Gradient-enriched machine learning control exemplified for shear flows in simulations and experiments

Guy Y. Cornejo Maceda

Thèse de doctorat

NNT: 2021UPAST036

université PARIS-SACLAY

Gradient-enriched machine learning control exemplified for shear flows in simulations and experiments

Contrôle par apprentissage automatique et méthodes de gradient appliqué aux écoulements cisaillés numériques et expérimentaux

Thèse de doctorat de l'université Paris-Saclay

École doctorale n° 579, Sciences mécaniques et énergétiques, matériaux et géosciences (SMEMAG)
Spécialité de doctorat: mécanique des fluides
Unité de recherche: Université Paris-Saclay, CNRS, Laboratoire Interdisciplinaire des Sciences du Numérique, 91400, Orsay, France.
Référent: Faculté des sciences d'Orsay

Thèse présentée et soutenue à Paris-Saclay, le 17 Mars 2021, par

Guy Y. CORNEJO MACEDA

Composition du jury:

Marc SCHOENAUER Directeur de Recherche, INRIA	Président
Laurent KEIRSBULCK Professeur, UVHC	Rapporteur & Examinateur
Christian Oliver PASCHEREIT Professeur, TU Berlin	Rapporteur & Examinateur
Shervin BAGHERI Professeur, KTH	Examinateur
Nathan KUTZ Professeur, UW	Examinateur
Bérengère PODVIN Chargée de Recherches HDR, LISN–CNRS	Examinatrice

Direction de la thèse:

Bernd R. NOACK Professeur, HIT	Directeur de thèse
François LUSSEYRAN Directeur de Recherche, LISN–CNRS	Co-Directeur de thèse

Dedicado a mis queridos padres,
.....

Deep in the human unconscious is a pervasive need for a logical universe that makes sense. But the real universe is always one step beyond logic.
– From 'The Sayings of Muad'Dib'
By the Princess Irulan
Frank Herbert.

Acknowledgments

First and foremost, I would like to express my sincere appreciation to my two PhD advisors B. R. Noack and F. Lusseyran for accompanying me throughout those years. I thank them deeply for their trust they put in my abilities and me. I would like also to thank them for sharing their enthusiasm and passion for research which has always been inspiring to me. They knew how to find the words to guide me and always push me forward in the good and the bad times. Working with them has been one of the greatest privileges of my PhD. I am sincerely grateful for the many opportunities they have offered me. Needless to say that without them, none of this work would have been possible without them.

I would like to express my deepest gratitude to the members of the jury: Prof. C. O. Paschereit, Prof. L. Keirsbulck, Prof. M. Schoenauer, Prof. B. Podvin, Prof. N. Kutz and Prof. S. Bagheri for accepting to be part of my thesis committee and for their interest in my work.

I am deeply indebted to M. Morzynski for his long-distance support and without whom none of this work would have been possible.

Special thanks to L. Pastur and N. Deng for whom I have only admiration and friendship.

I am also grateful to K. Taira for receiving me in his research team at UCLA during one month. His dedication has been truly inspiring.

I also thank all of our co-authors and expecially Y. Li for her hard work and insightful suggestions.

I cordially thank E. Varon, L. Mathelin, O. Semeraro, S. Caillou, P. Y. Passaggia and all the current and former members of the FlowCon project for the insightful exchanges during our meetings. I would also like to thank all our colleagues, fellow PhD students, interns for making LIMSI such a great place to be. Thanks also to all the fellow PhD students and researchers I met during the conferences for their interest in my work and their friendliness.

I am also grateful to my friends for their continuous moral support throughout the years.

I would also like to thank all the people I don't know I have to thank but who have secretly contributed to the success of this work.

I would like also to thank my family, my aunts and uncles, my cousins, my nephews, my niece and her parents for accompanying me morally and spiritually in this adventure. And finally I thank my parents for their unconditional support and for believing in me and to whom I dedicate these pages.

Résumé étendu

Chapitre 1: Introduction

Ce chapitre commence par évoquer les enjeux économiques et environnementaux du contrôle en mécanique des fluides. Nous rappelons notamment que les engagements européens en matière de réduction des émissions de dioxyde de carbone se traduisent notamment par le développement des énergies renouvelables et l'amélioration de l'efficacité énergétique de l'industrie du transport. Autant de domaines où le contrôle d'écoulement est à la fois un challenge et une opportunité pour les réductions de coûts énergétiques.

Nous rappelons les grandes lignes de la discipline qui sont: l'aérodynamisme ou l'optimisation des formes, le contrôle passif par la modification locale de la géométrie et bien sûr le contrôle actif qui nécessite l'injection en temps réel d'énergie dans l'écoulement. C'est grâce à sa flexibilité et à la possibilité de produire un contrôle qui dépend de l'état du système que le contrôle actif permet, a priori, les meilleures performances de contrôle. Mais le contrôle actif des fluides doit surmonter les obstacles qui sont: la très grande dimension des systèmes fluides qui rendent toutes simulations numériques coûteuses, les retards entre l'action, la mesure de l'état et la réponse à l'action sans oublier les interactions non-linéaires entre modes. Les deux derniers étant en général incompatibles avec une description linéaire de l'écoulement. C'est ce qui nous oriente vers un développement des approches sans modèles qui permettent de conserver toute la richesse des non-linéarités de l'écoulement. On considère alors le système sous contrôle comme une boîte noire avec des entrées (la commande de l'action) et des sorties (les données des capteurs). Le problème de contrôle consiste alors à déterminer la relation fonctionnelle qui lie les entrées et les sorties: la loi de contrôle. La reformulation de ce problème sous forme de problème d'optimisation fonctionnelle permet alors d'employer de puissant algorithme d'apprentissage automatique (machine learning) pour le résoudre. On s'intéresse notamment à la programmation génétique GP pour l'optimisation des lois de contrôle. On rappelle que la programmation génétique est employée avec succès depuis plusieurs années pour résoudre des problèmes de contrôle à la fois numérique et expérimental.

L'objectif principal de la thèse est alors d'accélérer le processus d'apprentissage de l'algorithme de programmation génétique à fin de pouvoir explorer des problèmes de contrôle de plus en plus complexes, comprenant, par exemple, un grand nombre d'actionneurs et de capteurs et pour lesquels les temps d'apprentissage actuels sont souvent rédhibitoires. Dans cet objectif, nous avons développé deux codes de machine learning, le `xMLC` basé sur les algorithmes existants de programmation génétique et le `gMLC`, basé sur un nouvel algorithme que nous proposons et qui combine programmation génétique et méthodes de gradients pour aboutir à un apprentissage accéléré. Nos algorithmes testés en simulations numériques sur le pinball fluidique et en expérience sur la cavité ouverte, sont tous les deux disponibles en ligne.

Partie 1: Contrôle de sillage par programmation génétique

Chapitre 2: L'algorithme de programmation génétique pour l'apprentissage de lois de contrôle (LGPC)

Dans ce chapitre, nous décrivons l'algorithme de programmation génétique linéaire dans le contexte de résolution des problèmes de contrôle (LGPC). LGPC est une méthode d'optimisation fonctionnelle basé sur le processus biologique d'évolution des espèces. L'idée principale étant l'essaie successif de solutions candidates à un problème. Dans ce contexte, ces solutions candidates sont aussi appelées individus ou encore lois de contrôle. Les principaux mécanismes biologiques dont s'inspire la méthode de programmation génétique sont: la sélection naturelle, la recombinaison aléatoire des individus (crossover) et la mutation des individus, également aléatoire. Les opérations de crossover et mutation sont les forces principales d'évolution des solutions candidates et permettent de construire des solutions de plus en plus performantes. L'algorithme de programmation génétique linéaire est alors un processus itératif et stochastique qui imite l'évolution naturelle des espèces en combinant et modifiant des solutions candidates à un problème au fur et à mesure des générations. D'un point de vue algorithmique, la représentation interne des solutions candidates se fait grâce à une matrice d'instruction qui «code» une expression mathématique. Ces expressions mathématiques sont en particulier construites à partir d'une bibliothèque donnée de fonctions de bases et de données du problème telles que l'état du système au cours du temps.

Dans une seconde partie, nous nous intéressons à la stabilisation d'un oscillateur amorti, dit de Landau, pour effectuer une étude paramétrique de LGPC. L'oscillateur de Landau est un système de deux équations différentielles ordinaires présentant un point fixe et un cycle limite. Le problème de contrôle est alors de ramener le système sur son point fixe en ajoutant un terme de forçage à la deuxième équation du système. La fonction de coût associée comprend à la fois la distance au point fixe et un terme de pénalisation de l'action. Un contrôle linéaire, calculé grâce à la fonction `fminsearch` de MATLAB, montre que le problème peut être résolu par un retour linéaire de l'état du système. Le but LGPC est donc de construire une loi de contrôle basée sur le retour d'état et des fonctions de bases définies par l'utilisateur, qui performerait mieux que la loi linéaire. Nous testons, notamment l'impact des probabilités de crossover, mutation et réplication sur la solution finale. Pour cela, nous effectuons 100 réalisations pour chaque combinaison de paramètres. Les intervalles des probabilités des opérateurs génétiques sont discrétisés par pas de 0.1 résultant en 66 combinaisons à tester. On montre alors que les opérations seules de crossover et mutation ne sont pas les combinaisons les plus performantes et que c'est bien la combinaison des deux qui permet les meilleurs résultats. Aussi, de manière surprenante, l'opération de réplication semble être nécessaire en faible proportion pour garantir une bonne optimisation et ce même si le problème ne présente, a priori, que peu de minima.

Nous étudions aussi le rôle de la taille de la population, le nombre d'instructions et le choix des fonctions de bases. Des tests exhaustifs montrent que les config-

urations qui permettent le plus d'étapes d'évolutions donnent les meilleurs résultats. On montre aussi que le nombre d'individus à évaluer croît avec le nombre de fonctions de bases. Finalement, parmi toutes les simulations effectuées, nous déterminons deux solutions particulières: une loi de contrôle linéaire plus performante que celle définie par `fminsearch`, révélant qu'il y a plus d'un minimum dans le sous-espace des lois de contrôle linéaire; et une loi non-linéaire surclassant toutes les lois apprises et qui agit uniquement lorsque c'est nécessaire. Ces études paramétriques ont permis de comprendre le rôle de chaque paramètre et de révéler des règles empiriques pour le choix des paramètres de LGPC. Ces règles sont notamment utilisées dans toute cette étude.

Enfin, l'ensemble des simulations ont été réalisées grâce à notre propre code développé sur le langage MATLAB et GNU Octave et librement accessible.

Chapitre 3: Réduction de traînée du pinball fluidic par LGPC

Ce chapitre s'appuie sur l'étude des méta-paramètres de LGPC du chapitre précédent pour réduire la puissance nette de traînée sur le problème 2D du pinball fluidique. Le pinball fluidique est un système fluide comprenant trois cylindres disposés aux sommets d'un triangle équilatéral pointant vers l'amont de l'écoulement. Malgré sa géométrie simple, le pinball fluidique conserve des caractéristiques présentes dans les écoulements réels telles que des bifurcations successives et des interactions non-linéaires entre modes. En effet, la configuration du pinball passe premièrement par une bifurcation de Hopf, responsable d'un lâché tourbillonnaire de type von Kármán, puis une bifurcation de type fourche qui brise la symétrie de l'écoulement et provoque une déflexion du jet à l'arrière des cylindres en aval de l'écoulement. C'est cet écoulement statistiquement asymétrique et périodique que nous souhaitons contrôler grâce à la méthode de programmation génétique.

L'action sur l'écoulement se fait par la rotation indépendante des cylindres et l'objectif visé est la réduction de la puissance nette de traînée. Pour ce faire, on minimise une fonction de coût qui comprend la puissance de traînée et un terme de pénalisation de la puissance de l'action. Une étude de contrôle en boucle ouverte explorant l'espace des lois de contrôle symétrique révèle que la meilleure stratégie pour réduire la traînée est la configuration dite de *boat tailing*. Dans cette configuration, le cylindre frontale est immobile et les deux cylindres arrières aspire l'écoulement à contre-courant.

LGPC est alors employé pour réduire la puissance nette de traînée du pinball en explorant trois types de contrôle:

- **forçage multi-fréquence:** on autorise l'emploi de fonction périodiques pour construire les lois de contrôle;
- **contrôle en boucle fermée:** on autorise l'emploi de données de vitesses prélevées en aval de l'écoulement;
- **contrôle généralisé:** on autorise à la fois les fonctions périodiques et les données de vitesse.

À fin de rester compatible avec des conditions expérimentales, l'optimisation a été limité à l'évaluation de 1000 individus répartis dans une population de 100 individus évoluant à travers 10 générations. L'optimisation des lois de contrôle dans ces trois catégories révèle dans un premier temps que le forçage périodique n'est pas une solution viable pour réduire la puissance nette de traînée. En effet, les deux stratégies, forçage multi-fréquence et forçage généralisé, ont exclus les fonctions périodiques des solutions finales. De fait, le forçage multi-fréquence a construit une loi de contrôle constante pour les trois cylindres et qui imite une configuration boat tailing légèrement asymétrique. Cette loi de contrôle a permis une réduction de la puissance nette de traînée supérieure à celle du boat tailing symétrique. L'ajout de données de vitesse comme entrée de la loi de contrôle a permis de réduire davantage la fonction de coût. La meilleure solution est obtenue pour le troisième type d'optimisation et combine un forçage de type boat tailing asymétrique ainsi qu'un retour en boucle fermée pour les trois cylindres qui ajoute au contrôle une composante dépendante du temps. Ce contrôle a pour conséquence de réduire l'intensité des tourbillons en aval de l'écoulement ainsi que de retarder leurs lâchers par rapport à l'écoulement naturel sans contrôle.

Le succès de LGPC est alors d'avoir découvert des mécanismes d'action de la littérature et de les combiner pour réduire significativement la fonction de coût, et ce, sans connaissance a priori de la physique de l'écoulement.

Partie 2: Contrôle de sillage à convergence rapide par combinaison de programmation génétique et de méthodes de gradient

Chapitre 4: L'*Explorative Gradient Method* (EGM) ou méthode d'optimisation paramétrique par exploration et méthodes de gradient

Dans ce chapitre, nous traitons un nouveau problème de contrôle, la stabilisation du pinball fluidique, grâce à une méthode d'optimisation paramétrique, l'*Explorative Gradient Method* (EGM) introduite par Li *et al.* (2021). EGM est un algorithme d'optimisation de paramètres basé sur l'équilibre entre exploration et exploitation de l'espace de recherche pour résoudre des problèmes d'optimisation non-convexe et avec de nombreux minima. EGM est une méthode itérative combinant exploration et exploitation. L'exploration se fait grâce à une méthode de remplissage d'espace (le latin hypercube sampling, LHS), dont l'objectif est de trouver les minima du problème en évaluant à chaque fois l'individu le plus «éloigné» de ceux déjà évalués. L'exploitation est, quant à elle, effectuée grâce à l'application de la *méthode du simplexe* de Nelder & Mead (1965) dont l'idée est de faire progresser un sous-ensemble d'individu vers un minimum grâce des opérations géométriques (réflexion, contraction, expansion) basées sur les performances des individus. De fait, la méthode du simplexe exploite les gradients locaux pour descendre en direction du minimum local. Les opérations d'exploitation

et d'exploration sont alors alternées pour équilibrer la recherche du voisinage du minimum global et la descente effective vers le minimum.

