whose center is at the vertical position $(e_0 - H)/2$.

The governing equations are the coupled Navier–Stokes and heat equations. Those are solved with zero buoyancy and radiative heat transfer (so the temperature ultimately behaves as a passive scalar) and no-slip isothermal conditions ($\mathbf{u} = \mathbf{0}$ and $T = T_w$), except at the injection exit planes ($\mathbf{u} = -V_{1,2}\mathbf{e}_y$ and $T = T_c$) and at the exhaust areas (zero pressure and velocity/temperature gradients). No thermal condition is imposed at the interface for heat exchange arise from the difference in the individual material properties provided in Table 7.1, that yield a Prandtl number $\text{Pr} = 2$ and a Reynolds number $\text{Re}_{1,2} = 200V_{1,2}$.

In the context of flow control, the quantities being optimized are the injector velocities V_1 and V_2 in the attempt to achieve both efficient and homogeneous cooling (i.e. this is a multi-objective test case, unlike those in chapter 5 that focused solely on homogeneous cooling, then proceeded to assess efficiency a posteriori). To this end, we distribute uniformly 15 probes in the solid domain, into $n_x = 5$ columns and $n_y = 3$ rows with resolutions $\Delta x = 0.09$ and $\Delta y = 0.075$. We then compute an efficiency estimator as the normalized, temperature

$$r_T = \frac{1}{n_x n_y} \frac{1}{T_h} \sum_{i,j} T_{ij}, \quad (7.1)$$

averaged over all probes, where we use subscripts i , j and ij to 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. Similarly, a homogeneity indicator is computed as

$$r_{\nabla T} = \frac{1}{n_x + n_y} \sum_{i,j} \langle \|\nabla_{\parallel} T\| \rangle_i + \langle \|\nabla_{\parallel} T\| \rangle_j, \quad (7.2)$$

where $\langle \|\nabla_{\parallel} T\| \rangle_i$ (resp. $\langle \|\nabla_{\parallel} T\| \rangle_j$) is the norm of the temperature gradient averaged over the i -th column (resp. the j -th row), defined as

$$\langle \|\nabla_{\parallel} T\| \rangle_i = \frac{2}{n_y - 1} \frac{1}{T_h} \left| \sum_{j \neq 0} \text{sgn}(j) \|\nabla T\|_{ij} \right|, \quad (7.3)$$

$$\langle \|\nabla_{\parallel} T\| \rangle_j = \frac{2}{n_x - 1} \frac{1}{T_h} \left| \sum_{i \neq 0} \text{sgn}(i) \|\nabla T\|_{ij} \right|. \quad (7.4)$$

In this regards, the asymmetry in the injectors distribution generates an artificial inhomogeneity that the agent must account for by relevantly selecting asymmetrical blowing velocities. Finally, the numerical reward fed to the DRL agent is

$$r = -w_1 r_T - w_2 r_{\nabla T}, \quad (7.5)$$

where $w_1 = 0.2$ and $w_2 = 4$ are weights set empirically for both components of the reward to be of the same order of magnitude.

In practice, all solutions are computed by marching in time the same initial guess (consisting of zero velocity and uniform temperature, except at the hot sidewall) with time step $\Delta = 0.1$, and the mesh is updated every 5 time steps under the constraint of a fixed number of elements $n_{el} = 20000$ using the level set, velocity, and temperature fields as multiple-component criterion. The temperature fields at the final simulation time $t_f = 200$ is used to compute the rewards (7.1-7.2). As will be seen in the following, the latter has not been chosen so as to yield accurate convergence to steady state, only to mimic a real cooling device (whose operating time is often used-defined) while putting a cap on the CPU cost.

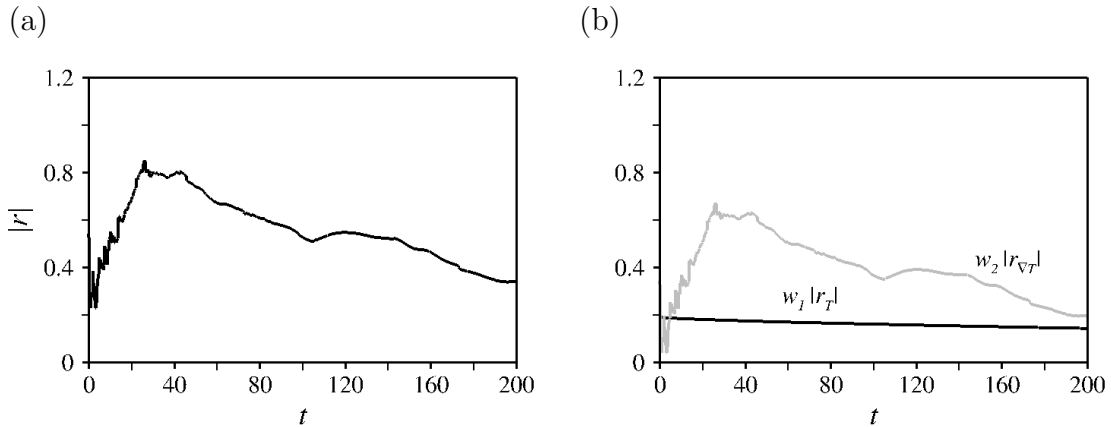


Figure 7.2: (a) Time evolution of the uncontrolled numerical reward with injectors velocity set to $V_1 = V_2 = 1$. (b) Same as (a) for the temperature (in black) and homogeneity (in gray) components of the reward.

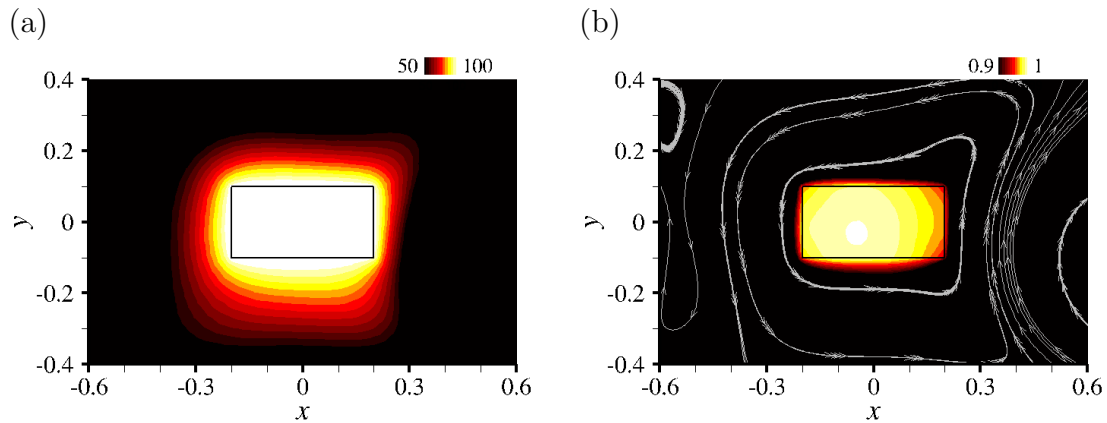


Figure 7.3: (a) Iso-contours of baseline temperature at the final time $t = 200$. (b) Same as (a) for the solid temperature normalized to $\max_{x,y} T = 1$, with superimposed streamlines of the underlying velocity field.

7.2.1 Uncontrolled flow

We characterize first the uncontrolled flow for which the injectors velocity is set to $V_1 = V_2 = 1$. The reward shown in figure 7.2(a) is seen to decrease over the first 50 time units, then increases steadily to reach $|r| = 0.35$ at the end of the simulation (this is the baseline reward to which the control results shall be compared in the following). The homogeneity component $w_2|r_{\nabla T}|$ shown in figure 7.2(b) displays a similar behavior to reach 0.20, meaning that the cooling becomes more homogenous over time. Meanwhile, the (weighted) temperature component $w_1|r_T|$ decreases over time (which reflects the workpiece cooling) down to the value 0.143

Mean	Variance	Correlation	Neural network
5×10^{-3}	\gg	10^{-3}	Learning rate
128	8	\gg	Nb. epochs
1	8	16	Nb. learning episodes
1	4	8	Nb. mini-batches
[2,2,2]	\gg	\gg	Architecture (hidden layers)

Table 7.2: Details of the PBO meta-parameters and network architectures.

reported in figure 7.2(b), that corresponds to a solid temperature 107° . The iso-contours of temperature shown in figure 7.3 shows that the baseline configuration achieves asymmetrical vertical cooling, owing to the formation of two large-scale, small velocity end vortices entraining heat laterally downwards. It also achieves asymmetrical horizontal cooling, as the strength of both vortices is different due to the asymmetrical injectors position with respect to the centerline of the domain.

7.2.2 Open-loop control using single-step DRL

For validation purposes, we report first reference results pertaining to the single-step control problem, for which we use the PBO algorithm; see chapter 2. We recall that PBO learns the mean, variance and correlation parameters of a multivariate normal search distribution from three separate neural networks, whose meta-parameters (number of epochs and of mini-batches, learning rate, history of learning episode for off-policy update) are provided in table 7.2.

A total of 150 episodes has been run with 7 parallel environments, each of which computes 200 time units with time step $\Delta t = 0.1$ to march the initial condition to steady state. This represents 1050 simulations, each of which lasts 12 mn using 12 cores, hence 30h of CPU cost. The network action output consists of two values $\xi_{j=1,2}$ in $[-1; 1]^2$, mapped into

$$V_j = \frac{1 - \hat{\xi}_j}{2} V_{min} + \frac{1 + \hat{\xi}_j}{2} V_{max}, \quad (7.6)$$

for each blowing velocity to vary in $[V_{min}; V_{max}]$ with $V_{min} = 0.5$ and $V_{max} = 1.5$ (hence, the Reynolds number in the exit plane of an injector is between 100 and 300). In practice, the blowing velocities remain set to their uncontrolled values $V_1 = V_2 = 1$ over the first 50 time units, then ramped up to the DRL value over 3 time units.

We show in figure 7.4(a) the evolution of the controlled reward, whose moving average (computed over the 50 latest episodes, equivalently the 50 latest values, or the

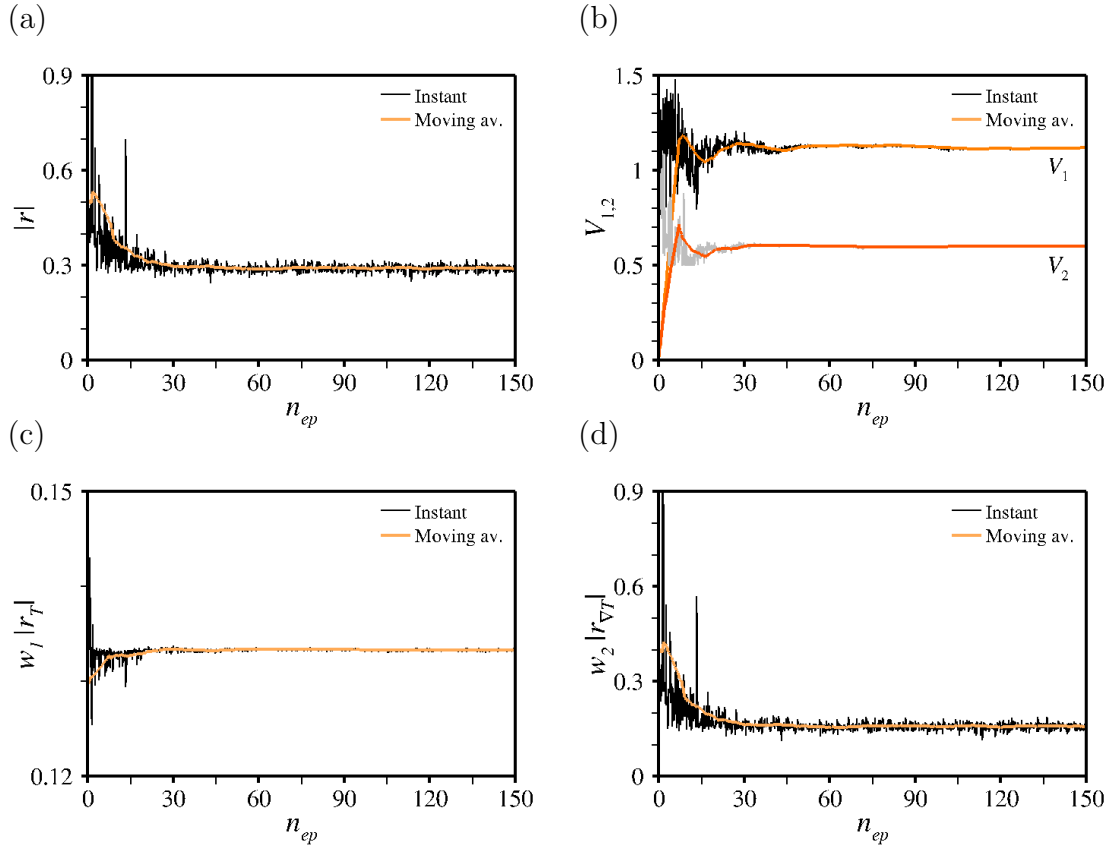


Figure 7.4: Single-step DRL control using PBO. (a) Evolution per episode for the instant (black line) and moving average (over episodes, light orange line) reward. (b-d) Same as (a) for the injector velocities, the (c) temperature and (d) the homogeneity components of the reward.

whole sample if it has insufficient size) increases monotonically and reaches a plateau after about 30 episodes, corresponding to $|r^*| = 0.29$, at which point the control velocities in figure 7.4(b) are $V_1^* = 0.60$ and $V_2^* = 1.12$. The same trends carry over to the homogeneity component of the reward, that reaches $w_2|r_{\nabla T}^*| = 0.176$ in 7.4(c). The temperature reward in figure 7.4(d) exhibits very limited variations, which may suggest at first sight that the control primarily act by improving homogeneity but fails at improving the efficiency. Nonetheless, the optimal value $w_1|r_T^*| = 0.133$ corresponds to a final temperature of 100° , which is still lower than the uncontrolled value by 6%. Another unplanned consequence of the control is best seen in figure 7.6 is that optimally controlled reward reaches its maximum much quicker, since all compound and individual rewards at $t = 150$ (that is, 100 time units after the control has been switched on) have improved over their uncontrolled values at $t = 200$. In short, the optimal control selected by PBO yields a more efficient and

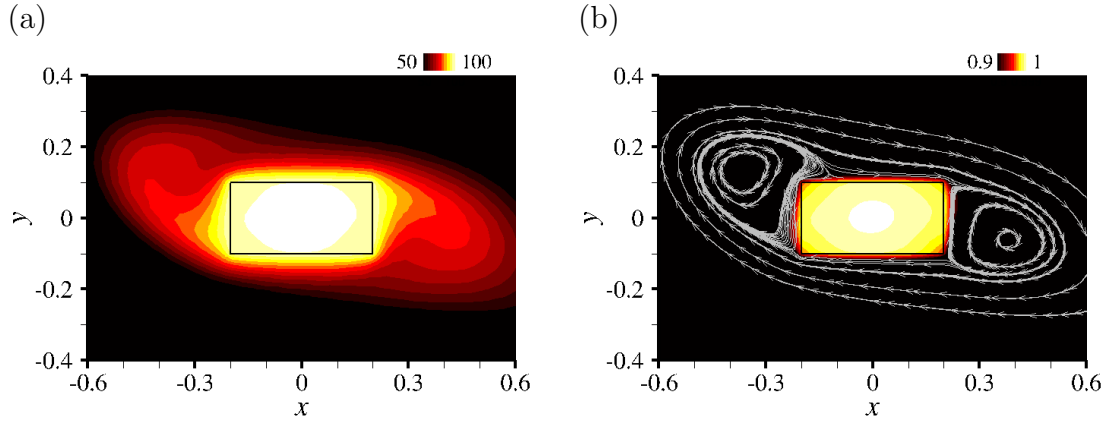


Figure 7.5: Same as figure 7.3 for the optimal single-step PBO temperature.

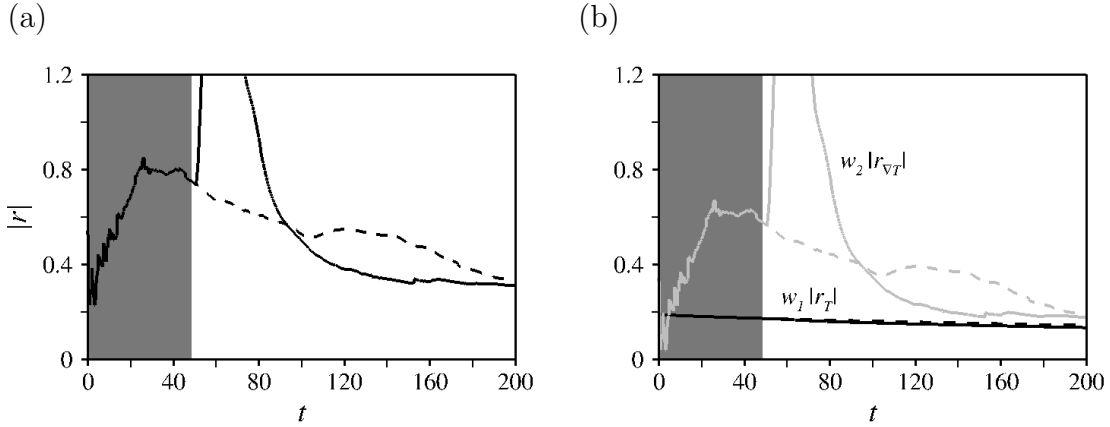


Figure 7.6: (a) Time evolution of the instantaneous reward over the course of an optimal PBO single-step learning episode. The controlled (resp. uncontrolled) values are shown as the solid (resp. dashed) lines. The dark gray shade marks the initial time interval during which control is forced to zero. (b) Same as (a) for the temperature and homogeneity components of the reward.

homogeneous cooling and reduces the time needed to do so by nearly 25%. As evidenced by the focus on the normalized temperature in figure 7.5, PBO has restored a near-perfect horizontal and vertical symmetry, as the workpiece is now at the core of a close recirculation that wraps almost perfectly around its surface, albeit itself asymmetric.

7.2.3 Closed-loop control using multistep-step DRL

The same problem is now revisited using the classical PPO algorithm to assess and validate the multi-step DRL/CFD implementation. The control strategy proceeds

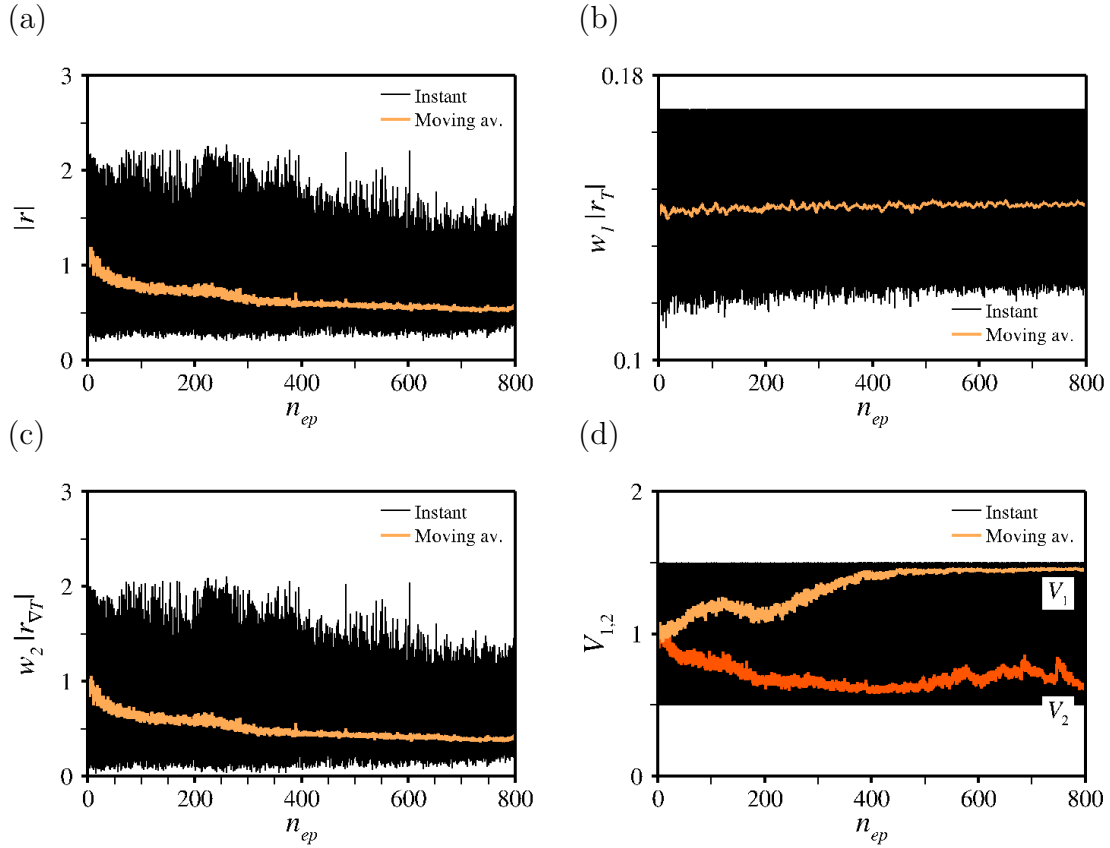


Figure 7.7: Multi-step DRL control using PPO. (a) Evolution per episode for the instant (black line) and moving average (over episodes, light orange line) reward. (b-d) Same as (a) for the injector velocities, the (c) temperature and (d) the homogeneity components of the reward.

here from multi-step learning episodes during which the network outputs a series of blowing velocities. The PPO agent is a fully connected network with two dense hidden layers, each with 128 neurons, with learning rate set to 2.5×10^{-4} and PPO loss clipping range to 0.2. The discount factor weighing the relative importance of present and future rewards is set to 0.99, meaning that the agent cares almost equally about all rewards along the current succession of states and actions. Temperature and temperature gradient information is extracted from the simulation and provided to the neural network from the 15 probes distributed in the solid (no velocity/pressure state is extracted, hence, the agent knows only about the thermal state, not the flow state). A total of 800 episodes has been run using 8 parallel environments. Each episode computes 200 time units with time step $\Delta t = 0.1$, with the first 50 time units simulated without any control, after which the network outputs a series of 50 actions over 150 time units, and the network is updated for 16

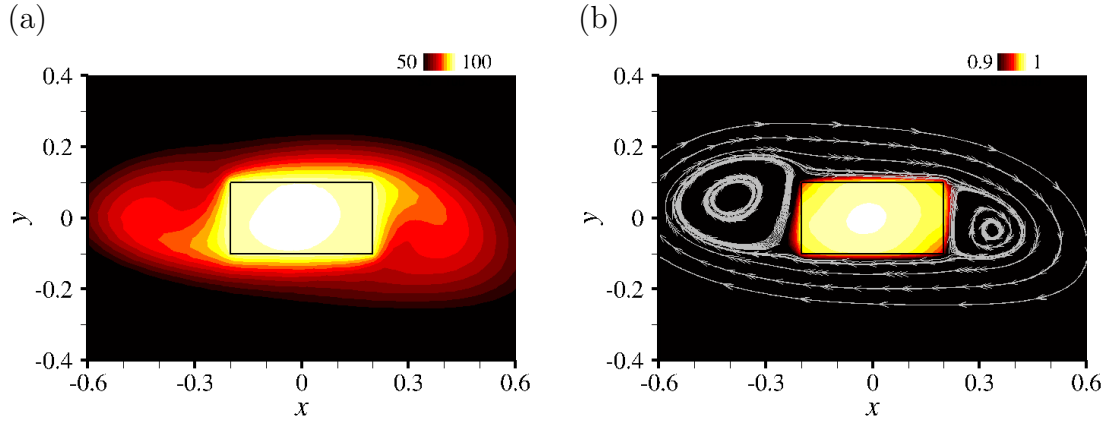


Figure 7.8: Same as figure 7.3 for the optimal multi-step PPO temperature.

epochs using 8 steps mini-batches. This yields an action time $\Delta t_{act} = 3$ and a ratio of the action to the simulation time-steps of 30 (i.e., each action is evaluated over 30 numerical time steps), very much in line with the values found in the literature.

For each action, the agent outputs two values $\xi_{j=1,2}$ in $[-1; 1]^2$, mapped into $[V_{min} = 0.5; V_{max} = 1.5]$ using (7.6). In order to avoid abrupt control changes in the environment (as this may cause numerical instabilities), the control is made continuous over time using

$$\tilde{V}_{j;k+1} = \tilde{V}_{j;k} + \alpha(V_j - \tilde{V}_{j;k}), \quad (7.7)$$

where $\tilde{V}_{j;k}$ is the effective blowing velocity for the j -th injector considered at the previous numerical time step, $\tilde{V}_{j;k+1}$ is the new control, and V_j is the target action set by the PPO agent for the current 30 time steps. Finally, α is a numerical parameter set to 0.3 for $\tilde{V}_{j;k}$ reaches $0.99V_{j;k}$ within 20 numerical time steps (equivalently 2 time units), before the network samples a new action. The numerical reward fed to the PPO agent is computed every Δt_{act} from the current temperature field, once a given action has been fully assessed.

The moving average reward in figure 7.7(a) computed over the 4 latest episodes (equivalently the 200 latest values) reaches a plateau after about 400 episodes, that correspond to $|r^*| = 0.53$, which is less than the uncontrolled value. This is true also of the homogeneity ($w_2|r_{\nabla T}^*| = 0.39$), and thermal component ($w_1|r_T^*| = 0.148$), which we believe is due to the presence of exploration noise, that makes the control less efficient than its deterministic counterpart. Meanwhile the optimal blowing velocities in figure 7.7(d) are $V_1^* = 0.70$ and $V_2^* = 1.45$, quite close to their PBO counterparts. Also, the time-evolution of the reward under optimal PPO actuation shown in figure 7.9 yields results reminiscent of PBO, with the main difference due to the homogeneity indicator, as the thermal indicators in figure 7.9(b) are nearly indistinguishable. At this stage, we believe these results can provide a first element of

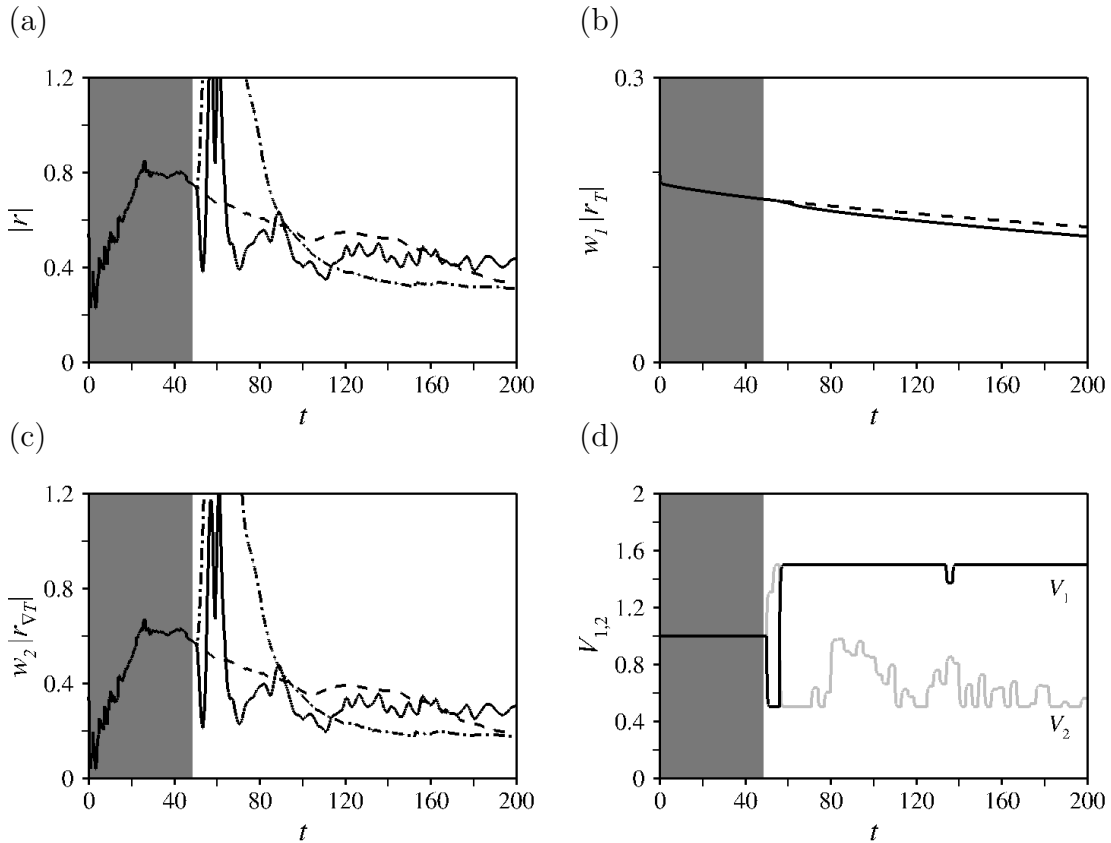


Figure 7.9: a) Time evolution of the instantaneous reward over the course of an optimal multi-step episode. The controlled (resp. uncontrolled) values are shown as the solid (resp. dashed) lines. The single-step PBO values reproduced from figure 7.6(a) are shown as the dash-dotted line. The dark gray shade marks the initial time interval during which control is forced to zero. (b-c) Same as (a) for the (b) temperature and (c) homogeneity reward components of the reward. (d) Same as (a) for the injector velocities.

validation of implemented PPO/CFD coupling, and that most discrepancies simply reflect the difficulty of learning a rigorously time-independent sequence of actions using an intrinsically time-dependent theoretical framework. Moreover, we expect closer results to be obtained by picking up the most likely action of the optimal policy from the mean of the multivariate normal search distribution (i.e., setting the variance to zero to remove the exploration noise, meaning that the neural network is used in full exploitation mode), which has not yet been undertaken due to a lack of time. Even though, it is worth noticing that the solid temperature shown in figure 7.8 is especially close to the single-step distributions, with maximum value of 101° , and improved horizontal and vertical symmetry. Also, as has been found

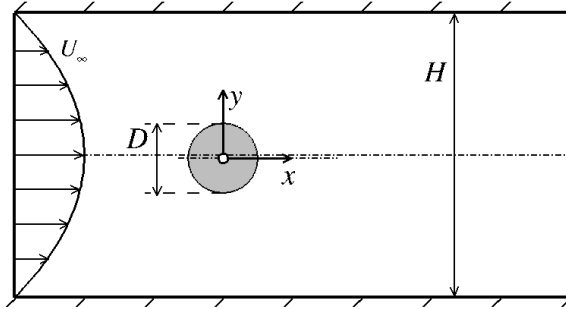


Figure 7.10: Schematic diagram of the two-dimensional drag reduction set-up.

using single-step DRL, the control yields a quicker cooling (with better efficiency and comparable homogeneity compared to the uncontrolled case) as the optimal reward converges much faster (by roughly 25%) to its maximum.