Cette méthode est alors appliquée au problème de stabilisation du pinball fluidique. Le but étant d'atteindre l'état symétrique stationnaire de l'écoulement. Pour cela, nous regardons la distance en norme L^2 à cet état symétrique stationnaire, le tout moyenné sur une fenêtre de temps suffisamment longue pour négliger la contribution du transitoire. Une première étude paramétrique des forçages symétriques en boucle ouverte montre que pour stabiliser l'écoulement, la configuration dite de *base bleeding*, où le fluide est éjecté à co-courant par les cylindres arrières, est favorable. Cette configuration permet notamment de re-symétriser l'écoulement. Par la suite, nous appliquons l'algorithme EGM à deux types de forçage:

- **périodique symétrique:** le cylindre frontal est immobile et les deux cylindres arrières sont pilotés par la même commande périodique au signe près.
- **libre et constant:** on autorise la rotation à vitesse constante des trois cylindres de manière indépendante.

Dans le premier cas, nous optimisons deux paramètres, l'amplitude et la fréquence de forçage tandis que dans le deuxième, nous devons déterminer trois constantes de vitesses, une pour chaque cylindre. Dans les deux cas, EGM réussit à trouver des solutions diminuant significativement la fonction de coût en moins de 50 individus évalués. Le cas du forçage périodique symétrique permet une stabilisation complète du sillage, mais au détriment d'un coût d'action très élevé. Le cas du forçage libre révèle, quant à lui, qu'un forçage asymétrique permet une meilleure stabilisation que le forçage symétrique. Le principe d'exploration et d'exploitation de l'espace de recherche introduit par EGM a servi d'inspiration pour accélérer la vitesse d'apprentissage de LGPC en y ajoutant une phase d'exploitation basée sur la méthode de simplexe. La nouvelle méthode est alors appelée *gradient-enriched machine learning method* (gMLC) et est détaillé en chapitre 5.

Chapitre 5: Le *Gradient-enriched machine learning control* (gMLC) ou apprentissage automatique de lois de contrôle par combinaison de programmation génétique et méthodes de gradient

Dans ce chapitre, nous présentons un nouvel algorithme d'apprentissage automatique original combinant, à l'instar d'EGM, une phase d'exploration et une phase d'exploitation, le *gradient-enriched machine learning control* (gMLC). L'algorithme se base sur le processus itératif et stochastique de LGPC, mais s'en distingue par l'absence d'une population qui évolue par générations successives. En effet, dans cette approche, on considère l'ensemble des solutions candidates testées. À cet ensemble s'ajoute par la suite de nouveaux individus issus des phases d'exploration et d'exploitation. L'exploration se fait grâce à une recombinaison stochastique des individus grâce aux opérateurs génétiques de crossover et de mutation. C'est

une étape qui s'inspire fortement de la méthode de programmation génétique. L'exploitation se fait grâce à une variante de la méthode du simplexe appliqué dans un sous-ensemble de l'espace des lois de contrôle. Une des innovations qui a permis ce mariage est, notamment, la mise en place d'une étape de reconstruction qui permet d'appliquer les opérateurs génétiques sur des individus issus d'une étape de simplexe.

Appliqué au problème de stabilisation du pinball fluidique, le gMLC a construit une loi de contrôle combinant un forçage asymétrique et un contrôle en boucle fermée qui surpasse toutes les solutions trouvées jusque-là. C'est une loi qui combine linéairement 14 autres lois et qui, en plus de permettre une diminution plus importante de la fonction de coût, le fait avec avec la plus faible puissance d'action. Les performances de gMLC comparées avec l'algorithme LGPC pour le problème de stabilisation montre que la nouvelle méthode permet, non seulement, de construire des solutions plus performantes, mais de plus s'obtiennent avec un nombre plus faible d'évaluations. En effet, déjà après 250 évaluations, les résultats de gMLC sont meilleurs qu'une optimisation sur 1000 individus de LGPC. L'accélération de l'apprentissage est confirmée par un test de répétabilité sur un système dynamique (le generalized mean-field model, GMFM). Sur 100 évaluations de chaque algorithme, la distribution des réalisations montre que gMLC surpasse LGPC à la fois en termes de vitesse de convergence ainsi que de la performance de la solution finale.

Dans la dernière partie, gMLC est testé en conditions expérimentales pour contrôler l'écoulement de cavité ouverte.

Partie 3: Démonstration expérimentale

Chapitre 6: Description du banc expérimental de la cavité ouverte

La dernière partie du travail consiste à confronter nos avancées méthodologiques d'apprentissage automatique à la réalité expérimentale. Nous présentons dans ce chapitre la configuration d'écoulement choisie, à savoir, l'écoulement d'une cavité ouverte en interaction avec une couche limite laminaire. Cet écoulement présente une dynamique riche pilotée par cinq paramètres en incompressible: les trois dimensions de la cavité, l'épaisseur de la couche limite et la vitesse incidente du fluide incident. Le contrôle de la cavité se fait grâce à un actionneur plasma DBD (décharge à barrière diélectrique), situé au bord de fuite amont de la cavité. Cet actionneur permet d'imiter l'action d'une force volumique le long de la cavité. Un capteur fil chaud en aval de l'écoulement mesure la vitesse du fluide en aval de la cavité et nous permet à la fois de caractériser l'écoulement et de contrôler l'actionneur plasma par retour de l'état instantané. Le tout est alors piloté grâce à un contrôleur en temps réel dSPACE.

Dans cette étude, nous nous plaçons à Reynolds $\mathrm{Re} \approx 10^4$, dans une configuration où la réponse fréquentielle présente une fréquence dominante qui reflète les

oscillations de la couche de mélange. Une étude en boucle ouverte avec une action constante a été réalisée. Elle révèle notamment qu'une action forte et constante tue la dynamique de la cavité et «noie» le pic principale par la destruction de l'amplificateur de Kelvin-Helmholtz. Une action modérée permet, quant à elle, de réduire la puissance associée au pic principale et fait resurgir un autre mode de l'écoulement. Nous limitons la gamme d'intensité de l'action entre le seuil d'ionisation de l'actionneur plasma (action nulle) et un maximum conservant la résonnance principale de la cavité. Enfin, soulignons que l'expérience est sujette à différentes dérives, telles que la température de la salle, la vitesse débitante de la soufflerie et l'action du plasma DBD. Ces dérives devant rester «acceptables» pour l'apprentissage, la durée de l'expérience sera nécessairement limité.

Chapitre 7: Contrôle de la cavité ouverte par LGPC et gMLC

Dans ce chapitre, nous appliquons les méthodes de LGPC et gMLC au contrôle de la cavité. L'objectif de contrôle cherche à réduire la puissance du pic de résonance de la cavité et celle des modes associés. Pour ce faire, nous construisons une fonction de coût qui mesure la valeur maximale du pic dans le spectre du fil chaud sur une fenêtre qui englobe l'ensemble des modes fréquentiels à l'ordre 1 (sans les harmoniques). Enfin, un terme de pénalisation de l'action est inclus pour orienter l'action vers un contrôle de l'état stationnaire instable et non vers un «soufflage» de la couche de mélange.

Les algorithmes, LGPC et gMLC, sont appliqués pour une évaluation totale de 1000 individus. Après 1000 évaluations, LGPC réussit à considérablement diminuer la puissance associé au premier pic (gain d'un facteur 1000 en puissance) avec un coût d'action quasi-négligeable mais au détriment d'une augmentation significative de la déviation standard du signal global. LGPC réduit le pic, mais augmente le bruit de fond. Gradient-enriched MLC, quant à lui, réussit à construire une loi de contrôle en boucle fermée en moins de 300 évaluations dont les performances sont similaires à la loi LGPC, mais avec une réduction des fluctuations de vitesse sur toutes les fréquences, conduisant à une diminution de la déviation standard de 14% par rapport au cas naturel sans contrôle. De plus, contrairement à LGPC, cette loi de contrôle diminue la déviation standard du signal. Par ailleurs, nous apportons la preuve que cette loi de contrôle est effectivement une loi en boucle fermée dont la rétroaction est essentielle bien que quasi-négligeable en amplitude.

Aussi, nous démontrons la capacité de nos algorithmes à opérer dans des conditions expérimentales réelles, incluant du bruit externe et des dérives. C'est notre nouvel algorithme gMLC qui présente le résultat le plus impressionnant, en réussissant à surclasser LGPC en termes de vitesse de convergence, solution finale et qualité de la solution.

Chapitre 8: Conclusion

Dans ce chapitre final, nous rappelons ce qui a pu être accompli durant ce travail de thèse. L'ensemble des efforts ont pu converger vers la principale innovation: le développement d'un nouvel algorithme de contrôle, le gMLC. L'efficacité de gMLC

a été démontré à la fois numériquement, avec le contrôle du pinball fluidique, et expérimentalement, avec le contrôle de la cavité ouverte. En intégrant, l'ensemble des bénéfices, vitesse de convergence et qualité de la solution finale, on peut estimer qu'une accélération d'un facteur au moins 10 a été obtenu entre gMLC et LGPC. Une telle accélération ouvre la porte à des expériences de contrôle plus complexes avec des temps d'expérience limités ou des espaces de recherches plus riches. En particulier, un apprentissage plus rapide permet de considérer des méthodes d'entraînements plus complexes en multipliant les évaluations d'une même loi de contrôle pour assurer une robustesse vis-à-vis des paramètres du problème et des conditions de fonctionnement.

Contents

Nomenclature

Re	Reynolds number
St	Strouhal number
i	Individual
N_{popsize}	Number of individuals in a generation or population size
n	Generation
N_G	Number of generations
N_{r}	Total number of registers
N_{vr}	Number of variable registers
N_{cr}	Number of constant registers
N_{inst}	Number of instructions
$N_{\text{inst,max}}$	Number maximum of instructions
N_{o}	Number of functions in the library or mathematical operators
N_{b}	Number of actuators
N_{h}	Number of time-dependent functions
N_{s}	Number of sensor signals
N_i	Number of evaluations
N_{tour}	Tournament selection size
P_{tour}	Tournament probability
P_{mut}	Mutation probability for an instruction
N_ρ	Number of realizations
ρ	Realization
P_c	Crossover probability
P_m	Mutation probability
P_r	Replication probability
N_e	Elitism number
N_{MC}	Number of Monte Carlo individuals
N_{sub}	Subspace size
N_{p}	Number of individuals per phase
$\mathcal{K}$	Control law space
$\boldsymbol{K}$	Control law
$\boldsymbol{a}$	State vector
$\boldsymbol{b}$	Actuation vector
$\boldsymbol{s}$	Sensor vector
J	Cost function
J_a, J_b	Cost function components
γ	Penalization parameter

Chapter 1

Introduction

Contents

1.1 Economic and industrial context

Flow control has been at the heart of many engineering advancements through time. One of the oldest examples of flow control being the use of the fletching or feathers on the foragers' arrows to stabilize the flight of their projectiles, thousands of years ago. Even if the feathers are responsible for about 35% of the drag (Meyer, 2015), the additional lateral surfaces provide a stability of the arrow throughout the entirety of its flight, thus improving significantly its objective: reaching the target. Such a technological advancement had a significant impact on the development and survival of the foragers. Nowadays, building on great theoretical advancements, more complex controllers and control strategies, capable to react in real-time, have been set up, enabling long-range intercontinental missiles that can travel up to 17 000 km as in the case of the Russian RS-28 Sarmat.

But flow control is not only about ballistics. Flow control is now a key opportunity for the next challenges of the century, as modelling, predicting and control of fluid flows are fundamentals problems for engineers in the industry. One of the most compelling examples being the transport field. Traffic alone profits from flow control via drag reduction of transport vehicles (Choi *et al.*, 2008), lift increase of wings (Semaan *et al.*, 2016), mixing control for more efficient combustion (Dowling & Morgans, 2005) and noise reduction (Jordan & Colonius, 2013). An

improvement of performances in one of those fields will be greatly beneficial allowing reduction of the energy consumption, increase of payloads reductions and facilitates the ecological transition.

In the last two decades, the total number of cars increased dramatically posing environmental and energy resources concerns. Indeed, in 2010 the number of motorized vehicles worldwide exceeded 1 billion and this number is expected to double by 2030 (Sousanis, 2011). In 2015, the annual global CO_2 emissions from road vehicles surpassed 22Mt and is expected to reach more than 30Mt by 2030 (Brunton & Noack, 2015). To limit the polluting impact of the increase of the car population and the emission of greenhouse gases and to achieve climate-neutral by 2050, the European Parliament and the Council adopted, on April 2019, the Regulation (EU) 2019/631 setting CO_2 emission performance standards for new passenger cars and for new vans in the EU (European Commission, 2019). This regulation includes 15% CO_2 reduction from 2025 on and 37.5% CO_2 reduction from 2030 on. Thus, developing flow control to be manageable for real life conditions is a key opportunity for improving industrial efficiency and reducing CO_2 emissions. And recent advances show that active flow control is already able to achieve up to 15% net power saving for 3D bluff bodies in experiments (Pfeiffer & King, 2012). Of course, the impact of flow control in the transport field is only an example among other fields such as: drag reduction for airborne, ground and sea transport, lift increase of wings (Semaan *et al.*, 2016), gust mitigation for wind power generation and heat transfer improvement to cite a few.

The aim of this study is then to advance the flow control field towards real life applications. In the following, we make an overview of turbulence flow control, the main challenges of the field, the framework used in this study and its main objectives.

1.2 Turbulence flow control

Turbulence flow control is the field that comprises decisions on the flow control system or plant, on the cost function that measures the objective, on the actuation and sensing and the control logic. In this following, we give a short overview of the field and more details on active flow control and the control logic.

1.2.1 Overview of the field

By manipulating the flow around a body or in a cavity, performance can be improved and energy consumption reduced, but there are different ways to act on the flow. In this section, we present an overview of the three main categories of flow control: aerodynamic shape optimization, passive control and active control.

The goal of aerodynamic shaping is to optimize the geometry of a body to improve its interaction with the flow. One can consider that the field of aerodynamic shaping started with Isaac Newton and his sine-squared law for the hydrodynamic force on an inclined surface, at the end of the 17th century. In his renowned *Principia*, Isaac Newton made the mathematical connection between geometry of

a body and the forces that are applied to it (Anderson, 2011). Modern examples of shape optimizing include airfoil design, heat exchangers, wind turbine blades and combustion chambers to cite a few examples. Historically, the field is based on potential flow theory pioneered 150 years ago, but, today, modern techniques rely on efficient adjoint-based methods but relying on costly numerical simulations of the full Navier-Stokes equations, such as the pioneering work of Reuther *et al.* (1996).

The second approach is passive control. While aerodynamic shaping aims to optimize the shape of the system, passive control adds features to further increase the performance. As described earlier, the feathers on the archer's arrow is an example of passive control, increasing the stability of the projectile but also increasing the drag. The same mechanism is exploited for the stabilization of rockets thanks to the lateral fins. On another example, is the presence of dimples on the surface of golf balls that facilitates the transition from a laminar to turbulent flow and thus reducing the drag force. The roughness of the surface combined with the rotation of the ball produces an increased Magnus lift. Those two mechanisms greatly increase the distance covered by golf balls and make that, now, dimpled golf balls a standard in the field. Other engineering applications of passive control are the turbulators on wings that facilitate the transition towards a turbulent flow and allow a lift increase; another example is the use of Gurney flaps at the trailing edge of airfoils that helps the boundary layer to stay attached until the trailing edge, increasing the maximum lift (Myose *et al.*, 1998). However, these benefits are often accompanied by drawbacks, such as drag increase, that are not easily avoided as these additional devices are not meant to be controlled in 'real time.' For academic flows, passive control can be achieved with the introduction of a plate behind a bluff to suppress the vortex shedding or control the aeroacoustic effects (Winkler *et al.*, 2012).

The third approach, active control, on the other hand, uses actuators to disturb favorably the flow. These include Coanda blowers, synthetic jets (You & Moin, 2008), plasma actuators (Moreau, 2007), rotating blades, fluidic oscillator (Guyot *et al.*, 2008; Bobusch *et al.*, 2013) and zero-net-mass-flux actuators (Zhang *et al.*, 2008) and piezoelectric actuators (Cattafesta *et al.*, 2001). Their purpose is to inject/suppress momentum into/from the flow with a time response of the same order of magnitude as the time scales of the flow. Such control requires a positive balance between the energy injected and the energy saved. However, as opposed to passive control, active control can be switched on and off following the conditions, limiting the negative impact of such control. Thus, actuators can be controlled by external commands, independently of the flow state, such as periodic blowing or suction; this control is referred to as *open-loop* control. An online adaption of the control is possible thanks to the feedback of sensor information; this control is referred to as *closed-loop* control. The sensors can be employed to feedback direct flow information such as velocity components or pressure, but it can also be employed to reconstruct the flow from sparse measurements (Podvin *et al.*, 2005). The position and number of sensors and actuators is a hard problem in itself. Sensitive regions can be derived thanks to adjoint based methods such as

in Giannetti & Luchini (2007) for the cylinder wake. Position of the sensors is crucial as two types of closed-loop control arise in convective flows: feedforward (sensors are located upstream of the actuators) and feedback (sensors are located downstream of the actuators). Feedback proved to efficiently reject disturbances and is able to compensate unmodeled dynamics (Belson *et al.*, 2013). Optimal positions of sensors have also been studied in a data-driven manner for dynamic processes in (Proctor *et al.*, 2014; Brunton *et al.*, 2014). The largest gains of performance happens to be realized with a closed-loop control where the actuation command is directly related to the flow state.

However, deriving the optimal control command for the actuators, that increases the performances and limits the negative impacts, is in general a difficult problem. Indeed, the nonlinear nature of the partial differential equations that describe the motion of a flow, the Navier-Stokes equations, bears fundamental challenges for modelling, predicting and real-time control:

high-dimensionality: to have an accurate description of all the spatial and time scales of the flow, such as in turbulent flows, a high number of virtual particles needs to be simulated, rendering such simulations very costly;

time-delayed response: there is in general a non-negligible transient time between actuation and response and between actuation and measure of the state;

frequency crosstalk or nonlinear interaction between modes: exciting a given mode with a given frequency may give a response at a different mode with a different frequency, making modelling and prediction delicate.

We can distinguish two approaches to derive the optimal control command that takes more or less into account these challenges. First is the model-based approach based on linear control theory. The main idea is to model the flow by linearizing its dynamics around a state of interest such as the mean flow or a fixed point to exploit powerful mathematical tools to control and predict the flow (Sipp *et al.*, 2010; Bagheri & Henningson, 2011). Examples of model-based control successes relate to first and second order dynamics, e.g., the quasi-steady response to quasi-steady actuation (Pfeiffer & King, 2012), opposition control near walls (Choi *et al.*, 1994; Fukagata & Nobuhide, 2003), energy attenuation for Tollmien-Schlichting waves and streaks in a transitional boundary layer (Semeraro *et al.*, 2011), phasor control of oscillations (Pastoor *et al.*, 2008), two-frequency crosstalk (Glezer *et al.*, 2005; Luchtenburg *et al.*, 2009) and robust control of an Ahmed body (Plumejeau *et al.*, 2019; Chovet *et al.*, 2020). The linearizing of the equations can be avoided by relying on models produced by input-ouput data thanks to the eigensystem realization algorithm (ERA) (Juang & Pappa, 1985). ERA has been used, for example, to build an optimal controller to suppress three-dimensional Tollmien-Schlichting waves in a transitional boundary layer (Semeraro *et al.*, 2013).

However, even though linear control theory may have some answers, it calls upon restrictive linear hypotheses that discard nonlinear interactions between

modes, which is often essential for fluid control. Indeed, examples of wake stabilization with high-frequency forcing and low-frequency forcing show that frequency crosstalk is a key enabler for control (Pastoor *et al.*, 2008; Luchtenburg *et al.*, 2009). Moreover, in addition to the nonlinear complexity of the flow, there are a number of constraints related to real-life experiments, such as the number, location and type of sensors and actuators required in a closed-loop system, such features are typically determined from engineering wisdom (Cattafesta & Shelpak, 2011). Finally, the use of non-intrusive sensors gives access only to very sparse information of the dynamics. Those additional constraints make linear control theory often unpractical for in-time control.

Thus, it is essential to consider another approach: the derivation of the optimal control command in a model-free framework with sparse knowledge of the system state. In this case, the system to control is considered as a black-box model only taking into account the actuation command and the sensor signals. Thus, the entirety of the dynamics are taken into account even sparsely in space and time but with all its richness including both linear and nonlinear aspects. Deriving the optimal control in such framework is a hard problem that needs powerful regression techniques inspired of machine learning methods.

Thus in this study, we aim to develop a model-free methodology to derive the optimal actuation command for active flow control. In the next sections, we present the active control framework, introducing notations and terminology used throughout the study.

1.2.2 Active control

In this section, we describe the framework of active control for fluid flows.

Throughout this study, the system to control will be referred as the *plant*. The plant can designate a(n):

dynamical system: such as the Lorenz system with its three ordinary differential equations;

numerical simulation: two-dimensional or three dimensional flows comprising bluff bodies, shear layers or other interesting geometries with full information of the state and no noise or disturbances a priori;

experiment: such as an open cavity, a mixing jet or an Ahmed body with controlled experimental conditions but with external noise, disturbances and measurements uncertainties.

real fluid flow: a car, a truck, a plane, a wind turbine or a heat exchanger in real-life conditions;

or any other system with actuator(s).

In fluid mechanics, the quantities of interest that we want to control, are the ones related to the aerodynamics. For example, increasing the lift of an airfoil or reducing the drag of a cylinder. But we can also be interested in the dynamics

of the flow and its coherent structures, for example, reducing the oscillations of a shear layer, stabilizing the wake behind a bluff body and reducing the fluid-structure interactions. Of course, controlling the flow and its coherent structures is intrinsically related to undesirable forces to mitigate. The plant is then the main system to control.

In order to control the plant, one or several *actuators* are needed. The actuators can be, for example, synthetic jets on an airfoil, Coanda blowers at the back of an Ahmed body, piezoelectric actuators, dielectric-barrier discharge (DBD) or rotating bodies. Their goal is to perturb favorably the flow by injecting or suppressing momentum, for blowers or vorticity, for rotating bodies. The intensity of the control and its 'form' (strong or weak jets, fast or slow rotation, mean value and frequencies) is directly related to an external input, we refer as the *actuation command* and is denoted by the vectorial quantity $\boldsymbol{b}$.

Solely with actuators, active flow control in an open-loop manner can be performed. Indeed, time-dependent functions can be employed as an actuation command to steady actuation (Mestiri *et al.*, 2014) as well as multi-frequency forcing (Li *et al.*, 2019). Throughout this study, the time-dependent functions are denoted as $\boldsymbol{h}$.

Now, to have a control that reacts to the flow behaviour, a feedback of the state is needed. For this, sensors can be introduced in the plant in order to retrieve information of the flow, such as velocity probes, hot-wires and pressure sensors. In the following, the sensor information is denoted by the vectorial quantity $\boldsymbol{s}$.

However, in order to effectively perform closed-loop control, the relationship between the actuation command and the sensor signals needs to be defined. In the most general case, the actuation command $\boldsymbol{b}$ is a function of sensor signals $\boldsymbol{s}$ and time-dependent functions $\boldsymbol{h}$. The controller then reads:

$$\boldsymbol{b}(t) = \boldsymbol{K}(\boldsymbol{s}(t), \boldsymbol{h}(t)). \tag{1.1}$$

The function $\boldsymbol{K}$ that maps $\boldsymbol{s}$, $\boldsymbol{h}$ and $\boldsymbol{b}$ is referred as the *control law*. $\boldsymbol{K}$ is an element of $\mathcal{K} : X \mapsto Y$, the space of all possible control laws, also called, in this study, *control law space*, *search space* or *control landscape*. Here, X is the input space, e.g., the space of sensor signals and Y is the output space for actuation commands.

To determine the optimal $\boldsymbol{K}$ that fulfills a given objective such as drag reduction or lift increase, a measure of the performance of the control is needed. This is achieved by the cost function J.

$$J = J_a(\boldsymbol{b}) + \gamma J_b(\boldsymbol{b}) \tag{1.2}$$

The cost function is a function that gives a scalar value related to the performance of the control. It is defined such as the optimal control regarding our objective is an extremum of the cost function. In this study, we choose the convention to minimize the cost function, i.e. the optimal control $\boldsymbol{K}^*$ corresponds to the minimum of the cost function J over the search space $\mathcal{K}$. The cost function J is usually the sum of two components, J_a and J_b, defining different aspects to optimize. J_a is generally directly related to the physical quantity to optimize, it

can be the lift or drag coefficient, the turbulent kinetic energy or the length of the recirculation bubble, for example. J_b, on the other hand, is a measure of the energy invested in the control, such as the momentum injected in the flow or the actuation power related to the rotation of some bodies. It is computed so that strong actuations are penalized. The parameter γ then serves to balance the objective and the actuation penalization term. The computation of the cost function is, in general, not directly related to the sensor signals and can be computed offline, meaning once the controlled is performed. J can be, for example, computed from time series measurements with an aerodynamic balance, flow fields measured with particle image velocimetry (PIV) or estimations of the flow state thanks to sensor signals.

Figure 1.1 gives a description of the components of the plant. The position of the actuators and the sensors in the figure do not assume their position in the plant. Indeed, sensors can be located at the leading edge of an airfoil and the actuators and the trailing edge and vice versa.

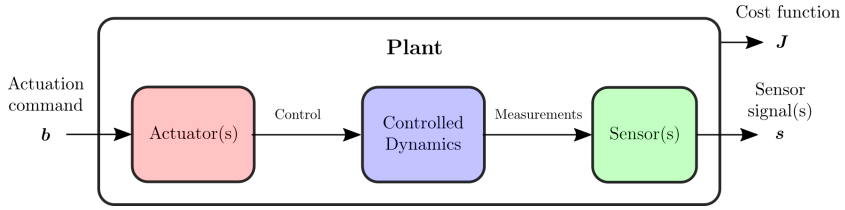


Figure 1.1: Configuration for a plant to control. The actuators receive an external actuation command and control and thus modify the natural flow. Then the sensors retrieve some flow state information. The cost function measures the performance of the control. The arrows do not assume the position of the actuator(s) and sensor(s).

Efficient methods that rely solely on the actuation command and the sensor signals exists, such as extremum-seeking control (Gelbert *et al.*, 2012), opposition control (Choi *et al.*, 1994; Fukagata & Nobuhide, 2003) or proportional-integral-derivative (PID) control (Krstic *et al.*, 1999). However, they all rely on a predetermined structure of the control law and tuning such control law is to tune a set of parameters. Moreover, the predefined structure of the control law then bounds the actuation to a subspace of $\mathcal{K}$, limiting the possibility to exploit the nonlinearities of the flow for control.

In order to take advantage of the nonlinear richness for the control, we consider a structure-free control law approach and our aim is to build the relationship between the inputs and the outputs from scratch. Such a control problem is hard to solve and require powerful methods of machine learning or artificial intelligence to build the mapping. A key enabler to employ such methods is the formulation of the control problem as an optimization problem: the goal is to find $\boldsymbol{K}^*$ that minimizes J.

1.2.3 Flow control as an optimization problem

The main approach to solving the model-free, structure-free control problem is to formulate it as a function optimization problem where the function to optimize is the control law itself, according to a cost function J to minimize. Equation 1.3 is the optimization problem to solve to find the optimal control law $\boldsymbol{K}^*$.

$$\boldsymbol{K}^* = \underset{\boldsymbol{K}\in\mathcal{K}}{\arg\min}\ J(\boldsymbol{K}) \tag{1.3}$$

with $\mathcal{K}$, the control law space as defined above. The resulting optimization problem is another non-convex optimization problem, which is difficult, but much more accessible to machine learning/artificial intelligence regression solvers. Indeed, equation 1.3 can also be interpreted as a regression problem of the second kind, meaning a regression problem where the optimal solution is unknown. The term regression model of the first and second kind has been introduced by Fisk (1967) in the field of statistics and designate two types of models. In the following, we detail the meaning in the control optimization context. In a regression problem, the goal is to determine the relationship that ties several sets of data. There are two types of regression problems: the first and the second kind.

For a regression problem of the first kind, we assume the knowledge of the response of all the possible actions for every state. So, we have access to the matrix $\boldsymbol{P}$ of the plant that connects the actuation to the sensing: $\boldsymbol{s} = \boldsymbol{P}\boldsymbol{b}$. The goal is then to build a function representing the inverse model $\boldsymbol{P}^{-1}$ of the plant by approximating a mapping between the sensors and the actuation command. The *quality* of the inverse model $\boldsymbol{P}^{-1}$ is evaluated thanks to a metric that compares the real action $\boldsymbol{b}$ and the reconstruction $\boldsymbol{P}^{-1}\boldsymbol{P}\boldsymbol{b}$. The inverse model is then enough to derive the optimal control for a given objective. Such approach has been successfully employed by Lee *et al.* (1997); Lorang *et al.* (2008) to reduce the drag in a turbulent channel flow, where the inverse model has been reconstructed thanks to neural networks. The reconstruction of the input-output relationship can also be realized in a data-driven manner, directly from time series or snapshots thanks to mode decomposition such as in Proctor *et al.* (2016), where the actuation input is included in the data. Regression problems of the first kind can be considered as part of supervised learning, as they rely on the prior knowledge of the plant behaviour. Linear regression can be seen as a regression problem of the first kind where an assumption of a linear model is made. Thus, inverting the model, based on the observations, is to determine the best fitting parameters of the model. With these parameters, it is then possible to interpolate, extrapolate the data beyond the observation range. Of course, the area of validity of the learned model depends on the data and type of problem.

On the other hand, regression problem of the second kind are only based on the control performance, without any knowledge of the plant nor the optimal solution. The control performance is, of course, computed thanks to the cost function J. The aim is to optimize/build better control laws solely based on their performance. As each new state is a consequence of the action-response relationship, what is learned is a curve or a trajectory of the system state. It is the trajectory towards

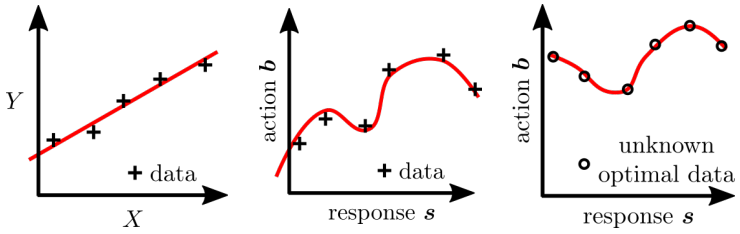
	Linear regression	regression problem of the first kind	regression problem of the second kind
visual representation			
mapping function	$Y = \alpha X + \beta$	$\boldsymbol{b} = \boldsymbol{P}^{-1}\boldsymbol{s}$	$\boldsymbol{b} = \boldsymbol{K}(\boldsymbol{s})$
unknown of the problem	best fitting parameters (α^*, β^*)	the *inverse plant* $\boldsymbol{P}^{-1}$	the optimal control law $\boldsymbol{K}^*$
optimization type	parametric	function optimization	curve/trajectory optimization
optimization criterion	coefficient of determination R^2	metric	cost function J

Table 1.1: Table summarizing the two types of regression problem. A comparison with regression linear is displayed. For regression problems of the first kind, the knowledge of the action-response relationship is assumed. For a regression of the second kind, the optimal solution is not known. Linear regression is a type of regression problem of the first kind where an assumption of the linear model is made.

the unknown optimal solution that is built. Table 1.1, illustrates linear regression and the differences between the two types of regression.

An alternative for solving control problems is also their reformulation as a Bellman equation where the principle is to break down an optimization problem in a sequence of subproblems. This is at the basis of reinforcement learning (RL) algorithm and has been proven to be efficient to derive feedback control laws in simulations and experiments (Fan *et al.*, 2020; Rabault *et al.*, 2019; Bucci *et al.*, 2019).