7.3 Drag reduction in the flow past a circular cylinder

The second test case is the two-dimensional drag reduction problem proposed by Rabault *et al.* [39] (that has subsequently been further analyzed in [40, 49]), that is best suited to check the implementation of multi-step DRL, due to the time-dependent, vortex-shedding behavior of all solutions. As sketched in figure 7.10, the case features a circular cylinder of diameter D confined between two parallel channel walls separated by a distance $H = 4.1D$, and is adapted from the CFD benchmark in [243]. The environment consists of CFD simulations in a Cartesian coordinate system with drag positive in the $+x$ direction. A laminar, time-dependent case at Reynolds number $\text{Re} = U_0 D / \nu = 100$ is modeled after the Navier–Stokes equations, where U_0 is the mean inflow velocity and ν the kinematic viscosity. The governing equations are discretized on a rectangular grid of dimensions $[-2; 20] \times [-2, 2.1]$. Open flow conditions are used, that consist of a parabolic inflow

$$U_{in}(y) = 1.5 \left(1 + \frac{y}{2}\right) \left(1 - \frac{y}{2.1}\right), \quad (7.8)$$

in the x direction, together with no-slip conditions at the walls (cylinder surface and parallel walls) and a stress-free outflow condition. The cylinder center is at $(0, 0)$, slightly off the centerline of the domain in the y direction to speed up the onset of vortex shedding.

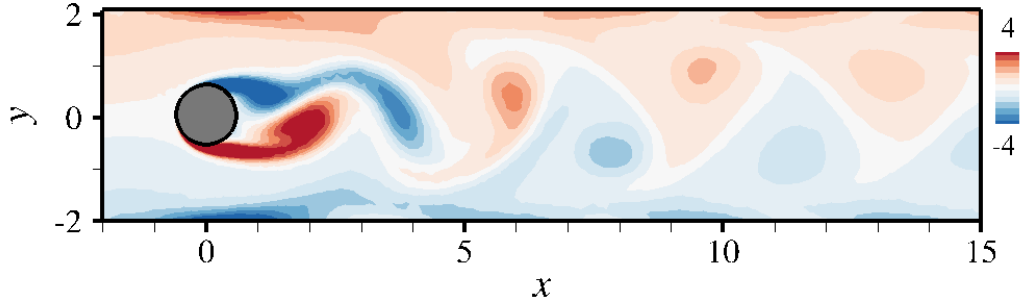


Figure 7.11: Iso-contours of the uncontrolled vorticity field in the time-periodic regime.

7.3.1 Uncontrolled flow

Unsteady computations are performed using a constant time step $\Delta t = 0.005$, as in [39]. The retained mesh is composed of 22000 elements and has been refined at the cylinder wall using a minimum mesh size $h = 0.004$. We have computed first a total of 100 time units of the uncontrolled solution, denoted in the following as the baseline flow, that follows the well-documented limit cycle associated with vortex shedding, illustrated in figure 7.11. The shedding frequency is found to be approximately 0.30, in good agreement with the value reported in [39]. A similar agreement is found for the mean drag coefficient, as the value $\bar{D} = 3.20$ computed by averaging over the 25 last time units (which represents about 7.5 shedding cycles) is within 1% of the reference value [243]. It is worth noticing that larger time-steps up to $\Delta t = 0.02$ have produced mean drag values identical within 2%, but have not been further considered to allow using the same action time step Δt_{act} as in [39].

7.3.2 Open-loop control using single-step DRL

In the context of flow control, and as shown in figure 7.12, injection or suction is achieved through two jets of width $\Delta\theta = 10^\circ$ located at the on the cylinder's poles (that is, at angles $\theta_0 = 90^\circ$ and 270° relative to the flow direction). The objective is to minimize the (time-averaged) mean drag by minimizing the compound reward function

$$r = -\bar{D} - \beta|\bar{L}|, \quad (7.9)$$

whose rightmost term is a penalization that prevents the network from achieving efficient drag reduction at the cost of a large induced lift, as it is damageable in many practical applications (we use here the same value $\beta = 0.2$ as in [39]). A given jet velocity is normal to the cylinder wall (to ensure that the observed drag reduction does not simply result from a mere injection of momentum) and follows

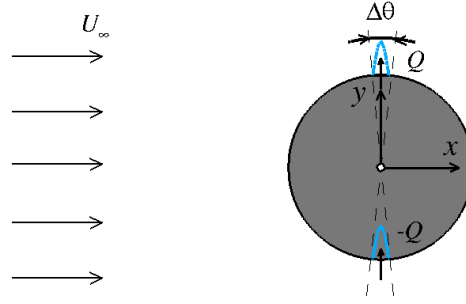


Figure 7.12: Synthetic jets configuration used for control purposes.

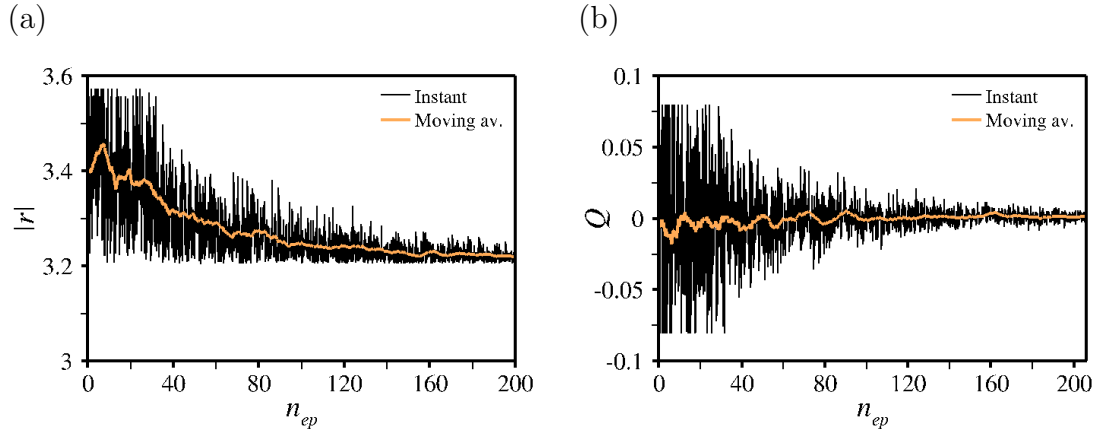


Figure 7.13: Single-step DRL control using PPO-1. (a) Evolution per episode for the instant (black line) and moving average (over episodes, light orange line) reward. (b) Same as (a) for the mass flow rate of the upper actuator.

the parabolic-like distribution going to zero at the edges of the jet

$$W = \frac{Q\pi}{\Delta\theta} \cos \frac{\pi}{\Delta\theta} (\theta - \theta_0), \quad (7.10)$$

where Q is the mass flow rate. A synthetic jets setting is used in which the two jets act reciprocally, i.e., the mass flow rate are Q for the top jet, and $-Q$ for the bottom, meaning that there is only one single control parameter and the instantaneous net mass flow rate is zero for each action (as this is more realistic than adding or subtracting mass from the flow, and has been found useful to avoid numerical issues [39]).

We report first results pertaining to the single-step control problem, for which we use the PPO-1 algorithm introduced in chapter 2. We recall that PPO-1 learns the mean and variance of a multivariate normal search distribution from a single network, under the assumption that all variables have the same variance and are

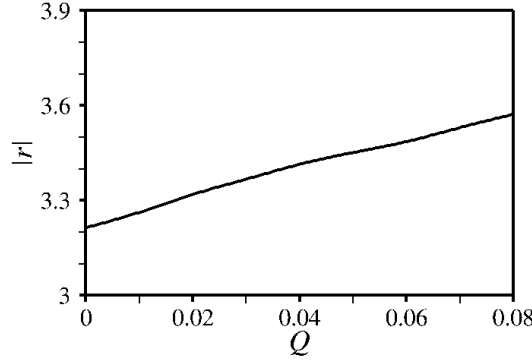


Figure 7.14: Evolution of the numerical reward against the mass flow rate, as obtained by direct numerical simulation.

uncorrelated. The agent is a fully-connected network with two hidden layers, each holding 2 neurons, with learning rate set to 2.5×10^{-3} and PPO loss clipping range to 0.2. A total of 200 episodes has been run using 8 parallel environments, each of which follows the exact same procedure as in the uncontrolled case, and computes 100 time units with time step $\Delta t = 0.005$ to march the initial condition to time-periodic state (in practice, the final baseline state is used as restart condition for all learning episodes). This represents 1600 simulations, each of which is performed on 12 cores and lasts 25mn, hence 85h of total CPU cost. The network outputs a single value $\hat{\xi} \in [-1, 1]$, mapped into

$$Q = \hat{\xi} Q_{max}, \quad (7.11)$$

for the mass flow rate to vary $[0; Q_{max}]$ with $Q_{max} = 0.08$ to prevent unphysically large actuation and associated numerical issues (the associated maximum centerline velocity is then $W_{max}(\theta = 0) = 1.45$). The mean drag and lift are computed by averaging over the 25 last time units, as has been done for the uncontrolled flow, after which the network is updated for 16 epochs using 4 steps mini-batches.

The main results are provided in figure 7.13 in terms of the instant reward and mass flow rate, together with their moving average computed over the 50 latest episodes (equivalently the 50 latest values) Although the moving average reward increases monotonically and reaches a plateau after about 80 episodes, the optimal mass flow rate converges to a close to zero value $Q^* \sim 0.001$. In return, the optimal reward is $|r^*| = 3.22$, with drag $\bar{D}^* = 3.2$ equal to the baseline, and lift weakly fluctuating around zero (not shown here). This is not ascribed to a failure of the method, though, as it only reflects the inability of the retained approach to reduce drag. Indeed, performing a series of numerical simulations with various control amplitudes yields the curve in figure 7.14 showing that the reward decreases

monotonically with Q (hence the baseline solution *is* somehow the optimal searched for by the PPO-1 agent).

7.3.3 Closed-loop control using multi-step DRL

In this section, we tackle the closed-loop problem using the classical PPO algorithm to design an efficient control strategy from multi-step learning episodes during which the network shall output a series of actions (not a single action). Here, the control objective is to minimize the mean drag by minimizing the compound reward function

$$r = -\langle D \rangle_{VS} - \beta |\langle L \rangle_{VS}|, \quad (7.12)$$

where $\langle \cdot \rangle_{VS}$ denotes the moving average over one shedding cycle of the baseline solution (hence $\langle D_0 \rangle_{VS} = \bar{D}_0 = 3.20$, where the 0 superscript indicates a baseline quantity), and we use the same penalization parameter $\beta = 0.2$ as in section 7.3.2. Compared to using the instantaneous values of drag and lift, averaging over a shedding cycle allows reducing the variability in the value of the reward function, which has been found in [39] to improve learning.

The PPO agent is a fully connected network with two dense hidden layers, each holding 512 neurons, with learning rate set to 2.5×10^{-4} , PPO loss clipping range to 0.2, and discount factor to 0.99. Note the substantially increased size of the network with respect to the single-step agent, as the number of neurons in the hidden layers has been increased by a factor of 256, as a result of the multi-step agent being tasked with learning a complex state-action relation, not a simple mapping from a constant input state to a given action. The agent operates under partial knowledge of the state, with pressure information extracted from the simulation and provided to the neural network from 156 probes distributed on the structured grid shown in figure 7.15. As has been mentioned at the beginning of this chapter, the number and location of this probes, as well as the quality of the information they collect, are key for the network to understand and find a possible control strategy of the vortex shedding pattern. The sensitivity of the learned control strategy to the probes distribution has been studied briefly in [39] and more thoroughly in [41]. The general picture for this case is that a relevant arrangement must feature of a sufficient number of probes distributed in the vicinity of the cylinder as well as in its wake region (this has been confirmed by preliminary DRL runs in which probes clustered in the vicinity of the cylinder ultimately yielded poor control efficiency). Meanwhile, the details of said arrangement (the exact number, or the exact nature of the collected information, for instance velocity rather than pressure) have only a limited impact on the learning performance. One difficulty is to appropriately distribute the wake probes, as no learning could be obtained in preliminary runs where the probes spanned only the very near wake, but it is reported in [41] that the

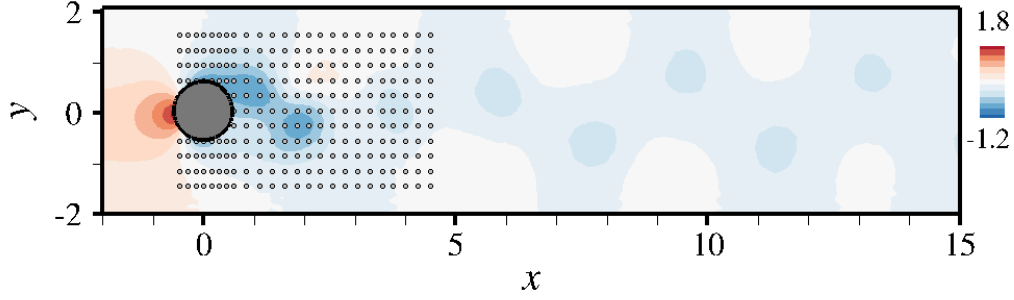


Figure 7.15: Probe array for observation collection in the context of multi-step drag reduction. The background field in the uncontrolled pressure field associated with the vorticity distribution shown in figure 7.11.

information provided by probes positioned too far downstream has little relevance (likely because, by observing the flow closely downstream of the cylinder, the agent is able to observe the consequences of its actions right after they were taken, while more distant probes provide a delayed feedback that can be more difficult to interpret).

As in the single-step case, the final baseline state is used as consistent restart condition for all learning episode. Each episode computes 25 time units with time step $\Delta t = 0.005$, with the first 5 time units simulated without any control, after which the network outputs a series of 80 actions over 20 time units (about 6.5 baseline shedding cycles) and the network is updated for 16 epochs using 8 steps mini-batches. This yields an action time step $\Delta t_{act} = 0.25$ and a ratio of the action to the simulation time-steps of 50 (i.e., each action is evaluated over 50 numerical time steps). For each action, the agent outputs a single value $\hat{\xi}_j \in [-1, 1]$, mapped into $Q_j = \hat{\xi}_j Q_{max}$ with $Q_{max} = 0.08$. In order to avoid discontinuities in the pressure and velocity fields every time a new action is sampled, the control is made continuous over time using

$$\tilde{Q}_{j;k+1} = \tilde{Q}_{j;k} + \alpha(Q_j - \tilde{Q}_{j;k}), \quad (7.13)$$

where $\tilde{Q}_{j;k}$ is the effective control considered at the previous numerical time step, $\tilde{Q}_{j;k+1}$ is the new control, and Q_j is the target action set by the PPO agent for the current 50 time steps. Finally, α is a numerical parameter set to 0.1, which guarantees that \tilde{Q}_j reaches $0.99Q_j$ within 45 numerical time steps (equivalently 0.225 time units), before the network samples a new action. The numerical reward fed to the PPO agent is computed every Δt_{act} once a given action has been fully assessed, by averaging over the latest vortex shedding period (hence it assesses the effect of the latest 12 actions).