In this study, to solve the control problem, we will focus on genetic programming which is regression solver of the second second kind. Genetic programming control (GPC) has been pioneered by Dracopoulos (1997) over 20 years ago and has been proven to be particularly successful for nonlinear feedback turbulence control in experiments. Examples include the drag reduction of the Ahmed body

(Li *et al.*, 2018) and the same obstacle under yaw angle (Li *et al.*, 2019), mixing layer control (Parezanović *et al.*, 2016), separation control of a turbulent boundary layer (Debien *et al.*, 2016) recirculation zone reduction behind a backward facing step (Gautier *et al.*, 2015), and jet mixing enhancement (Zhou *et al.*, 2020), just to name a few. GPC has consistently outperformed existing optimized control approaches, often with unexpected frequency crosstalk mechanisms (Noack, 2019; Ren *et al.*, 2020). GPC has a powerful capability to find new mechanisms, thanks to *exploration* and populate the best minima, thanks to *exploitation*. Yet, the exploitation is inefficient leading to increasing redundant testing of similar control laws with poor convergence to the minimum. This challenge is well known and is the main objective of this study. Indeed, the acceleration of the learning is crucial for future experiments. As it will truly multiple the number of control experiments. The acceleration of the learning will, among other things, enable to control experiments with many actuators and sensors and are subject to the curse of dimensionality. It will also allow complex training requiring multi-parameter testing for robustness and it will make possible to control costly experiments with a reduced experiment time. The acceleration of the learning is also part of a broader perspective concerning learning and what performance is accessible from our current knowledge. This resonates with the concepts of exploration, to acquire more knowledge and exploitation, to build something new from our current knowledge. It is not a surprise that, exploration and exploitation are the two main forces behind the learning process of genetic programming. Their role is further detailed in this study as they have a primordial place in the speed up achieved by our new algorithm, the *gradient-enriched machine learning control*.

In (Schoenauer *et al.*, 1996), the authors introduce a hill-climbing stage in genetic programming to improve the learning rate by exploring local minima in model identification for a material. Such improvement proved to be efficient but only for a given set of meta-parameters. Our work is part of the same perspective but the local minima are explored thanks to gradient-based methods instead of hill-climbing.

1.3 Our plants: the fluidic pinball and the open cavity

Our algorithmic development needs to be tested on relevant plants. However real flows are complex. Indeed, reducing the drag of a real car or increasing the lift of an airplane implies considering the interaction of different physical mechanisms due to the complexity of the geometries. That is why, real systems are broken down into canonical geometries that represent simplified versions of the system or parts of it. In this work, we focus on two plants: a numerical simulation of a fluidic pinball and an experiment on an open cavity.

As main benchmark control problem, we choose to study the fluidic pinball, a configuration of three cylinders situated at the vertices of an equilateral triangle pointing in the upstream direction (Noack *et al.*, 2016; Deng *et al.*, 2020;

Chen *et al.*, 2020). The actuation is performed by the independent rotation of the three cylinders and the flow is monitored by 9 velocity probes downstream. This choice is motivated by several reasons. First, already the unforced fluidic pinball shows a surprisingly rich dynamics. With increasing Reynolds number the steady wake becomes successively unstable and goes through a Hopf bifurcation, a pitchfork bifurcation and another Hopf bifurcation before, eventually, a chaotic state is reached (Deng *et al.*, 2020). Second, the cylinder rotations may encapsulate the most common wake stabilization approaches, like Coanda forcing (Geropp & Odenthal, 2000), base bleed (Wood, 1964; Bearman, 1967), low-frequency forcing (Pastoor *et al.*, 2008), high-frequency forcing (Thiria *et al.*, 2006), phasor control (Roussopoulos, 1993) and circulation control (Cortelezzi *et al.*, 1994). Third, the rich unforced and controlled dynamics mimic nonlinear behaviour of turbulence while the computation of the two-dimensional flow is manageable on workstations. To summarize, the fluidic pinball is an attractive all-weather plant for non-trivial multiple-input multiple-output control dynamics.

As for the open cavity, its choice is motivated by both the simplicity of its geometry and the richness of its dynamics. It is a well-known configuration presenting a wide range of dynamics accessible by tuning only two parameters: the aspect ratio and the Reynolds number. Thus, the dynamics may go from a single main mode to a rich spectrum comprising a various coupled modes (Rowley & Williams, 2006; Basley *et al.*, 2014). Experimentally, the cavity has been controlled both in a passive way, with the introduction of a cylinder upstream (Keirsbulck *et al.*, 2008; El Hassan & Keirsbulck, 2017) or in an active way, with synthetic jets (Kourta & Vitale, 2008) for example, suppressing of the cavity resonance or significantly reducing the noise level. Finally, the dynamics of the cavity have been thoroughly studied throughout the years in our laboratory, strengthening our choice as a control benchmark .

1.4 Objectives of the thesis

The main goal of this thesis it to push further the methodology of model-free real-time control for numerical simulations and experiments. We focus especially on the algorithmic development for the acceleration of the learning process of genetic programming control, while keeping the possibility to build complex control laws. For this, we developed our own machine learning codes: 1) `xMLC` based on linear genetic programming and 2) `gMLC`, based on our new algorithm, the gradient-enriched machine learning control (gMLC). Our codes have been been demonstrated both on numerical and experimental plants revealing efficient and rich control mechanisms for control. The `xMLC` and `gMLC` source code are both available online.

The thesis is organized in three parts. In the first part, we describe the genetic programming methodology in the context of flow control and is illustrated with the control of the fluidic pinball. In chapter 2, we give a full presentation of the genetic programming control algorithm and the 'forces' that drive the learning process. In particular, we illustrate the learning capability of GPC with an ex-

haustive parametric study, revealing key operating combinations in the space of meta-parameters. We conclude the first part, by a net drag power optimization carried out on the fluidic pinball in chapter 3 in which we explore three different control spaces and reveal some actuation mechanisms behind net drag reduction. In the second part, we introduce our new algorithm for fast learning of feedback control laws, the gradient-enriched machine learning control (gMLC). But first, in chapter 4, we focus on the Explorative Gradient Method (EGM)(Li *et al.*, 2021), a new algorithm that exploits the principles of exploration and exploitation for parametric optimization. The method is fully described and applied to the fluidic pinball as it largely inspired our new algorithm. In chapter 5, we describe in detail gMLC algorithm and employ it for the stabilization of the fluidic pinball. gMLC manages to build a non-trivial feedback control law outperforming GPC in terms of learning rate and final solution. The third part focuses on the deployment of our algorithms in experimental conditions: For this, we give a description of the open cavity setup, its means of action and sensing and its response to actuation (chapter 6).Finally, we present and compare experimental results of GPC and gMLC applied to the open cavity (chapter 7). We note, in particular, the same learning speed up observed for the stabilization of the fluidic pinball. It has been observed that the learning process has been accelerated by at least a factor 10, in both, the numerical simulations and the experiment. A closing chapter will summarizes the achievements of this work and opens on the future challenges of the field (chapter 8).

Part I

Wake control with genetic programming

Chapter 2

Linear genetic programming control—Description and parametric study

The first part of this thesis aims to describe the linear genetic programming methodology and its application to control and especially on the net drag reduction of the fluidic pinball. This chapter focuses on the description of the genetic programming algorithm in a control framework and the understanding of the internal learning process.

First, we introduce the field of evolutionary algorithms and give a complete description of the linear genetic programming control (LGPC) algorithm in section 2.1. Secondly, in section 2.2, we illustrate the learning process of LGPC with the stabilization of a two-equation dynamical system—the Landau oscillator. Finally in section 2.3, we carry out an extensive study of the meta-parameters of the algorithm before a demonstration on the fluidic pinball in chapter 3.

Contents

2.1 An evolutionary algorithm

Genetic programming control (GPC) is a technique that is part of the machine learning control (MLC) field, it is an evolutionary algorithm that builds control laws, such as K in equation 1.1, to solve control problems. As all evolutionary algorithms, GPC is an optimization algorithm that relies on mechanisms inspired by biological evolution to build candidates solutions for the optimization problem in an iterative and stochastic manner.

The main idea of evolutionary algorithms is based on the evolution or progression of a set of candidate solutions throughout generations thanks to selected recombinations. Following the biological terminology, a candidate solution is also called an individual and a set of individuals, a population, thus, in the following, candidate solutions build in the context of genetic programming will also be referred as *individuals*. The theory of evolution conceived by Charles Darwin in the mid-19th century and augmented/completed throughout the years describes the forces and mechanisms of the evolution of species. We shall not enter in detail in the theory, but only detail key elements that are at the heart of evolutionary algorithms such as:

the survival of the fittest: it is the selection of the most fitting individual, or the most efficient individual according to the environment, to pass on the next generation. This mechanism assures that at each new generation the 'best' individuals are at least as good as the 'best' individuals of the past generation, thus, assuring that the quality of the population improves through time;

crossover: it is one of the two forces of evolution that brings diversity to the population and gives opportunity to improve individuals; crossover is able to exploit the strengths of individuals by recombining two or more individuals and generating one or more offspring build from their 'parents'.

mutation: it is the second force of evolution; it is the force that brings novelty to the population; new features unknown to the individuals are likely to appear thanks to mutation.

It is worth noting that both crossover and mutation are stochastic mechanisms. Indeed, the recombination and the mutation of given individuals are random processes that, in general, give always different results. An evolutionary algorithm is, then, a stochastic process that aims to build better solutions to a problem by making a population of individuals or candidate solutions, evolve by breeding and mutating the best of them.

It is thanks to a similar process, but much more complex, that after millions of years of evolution, the eagle, figure 2.1, evolved as a bird of prey able to fly and land in calm, as well as, harsh, windy conditions thanks to the skillful maneuvering of its wings. A success that is the envy of many, starting with drone manufacturers. Examples of this kind are numerous in nature. Of course, the environment plays a decisive role in the definition and selection of the fittest individual. Indeed

following the environment, animals evolved in different ways, exploiting different features of their environment. Thus, insects and birds exploit different actuation mechanisms to fly. Insects, due to their small size and slow speed, fly at laminar regimes, at Reynolds numbers close Re $\approx$ 200, where they are able to eject fluid momentum to propel themselves. Whereas, birds, which are bigger and fly at higher speed, navigate at regimes close to Re $\approx$ 20000, where they have to deal with turbulence and wingtip vortices to maneuver. It is worth noting that for viscous dominant regimes such as Re $\approx$ 20, there is no possibility to exploit a reaction to a momentum variation of the fluid as the fluid remains sticked to the body steadily, thus this Reynolds number range is also known as the 'death' region where no animal can move and live. This shows the importance of the environment in the process of evolution.

In the case of solving optimization problems with evolutionary algorithms, the environment is imitated by the cost function J, that assess the performance/quality of an individual.

Figure 2.1: The bald eagle, *Haliaeetus leucocephalus*, sharpened its skills through millions of years of evolution.

The beginning of the field can be attributed to Alan Turing, who used the denomination 'evolutionary' in Turing (1950). But it is Ingo Rechenberg, that pioneered the field of evolutionary computation and artificial evolution. Indeed, in the 1960s and 1970s, him and his team, developed a highly influential set of optimization methods known as evolution strategies. One of the key successes of their approach is the design of aerodynamic wings.

Evolutionary algorithms comprise plethora of techniques to solve different kind of problems. For optimization problems in control, one can, for example, perform parametric optimization with evolutionary algorithms. Examples of such algorithms are genetic algorithms (Holland, 1975) and strategy with covariance matrix adaptation evolution (CMA-ES) (Hansen & Ostermeier, 1996). Complex controls involving high-dimensional parameter optimization has been possible thanks genetic algorithms (Benard *et al.*, 2016). Evolutionary algorithms are also useful to build complex, approximate solutions to problems where the form of the exact solution is not known or very hard to compute(Koumoutsakos *et al.*, 2001). They can also be employed for multi-objectives optimization problems (Paschereit *et al.*, 2003). Genetic programming (GP) (Koza, 1992) is one of those techniques.

The idea of GP is to describe the candidate solutions under the form of a computer program. By making an analogy with biology, those computer programs can be considered as the genome of the individual and the operations of mutation and crossover are applied directly on the computer programs. In this context, we refer the mutation and crossover as *genetic operators*. The computer programs can represent solutions to all kind of problems, they can be functions for surface fitting, control laws and computer programs including boolean operations and conditional branches (Brameier & Banzhaf, 2006). GP is then capable to approximate solutions for vast number of problems such as: symbolic regression, classification, data modelling, path finding and, in our case, control (White *et al.*, 2013). For more information on the achievement of GP, we refer to Koza (2010), where the author lists 76 examples where GP's results are competitive to human-produced results.

In the field of flow control, GP is used a regression solver of the second kind to build a mapping between sensor signals $\boldsymbol{b}$/time-dependent functions $\boldsymbol{h}$ and actuation command $\boldsymbol{b}$, minimizing a cost function J. The mapping is referred is a control law $\boldsymbol{K}$ and is a function, living in a infinite dimension, Hilbert function space.

In the following, we will describe the genetic programming control algorithm in the control framework. First, we will present the internal representation of the control laws and how we operate on them to generate new control laws.

2.1.1 Internal representation

To be able to combine and mutate the control laws throughout the generations, an internal representation of a mathematical function is needed. There are mainly two types of genetic programming to build functions:

tree-based genetic programming: the function or control law is represented by a tree, where the nodes are the operators ($+$, $-$, $\times$, $\div$, sin, exp, etc.) and the leaves are the operands, meaning the arguments of the operators (Duriez *et al.*, 2016).

linear genetic programming (LGP): the function is decomposed in a sequence of unitary or binary mathematical operations. The sequence is then encoded in a four column matrix and read sequentially (Brameier & Banzhaf, 2006).

There is, of course, an equivalence between the two representations. However, we prefer the linear genetic programming representation for its simplicity:

easier for multiple-input control: in the case of multiple-input control, i.e. when there are more than one controller to command, linear genetic programming has the benefit to represent all the associated control law with only one matrix. Thus, the relationship between the control laws is inherent to the representation. The definition of several control laws with only one matrix is detailed lower. Whereas, for tree-based genetic programming, one tree will be needed for each controller, making the interaction between the control laws more difficult to build.

easier to define the genetic operators: because mutation and crossover are defined only on one matrix, instead of several trees or subtrees for tree-based GP. Indeed, sharing substructures between individuals need to be carefully defined, adding more complexity to the genetic operators.

In LGP, the individuals are considered as little computer programs, using a finite number N_{inst} of instructions, a given register of variables and a set of constants. The instructions employ basic operations ($+$, $-$, $\times$, $\div$, cos, sin, tanh, etc.) using inputs (h_i time-dependent functions and s_i sensor signals) and yielding the control commands as outputs. A matrix representation conveniently comprises the operations of each individual. Every row describes one instruction. The first two columns define the register indices of the arguments, the third column the index of the operation and the fourth column the output register. Before execution, all registers are zeroed. Then, the last registers are initialized with the input arguments, while the output is read from the first registers after the execution of all instructions. This leads to a $N_{\text{inst}} \times 4$ matrix representing the control law $\boldsymbol{K}$. The name 'linear' refers to the sequential execution of the instructions. If the operation to execute only requires one operand, only the first column is considered and the second one is ignored. Each column of the matrix has its own range of values following what it codes. For single input control, i.e. when there is only one controller, the control law is read in the first register. For N_{b} controllers, the control laws are read in the first N_{b} registers. Finally, to avoid definition problems, the operators such as division and logarithm are protected to be defined on $\mathbb{R}$ the space of all the real numbers, see Duriez *et al.* (2016).

The registers play the role of memory slots. We distinguish two types of registers:

variable registers: they are registers that can be overwritten while executing an instruction. They help to store intermediate calculations.

constant registers: they are registers that are protected during the reading of the matrix. They are used to store random constants or data of the problem.

Figure 2.2, illustrates how a single instruction is represented in matrix form.

It is worth noting that for a given mathematical expression, there is more than one matrix representation. Indeed, as stated before, there are instructions in the matrix that have no impact in the output registers. Also, the matrix representation takes into account the order of the operations even for operators that are commutative: the control laws $b = s_1 + s_2$ and $b = s_1 + s_2$ will be have different representation while they are the same control laws. As consequence, several instances of the same individual can be present in the population. In order to accelerate the learning and avoid to evaluate redundant individuals, we carry out a numerical test that removes such individuals. section 2.4 gives more details on how we detect and remove redundant individuals.

Figure 2.3 depicts how a matrix of instructions is read to build a control law. We notice that in the instruction matrix, not all instruction lines are useful. Indeed, if an instruction line does not affect one of the output registers then it is,

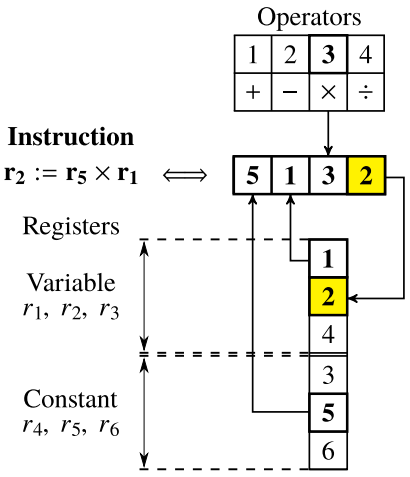


Figure 2.2: Matrix representation of one single operation. The operations and the arguments are numbered separately. This representation is inspired of reverse polish notation. The vertical column are the indices of the registers. In this figure, the result of the instruction is stored in the variable register r_2, overwriting its previous value.

in reality, useless. However, in the process of recombination or mutation, these instructions lines can be 'activated', changing the final control law. Following Brameier & Banzhaf (2006), these 'useless' instruction lines, also called *introns*, play a major in the process of building relevant structures.

With enough instructions and operators, any function can be represented in matrix form. LGP can, for example, reproduce the Taylor expansion of any function until an arbitrary order by deriving the coefficients of the power series. Also, using the matrix representation, we do not constrain, a priori, the structure of the control laws. Of course, the solutions built strongly depend on the library of operators and control inputs given to the algorithm. Indeed, the richness of these libraries defines the complexity of the search space for the optimization problem. The choice of the function libraries is studied in section 2.3 on a dynamical system whereas different sets of control inputs are tested for the control of the fluidic pinball in chapter 3.

Before giving the genetic algorithm in its final form, we first describe its first, the Monte Carlo step, as it is an optimization algorithm on its own.

2.1.2 Monte Carlo optimization

A starting point for genetic programming is the random generation of the first set of individuals. This operation can be seen as a Monte Carlo optimization process. In the LGPC framework, to define a control law is to chose a library for the operators (+, −, cos, etc.), a library for the inputs (a_1, a_2, etc.), the maximum number of instruction $N_{\text{inst,max}}$, or the number of rows in the matrices, the number of variable registers N_{vr} and the number of constant registers N_{cr}. From these parameters, we can generate random matrices that are then read sequentially to

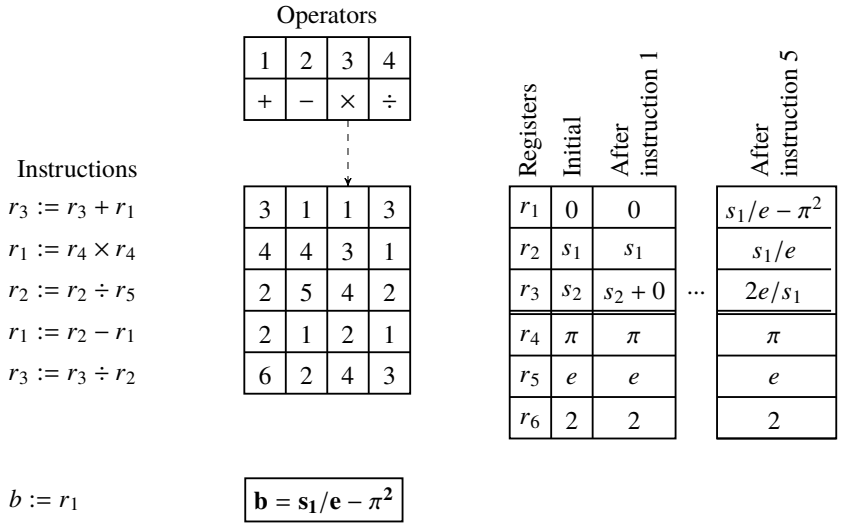


Figure 2.3: Representation of the transcription of a matrix of instructions into a control law (expression framed in the bottom). The matrix (middle) has five instructions. The instruction are displayed on the left and the evolution of the registers after the execution of the first and fifth instruction on the right. The library of operators ($+$, $-$, $\times$, $\div$) and there corresponding index are displayed on the top. The first three registers (r_1, r_2, r_3) are the variable register and the last three (r_4, r_5, r_6) are the constant register. The control law is derived from the expression stored in the first register. In case of multiple input problems the others control laws would be derived from the following registers.

form control laws. The number of instructions for each matrix is randomly drawn from an uniform distribution between 1 and $N_{\text{inst,max}}$. In theory, a Monte Carlo process is enough to solve equation (1.3) but a very large number of individuals might be needed to reach the global optimum of the problem, especially for search spaces of infinite dimension. For pragmatic reasons and also to emulate limited experiment time, we fixed the total number of individuals tested N_i. N_i can also be seen as the total number of cost function callings and also the total number of experiments to run. Figure 2.4 illustrates the Monte Carlo optimization process. N_i individuals are generated randomly, they are all tested and sorted following their cost. The final result of the algorithm is the individual with the lowest cost J, thus the most performing following the cost function criterion. Figure 2.4 depicts the Monte Carlo process for controlling a plant P (framed in the figure).

In the next section, we describe how to create the next generations of individuals from a set of individuals generated thanks to a Monte Carlo process.

2.1.3 The evolution process—Selection and genetic operators

In the following sections, we will describe how the natural selection is emulated and the implementation of the genetic operators. In this section, we detail the steps carried out to create a new population based on a previous one.

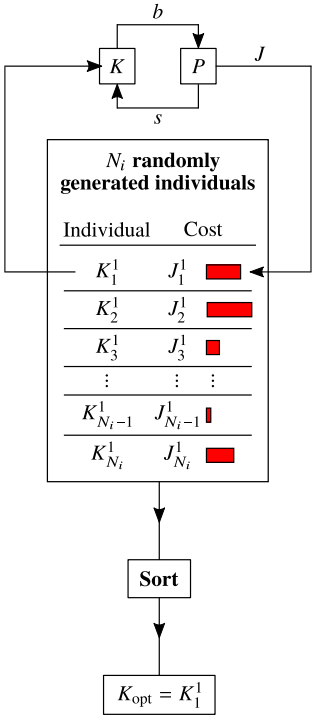


Figure 2.4: A randomly generated set of control laws is evaluated and sorted following their performances. The red bar on the right of each individual is a visual representation of the performance. The smaller the bar is, the better the individual performs. The superscript 1 on the individuals K_i signify that they belong to the first generation of individuals. The top part of the figure, represents the controller K, receiving the sensors signals $\boldsymbol{s}$ as input and giving back the actuation command $\boldsymbol{b}$ to the plant P.

Selection

To create the next generation of individuals, we need, first, to select the most performing individuals to be combined and mutated. The operation of selection is carried out thanks to a tournament selection. The idea of a tournament selection is to select N_{tour} individuals among the N_i individuals in the population. Among the N_{tour} individuals selected, the best one is selected with a probability of P_{tour}. If the best one is not chosen, the second best is chosen with the same probability P_{tour} and so on for all the selected individuals. At the end, if no other individual is chosen, the least performing among the N_{tour} is selected. The choice of N_{tour} and P_{tour} influence the extent to which well-performing individuals are preferred over least-performing ones. This feature is called *selection pressure* and is developed in detail in Wahde (2008). In this study, we choose to follow the recommendations of (Duriez *et al.*, 2016) and set $N_{\text{tour}} = 7$ for a population of $N_i = 100$ individuals and $P_{\text{tour}} = 1$.

Crossover for exploitation

Crossover is the operation of recombination of individuals. It has the potential to extract and combine relevant structures in the individuals. That is why, we refer this genetic operator as the *exploitation operator*. To combine the individuals, two individuals are selected in the population, and their matrices are split in two and the parts are swapped to generate two new individuals, also referred as *offsprings*. Figure 2.5 illustrates the crossover operation between two individuals. It is worth

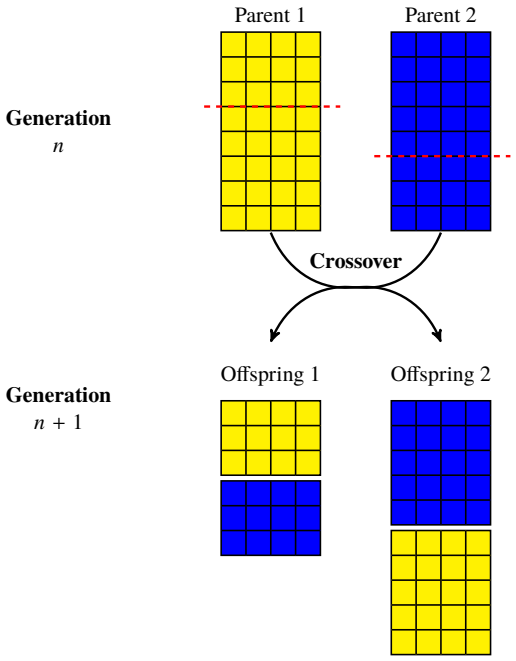


Figure 2.5: Exploitation of the genetic material by recombining two individuals. The parts of the matrices are switched to build new matrices.

noting that, the crossover operation is defined such as the length of the matrices may increase or decrease. To avoid that the size of the matrices explodes, we set a upper limit to the number of rows in the matrix. In practice, this limit is the same $N_{\text{inst,max}}$. If this limit is exceeded, then the operation is restarted until offspring with lesser instructions are built.

Mutation for exploration

Mutation is the operator that generates new sequences in the matrices. The role of this operator is to find new structures, unknown to the population, to improve the solutions. For the mutation of one individual, each row of the corresponding matrix representation has a probability of P_{mut} to be completely changed. The parameter P_{mut} is chosen such as at least one row is changed in the matrix. The change of one line can either have no consequences in the final output, if the instruction stays an intron, or it can also completely change the final output. To improve our exploration potential, we choose to restart the mutation operation

when the mutated individual is identical to the original one. Figure 2.6, depicts the process of mutation for an individual.

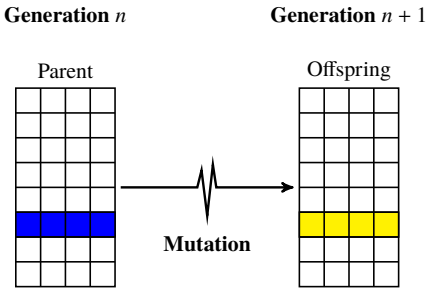


Figure 2.6: Exploration of the search space by creating new instructions and thus new structures.

Of course, there are several ways to define the crossover and mutation operators but we choose to realize the simplest implementation.

From an optimization point of view, crossover is the operator that improves existing solution. Its role is to 'explore' the neighbourhood of a minimum, while mutation is the one that explores the control landscape to discover new minima. The learning principles are illustrated in figure 2.7.

Replication and elitism for memory

In addition to crossover and mutation, we also consider two other operators: replication and elitism. With replication an identical copy of one individual is copied to the next generation, assuring memory of good individuals and allowing future recombination. This elitism operation assures that the best individual is always in the latest generation so that 'the winner does not get lost' throughout the generations.

Figure 2.8, illustrates the complete LGPC algorithm. The first generation of individuals is generated thanks to a random sampling of the individuals (Monte Carlo method). Then, from a generation n, the individuals are all evaluated and sorted following their performances. The best individuals are then selected, thanks to a tournament method, to be modified and recombined with crossover and mutation. Replication and elitism assures a memory of the good individuals. The choice of crossover, mutation or replication to populate the next generation is controlled by the probabilities P_c, the crossover probability, P_m, the mutation probability and P_r the replication probability. They are chosen such as $P_c + P_m + P_r = 1$. The balance between crossover, mutation and replication is thoroughly analyzed in section 2.2.3.

There are several variations of the genetic programming algorithm, where the genetic operators are not separated but applied one after the other and where the offspring replaces the 'parent' individual in the population only if it is better than it. In this study, we choose to follow the classical evolutionary algorithm described in Brameier & Banzhaf (2006) and also employed by Duriez *et al.* (2016).

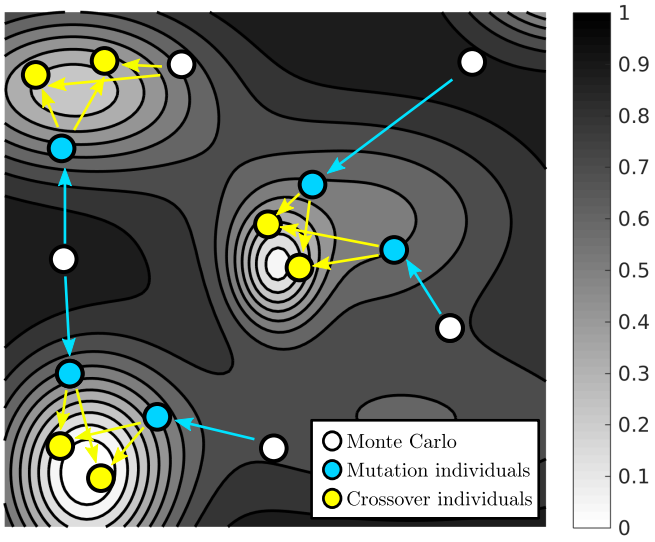


Figure 2.7: Scheme of principle of the exploitation/exploration potential of crossover and mutation. The background is a 2D representation of a control landscape. White regions denoting good performances and dark regions poor performances. Three minima are depicted. After an a priori exploration of the control landscape with Monte Carlo, crossover and mutation improve the evaluated individuals. Crossover individuals have the potential to explore the neighbourhood of a minimum whereas mutation discovers new minima by combining good individuals.

In the next section, we will illustrate the learning process of LGPC and analyze the role of some parameters by stabilizing a Landau oscillator.

2.2 Stabilization of the Landau oscillator

To illustrate the linear genetic programming learning methodology, we optimize a controller for a damped Landau oscillator. First, we present the dynamical system used for this study. Then, we describe the learning mechanisms of LGPC by stabilizing the Landau oscillator with increasingly more complex algorithms. We start with a Monte Carlo optimization then we stabilize the oscillator with the LGPC algorithm described in fig. 2.8. In this section, we focus on the meta-parameters (P_c, P_m, P_r) to analyze the role of crossover, mutation and replication in the learning process. We reveal, among other things, that there is sweet spot in the meta-parameter space.

All the simulations have been carried out thanks to our own LGPC code developed on MATLAB and also available for the free software GNU Octave.