A total of 500 episodes has been run using one single environment. This represents 500 simulations, each of which is performed on 12 cores and lasts 20mn,

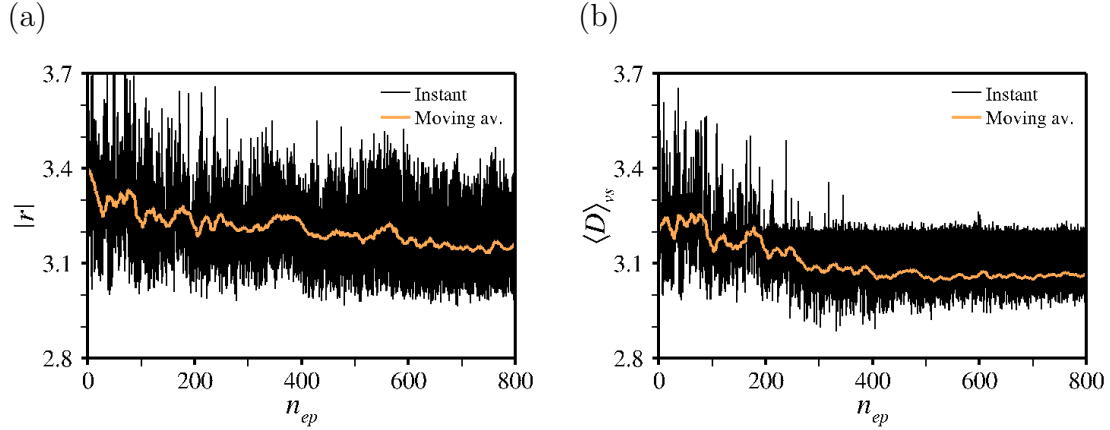


Figure 7.16: Multi-step DRL control using PPO-1. (a) Evolution per episode for the instant (black line) and moving average (over episodes, light orange line) reward. (b) Same as (a) for the drag components of the reward.

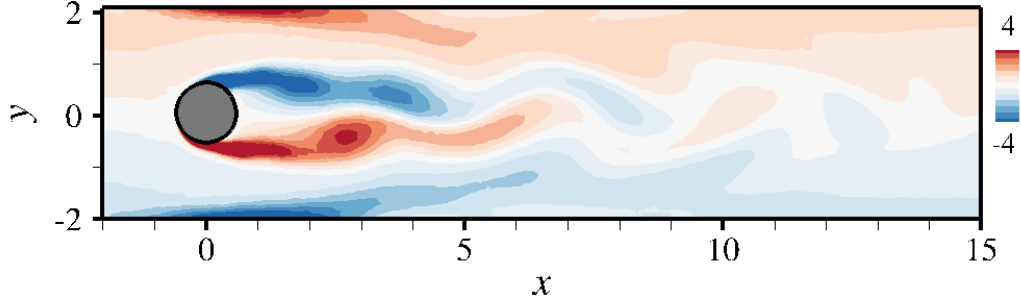


Figure 7.17: Iso-contours of the vorticity field optimally controlled using PPO multi-step DRL control. The snapshot corresponds to the established pseudo-periodic modified regime, attained after the initial transient control.

hence 165h of total CPU cost. As illustrated in figure 7.16 showing the evolution per episode of the numerical reward, a robust learning is achieved after 250 episodes. The moving average reward computed over the 10 latest episodes (equivalently the 800 latest values) slowly decreases to reach the optimum $|r^*| = 3.16$, with some noises due to the exploration characteristic. The corresponding drag coefficient is $\langle D \rangle_{VS}^* = 3.05$ (meaning that the lift component of the reward is not zero), but the instant value reaches a minimum of 2.96 representing a drag reduction of about 8%. These values, as well as the associated flow field shown in figure 7.17 are very consistent with those reported in [39] and assess the validity of the present PPO/CFD implementation. The evolution of drag under optimal PPO actuation shown in figure 7.18(a) shows that the controlled value decreases to $\langle D \rangle_{VS}^*$ within roughly 5 time units, which represents a series of 20 actions, out of the 80 actions sampled

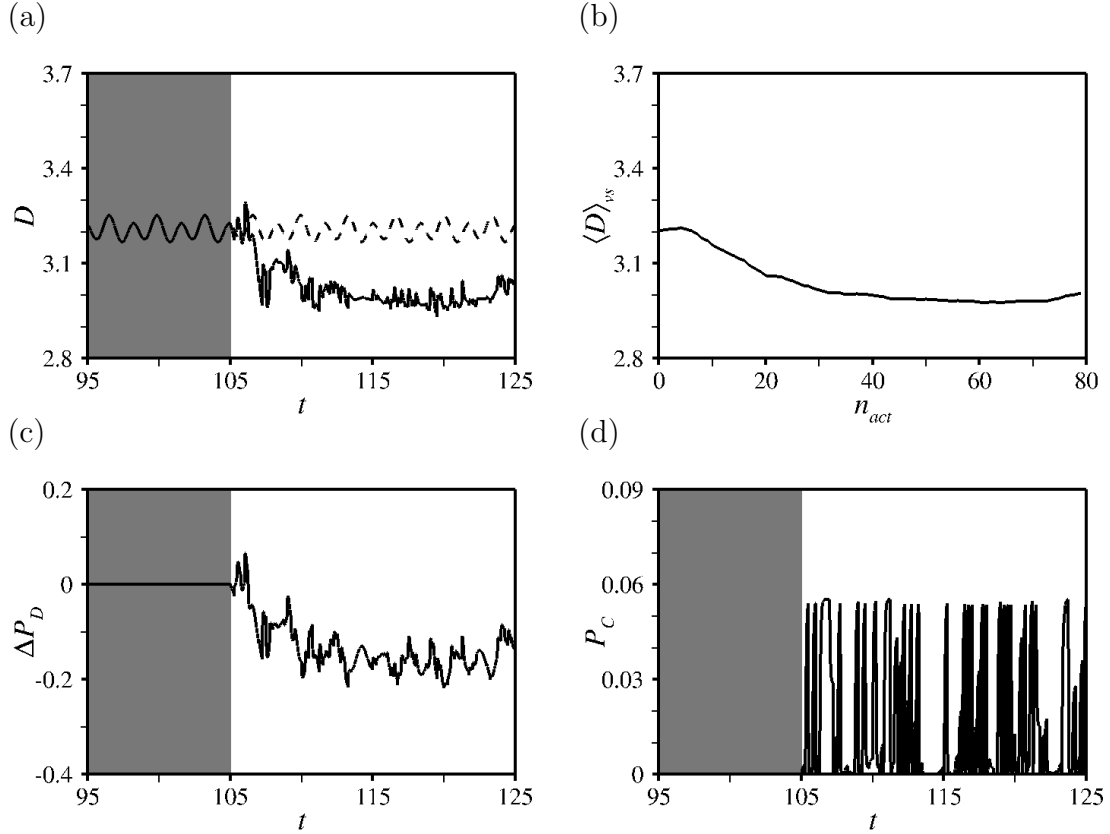


Figure 7.18: (a) Time evolution of the instantaneous reward over the course of an optimal PPO multi-step episode. The controlled (resp. uncontrolled) values are shown as the solid (resp. dashed) lines. The dark gray shade marks the initial time interval during which control is forced to zero. (b) Evolution of drag per DRL action along the learning episode represented in (a). (c-d) Same as (a) for the (c) gain in power drag, and (d) actuation power of a single jet.

by the agent over the course of the episode. At this point, the average value over a shedding period used to determine remains essentially constant, as evidenced in figure 7.18(b).

Another important indicator of the performance of the control is the energy required for drag reduction, defined as

$$P_C = \frac{1}{4} \int_{-\frac{\Delta\theta}{2}}^{\frac{\Delta\theta}{2}} W^2 |W| d\theta, \quad (7.14)$$

and whose evolution against time is reported for a single jet in figure 7.18(d). The actuation power averaged in time over the second half of the optimal PPO episode (to get rid of the initial transient) is $P_C = 0.018$, which is only 8% of the time-

averaged baseline flow drag power

$$P_{D0} = \frac{1}{2} \int_{-\frac{1}{2}}^{\frac{1}{2}} U_{in} \bar{D}_0 dy = \frac{1}{2} Q_{in} \bar{D}_0 = \frac{3}{4} \left(1 - \frac{1}{3H^2} \right) \bar{D}_0, \quad (7.15)$$

where Q_{in} is the reference mass flow rate intercepting the cylinder. Another interesting indicator is the power Saving Ratio (PSR) introduced by [244], defined as the ratio of the gain in drag power (figure 7.18(c)) to the control power, hence

$$\text{PSR} = \frac{3}{4} \left(1 - \frac{1}{3H^2} \right) \frac{\bar{D} - \bar{D}_0}{P_C}. \quad (7.16)$$

We obtain here a PSR of 6, meaning that the control, albeit efficient, is not highly energy-efficient. This is because the PPO agent selects rather high actuation amplitudes, as the mass flow rate regularly peaks at its maximum allowed value (not shown here). This contrasts with the results reported in [39], whose actuation amplitude is indeed large during the initial transient, but then drops to small values representing only about 10% percent of the maximum value (a similar behavior is documented in [41], albeit on a different case without the lateral channel walls). The reasons for the observed differences are unclear at the time being, as we still need to consolidate our results by removing the exploration noise and by running the neural network in full exploitation mode. A possible explanation may lie in the maximum allowed actuation amplitude itself, as the latter varies from study to study (with sometimes different and intricate definitions), which hinders reproducibility. For instance, the maximum energy allowed for drag reduction is 0.057 in the present case, but 0.21 in [41], and allowing for larger overall actuation amplitudes during may ultimately help save actuation power if more energy is spent to bring the flow in low-drag state, but less energy is required to have it stay on it. Even though, we believe these results assess at the same time the high potential of multi-step DRL to improve the control performance by resorting to closed-loop actuation, and the need for additional fine tuning in order to consistently outperform the current reference methods. To this end, it should however be kept in mind that efficiency comparisons between different actuation techniques should be performed carefully, as [245] use cylinder rotation to achieve an extremely energy-efficient control that outperforms the present study (and all other related studies) in terms of PSR and net drag reduction, but do not consider the inertia of the rotating cylinder for their power-based comparisons.

7.4 Conclusion

This chapter has focused on the coupling of Cimlib-CFD with the multi-step DRL approach that is widely used in the literature. The approach is relevant to closed-

loop flow control, as the agent is then tasked to optimize a complete state-action relation, not just a mapping from constant state to optimal action. Two validation test cases have been implemented and studied with the PPO algorithm, a first steady-state thermal problem whose optimal is open-loop, and a second time-dependent problem whose optimal is closed-loop. Promising results have been obtained, with the drag reduction achieved on the second test-case problem fully consistent with that reported in the literature, but further investigations are needed to assess the impact of using the agent in full exploration mode instead of exploitation, as this may ultimately substantially hinder the energetic efficiency of the optimal control.

Chapter 8

Conclusion

Contents

8.1 Summary	206
8.2 Perspectives	208

8.1 Summary

The objective of this thesis was to implement a massively parallel numerical platform bringing together the recent advances achieved at the CEMEF research center in the fields of Computational Fluid Dynamics (CFD) and Deep Reinforcement Learning (DRL). It should be considered as a first step towards the long-term ambition to revisit the current design of control strategies for computationally-expensive flow problems representative of industrial applications. The first objective was to further shape the capabilities of Deep Reinforcement Learning (DRL) for closed-loop control, essentially by adding novel control cases of upscaled complexity compared to the existing literature. The second objective was to gauge the ability of a neural network to be used as an efficient black-box optimizer for open-loop control problems, by having it interact only once per episode with its environment to learn the mapping from a constant input state to an optimal action (hence, single-step episodes, and by extension, single-step DRL).

Below is a summary of the work undertaken during the thesis. The latter has taken advantage of previous and current developments by the Computing & Fluids group of CEMEF, regarding both the CFD solver (the in-house Cimlib-CFD finite elements library) and the DRL algorithms (in-house implementation of classical algorithms: PPO, as well as in-house algorithms: PPO-1, PBO). Recent efforts have allowed to apply the multi-step approach to closed-loop thermal control, which indeed represents a novelty with respect to the existing literature (although a simplified steady-state case has been considered for validation purposes). Nonetheless, most of the work has ultimately been devoted to developing the single-step approach, as it has led to promising early results and the scope of applications that can be tackled is extremely wide, as evidenced by the many projects currently underway at CEMEF using the methodology.

Chapter 2: we introduce the basic concepts of Deep Reinforcement Learning, the advanced branch of machine learning that train neural networks in solving decision-making problems. The most represented value-based and policy-based methods used so far in fluid mechanics are briefly reviewed, after which the algorithms selected for this thesis are presented in further details. This includes the celebrated Proximal Policy Optimization (PPO) algorithm, used in chapter 7 for closed-loop control, its tweaked, single-step version called PPO-1, used in chapters 4-5 for open-loop control, and a refined single-step algorithm called Policy Based Optimization (PBO), used in chapter 6 for shape optimization. The main difference between PPO-1 and PBO is that PPO-1 proceeds from the standard PPO algorithm and therefore uses trust regions while sampling actions isotropically from scalar covariance matrices, while PBO uses a variant of the vanilla policy gradient method while sampling actions anisotropically from full covariance matrices.

Chapter 3 we describe the Cimlib-CFD finite elements library used to solve the incompressible Navier–Stokes equations. The latter is based on the Variational Multiscale (VMS) method, that introduces an a priori decomposition of the unknown variables into coarse and fine components, that correspond to different scales of resolution. Only the large scales of the flow field are fully solved at the discrete levels, while the effect of the small scale is represented by consistently derived source terms proportional to the residual of the large scale solution. The extension to the Reynolds-Averaged Navier–Stokes approach with Spalart-Allmaras turbulence model as closure is also provided for which the stabilization proceeds from that of the Convection Diffusion Reaction (CDR) equation. Finally, we present the monolithic Immersed Volume Method (IVM) used to solve a single set of equations on a unique computational domain encompassing generic solid bodies immersed in the domain where the fluid flows, and the anisotropic mesh adaptation procedure used to ensure that all solid/fluid interfaces are accurately represented. This is of great importance for DRL problems in which the mesh depends on the sampled action (for instance if a neural network is tasked with optimizing the position of a solid body, as is done here on several occasions), as it allows ensuring that all actions are assessed with comparable accuracy.