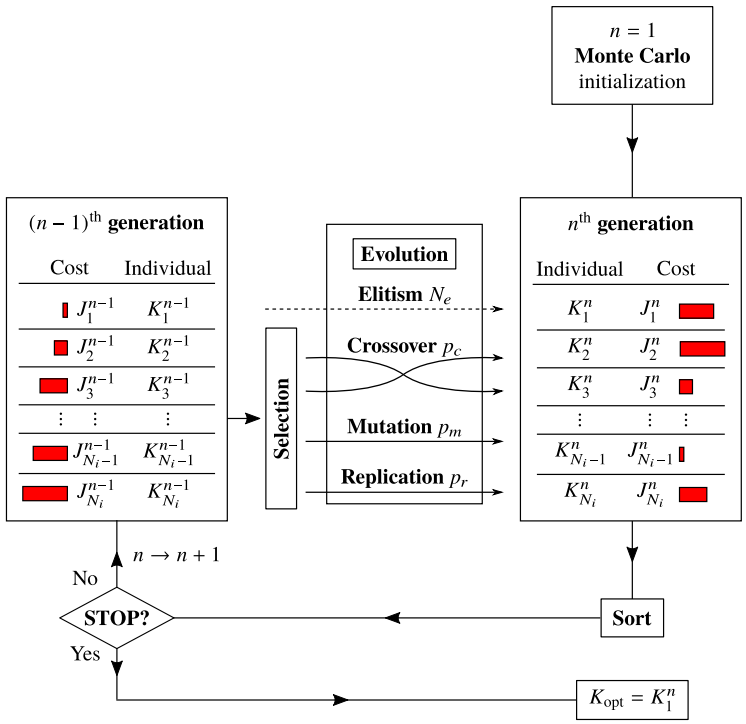


Figure 2.8: Linear genetic programming algorithm. Each generation is built from the previous one thanks to the genetic operators.

2.2.1 The damped Landau oscillator

The controlled dynamical system

The damped Landau oscillator is a system of two coupled ordinary differential equations with a nonlinear damping of the growth rate. Despite its simplicity, it describes a fundamental oscillatory process at the heart of physical mechanisms such as the von Kármán vortex shedding behind a cylinder (Luchtenburg *et al.*, 2009). To control the oscillator a forcing term b is introduced in the second equation. The systems reads:

$$\begin{cases} \dot{a}_1 &= \sigma a_1 - a_2 \\ \dot{a}_2 &= \sigma a_2 + a_1 + b \\ \sigma &= (1 - a_1^2 - a_2^2) \end{cases} \tag{2.1}$$

For $b = 0$, we have an oscillator of growth rate σ, angular frequency 1, period $T = 2\pi$ and fixed point $(0, 0)$. For an initial condition close to the fixed point, the quadratic terms in σ are negligible, leading to an exponential growth. When the system is far from the fixed point, the growth is damped due to the quadratic terms, stabilizing the oscillator to the limit cycle of radius $\sqrt{1} = 1$. The same reasoning for an initial condition outside the circle of radius 1 shows that the limit cycle is globally stable. The uncontrolled dynamics are depicted in figure 2.9 a) and b). The control on the second equation has the effect of pushing the system upwards or downwards following the sign of b. Upwards if $b > 0$ and downwards if $b < 0$.

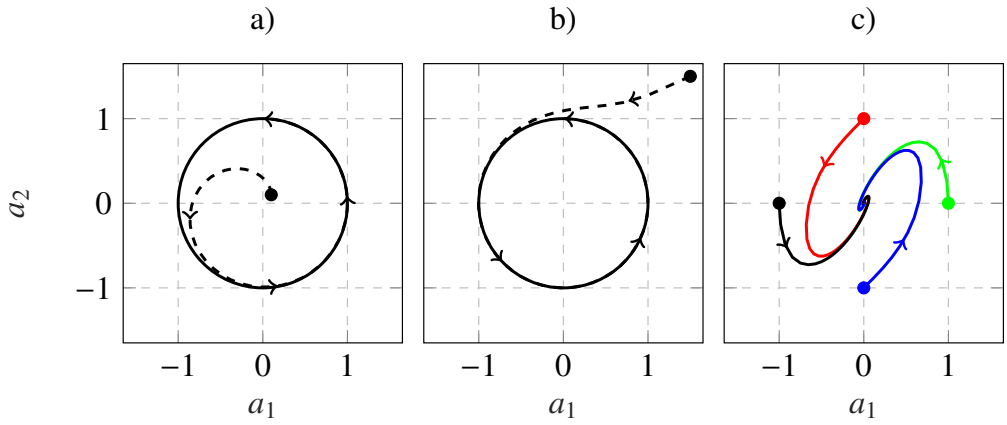


Figure 2.9: a) Phase portrait of the oscillator with no actuation ($b = 0$) with $(a_1, a_2)_{t=0} = (0.1, 0.1)$ as initial condition. b) Phase portrait for the initial condition $(a_1, a_2)_{t=0} = (1.5, 1.5)$. In both cases the system converges towards the limit cycle of radius 1. c) Optimal linear control for four different initial conditions.

Objective and cost function

Our control objective is to bring the system to the fixed point $(a_1, a_2) = (0, 0)$ from the limit cycle. For the cost function, we consider the two terms J_a and J_b of equation 1.2:

$$\begin{cases} J_a &= \overline{a_1^2 + a_2^2} \\ J_b &= \overline{b^2} \end{cases} \tag{2.2}$$

where

$$\overline{f(t)} = \int_0^{20\pi} f(t)dt.$$

J_a is the integral of the distance to the fixed point and J_b measures the total energy delivered for the control. Both are integrals quantities over 10 periods as we are interested not only in the final solution but also on the trajectory. In order to assure a general solution, we consider four initial conditions on the limit cycle: $(1, 0)$, $(0, 1)$, $(-1, 0)$, $(0, -1)$. The cost function J is then a mean value between these four initial conditions.

For the uncontrolled dynamics, we have $a_1^2 + a_2^2 = 1$ and $b^2 = 0$, therefore $J_a = 20\pi \approx 63$ and $J_b = 0$. The unforced cost is then $J_0 = 20\pi$. J_0 is our reference for future controls.

Actuation limitation

For the control, we decide to add a saturation function for the actuation command. This motivated by two reasons: first, we do not want to explore actuations with excessive values that we know they won't work. As a result we restrict the search space and accelerate the learning. The second reason is that with a saturated control we are closer to experimental conditions where the maximum power for control is limited. The one-dimensional saturation function is described in

equation 2.2.1. Of course, different thresholds can be set for each actuators.

$$h: \begin{array}{ccc} \mathbb{R} & \longrightarrow & \mathbb{R} \\ x & \longmapsto & \begin{cases} b_{min} & \text{if } x < b_{min}, \\ x & \text{if } b_{min} \leq x \leq b_{max}, \\ b_{max} & \text{if } x > b_{max}. \end{cases} \end{array}$$

In the following, we will not differentiate the actuation command $\boldsymbol{b}$ and the bounded actuation command $\boldsymbol{h}(\boldsymbol{b})$ to alleviate the notations.

Optimal linear solution

We compute the optimal linear control of the control problem thanks to the `fminsearch` function of MATLAB. `fminsearch` is a derivative-free method based on Nelder-Mead simplex method (Nelder & Mead, 1965). For this problem, we optimize the parameters of a linear control to stabilize the Landau oscillator. The optimal linear control is detailed in equation 2.3. This solution has been found with a random initialization of the `fminsearch` function. We do not know if this solution is the optimal one but it is the best one so far, thus we refer to it as 'the optimal linear control'. In fact, section 2.2.3 reveals that this solution is not the optimal linear control as another linear solution performs better. For now, we only focus on the solution in equation 2.3, that is only a local minimum. The associated cost is $J_{opt} = 3.3403$ which corresponding to a $\Delta J_{\text{opt}}/J_0 = 94.68\%$ reduction of the unforced cost.

$$b_{\text{opt}} = 2.4061a_1 - 3.0984a_2 \tag{2.3}$$

Figure 2.10 shows the controlled Landau oscillator and the actuation map for the control 2.3. We notice that the symmetry of the problem is respected and that the system is successfully brought to the fixed point $(a_1, a_2) = (0, 0)$ in less than a period. The strategy employed by the optimal control is to vigorously push the system towards the fixed point in the zones where the dynamics naturally lead them closer in terms of a_2. Once the system is close enough to the fixed point, the intensity of the actuation decreases. The slope of the actuation is determined by the cost function of the problem, with different cost functions, different slopes are expected. Indeed, as it is defined, solutions reaching the fixed point faster and with least actuation are favored. The linear optimal control is then the fastest trajectory, with the given constraints, to stabilize the oscillator. It is worth noting that the optimal control leads the system beyond the limit cycle for a short period of time. Evidently, if the control would be on the first equation instead of the second, the results would be symmetrical to the $a_1 = a_2$ axis.

Our goal is then to stabilize the Landau oscillator with LGPC and to find a better solution than the optimal linear control.

2.2.2 Monte Carlo reference

In this section, we stabilize the Landau oscillator with a Monte Carlo optimization process. This optimization is used as a reference to compare the learning process of LGPC.

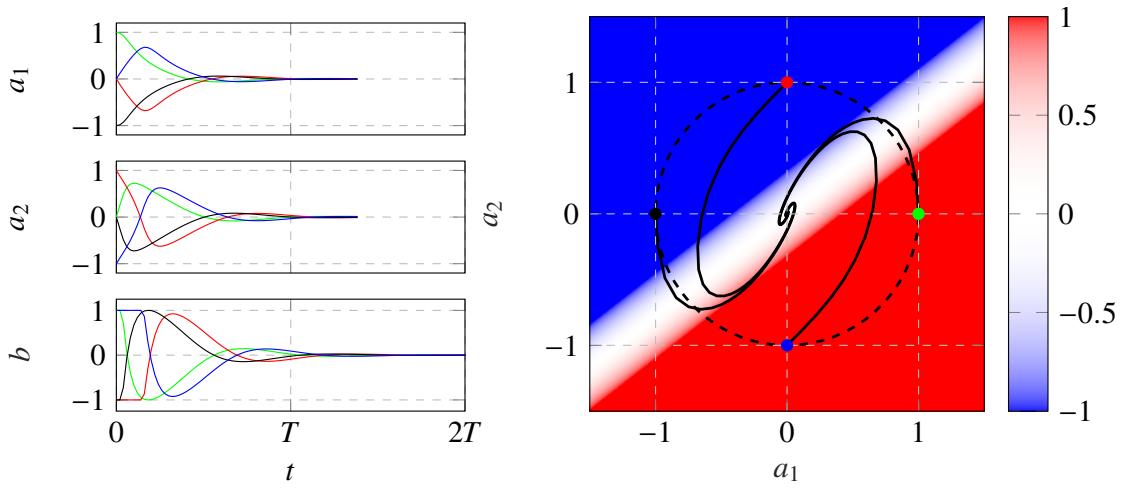


Figure 2.10: Visualization of the linear optimal control law and actuation map. Time series of a_1, a_2 and b for the two first periods is presented on the left. On the right, is depicted the trajectories of the controlled system for the four initial conditions and the actuation map on the background. Red regions mean that the system is pushed upwards and blue regions downwards.

The search space

In the linear genetic programming framework, the search space is defined by the function and inputs library. In the following, we designate by parameters, the meta-parameters of the algorithm as we do not consider other configurations for the Landau oscillator. The other parameters than can influence the exploration of the landscape are the number of variable registers N_{vr}, the number of constant registers N_{cr} and the maximum number of instructions $N_{\text{inst,max}}$. Indeed if N_{vr}, N_{cr} and $N_{\text{inst,max}}$ are small then the control laws are bound to contain only few operations, thus complex expressions become inaccessible. This aspect is further studied in section 2.3.

Table 2.1 summarizes the parameters chosen for the Monte Carlo optimization. From these parameters, we can compute the total number of control laws in the

parameter	description	value
	function library	$F_1 = \{+, -, \times, \div\}$
$\boldsymbol{s}$	controller inputs	a_1, a_2
N_{vr}	number of variable registers	3
N_{cr}	number of constant registers	3
$N_{\text{inst,max}}$	max number of instructions	5

Table 2.1: Parameters for the control ansatz.

search space by listing all the possible combinations. For this, we multiple the possible values for each columns of the instruction matrix and elevate the result at the power of the number of rows in the matrix. As we allow matrices of different sizes, we need to add the results for each possible number of rows. Expression 2.4 gives the order of magnitude of the number of possible control laws with the chosen parameters.

$$\sum_{q=1}^{N_{\text{inst,max}}} [N_{\text{r}} \times N_{\text{r}} \times N_{\text{o}} \times N_{\text{vr}}]^q \approx 1.5 \times 10^{13} \tag{2.4}$$

where

$N_{\text{inst,max}}$: maximum number of instructions.
N_{r}: total number of registers: $N_{\text{r}} = N_{\text{vr}} + N_{\text{cr}}$.
N_{o}: number of functions in the library or mathematical operators.
N_{vr}: number of variable registers, where to store intermediate calculations.

This number 1.5×10^{13} represents the total number of matrices. We assumed that all registers where initialized with different values which is often the case in practice, unless we want to favor a given information. Table 2.2 presents the initialization of the registers for the oscillator stabilization problem before the execution of the instruction matrices.

variable registers		constant registers	
r_1	0	r_4	$c_1 = -0.94$
r_2	s_1	r_5	$c_2 = -0.05$
r_3	s_2	r_6	$c_3 = -0.72$

Table 2.2: Initialization of the registers. The constants c_i have been randomly taken in the range $[-1, 1]$.

Also, among the 1.5×10^{13}, there are, of course, a lot of identical and equivalent control laws, due to overwriting, useless instructions, etc. This feature is valuable as the expression of the global minimum is no longer unique and can be built in different ways.

A randomly generated control law

Before performing a Monte Carlo optimization, let's first generate a random control law. Figure 2.11 shows an instruction matrix randomly generated and depicts the process of translation to a control law. The resulting control is a constant beyond the actuation limits, thus this control law is equivalent to constant forcing at $b = 1$. Figure 2.12 gives a visualization of the actuation. We notice that with such action, the point $(a_1, a_2) = (-1, 0)$ becomes a fixed point. Indeed, without control this point is the unique point where $(\dot{a}_1, \dot{a}_2) = (0, -1)$, thus with the actuation, this point becomes a fixed point.

Operators

1	2	3	4
+	−	×	÷

Instructions

$r_1 := r_2 \times r_4$

$r_2 := r_3 \div r_4$

$r_1 := r_4 \div r_6$

2	4	3	1
3	4	4	2
4	6	4	1

Registers	Initial	Instruction 1	Instruction 1	Instruction 3
r_1	0	$c_1.s_1$	$c_1.s_1$	c_1/c_3
r_2	s_1	s_1	s_1	s_2/c_1
r_3	s_2	s_2	s_2	s_3
r_4	c_1			
r_5	c_2			
r_6	c_3			

$b := r_1$ **b = 1.30**

Figure 2.11: Matrix of the randomly generated control law and translation.