Chapter 4: we assess the relevance of single-step DRL for cases where the policy to be learnt by a neural network is independent of state. The PPO-1 algorithm is applied to several open-loop control problems whose parameter spaces are large enough to dismiss forward optimization by direct numerical simulation. It is found to successfully reduce the drag of laminar and turbulent cylinder flows by identifying the best positions for placement of a small rod, the achieved reduction ranging from 2% at $Re = 40$ using a circular main cylinder, up to 30% at $Re=22000$ using a square main cylinder. The method also reduces the drag of the fluidic pinball, a more complex arrangement of three rotating circular cylinders in a turbulent stream, for which the optimal reduction is by almost 60%.

Chapter 5: the scope of the single-step approach is extended to the control of conjugate heat transfer systems governed by the coupled Navier–Stokes and heat equations. The approach is found capable of improving the homogeneity of temperature across the surface of two and three-dimensional hot workpieces under impingement cooling. Several control strategies are compared, for which the position of multiple cold air injectors is optimized relative to a fixed workpiece position. The flexibility of the numerical framework also allows optimizing the workpiece position relative to a fixed, symmetrical actuation injector distribution. The optimal position turns to be offset from the symmetry axis, which had not been anticipated, and stresses that PPO-1 (and DRL in general) can effectively explore and discover new solutions

from unforeseen parameter combinations.

Chapter 6: single-step DRL is applied to aerodynamic shape optimization problems. Several cases are documented, for which two- and three-dimensional shapes are generated by varying a chord length and a symmetric thickness distribution (and possibly extruding in the off-body direction) at a discrete set of control points connected by cubic Bézier curves from local position and curvature information. The PBO algorithm is used to find the shapes maximizing the lift-to-drag ratio under constant incidence of 30° and constant chord length. By doing so, the performance of the equivalent ellipse (that is, the ellipse of same cross-sectional area) is increased by 13% in 2-D and 5% in 3-D. Moreover, the optimal is found to perform just as well as a conventional airfoil, despite DRL starting from scratch and having no priori knowledge of aerodynamic concepts.

Chapter 7: we ultimately return to multi-step DRL, i.e., the one approach that has been used in all the literature available on this topic. The main difference is that single-step DRL optimizes a transformation from a constant input state to a given action, which allows using very small networks consisting (here, 2 up to 4 neurons per hidden layer), while multi-step DRL optimizes a complex state-action relation more relevant to closed-loop control problems, but one that requires much larger networks (here, 512 neurons per hidden layers). Two problems are tackled with the PPO algorithm: the first one is an original thermal control problem intended for comparison with dedicated single-step results, for which we successfully improve the homogeneity and final temperature across the surface of two-dimensional hot workpiece under slightly asymmetrical impingement cooling (although further investigation is needed to truly assess the level performance by removing the exploration noise). The second one is the classical drag reduction problem introduced by Rabault *et al.* [39], for which we report a comparable reduction by 8% after a few hundreds episodes, each of which assesses a series of 80 actions.

8.2 Perspectives

Many perspectives can be drawn from this work, as fluid dynamicists have just begun to assess the relevance of DRL to assist the design of optimal flow control strategies. The one advantage of DRL is the scope and applicability, as the only prerequisite is the ability to compute accurate numerical solutions. Obviously, this is up to the CFD solver, not the RL algorithm. Cimlib-CFD is currently successfully coupled with state-of-the-art multi-step and single-step DRL algorithms, and computations can be run on various architectures ranging from small-scale, local installations (180 cores) to large-scale, high performance computing clusters (2176

cores). Therefore, we would not anticipate any additional numerical developments (besides some adjustments to fully benefit from massively parallel architectures) before tackling more complex flow problems already in the scope of the CFD solver, such as non-newtonian and/or multiphase flows, or fluid-structure interaction problems.

Despite these achievements, further development, characterization and fine-tuning are needed to consolidate the acquired knowledge. The one challenge to overcome in the future is improving the sampling efficiency, that is, reducing the number of calls to the CFD solver necessary to achieve learning, as it represents the most time-consuming part of the method (CFD environments routinely involve tens or hundreds of millions of degrees of freedom and are thus very resource expensive). Meanwhile, classical reinforcement learning methods are known to have low sample efficiency, i.e. many attempts are required for an agent to learn the desired behavior. This is likely a huge reason why complex problems involving time-dependent, three-dimensional solutions have received little to no interest so far. Particular attention should thus be paid to reducing the amount of calls to the CFD solver necessary to achieve learning, as it represents the most time-consuming part of the method. A detailed comparison between off-policy and on-policy algorithms would be helpful in this regards, as on-policy methods (including the PPO algorithm used in this work) are supposed to have lower sample efficiency in the context of CPU-expensive environments, which is not supported by the available results.

Transfer learning, defined as the ability to use pre-trained deep learning models, is another emerging concept that may lead to substantial performance improvement. Promising results in this regards have been reported in the literature, for instance it has been shown possible in [49] to achieve robust drag reduction by training simultaneously a single agent at four different Reynolds numbers distributed between 100 and 400. After training, the agent efficiently reduces drag regardless of the Reynolds numbers in the range from 60 to 400, although the performance for a given Reynolds number is a tad below that of an agent specifically trained at this value of Re. Nonetheless, such an approach remains limited to cases with similar dynamics, as the use the pre-trained agent failed to reduce drag at $Re = 1000$ due to too different flow patterns.

Bibliography

- [1] H. Choi, W.-P. Jeon, J. Kim, Control of flow over a bluff body, *Annu. Rev. Fluid Mech.* 40 (2008) 113–139. [2](#), [6](#), [56](#)
- [2] L. N. Cattafesta, M. Sheplak, Actuators for active flow control, *Annu. Rev. Fluid Mech.* 43 (2011) 247–272. [2](#), [6](#), [56](#)
- [3] W. Schoppa, F. Hussain, A large-scale control strategy for drag reduction in turbulent boundary layers, *Physics of Fluids* 10 (1998) 1049–1051. [2](#), [9](#)
- [4] W. Rawat, Z. Wang, Deep convolutional neural networks for image classification: A comprehensive review, *Neural computation* 29 (2017) 2352–2449. [2](#), [9](#)
- [5] A. B. Nassif, I. Shahin, I. Attili, M. Azzeh, K. Shaalan, Speech recognition using deep neural networks: A systematic review, *IEEE access* 7 (2019) 19143–19165. [2](#), [9](#)
- [6] J. Gui, Z. Sun, Y. Wen, D. Tao, J. Ye, A review on generative adversarial networks: Algorithms, theory, and applications, *IEEE Transactions on Knowledge and Data Engineering*. [2](#), [9](#)
- [7] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, P. Abbeel, Asymmetric actor critic for image-based robot learning, *arXiv preprint arXiv:1710.06542*. [3](#), [9](#), [17](#)
- [8] D. Bahdanau, P. Brakel, K. Xu, A. Goyal, R. Lowe, J. Pineau, A. Courville, Y. Bengio, An actor-critic algorithm for sequence prediction, *arXiv preprint arXiv:1607.07086*. [3](#), [9](#), [17](#)
- [9] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, Playing atari with deep reinforcement learning, *arXiv preprint arXiv:1312.5602*. [3](#), [9](#), [17](#)

- [10] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, A. Shah, Learning to drive in a day, in: 2019 International Conference on Robotics and Automation (ICRA), IEEE, 2019, pp. 8248–8254. [3](#), [9](#), [17](#), [147](#)
- [11] W. Knight, Google just gave control over data center cooling to an AI, <http://www.technologyreview.com/s/611902/google-just-gave-control-over-data-center-cooling-to-an-ai/> (2018). [3](#), [9](#), [147](#)
- [12] V. Belus, J. Rabault, J. Viquerat, Z. Che, E. Hachem, U. Reglade, Exploiting locality and translational invariance to design effective deep reinforcement learning control of the 1-dimensional unstable falling liquid film, *AIP Advances* 9 (12) (2019) 125014. [3](#), [9](#), [57](#), [102](#), [147](#)
- [13] M. A. Bucci, O. Semeraro, A. Allauzen, G. Wisniewski, L. Cordier, L. Mathelin, Control of chaotic systems by deep reinforcement learning, *Proceedings of the Royal Society A* 475 (2231) (2019) 20190351. [24](#)
- [14] G. Novati, L. Mahadevan, P. Koumoutsakos, Controlled gliding and perching through deep-reinforcement-learning, *Physical Review Fluids* 4 (9) (2019) 093902. [3](#), [9](#), [57](#), [102](#)
- [15] D. Fan, L. Yang, Z. Wang, M. S. Triantafyllou, G. E. Karniadakis, Reinforcement learning for bluff body active flow control in experiments and simulations, *Proc. Natl. Acad. Sci. U.S.A.* 117 (2020) 26091–26098. [3](#), [9](#), [102](#), [142](#), [147](#)
- [16] X. Y. Lee, A. Balu, D. Stoecklein, B. Ganapathysubramanian, S. Sarkar, A case study of deep reinforcement learning for engineering design: Application to microfluidic devices for flow sculpting, *Journal of Mechanical Design* 141 (11). [3](#), [9](#)
- [17] G. Beintema, A. Corbetta, L. Biferale, F. Toschi, Controlling rayleigh–bénard convection via reinforcement learning, *Journal of Turbulence* 21 (9-10) (2020) 585–605. [3](#), [9](#), [102](#), [103](#), [118](#), [119](#), [121](#), [122](#), [142](#), [185](#)
- [18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, arXiv preprint arXiv:1707.06347. [3](#), [12](#), [28](#), [57](#), [58](#), [102](#), [113](#), [147](#), [148](#)
- [19] N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, P. K. Tucker, Surrogate-based analysis and optimization, *Prog. Aerosp. Sci.* 41 (2005) 1–28. [4](#), [12](#), [146](#)

- [20] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, N. De Freitas, Learning to learn by gradient descent by gradient descent, *Advances in neural information processing systems* 29. 4, 12
- [21] X. Yan, J. Zhu, M. Kuang, X. Wang, Aerodynamic shape optimization using a novel optimizer based on machine learning techniques, *Aerospace Science and Technology* 86 (2019) 826–835. 4, 11, 12, 57, 102, 147
- [22] P. Hämmäläinen, A. Babadi, X. Ma, J. Lehtinen, Ppo-cma: Proximal policy optimization with covariance matrix adaptation, in: *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*, IEEE, 2020, pp. 1–6. 4, 12
- [23] J. J. Corbett, H. W. Koehler, Updated emissions from ocean shipping, *J. Geophys. Res.* 108 (2003) 4650–64. 6, 56, 146
- [24] How vortex generators boost wind-turbine performance and AEP. From <https://www.windpowerengineering.com/vortex-generators-boost-wind-turbine-performance-aep>. 6
- [25] AIRCOAT: Flying on water. From <https://aircoat.eu>. 6
- [26] T. Igarashi, Drag reduction of a square prism by flow control using a small rod, *J. Wind Eng. Ind. Aerodyn.* 69 (1997) 141–153. 7, 80, 81
- [27] T. C. Corke, C. L. Enloe, S. P. Wilkinson, Dielectric barrier discharge plasma actuators for flow control, *Annu. Rev. Fluid Mech.* 42 (2010) 505–529. 7, 56
- [28] S. Koziel, Y. Tesfahunegn, A. Amrit, L. T. Leifsson, Rapid multi-objective aerodynamic design using co-kriging and space mapping, in: *57th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 2016, p. 0418. 7
- [29] M. C. Hall, Application of adjoint sensitivity theory to an atmospheric general circulation model, *Journal of Atmospheric Sciences* 43 (22) (1986) 2644–2652. 8, 56, 146
- [30] B. Mohammadi, O. Pironneau, Shape optimization in fluid mechanics, *Annu. Rev. Fluid Mech.* 36 (2004) 255–279. 8, 56
- [31] S. Scott Collis, K. Ghayour, M. Heinkenschloss, M. Ulbrich, S. Ulbrich, Optimal control of unsteady compressible viscous flows, *International journal for numerical methods in fluids* 40 (2002) 1401–1429. 8

- [32] S. N. Skinner, H. Zare-Behtash, State-of-the-art in aerodynamic shape optimisation methods, *Appl. Soft Comput.* 62 (2018) 933–962. [8](#), [9](#), [146](#)
- [33] R. P. Dwight, J. Brezillon, Effect of approximations of the discrete adjoint on gradient-based optimization, *AIAA journal* 44 (12) (2006) 3022–3031. [8](#)
- [34] C. C. Margossian, A review of automatic differentiation and its efficient implementation, *Wiley interdisciplinary reviews: data mining and knowledge discovery* 9 (2019) e1305. [8](#)
- [35] S. Sivanandam, S. Deepa, Genetic algorithms, in: *Introduction to genetic algorithms*, Springer, 2008, pp. 15–37. [8](#)
- [36] R. Hassan, B. Cohanım, O. De Weck, G. Venter, A comparison of particle swarm optimization and the genetic algorithm, *AIAA* 2005-1897. [9](#), [146](#)
- [37] G. Novati, S. Verma, D. Alexeev, D. Rossinelli, W. M. Van Rees, P. Koumoutsakos, Synchronisation through learning for two self-propelled swimmers, *Bioinspiration & biomimetics* 12 (3) (2017) 036001. [10](#), [57](#), [102](#), [147](#)
- [38] S. Verma, G. Novati, P. Koumoutsakos, Efficient collective swimming by harnessing vortices through deep reinforcement learning, *Proceedings of the National Academy of Sciences* 115 (23) (2018) 5849–5854. [10](#), [24](#), [57](#), [102](#), [147](#)
- [39] J. Rabault, M. Kuchta, A. Jensen, U. Réglade, N. Cerardi, Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control, *Journal of fluid mechanics* 865 (2019) 281–302. [10](#), [11](#), [24](#), [57](#), [60](#), [102](#), [147](#), [184](#), [194](#), [195](#), [196](#), [198](#), [200](#), [202](#), [208](#)
- [40] J. Rabault, A. Kuhnle, Accelerating deep reinforcement learning strategies of flow control through a multi-environment approach, *Phys. Fluids* 31 (2019) 094105. [10](#), [194](#)
- [41] R. Paris, R. Beneddine, J. Dandois, Robust flow control and optimal sensor placement using deep reinforcement learning, *arXiv preprint arXiv:2006.11005*. [10](#), [57](#), [102](#), [198](#), [202](#)
- [42] M. Tokarev, E. Palkin, R. Mullyadzhyanov, Deep reinforcement learning control of cylinder flow using rotary oscillations at low reynolds number, *Energies* 13 (22) (2020) 5920. [10](#), [17](#), [184](#)
- [43] H. Xu, W. Zhang, J. Deng, J. Rabault, Active flow control with rotating cylinders by an artificial neural network trained by deep reinforcement learning, *J. Hydrodynam.* 32 (2020) 254–258. [10](#), [57](#), [102](#)