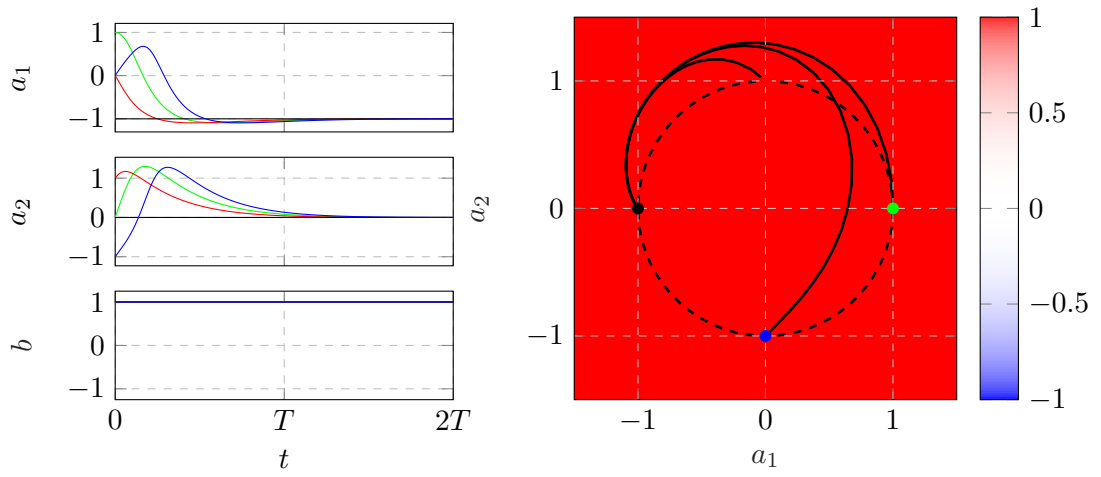


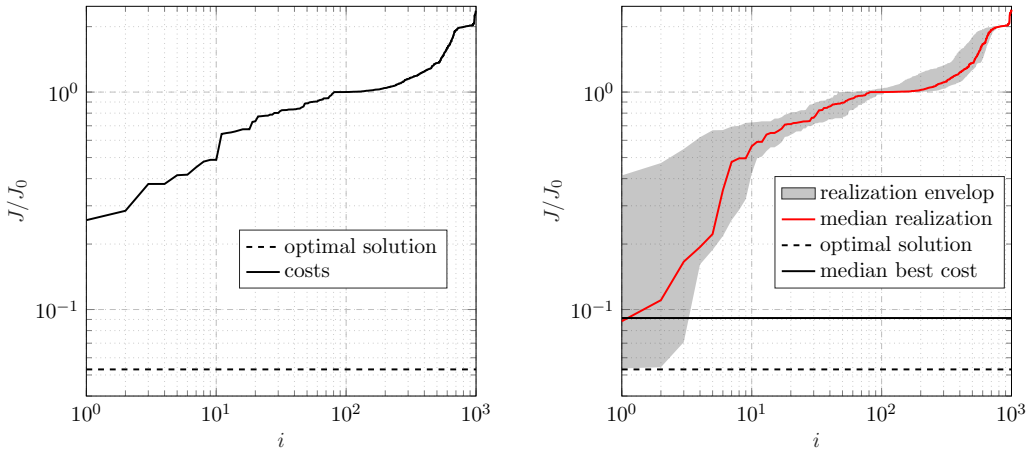
Figure 2.12: Visualization of the randomly generated control law and time series of a_1, a_2 and b.

Monte Carlo optimization

Now, let's perform a Monte Carlo optimization with the parameters of 2.1. In order to emulate experimental conditions, we only perform $N_i = 1000$ evaluations of the cost functions. Thus, we only generate 1000 control laws to be evaluated. Our Monte Carlo implementation is optimized with the screening of redundant individuals. More informations are provided in section 2.4. Figure 2.13a shows the distribution of costs in log-log scale. The values are sorted and normalized with the unforced cost J_0. We notice in figure 2.13a, that the performance of the best solution after 1000 evaluated control laws is not as good as the optimal linear solution. The associated cost reduction is $\Delta J/J_0 = 74\%$.

However, as we rely on a stochastic process, only one realization of the optimization process is not enough to estimate the performance of the method. That is why, we performed $N_\rho = 100$ realizations of Monte Carlo to have statistically

relevant realizations. Figure 2.13b shows the envelop of the one hundred realizations of Monte Carlo. The cost reduction of the best individuals in the realization envelop goes from 58.55% to 94.64%, and the median performance reduces the cost by $\Delta J/J_0 = 91.17\%$. The median performance is defined as the $\rho = 50$th best realization.



(a) (b)

Figure 2.13: (a) Distribution of the cost of 1000 randomly generated control laws for the Landau oscillator. (b) Envelop of 100 realizations of Monte Carlo. The median realization ($\rho = 50$th best) is depicted in red. In both (a) and (b) the individuals are sorted following their cost. The black horizontal line represents the cost of the best individual of the median realization. The associated cost reduction is $\Delta J/J_0 = 91.17\%$. The dashed line represents the cost of the linear optimal solution ($\Delta J_{\text{opt}}/J_0 = 94.68\%$). The vertical and horizontal axis are in log scale.

Figure 2.14 displays the best individuals for each one the one hundred realizations. We notice that the first twenty-three control laws are all similar. They resemble the linear optimal control with different band width. Starting from the 33th run, the best individuals takes more than two periods to reach the fixed point. We remark a majority of linear-like solutions, especially among the best ones. The first nonlinear solution is ranked 49th and the other ones are ranked beyond the 80th realizations. We also notice that those nonlinear solutions still present a symmetry in their actuation map.

The best control law of the median realization is:

$$b_{\text{MC,median}} = 2.0907(a_1 - a_2)$$

and the best control law of the best realization is:

$$b_{\text{MC,best}} = 3,7747(a_1 - a_2).$$

The expressions have been simplified compared to the expressions computed by LGPC. The two controls have the same form but with a different coefficient. The

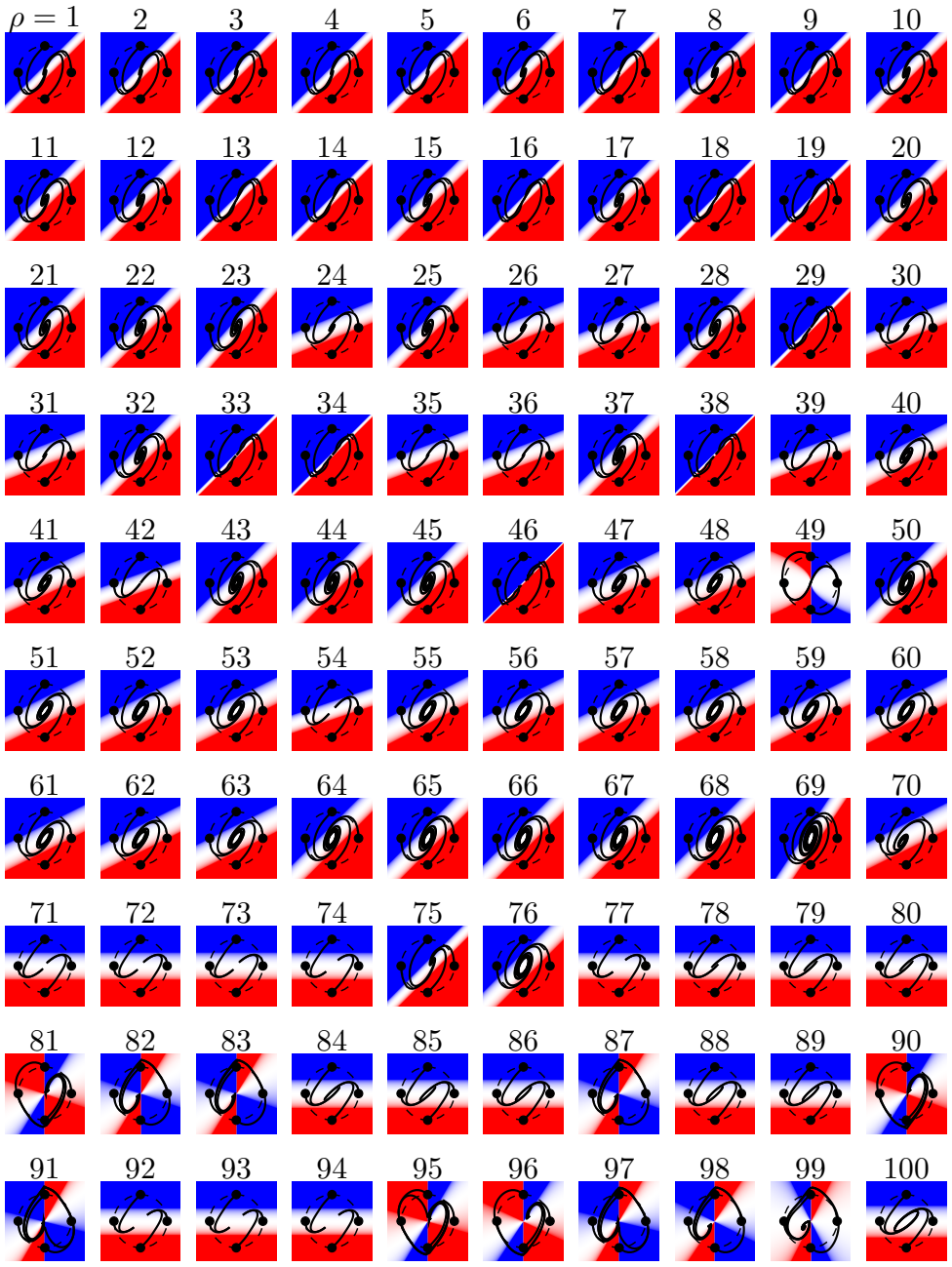


Figure 2.14: Visualization of the best control laws for one hundred 100 Monte Carlo realizations. The realizations are sorted by the cost of their best individual. The numbering corresponds to the different realizations. 1 being the best realization and 100 the worst. The trajectories for the four initial conditions are plotted for two periods.

coefficients are related to the width of the control band and thus at the speed of convergence of the oscillator towards the fixed point. This difference can be appreciated in figures 2.15 and 2.16

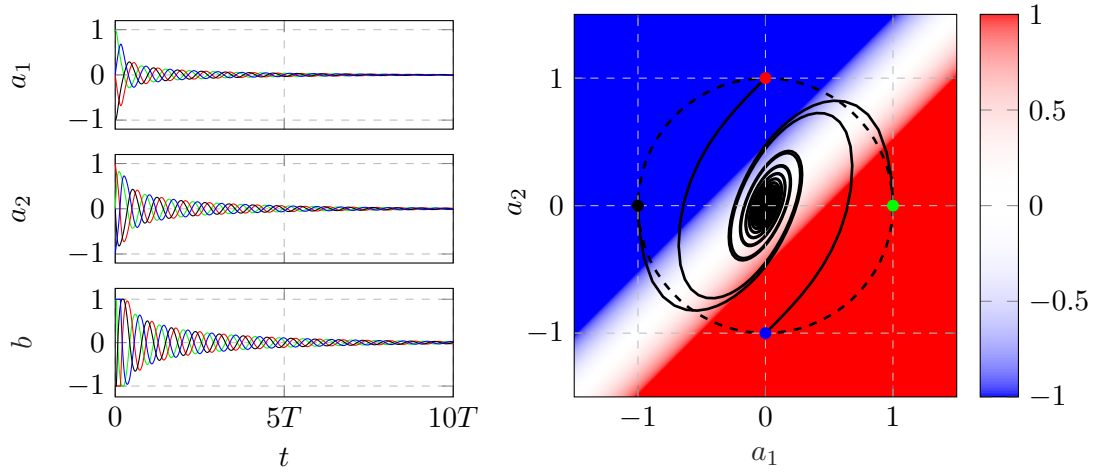


Figure 2.15: Visualization of the best control law for the median Monte Carlo realization. The Landau oscillator is controlled by $b_{\text{median}} = 2.0907(a_1 - a_2)$

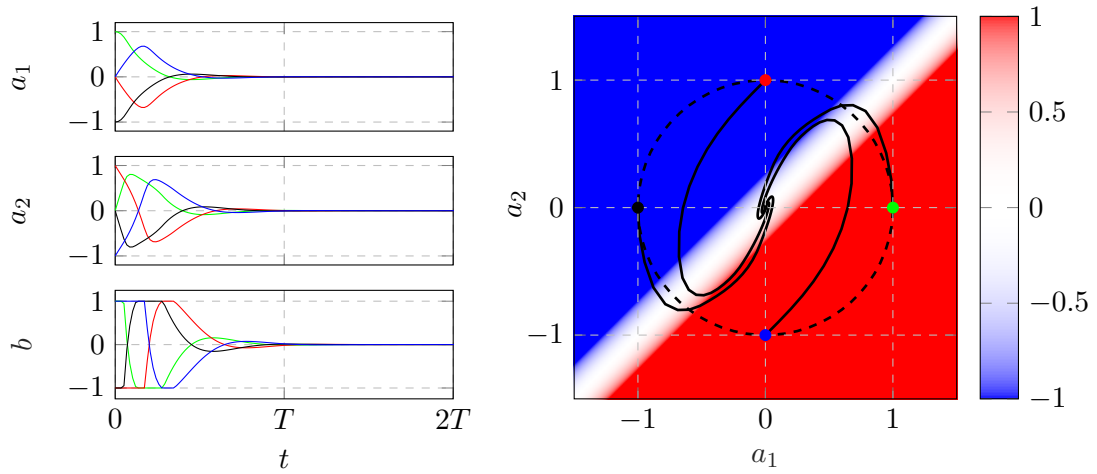


Figure 2.16: Visualization of the best control law for the best Monte Carlo realization. The Landau oscillator is controlled by $b_{\text{best}} = 3,7747(a_1 - a_2)$

2.2.3 LGPC optimization

In this section, we stabilize the Landau oscillator with the LGPC algorithm. We especially study the influence of the operators probability (P_c, P_m, P_r). For this, we ran LGPC optimizations with different combinations of (P_c, P_m, P_r). We start from $(P_c, P_m, P_r) = (1, 0, 0)$, full crossover, and we increase and decrease each parameter with a step of 0.1. We recall that $P_c + P_m + P_r = 1$. There are in total 66 combination of parameters to test. Also, as in the Monte Carlo optimization, only one realization is not enough to estimate the performance of the algorithm, thus we run $N_\rho = 100$ realizations for each combination of probabilities. To have a fair comparison between Monte Carlo and LGPC, we also evaluate $N_i = 1000$ individuals distributed in a population of $N_{\text{popsize}} = 100$ individuals that evolves 9 times for a total of $N_\rho = 10$ generations. The parameters are summarized in

table 2.3. Figure 2.17 summarizes the parameters for one combination of probabilities. The parameters used for the description of the search space are the same as the ones used for Monte Carlo.

parameter	description	value
	function library	$F_1 = \{+, -, \times, \div\}$
$\boldsymbol{s}$	controller inputs	a_1, a_2
N_{vr}	number of variable registers	3
N_{cr}	number of constant registers	3
$N_{\text{inst,max}}$	max number of instructions	5
N_{popsize}	population size	100
N_G	number of generations	10
N_{tour}	tournament size	7
N_e	elitism	1
P_c	**crossover probability**	P_c
P_m	**mutation probability**	P_m
P_r	**replication probability**	$1 - P_c - P_m$

Table 2.3: Parameters for LGPC. The operators probability (P_c, P_m, P_r) are discussed in section 2.2.3.

Influence of genetic probabilities P_c, P_m and P_r

Figure 2.18 presents the results of the 66 tests.

The replication-only is, as expected, the worse configuration. Indeed, this is equivalent to run a Monte Carlo optimization with only $N_i = 100$ individuals and no more individuals. The addition of crossover and mutation progressively improves the replication-only optimization. We notice that both crossover-only and mutation-only LGPC are less efficient than the combination of crossover, mutation and replication. Also from configuration 52 and lower, $(P_c, P_m, P_r) = (0.4, 0.1, 0.5)$, the median cost is lower than the median cost of Monte Carlo optimization. We note, in particular, that for a replication probability less than 0.5, LGPC performs better than Monte Carlo regardless of the crossover and mutation probability. This shows that crossover and mutation allow to effectively explore the search space. The operations of crossover and mutation, defined on the control law representations, manage to improve the individuals throughout the generations. Also it is the combination of crossover and mutation that gives the best results. It is worth noting that crossover-only LGPC (ranked 43) is better than mutation-only LGPC (ranked 47). This indicates that crossover and exploration has a better learning potential than the explorative power of mutation. This feature may be related to the choice of the problem as only few minima are expected, rendering the exploration operator less useful.