- [44] M. A. Elhawary, Deep reinforcement learning for active flow control around a circular cylinder using unsteady-mode plasma actuators, arXiv preprint arXiv:2012.10165. [10](#)
- [45] F. Ren, J. Rabault, H. Tang, Applying deep reinforcement learning to active flow control in weakly turbulent conditions, *Physics of Fluids* 33 (3) (2021) 037121. [11](#), [147](#)
- [46] R. Li, Y. Zhang, H. Chen, Learning the aerodynamic design of supercritical airfoils through deep reinforcement learning, *AIAA Journal* 59 (10) (2021) 3988–4001. [11](#), [147](#)
- [47] S. Qin, S. Wang, L. Wang, C. Wang, G. Sun, Y. Zhong, Multi-objective optimization of cascade blade profile based on reinforcement learning, *Appl. Sci.* 11 (2021) 106. [11](#), [147](#)
- [48] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al., Mastering the game of go without human knowledge, *nature* 550 (7676) (2017) 354–359. [17](#), [24](#), [57](#), [102](#), [147](#)
- [49] H. Tang, J. Rabault, A. Kuhnle, Y. Wang, T. Wang, Robust active flow control over a range of reynolds numbers using an artificial neural network trained through deep reinforcement learning, *Physics of Fluids* 32 (5) (2020) 053605. [17](#), [57](#), [102](#), [147](#), [184](#), [194](#), [209](#)
- [50] R. S. Sutton, A. G. Barto, Reinforcement learning: An introduction, MIT press, 2018. [17](#), [18](#), [110](#), [111](#)
- [51] M. H. Kalos, P. A. Whitlock, Monte carlo methods, John Wiley & Sons, 2009. [21](#)
- [52] D. Precup, Eligibility traces for off-policy policy evaluation, Computer Science Department Faculty Publication Series (2000) 80. [22](#)
- [53] R. S. Sutton, Learning to predict by the methods of temporal differences, *Machine learning* 3 (1) (1988) 9–44. [22](#)
- [54] C. J. Watkins, P. Dayan, Q-learning, *Machine learning* 8 (3-4) (1992) 279–292. [23](#)
- [55] R. J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, *Machine learning* 8 (3) (1992) 229–256. [24](#)

- [56] M. Moravčík, M. Schmid, N. Burch, V. Lisỳ, D. Morrill, N. Bard, T. Davis, K. Waugh, M. Johanson, M. Bowling, Deepstack: Expert-level artificial intelligence in heads-up no-limit poker, *Science* 356 (6337) (2017) 508–513. [24](#), [57](#), [147](#)
- [57] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, M. Hutter, Learning agile and dynamic motor skills for legged robots, *Science Robotics* 4 (26). [24](#), [57](#), [102](#), [147](#)
- [58] Y. Deng, F. Bao, Y. Kong, Z. Ren, Q. Dai, Deep direct reinforcement learning for financial signal representation and trading, *IEEE transactions on neural networks and learning systems* 28 (3) (2016) 653–664. [24](#), [57](#), [147](#)
- [59] A. Bernstein, E. Burnaev, Reinforcement learning in computer vision, in: Tenth International Conference on Machine Vision (ICMV 2017), Vol. 10696, International Society for Optics and Photonics, 2018, p. 106961S. [24](#), [57](#), [102](#), [147](#)
- [60] X. Pan, Y. You, Z. Wang, C. Lu, Virtual to real reinforcement learning for autonomous driving, arXiv preprint arXiv:1704.03952. [24](#), [57](#)
- [61] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, arXiv preprint arXiv:1509.02971. [24](#), [27](#), [102](#)
- [62] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning representations by back-propagating errors, *nature* 323 (6088) (1986) 533–536. [25](#), [59](#), [101](#), [111](#)
- [63] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, MIT press, 2016. [25](#), [108](#)
- [64] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, *nature* 518 (7540) (2015) 529–533. [25](#), [56](#), [102](#)
- [65] M. G. Bellemare, Y. Naddaf, J. Veness, M. Bowling, The arcade learning environment: An evaluation platform for general agents, *Journal of Artificial Intelligence Research* 47 (2013) 253–279. [25](#)
- [66] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, W. Zaremba, Hindsight experience replay, *Advances in neural information processing systems* 30. [25](#)

- [67] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M. Riedmiller, Deterministic policy gradient algorithms, in: International conference on machine learning, PMLR, 2014, pp. 387–395. [27](#)
- [68] J. Schulman, S. Levine, P. Abbeel, M. Jordan, P. Moritz, Trust region policy optimization, in: International conference on machine learning, PMLR, 2015, pp. 1889–1897. [27](#), [112](#)
- [69] H. Ghraieb, J. Viquerat, A. Larcher, P. Meliga, E. Hachem, Single-step deep reinforcement learning for open-loop control of laminar and turbulent flows, *Physical Review Fluids* 6 (5) (2021) 053902. [29](#), [31](#), [54](#), [103](#), [142](#), [147](#), [148](#), [151](#)
- [70] J. Viquerat, R. Duvigneau, P. Meliga, A. Kuhnle, E. Hachem, Policy-based optimization: single-step policy gradient method seen as an evolution strategy, arXiv preprint arXiv:2104.06175. [30](#), [34](#), [148](#), [151](#)
- [71] T. Degris, M. White, R. S. Sutton, Off-policy actor-critic, <https://arxiv.org/abs/1205.4839> (2013). [32](#)
- [72] K. Numpacharoen, A. Atsawarungrangkit, Generating correlation matrices based on the boundaries of their coefficients, *PLoS One* 7 (11) (2012) e48902. [33](#), [151](#), [155](#)
- [73] A. Masud, R. Calderer, A variational multiscale method for incompressible turbulent flows: Bubble functions and fine scale fields, *Computer Methods in Applied Mechanics and Engineering* 200 (33-36) (2011) 2577–2593. [37](#)
- [74] O. Guasch, R. Codina, Statistical behavior of the orthogonal subgrid scale stabilization terms in the finite element large eddy simulation of turbulent flows, *Computer methods in applied mechanics and engineering* 261 (2013) 154–166.
- [75] U. Rasthofer, V. Gravemeier, Multifractal subgrid-scale modeling within a variational multiscale method for large-eddy simulation of turbulent flow, *Journal of Computational Physics* 234 (2013) 79–107. [37](#)
- [76] J. Donea, A. Huerta, *Finite element methods for flow problems*, John Wiley & Sons, 2003. [37](#)
- [77] T. E. Tezduyar, S. Mittal, S. Ray, R. Shih, Incompressible flow computations with stabilized bilinear and linear equal-order-interpolation velocity-pressure elements, *Computer Methods in Applied Mechanics and Engineering* 95 (2) (1992) 221–242. [37](#)

- [78] T. J. Hughes, Multiscale phenomena: Green's functions, the dirichlet-to-neumann formulation, subgrid scale models, bubbles and the origins of stabilized methods, *Computer methods in applied mechanics and engineering* 127 (1-4) (1995) 387–401. [37](#), [39](#)
- [79] R. Codina, Stabilization of incompressibility and convection through orthogonal sub-scales in finite element methods, *Computer methods in applied mechanics and engineering* 190 (13-14) (2000) 1579–1599. [37](#), [61](#), [106](#), [152](#)
- [80] R. Codina, J. Blasco, Stabilized finite element method for the transient navier–stokes equations based on a pressure gradient projection, *Computer Methods in Applied Mechanics and Engineering* 182 (3-4) (2000) 277–300. [37](#)
- [81] V. Gravemeier, W. A. Wall, E. Ramm, A three-level finite element method for the instationary incompressible navier–stokes equations, *Computer Methods in Applied Mechanics and Engineering* 193 (15-16) (2004) 1323–1366. [37](#)
- [82] V. Gravemeier, A consistent dynamic localization model for large eddy simulation of turbulent flows based on a variational formulation, *Journal of Computational Physics* 218 (2) (2006) 677–701.
- [83] V. Gravemeier, Scale-separating operators for variational multiscale large eddy simulation of turbulent flows, *Journal of Computational Physics* 212 (2) (2006) 400–435. [37](#)
- [84] R. D. Moser, J. Kim, N. N. Mansour, Direct numerical simulation of turbulent channel flow up to $re \tau = 590$, *Physics of fluids* 11 (4) (1999) 943–945. [38](#)
- [85] S. Badia, R. Codina, Stabilized continuous and discontinuous galerkin techniques for darcy flow, *Computer Methods in Applied Mechanics and Engineering* 199 (25-28) (2010) 1654–1667. [41](#)
- [86] R. Codina, Finite element approximation of the three-field formulation of the stokes problem using arbitrary interpolations, *SIAM Journal on Numerical Analysis* 47 (1) (2009) 699–718. [41](#)
- [87] R. Codina, J. Principe, O. Guasch, S. Badia, Time dependent subscales in the stabilized finite element approximation of incompressible flow problems, *Computer Methods in Applied Mechanics and Engineering* 196 (21-24) (2007) 2413–2430. [41](#)
- [88] S. Badia, R. Codina, On a multiscale approach to the transient stokes problem: Dynamic subscales and anisotropic space–time discretization, *Applied Mathematics and Computation* 207 (2) (2009) 415–433. [41](#)

- [89] J. Sari, F. Cremonesi, M. Khalloufi, F. Cauneau, P. Meliga, Y. Mesri, E. Hachem, Anisotropic adaptive stabilized finite element solver for rans models, *Int. J. Numer. Methods Fluids* 86 (2018) 717–736. [43](#), [62](#), [142](#), [153](#)
- [90] G. Guiza, A. Larcher, A. Goetz, L. Billon, P. Meliga, E. Hachem, Anisotropic boundary layer mesh generation for reliable 3D unsteady RANS simulations, *Finite Elem. Anal. Des.* 170 (2020) 103345. [43](#), [62](#), [142](#), [153](#)
- [91] J. Boussinesq, *Théorie de l'écoulement tourbillonnant et tumultueux des liquides dans les lits rectilignes a grande section*, Vol. 1, Gauthier-Villars, 1897. [44](#)
- [92] B. Baldwin, H. Lomax, Thin-layer approximation and algebraic model for separated turbulentflows, in: *16th aerospace sciences meeting*, 1978, p. 257. [44](#)
- [93] A. Smith, T. Cebeci, Numerical solution of the turbulent-boundary-layer equations, Tech. rep., DOUGLAS AIRCRAFT CO LONG BEACH CA AIRCRAFT DIV (1967). [44](#)
- [94] D. C. Wilcox, et al., *Turbulence modeling for CFD*, Vol. 2, DCW industries La Canada, CA, 1998. [44](#)
- [95] B. Baldwin, T. Barth, A one-equation turbulence transport model for high reynolds number wall-bounded flows, in: *29th aerospace sciences meeting*, 1991, p. 610. [45](#)
- [96] B. E. Launder, D. B. Spalding, The numerical computation of turbulent flows, in: *Numerical prediction of flow, heat transfer, turbulence and combustion*, Elsevier, 1983, pp. 96–116. [45](#)
- [97] D. C. Wilcox, Reassessment of the scale-determining equation for advanced turbulence models, *AIAA journal* 26 (11) (1988) 1299–1310. [45](#)
- [98] P. Spalart, S. Allmaras, A one-equation turbulence model for aerodynamic flows, in: *30th aerospace sciences meeting and exhibit*, 1992, p. 439. [45](#)
- [99] S. R. Allmaras, F. T. Johnson, Modifications and clarifications for the implementation of the spalart-allmaras turbulence model, in: *Seventh international conference on computational fluid dynamics (ICCFD7)*, Vol. 1902, Big Island, HI, 2012. [45](#), [62](#), [152](#)
- [100] S. Badia, R. Codina, Analysis of a stabilized finite element approximation of the transient convection-diffusion equation using an ale framework, *SIAM Journal on Numerical Analysis* 44 (5) (2006) 2159–2197. [46](#), [62](#), [107](#), [152](#)

- [101] 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. [47](#), [106](#), [107](#), [117](#)
- [102] J. Bruchon, H. Digonnet, T. Coupez, Using a signed distance function for the simulation of metal forming processes: Formulation of the contact condition and mesh adaptation. from a lagrangian approach to an eulerian approach, *International journal for numerical methods in engineering* 78 (8) (2009) 980–1008. [47](#), [62](#), [152](#)
- [103] T. Coupez, E. Hachem, Solution of high-reynolds incompressible flow with stabilized finite element and adaptive anisotropic meshing, *Comput. Methods Appl. Mech. Engrg.* 267 (2013) 65–85. [49](#), [61](#), [62](#), [104](#), [105](#), [151](#), [152](#)
- [104] S. Van der Pijl, A. Segal, C. Vuik, P. Wesseling, A mass-conserving level-set method for modelling of multi-phase flows, *International journal for numerical methods in fluids* 47 (4) (2005) 339–361. [49](#)
- [105] S. V. Patankar, *Numerical heat transfer and fluid flow*, CRC press, 2018. [50](#), [105](#)
- [106] M. R. Khorrami, M. E. Berkman, M. Choudhari, Unsteady flow computations of a slat with a blunt trailing edge, *AIAA J.* 38 (2000) 2050–2058. [56](#)
- [107] C. Rowley, T. Colonius, A. Basu, On self-sustained oscillations in two-dimensional compressible flow over rectangular cavities, *J. Fluid Mech.* 455 (2002) 315–346. [56](#)
- [108] J. Knight, Honey, i shrank the lab, *Nature* 418 (2002) 474–475. [56](#)
- [109] N. Syred, J. M. Beér, Combustion in swirling flows: A review, *Combust. Flame* 23 (1974) 143–201. [56](#)
- [110] M. Gad-el Hak, Modern developments in flow control, *Appl. Mech. Rev.* 49 (7) (1996) 365–379. [56](#)
- [111] J. Lumley, P. Blossey, Control of turbulence, *Annu. Rev. Fluid Mech.* 30 (1998) 311–327.
- [112] A. Glezer, M. Amitay, Synthetic jets, *Annu. Rev. Fluid Mech.* 34 (1) (2002) 503–529.

- [113] S. S. Collis, R. D. Joslin, A. Seifert, V. Theofilis, Issues in active flow control: theory, control, simulation, and experiment, *Prog. Aerosp. Sci.* 40 (2004) 237–289.
- [114] J. Kim, T. R. Bewley, A linear systems approach to flow control, *Annu. Rev. Fluid Mech.* 39 (2007) 383–417.
- [115] A. Seifert, Boundary layer separation control: Experimental perspective and modeling outlook, *Annu. Rev. Fluid Mech.* 50 (2018) null. [56](#)
- [116] A. Jameson, Aerodynamic design via control theory, *Journal of scientific computing* 3 (3) (1988) 233–260. [56](#), [101](#)
- [117] A. Jameson, L. Martinelli, N. A. Pierce, Optimum aerodynamic design using the navier–stokes equations, *Theoretical and computational fluid dynamics* 10 (1) (1998) 213–237. [146](#)
- [118] M. D. Gunzburger, *Perspectives in flow control and optimization*, SIAM, Philadelphia. [56](#), [101](#), [146](#)
- [119] D. Hill, A theoretical approach for analyzing the restabilization of wakes, in: *30th Aerospace Sciences Meeting and Exhibit*, 1992, p. 67. [56](#)
- [120] F. Giannetti, P. Luchini, Structural sensitivity of the first instability of the cylinder wake, *J. Fluid Mech.* 581 (2007) 167–197.
- [121] O. Marquet, D. Sipp, L. Jacquin, Sensitivity analysis and passive control of cylinder flow, *J. Fluid Mech.* 615 (2008) 221–252.
- [122] J. O. Pralits, L. Brandt, F. Giannetti, Instability and sensitivity of the flow around a rotating circular cylinder, *J. Fluid Mech.* 650 (2010) 513–536.
- [123] D. Sipp, O. Marquet, P. Meliga, A. Barbagallo, Dynamics and control of global instabilities in open-flows: a linearized approach, *Applied Mechanics Reviews* 63 (3).
- [124] P. Meliga, E. Boujo, G. Pujals, F. Gallaire, Sensitivity of aerodynamic forces in laminar and turbulent flow past a square cylinder, *Physics of Fluids* 26 (10) (2014) 104101. [56](#), [73](#), [77](#), [78](#), [80](#), [81](#), [90](#), [96](#)
- [125] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, *Advances in neural information processing systems* 25 (2012) 1097–1105. [56](#), [102](#)

- [126] J. Kober, J. A. Bagnell, J. Peters, Reinforcement learning in robotics: A survey, *The International Journal of Robotics Research* 32 (11) (2013) 1238–1274. [56](#), [102](#)
- [127] B. Lusch, J. N. Kutz, S. L. Brunton, Deep learning for universal linear embeddings of nonlinear dynamics, *Nature communications* 9 (1) (2018) 1–10. [56](#), [102](#)
- [128] M. Raissi, A. Yazdani, G. E. Karniadakis, Hidden fluid mechanics: A navier-stokes informed deep learning framework for assimilating flow visualization data, *arXiv preprint arXiv:1808.04327*. [57](#), [102](#)
- [129] A. D. Beck, D. G. Flad, C.-D. Munz, Deep neural networks for data-driven turbulence models, *arXiv preprint arXiv:1806.04482*. [57](#), [102](#)
- [130] Z. Wang, D. Xiao, F. Fang, R. Govindan, C. C. Pain, Y. Guo, Model identification of reduced order fluid dynamics systems using deep learning, *International Journal for Numerical Methods in Fluids* 86 (4) (2018) 255–268. [57](#)
- [131] N. Gautier, J.-L. Aider, T. Duriez, B. Noack, M. Segond, M. Abel, Closed-loop separation control using machine learning, *Journal of Fluid Mechanics* 770 (2015) 442–457. [57](#)
- [132] C. Raibaud, P. Zhong, B. R. Noack, R. J. Martinuzzi, Machine learning strategies applied to the control of a fluidic pinball, *Physics of Fluids* 32 (1) (2020) 015108. [57](#), [63](#), [86](#)
- [133] Y. Lee, H. Yang, Z. Yin, PIV-DCNN: cascaded deep convolutional neural networks for particle image velocimetry, *Exp Fluids* 58 (2017) 171. [57](#)
- [134] J. Rabault, J. Kolaas, A. Jensen, Performing particle image velocimetry using artificial neural networks: a proof-of-concept, *Meas. Sci. Technol.* 28 (2017) 125301.
- [135] S. Cai, J. Liang, Q. Gao, C. Xu, R. Wei, Particle image velocimetry based on a deep learning motion estimator, *IEEE Trans. Instrum. Meas.* 69 (2020) 3538–3554. [57](#)
- [136] S. L. Brunton, B. R. Noack, P. Koumoutsakos, Machine learning for fluid mechanics, *Annual Review of Fluid Mechanics* 52 (2020) 477–508. [57](#), [102](#)
- [137] X. Y. Lee, A. Balu, D. Stoecklein, B. Ganapathysubramanian, S. Sarkar, Flow shape design for microfluidic devices using deep reinforcement learning, *arXiv preprint arXiv:1811.12444*. [57](#)

- [138] J. Viquerat, J. Rabault, A. Kuhnle, H. Ghraieb, A. Larcher, E. Hachem, Direct shape optimization through deep reinforcement learning, *Journal of Computational Physics* 428 (2021) 110080. [57](#), [58](#), [60](#), [102](#), [103](#), [113](#), [147](#), [148](#), [151](#), [152](#)
- [139] P. Ma, Y. Tian, Z. Pan, B. Ren, D. Manocha, Fluid directed rigid body control using deep reinforcement learning, *ACM Transactions on Graphics (TOG)* 37 (4) (2018) 1–11. [57](#), [102](#)
- [140] L. Biferale, F. Bonaccorso, M. Buzzicotti, P. Clark Di Leoni, K. Gustavsson, Zermelo’s problem: Optimal point-to-point navigation in 2d turbulent flows using reinforcement learning, *Chaos: An Interdisciplinary Journal of Nonlinear Science* 29 (10) (2019) 103138.
- [141] F. Ren, H. Hu, H. Tang, Active flow control using machine learning: A brief review, *J. Hydrodynam.* 32 (2020) 247–253. [57](#), [102](#), [147](#)
- [142] F. Ren, J. Rabault, H. Tang, Applying deep reinforcement learning to active flow control in turbulent conditions, arXiv preprint arXiv:2006.10683. [57](#)
- [143] Y. Wang, H. He, X. Tan, Y. Gan, Trust region-guided proximal policy optimization, arXiv preprint arXiv:1901.10314. [60](#), [113](#)
- [144] T. J. Hughes, G. R. Feijóo, L. Mazzei, J.-B. Quincy, The variational multiscale method—a paradigm for computational mechanics, *Computer methods in applied mechanics and engineering* 166 (1-2) (1998) 3–24. [61](#), [106](#), [152](#)
- [145] Y. Bazilevs, V. M. Calo, J. A. Cottrell, T. J. R. Hughes, A. Reali, G. Scovazzi, Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows, *Comput. Methods Appl. Mech. Engrg.* 197 (2007) 173–201. [61](#), [106](#), [152](#)
- [146] R. Codina, Comparison of some finite element methods for solving the diffusion-convection-reaction equation, *Comput. Methods Appl. Mech. Engrg.* 156 (1998) 185–210. [62](#), [107](#), [152](#)
- [147] E. Hachem, B. Rivaux, T. Kloczko, H. Digonnet, T. Coupez, Stabilized finite element method for incompressible flows with high Reynolds number, *J. Comput. Phys.* 229 (23) (2010) 8643–8665. [62](#), [106](#), [153](#)
- [148] T. Coupez, G. Jannoun, N. Nassif, H. C. Nguyen, H. Digonnet, E. Hachem, Adaptive time-step with anisotropic meshing for incompressible flows, *J. Comput. Phys.* 241 (2013) 195–211. [62](#), [153](#)

- [149] 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* 22 (2012) 718–741. [62](#), [107](#), [153](#)
- [150] E. Hachem, S. Feghali, R. Codina, T. Coupez, Immersed stress method for fluid-structure interaction using anisotropic mesh adaptation, *Int. J. Numer. Meth. Eng.* 94 (2013) 805–825. [62](#), [107](#), [153](#)
- [151] W. Rodi, Comparison of LES and RANS calculations of the flow around bluff bodies, *J. Wind Eng. Ind. Aerodyn.* 69–71 (1997) 55–75. [63](#)
- [152] V. John, Parallele Lösung der inkompressiblen Navier–Stokes Gleichungen auf adaptiv verfeinerten Gittern, Otto-von-Guericke-Universität Magdeburg, Fakultät für Mathematik. [64](#), [155](#)
- [153] V. John, Reference values for drag and lift of a two-dimensional time-dependent flow around a cylinder, *International Journal for Numerical Methods in Fluids* 44 (7) (2004) 777–788. [64](#), [155](#)
- [154] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, Stable baselines, <https://github.com/hill-a/stable-baselines> (2018). [64](#), [115](#)
- [155] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba, Openai gym (2016). [arXiv:arXiv:1606.01540](https://arxiv.org/abs/1606.01540). [64](#), [115](#)
- [156] S. Mittal, V. Kumar, A. Raghuvanshi, Unsteady incompressible flows past two cylinders in tandem and staggered arrangements, *Int. J. Numer. Meth. Fl.* 25 (1997) 1315–1344. [69](#)
- [157] J. R. Meneghini, F. Saltara, C. L. R. Siqueira, J. A. Ferrari, Numerical simulation of flow interference between two circular cylinders in tandem and side-by-side arrangements, *J. Fluids Struct.* 15 (2001) 327–350.
- [158] B. Sharman, F.-S. Lien, L. Davidson, C. Norberg, Numerical predictions of low reynolds number flows over two tandem circular cylinders, *Int. J. Numer. Meth. Fl.* 47 (2005) 423–447.
- [159] K. Lee, K.-S. Yang, D.-H. Yoon, Flow-induced forces on two circular cylinders in proximity, *Comput. Fluids* 38 (2009) 111–120. [69](#)
- [160] X. Mao, Sensitivity of forces to wall transpiration in flow past an aerofoil, *Proc. R. Soc. A* 471 (2015) 20150618. [73](#)

- [161] P. Meliga, E. Boujo, M. Meldi, F. Gallaire, Revisiting the drag reduction problem using adjoint-based distributed forcing of laminar and turbulent flows over a circular cylinder, *Eur. J. Mech. B-Fluid* 72 (2018, accepted) 123–134. [73](#)
- [162] B. Fornberg, A numerical study of steady viscous flow past a circular cylinder, *J. Fluid Mech.* 98 (1980) 819–855. [74](#)
- [163] R. D. Henderson, Details of the drag curve near the onset of vortex shedding, *Phys. Fluids* 7 (1995) 2102–2104. [74](#), [76](#)
- [164] P. Meliga, Computing the sensitivity of drag and lift in flow past a circular cylinder: time-stepping vs. self-consistent analysis, *Phys. Rev. Fluids* 2 (2017) 073905. [76](#), [79](#), [80](#)
- [165] H. Sakamoto, H. Haniu, Optimum suppression of fluid forces acting on a circular cylinder, *J. Fluids Eng.* 116 (1994) 221–227. [78](#), [79](#)
- [166] F. Pereira, G. Vaz, L. Eça, Flow past a circular cylinder: a comparison between rans and hybrid turbulence models for a low Reynolds number, *OMAE 2015-41235*. [77](#)
- [167] G. Iaccarino, A. Ooi, P. A. Durbin, M. Behnia, Reynolds averaged simulation of unsteady separated flow, *Int. J. Heat Fluid Flow* 24 (2003) 147–156. [81](#)
- [168] W. Rodi, J. H. Ferziger, M. Breuer, M. Pourquie, Status of large-eddy simulation: Results of a workshop, *J. Fluids Eng.* 119 (1997) 248–262. [81](#)
- [169] C. Raibaud, P. Zhong, R. J. Martinuzzi, B. R. Noack, Open and closed-loop control of a triangular bluff body using rotating cylinders, *IFAC-PapersOnLine* 50 (2017) 12291–12295. [83](#)
- [170] S. Kang, H. Choi, S. Lee, Laminar flow past a rotating circular cylinder, *Phys. Fluids* 11 (1999) 3312–3321. [87](#)
- [171] Y. Deng, L. Pastur, M. Morzyński, B. R. Noack, Route to chaos in the fluidic pinball, *Procs. of the ASME 2018 5th Joint US-European Fluids Engineering Division Summer Meeting*. [92](#)
- [172] A. Griewank, A. Walther, An implementation of checkpointing for the reverse or adjoint mode of computational differentiation, *ACM T. Math. Software* 26 (2000) 19–45. [95](#)
- [173] C. Tsitouras, Runge–Kutta pairs of order 5(4) satisfying only the first column simplifying assumption, *Comput. Math. with Appl.* 62 (2011) 770–775. [95](#)

- [174] E. Arian, M. D. Salas, Admitting the inadmissible: adjoint formulation for incomplete cost functionals in aerodynamic optimization, *AIAA J.* 37 (1999) 37–44. [95](#)
- [175] D. J. Lea, M. Allen, T. W. N. Haines, Sensitivity analysis of the climate of a chaotic system, *Tellus A* 52 (2000) 523–532. [95](#)
- [176] Q. Wang, Forward and adjoint sensitivity computation of chaotic dynamical systems, *J. Comput. Phys.* 235 (2013) 1–13. [95](#)
- [177] N. Chandramoorthy, Z.-N. Wang, Q. Wang, P. Tucker, Toward computing sensitivities of average quantities in turbulent flows, arXiv preprint arXiv:1902.11112. [95](#)
- [178] T. Barth, On the role of error and uncertainty in the numerical simulation of complex fluid flows, presented at the 2010 SIAM Annual Meeting, SIAM, Philadelphia. [96](#)
- [179] M. Nazarov, J. Hoffman, On the stability of the dual problem for high Reynolds number flow past a circular cylinder in two dimensions, *SIAM J. Sci. Comput.* 34 (2012) 1905–1924. [96](#)
- [180] Q. Wang, J.-H. Gao, The drag-adjoint field of a circular cylinder wake at reynolds numbers 20, 100 and 500, *J. Fluid Mech.* 730 (2013) 145–161. [96](#)
- [181] J. Hoffman, Computation of mean drag for bluff body problems using adaptive DNS/LES, *SIAM J. Sci. Comput.* 27 (2005) 184–207. [96](#)
- [182] J. Hoffman, Adaptive simulation of the turbulent flow past a sphere, *J. Fluid Mech.* 568 (2006) 77–88.
- [183] N. Jansson, J. Hoffman, M. Nazarov, Adaptive simulation of turbulent flow past a full car model, Proceedings of the SC '11, ACM International Conference for High Performance Computing, Networking, Storage and Analysis (2011) 20:1–20:8. [96](#)
- [184] 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. [99](#), [148](#)
- [185] T. Bewley, Flow control : new challenges for a new renaissance, *Prog. Aerosp. Sci.* 37 (2001) 21–58. [101](#)

- [186] K. Momose, K. Abe, H. Kimoto, Reverse computation of forced convection heat transfer for optimal control of thermal boundary conditions, *Heat Tran. Asian Res.* 33 (2004) 161–174. [101](#)
- [187] A. Belmiloudi, Robin-type boundary control problems for the nonlinear Boussinesq type equations, *J. Math. Anal. Appl.* 273 (2002) 428–456. [101](#)
- [188] G. Bärwolff, M. Hinze, Optimization of semiconductor melts, *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik: Applied Mathematics and Mechanics* 86 (6) (2006) 423–437.
- [189] J. Boldrini, E. Fernández-Cara, M. Rojas-Medar, An optimal control problem for a generalized Boussinesq model: The time dependent case, *Rev. Mat. Comput.* 20 (2007) 339–366.
- [190] H. Karkaba, T. Dbouk, C. Habchi, S. Russeil, T. Lemenand, D. Bougeard, Multi objective optimization of vortex generators for heat transfer enhancement using large design space exploration, *Chem. Eng. Process.* [101](#)
- [191] P. Meliga, J.-M. Chomaz, Global modes in a confined impinging jet: application to heat transfer and control, *Theor. Comput. Fluid Dyn.* 25 (1) (2011) 179–193. [101](#)
- [192] K. Lee, S.-A. Kim, J. Choi, S.-W. Lee, Deep reinforcement learning in continuous action spaces: a case study in the game of simulated curling, in: *International conference on machine learning*, PMLR, 2018, pp. 2937–2946. [102](#), [118](#)
- [193] H. Kazmi, F. Mehmood, S. Lodeweyckx, J. Driesen, Gigawatt-hour scale savings on a budget of zero: Deep reinforcement learning based optimal control of hot water systems, *Energy* 144 (2018) 159–168. [102](#)
- [194] T. Zhang, J. Luo, P. Chen, J. Liu, Flow rate control in smart district heating systems using deep reinforcement learning, *arXiv preprint arXiv:1912.05313*. [102](#)
- [195] S. V. Patankar, A numerical method for conduction in composite materials, flow in irregular geometries and conjugate heat transfer, in: *Procs. of the 6th International Heat Transfer Conference*, 1978, pp. 297–302. [105](#)
- [196] R. Codina, Stabilized finite element approximation of transient incompressible flows using orthogonal subscales, *Comput. Methods Appl. Mech. Engrg.* 191 (2002) 4295–4321. [107](#)

- [197] A. N. Brooks, T. J. R. Hughes, Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations, *Comput. Methods Appl. Mech. Engrg.* 32 (1982) 199–259. [107](#)
- [198] A. C. Galeão, E. G. D. Do Carmo, A consistent approximate upwind Petrov-Galerkin method for convection-dominated problems, *Comput. Methods Appl. Mech. Engrg.* 68 (1988) 83–95. [107](#)
- [199] E. Hachem, G. Jannoun, J. Veysset, M. Henri, R. Pierrot, I. Poitroult, E. Massoni, T. Coupez, Modeling of heat transfer and turbulent flows inside industrial furnaces, *Simul. Model. Pract. Th.* 30 (2013) 35–53. [107](#)
- [200] A. Kakade, A natural policy gradient, *Adv. Neural Inf. Process Syst.* 14 (2001) 1531–1538. [112](#)
- [201] G. de Vahl Davis, I. Jones, Natural convection in a square cavity: a comparison exercise, *Int. J. Numer. Methods Fluids* 3 (1983) 227–248. [118](#)
- [202] H. Dixit, V. Babu, Simulation of high Rayleigh number natural convection in a square cavity using the lattice Boltzmann method, *Int. J. Heat Mass Transfer* 49 (2006) 727–739. [118](#)
- [203] N. Markatos, K. Pericleous, Laminar and turbulent natural convection in an enclosed cavity, *Int. J. Heat Mass Transfer* 27 (1984) 772–775. [118](#)
- [204] G. Barakos, E. Mitsoulis, Natural convection flow in a square cavity revisited: laminar and turbulent models with wall functions, *Int. J. Numer. Methods Fluids* 18 (1994) 695–719. [118](#)
- [205] K. Khanafer, K. Vafai, M. Lightstone, Buoyancy-driven heat transfer enhancement in a two-dimensional enclosure utilizing nanofluids, *Int. J. Heat Mass Transfer* 46 (2003) 3639–3653. [118](#)
- [206] A. Lazaric, M. Restelli, A. Bonarini, Reinforcement learning in continuous action spaces through sequential Monte Carlo methods, in: *Procs. of the 35th International Conference on Machine Learning*, 2018, pp. 4587–4596. [118](#)
- [207] P. Meliga, E. Hachem, Time-accurate calculation and bifurcation analysis of the incompressible flow over a square cavity using variational multiscale modeling, *J. Comput. Phys.* 376 (2019) 952–972. [142](#)
- [208] P. Gangl, U. Langer, A. Laurain, H. Meftahi, K. Sturm, Shape optimization of an electric motor subject to nonlinear magnetostatics, *SIAM J. Sci. Comput.* 37 (2015) B1002–B1025. [146](#)

- [209] R. Udawalpola, M. Berggren, Optimization of an acoustic horn with respect to efficiency and directivity, *Int. J. Numer. Methods Eng.* 73 (2008) 1571–1606. [146](#)
- [210] M. Hintermüller, W. Ring, A second order shape optimization approach for image segmentation, *SIAM J. Appl. Math.* 64 (2004) 442–467. [146](#)
- [211] J. Pinzon, M. Siebenborn, A. Vogel, Parallel 3d shape optimization for cellular composites on large distributed-memory clusters, *Adv. Model. Simul. Eng. Sci.* 7 (2020) 117–135. [146](#)
- [212] P.-I. Schneider, X. G. Santiago, V. Soltwisch, M. Hammerschmidt, S. Burger, C. Rockstuhl, Benchmarking five global optimization approaches for nano-optical shape optimization and parameter reconstruction, *arXiv preprint arXiv:1809.06674*. [146](#)
- [213] O. Pironneau, On optimum profiles in stokes flow, *J. Fluid Mech.* 59 (1973) 117–128. [146](#)
- [214] O. Pironneau, On optimum design in fluid mechanics, *J. Fluid Mech.* 64 (1974) 97–110. [146](#)
- [215] A. L. Marsden, M. Wang, B. Mohammadi, P. Moin, Shape optimization for aerodynamic noise control, *Center for Turbulence Research Annual Brief* (2001). [146](#)
- [216] I. Rodriguez-Eguia, I. Errasti, U. Fernandez-Gamiz, J. M. Blanco, E. Zulueta, A. Saenz-Aguirre, A parametric study of trailing edge flap implementation on three different airfoils through an artificial neuronal network, *Symmetry* 12 (2020) 828. [146](#)
- [217] J. H. Holland, Genetic algorithms, *Sci. Am.* 267 (1992) 66–73. [146](#)
- [218] J. Kennedy, R. Eberhart, Particle swarm optimization, in: *Procs. of ICNN'95-international conference on neural networks, 1995*, pp. 1942–1948. [146](#)
- [219] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, E. Teller, Equation of state calculations by fast computing machines, *J. Chem. Phys* 21 (1953) 1087–1092. [146](#)
- [220] Z. Han, C. Xu, L. Zhang, Y. Zhang, K. Zhang, S. Wenping, Efficient aerodynamic shape optimization using variable-fidelity surrogate models and multi-level computational grids, *Chinese J. Aeronaut.* 33 (2020) 31–47. [146](#)

- [221] O. Chernukhin, D. W. Zingg, Multimodality and global optimization in aerodynamic design, *AIAA journal* 51 (2013) 1342–1354. [147](#)
- [222] A. Bewley, J. Rigley, Y. Liu, J. Hawke, R. Shen, V.-D. Lam, A. Kendall, Learning to drive from simulation without real world labels, arXiv preprint arXiv:1812.03823. [147](#)
- [223] J. Viquerat, P. Meliga, E. Hachem, A review on deep reinforcement learning for fluid mechanics: an update, arXiv preprint arXiv:2107.12206. [147](#)
- [224] M. Holm, Using deep reinforcement learning for active flow control, Ph.D. thesis, Master Thesis University of Oslo (2020). [147](#), [184](#)
- [225] J. Rabault, F. Ren, W. Zhang, H. Tang, H. Xu, Deep reinforcement learning in fluid mechanics: a promising method for both active flow control and shape optimization, *J. Hydrodynam.* 32 (2020) 234–246.
- [226] S. Qin, S. Wang, G. Sun, An application of data driven reward of deep reinforcement learning by dynamic mode decomposition in active flow control, arXiv preprint arXiv:arXiv:2106.06176. [147](#)
- [227] X. Hui, H. Wang, W. Li, J. Bai, F. Qin, G. He, Multi-object aerodynamic design optimization using deep reinforcement learning, *AIP Advances* 11 (8) (2021) 085311. [147](#)
- [228] R. Rebonato, P. Jäckel, The most general methodology to create a valid correlation matrix for risk management and option pricing purposes, Available at SSRN 1969689 135. [151](#), [155](#)
- [229] D. P. Kingma, J. L. Ba, Adam: A method for stochastic optimization, 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings (2015) 1–15. [151](#)
- [230] J. Viquerat, E. Hachem, A supervised neural network for drag prediction of arbitrary 2d shapes in laminar flows at low reynolds number, *Comput. Fluids* 210 (2020) 104645. [152](#)
- [231] M. Sussman, P. Smereka, S. Osher, A level set approach for computing solutions to incompressible two-phase flow, *J. Comput. Phys.* 114 (1994) 146–159. [155](#)
- [232] N. Hansen, The CMA Evolution Strategy: a tutorial, arXiv preprint arXiv:1604.00772. [155](#)

- [233] T. Kondoh, T. Matsumori, A. Kawamoto, Drag minimization and lift maximization in laminar flows via topology optimization employing simple objective function expressions based on body force integration, *Structural and Multidisciplinary Optimization* 45 (5) (2012) 693–701. [158](#), [161](#)
- [234] S. Richardson, Optimum profiles in two-dimensional Stokes flow, *Proc. R. Soc. A* 450 (1995) 603–622. [159](#)
- [235] J.-Y. Andro, G. Dergham, R. Godoy-Diana, L. Jacquin, D. Sipp, Conditions critiques de déclenchement du lâcher tourbillonnaire au cours du vol des insectes, in: *Procs. of the 19ème Congrès Français de Mécanique*, 2009. [170](#)
- [236] S. Sunada, T. Yasuda, K. Yasuda, K. Kawachi, Comparison of wing characteristics at an ultralow Reynolds number, *J. Aircr.* 39 (2002) 331–338. [171](#)
- [237] D. Funda Kurtulus, Vortex flow aerodynamics behind a symmetric airfoil at low angles of attack and reynolds numbers, *Int. J. Micro Air Veh.* 13 (2021) 17568293211055653. [171](#)
- [238] S. Wang, Y. Zhou, M. Mahbub Alam, H. Yang, Turbulent intensity and reynolds number effects on an airfoil at low reynolds numbers, *Phys. Fluids* 26 (2014) 115107. [171](#)
- [239] M. Breuer, Large eddy simulation of the subcritical flow past a circular cylinder: Numerical and modeling aspects, *Int. J. Numer. Meth. Fl.* 28 (1998) 1281–1302. [171](#)
- [240] B. Kulfan, J. Bussolletti, “Fundamental” parameteric geometry representations for aircraft component shapes, in: *Procs. of the 11th AIAA/ISSMO multidisciplinary analysis and optimization conference*, 2006, p. 6948. [172](#)
- [241] K. Zhang, S. Hayostek, M. Amitay, W. He, V. Theofilis, K. Taira, On the formation of three-dimensional separated flows over wings under tip effects, *J. Fluid Mech.* 895. [173](#)
- [242] A. Barbagallo, D. Sipp, P. J. Schmid, Closed-loop control of an open cavity flow using reduced-order models, *Journal of Fluid Mechanics* 641 (2009) 1–50. [183](#)
- [243] M. Schäfer, S. Turek, F. Durst, E. Krause, R. Rannacher, Benchmark computations of laminar flow around a cylinder, in: *Flow simulation with high-performance computers II*, Springer, 1996, pp. 547–566. [194](#), [195](#)
- [244] B. Protas, J. E. Wesfreid, Drag force in the open-loop control of the cylinder wake in the laminar regime, *Physics of Fluids* 14 (2) (2002) 810–826. [202](#)

- [245] B. Protas, A. Styczek, Optimal rotary control of the cylinder wake in the laminar regime, *Physics of Fluids* 14 (7) (2002) 2073–2087. [202](#)

RÉSUMÉ

Cette thèse évalue la pertinence des techniques d'apprentissage par renforcement profond (DRL) pour le contrôle optimal en mécanique des fluides. L'apprentissage par renforcement (RL) est le processus par lequel un agent apprend par essai et erreur les actions à prendre de façon à optimiser une récompense quantitative au cours du temps. Dans un contexte d'apprentissage par renforcement profond (deep RL ou DRL), l'agent est un réseau de neurones profond imitant les circuits formés par les neurones du cerveau humain. Le couplage entre les algorithmes de DRL et les codes de mécanique des fluides numérique (CFD) à la pointe de l'état de l'art, ainsi que leur implémentation dans un contexte de calcul haute performance, constituent les nouveautés et l'objectif principal de la thèse. L'environnement CFD utilisé pour calculer la récompense fournie au DRL est basé sur la méthode des éléments finis stabilisés multi-échelles de type Variational Multiscale (VMS), dans laquelle la solution est décomposée a priori en une grande échelle résolue et une petite échelle modélisée au travers de termes sources proportionnels aux résidus des équations du problème grande échelle. En ce qui concerne les algorithmes DRL, deux approches différentes sont considérées. La première, dans laquelle l'agent interagit avec son environnement une fois par épisode dans le but d'apprendre le mapping d'un état d'entrée constant à une action optimale (single-step DRL), vise les problèmes de contrôle en boucle ouverte, dans lesquels une quantité est optimisée via des paramètres d'actuation pré-définis (par exemple, une vitesse d'entrée constante). La seconde, dans laquelle l'agent interagit plusieurs fois par épisode afin d'apprendre une relation état-action plus complexe (multi-step DRL), est plus pertinente pour les problèmes de contrôle en boucle fermée, où des mesures de l'écoulement sont utilisées afin d'ajuster en permanence les paramètres d'actuation. Plusieurs cas-tests en deux et trois dimensions (en régime d'écoulement laminaire et turbulent) sont présentés afin d'évaluer la pertinence, la précision et les performances de ces méthodes, en particulier pour les problèmes de réduction de traînée et de contrôle thermique. Les résultats obtenus soulignent le potentiel élevé de l'approche DRL-CFD devraient permettre d'accélérer le développement du DRL et son application à des problématiques concrètes d'intérêt industriel.

MOTS CLÉS

Apprentissage par Renforcement Profond; Réseaux de neurones; Mécanique des fluides numérique; Éléments finis stabilisés VMS; Contrôle d'écoulements; Réduction de traînée; Contrôle thermique.

ABSTRACT

This thesis gauges the relevance of deep reinforcement learning (DRL) techniques for the optimal control of fluid mechanical systems. Reinforcement learning (RL) is the process by which an agent learns by trial and error interactions with its environment the succession of actions maximizing its cumulative reward over time. In a deep reinforcement learning context (deep RL or DRL), the agent is a deep neural network based on the neural circuits formed by neurons in the human brain. The coupling between state-of-the-art DRL algorithms and computational fluid dynamics (CFD) solvers and their implementation in a high performance computing context make for the novelties and main objective of the thesis. The CFD resolution framework used to compute the reward provided to the DRL agent relies on the Variational Multiscale (VMS) stabilized finite element method. The latter introduces an a priori decomposition of the numerical solution into large and small-scale components, the general picture being that only the large scales are resolved at the discrete level, while the effect of the small scales is modeled after consistently derived source terms proportional to the residual of the large scale solution. Regarding the DRL algorithms, two different frameworks are considered. The first one has the agent interact only once per episode with its environment to learn the mapping from a constant input state to an optimal action (hence, single-step episodes, and by extension, single-step DRL), and is thus relevant to open-loop control, where a desired output is optimized under pre-determined actuation parameters (for instance, a constant inlet velocity). The second one has the agent interact multiple times per episode to learn a more complex state-action relation (hence, multi-step DRL) and is more relevant to closed-loop control, where the output is optimized by continuously adjusting the design parameters to flow measurements. Several test-cases in two and three dimensions (both in laminar and turbulent flow regimes) are successfully tackled and presented to assess the relevance, accuracy and performance of the proposed methodologies, with particular emphasis put on drag reduction and thermal control applications. The obtained results emphasize the high potential of the DRL-CFD framework, and are expected to contribute to further progress towards improved and faster design and control of industrial fluid mechanical systems.

KEYWORDS

Deep Reinforcement Learning; Neural networks; Computational fluid dynamics; VMS stabilized finite elements; Flow control; Drag reduction; Thermal control.