Also, we highlight that the median cost of the 21 first configurations are all equal. Indeed, the median realizations of these 21 configurations all manage to

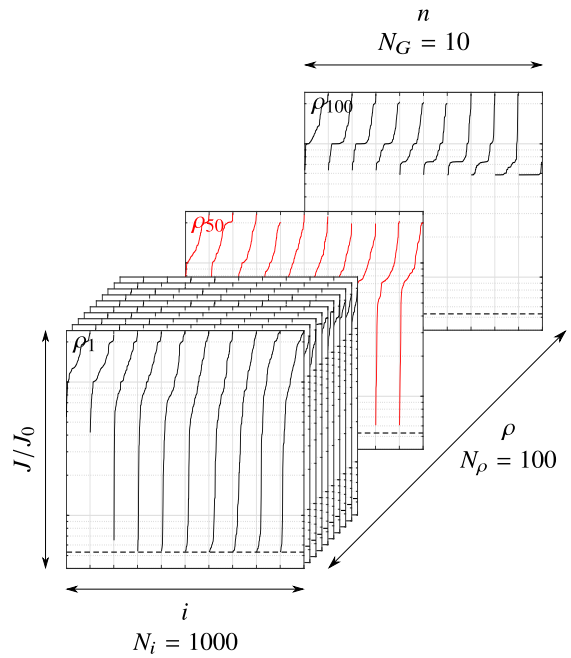


Figure 2.17: Conceptual figure describing all the parameters for the $N_\rho = 100$ realizations of one probability configuration. The realizations or indexed by ρ and the generations by n. For each generation, the individuals are sorted following their cost.

build the same controller, in 13 different ways:

$$b_{\text{LGPC,median}} = 2a_1 - 3a_2$$

that reduces the cost by $\Delta J/J_0 = 94.62\%$, see figure 2.19. We recall that the linear optimal control reduces the cost by $\Delta J_{\text{opt}}/J_0 = 94.68\%$. Surprisingly, even for such simple configuration, the replication operator also plays a non-negligible part as the 21 first configurations are all equivalent. We can conclude that memory is beneficial to explore a search space even when it is expected to have few minima.

Among all these configurations, there are 47 configurations that managed to build control laws that are better than the linear optimal control. Figure 2.20 depicts the number of realizations that managed to build a better control law than the optimal linear control. We notice that there is a clear region in the parameter space that favors better solutions, specially around $(P_c, P_m, P_r) = (0.6, 0.2, 0.2)$ and $(P_c, P_m, P_r) = (0.6, 0.3, 0.1)$.

Once again, we notice that crossover-only is better than mutation-only. Also replication seems essential to learn complex control laws.

The best control law all probability combinations and realizations combined is found for the configuration $(P_c, P_m, P_r) = (0.5, 0.4, 0.1)$ and reads after simplification:

$$b_{\text{best}} = \frac{(a_1 - 2a_2)^2(a_1 - a_2)}{a_1}.$$

It allows a cost reduction of $\Delta J_{best}/J_0 = 94.78\%$, better than the optimal linear control, revealing that the linear control is in fact not the global minimum of the

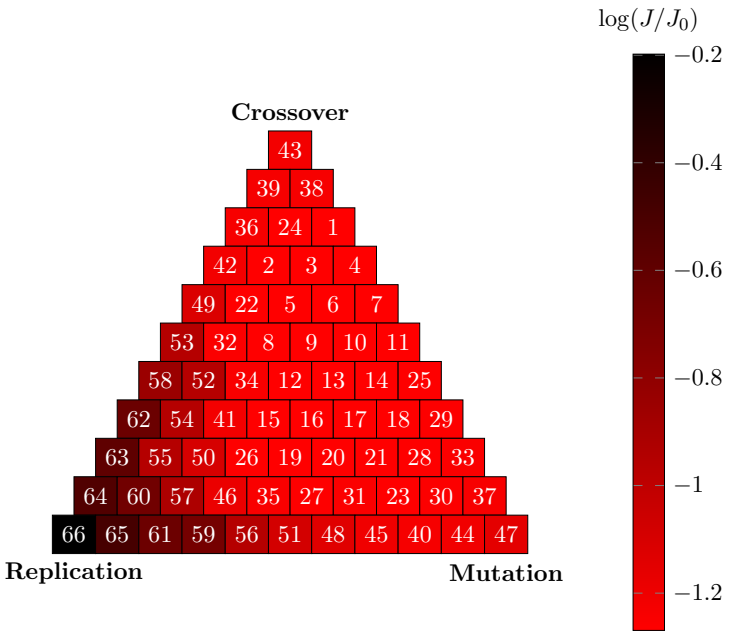


Figure 2.18: Performance of LGPC regarding the parameters ($P_c, P_m, P_r = 1 - P_c - P_m$). Each block of the triangle corresponds to one probability configuration. A neighbour block means a variation of 0.1 in the genetic operator probabilities. Thus the top block is crossover only $(1, 0, 0)$, the bottom-left block corresponds to replication only $(0, 0, 1)$ and the bottom-right block corresponds to mutation only $(0, 1, 0)$. For each configuration 100 realizations have been carried out. The color code symbolize the performance of the median cost for each configuration. The configurations are ranked following their relative performance, 1 being the best configuration and 66 the worst. The first 21 configurations have all the same minimum value $\log_{10}(J/J_0) = -1.26$. From the 52th rank, the configurations has a higher median cost than the median cost of Monte Carlo optimization.

problem. It is worth noting that b_{best} is built only from a_1 and a_2. No constants have been used to 'adjust' the control law. This may be explained by the fact that tuning constants from the initial random ones is costly in terms of instructions. Figure 2.21 shows the phase portrait of the controlled oscillator with the best control found. The actuation map displays the nonlinear effect of the control. The actuation is globally similar to the linear control, however the control intensifies near the fixed point. An absence of control in the diagonal direction prevents the system to go beyond the limit cycle unlike the optimal linear control and thus converges even faster. The curvature of the lobes that delimit the control must play a role in the fast convergence of the system towards the fixed point. We note that the oscillator converges linearly when close enough to the fixed point. Thus, LGPC manages to find an unexpected structure for the control law that performs better than the linear optimal control. We can expect that this nonlinear control law can be improved with adequate constant tuning.

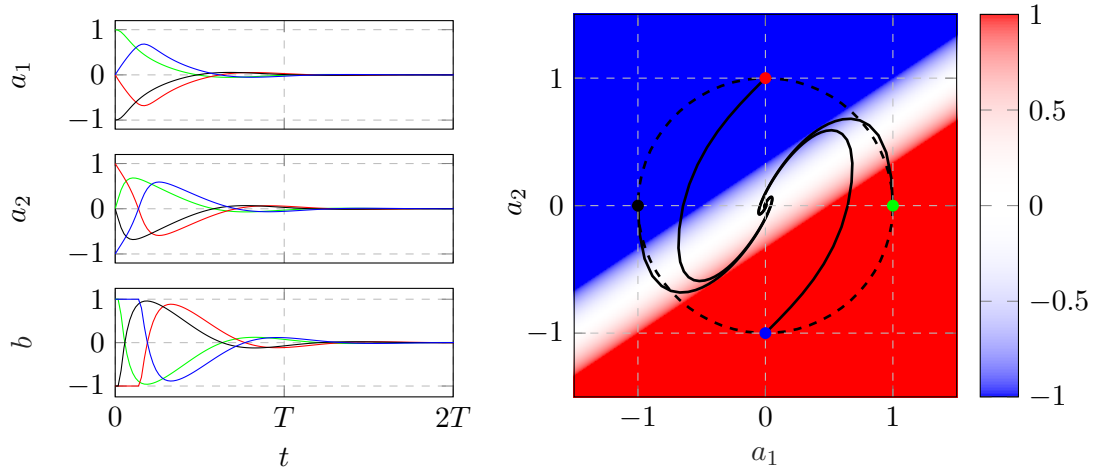


Figure 2.19: Visualization of the controlled Landau oscillator with $b_{\text{LGPC,median}} = 2a_1 - 3a_2$.

If we take a look at the best individual of the $(P_c, P_m, P_r) = (0.6, 0.3, 0.1)$ combination, all realizations combined, we notice that the best control law is not only linear:

$$b_{(0.6,0.3,0.1),\text{best}} = 2.4353a_1 - 3.1238a_2$$

but the associated cost reduction $\Delta J/J_0 = 94.70\%$ is also higher than the 'optimal' linear control found with `fminsearch` $\Delta J_{\text{opt}}/J_0 = 94.68\%$. This shows that the solution (equation 2.3) is, in fact only a local minimum of the space of linear control laws. When we run a `fminsearch` with this new control law as initial condition, we do not observe any improvement thus we can assume that this new solution is another local minimum in the space of linear control laws. One explanation of the presence of several minima in such simple dynamical system may be the nonlinear saturation of the actuation command. We do not display the phase portrait of the best linear control found with the $(P_c, P_m, P_r) = (0.6, 0.3, 0.1)$ combination because it is almost identical to figure 2.10 and no differences can be seen with the naked eye.

Here, we shall stop the analysis of the influence of the probability operators (P_c, P_m, P_r) to focus on other meta-parameters. In the following, we will consider the $(P_c, P_m, P_r) = (0.6, 0.3, 0.1)$ combination as it is one of the configurations that have the lowest median cost value and is the one that have the most realizations with better solutions that the linear 'optimal' control.

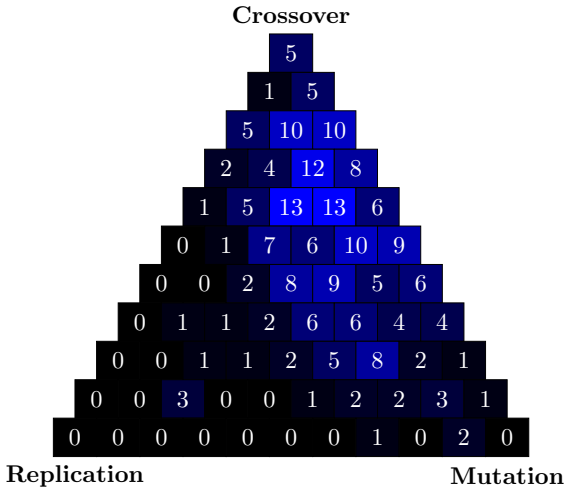


Figure 2.20: Number of realizations that managed to build a control law more efficient than the linear optimal control. The configurations are distributed as in figure 2.18. The color intensity indicates the number of individuals.

2.3 Parametric study of LGPC

In this section, we present the influence of other parameters on the learning process of LGPC. In particular, we investigate the role of the population size N_{popsize}, the maximum number of instructions $N_{\text{inst,max}}$ and the choice of the function library. For a fair comparison between the parameters, we run all LGPC optimizations with $N_i = 1000$ individuals. Seven population sizes are tested: 10, 20, 50, 100, 200, 500, 1000 with the adequate number of generations: 100, 50, 20, 10, 5, 2, 1. The 1000 individual run is equivalent to a Monte Carlo optimization, Also, we vary the maximum number of instructions $N_{\text{inst,max}}$, nine values are tested: 2, 5, 10, 20, 50, 100, 200, 500, 1000. This means that for $N_{\text{inst,max}} = 2$, the instruction matrix only contains two rows, and one thousand rows for $N_{\text{inst,max}} = 1000$. Finally, we look at the impact of the function library in the optimization process. We investigate two libraries:$F_1 = \{+, -, \times, \div\}$ and $F_2 = \{+, -, \times, \div, \exp, \tanh, \sin, \cos, \log\}$. The rest of the parameters are the same as the previous section and are summarized in table 2.4. The operator probabilities $(\Pr, P_c, P_r)$ are chosen according to section 2.2.3.

To have a relevant estimation of the performance of each combination of parameters, we realize $N_\rho = 100$ runs for each combination of parameters.

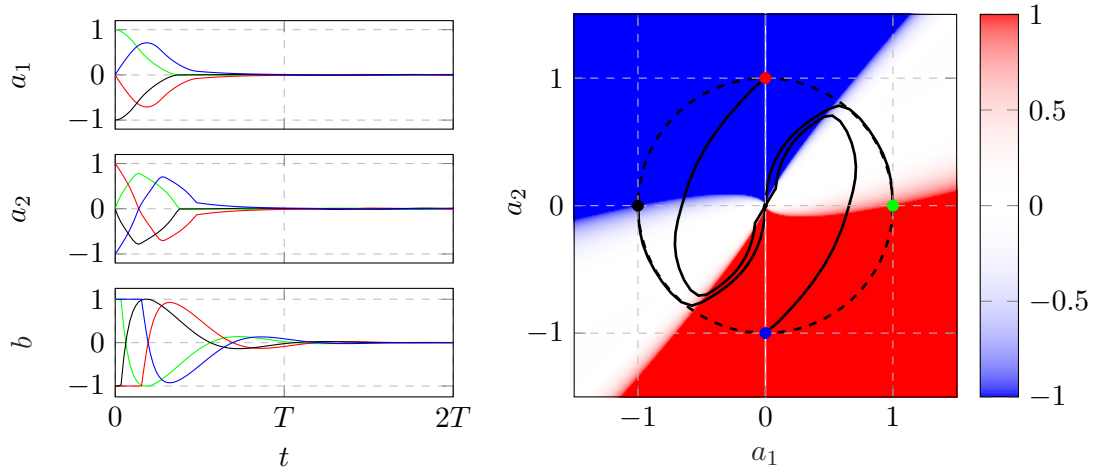


Figure 2.21: Visualization of the best control law derived thanks to LGPC with the $(P_c, P_m, P_r) = (0.5, 0.4, 0.1)$ combination.

2.3.1 $F_1 = \{+, -, \times, \div\}$

Let's first look at the results for the library employed in section 2.2.3. In figure 2.22, we notice, first, that there is a favored region in the parameter space for a best median performance. This region includes $N_{\text{inst,max}} \in [5, 10]$ and $N_{\text{popsize}} \in [50, 100]$ and the performance decreases smoothly beyond this region. The best configuration of all is $(N_{\text{inst,max}}, N_{\text{popsize}}) = (5, 100)$.

We remark that for a fixed population size, performance decreases when we increase the maximum number of instructions. This can be explained by the fact that as we allow more instructions, the search space becomes all the more bigger. Indeed, with more instructions, more and more complex control laws can be built, with the possibility to have a high level of function nesting. When a large number of instructions is employed, a bigger number of generations is preferred. Indeed, as the matrices have much more rows, several rounds of crossover and mutation will be needed to shape the control law by selecting the adequate operators, registers and avoiding that further instructions destroy the intermediate expressions. This also shows that too much instructions can be a hindrance for the learning process. For the 'simple' problem that is the stabilization of the oscillator where rather simple control laws are expected to work, a small number of instructions seems to be favored. However, to chose too few instructions is not advantageous as we lose too much in complexity of the control laws. The maximum number of instructions should be large enough to include relevant solutions but not too much otherwise, the search space becomes too large to be effectively explored in a reasonable number of generations.

Concerning the population size, we notice that population sizes around 50 and 100 are favored. Also, some Monte Carlo optimizations, especially with $N_{\text{inst,max}} = 2$ and 5, performed better than LGPC with $N_{\text{popsize}} = 10$. This is because small populations are more vulnerable to random variations of the population due to mutation; a given structure can more easily take over the population even if it is

parameter	description	value
	function library	$F_1 = \{+, -, \times, \div\}$ $F_2 =$ $F_1 \cup \{\exp, \tanh, \sin, \cos, \log\}$
$\boldsymbol{s}$	controller inputs	$a_1,\ a_2$
N_{vr}	number of variable registers	3
N_{cr}	number of constant registers	3
$N_{\text{inst,max}}$	**max number of instructions**	[2,5,10,20,50,100,200,500,1000]
N_{popsize}	**population size**	[10,20,50,100,200,500,1000]
N_G	**number of generations**	[100,50,20,10,5,2,1]
N_{tour}	tournament size	7
N_e	elitism	1
P_c	crossover probability	0.6
P_m	mutation probability	0.3
P_r	replication probability	0.1

Table 2.4: Parameters for the parametric study of LGPC. The studied parameters are in bold. Parameters are separated in two sets: (top) parameters that define the control laws, (bottom) parameters that define the learning process.

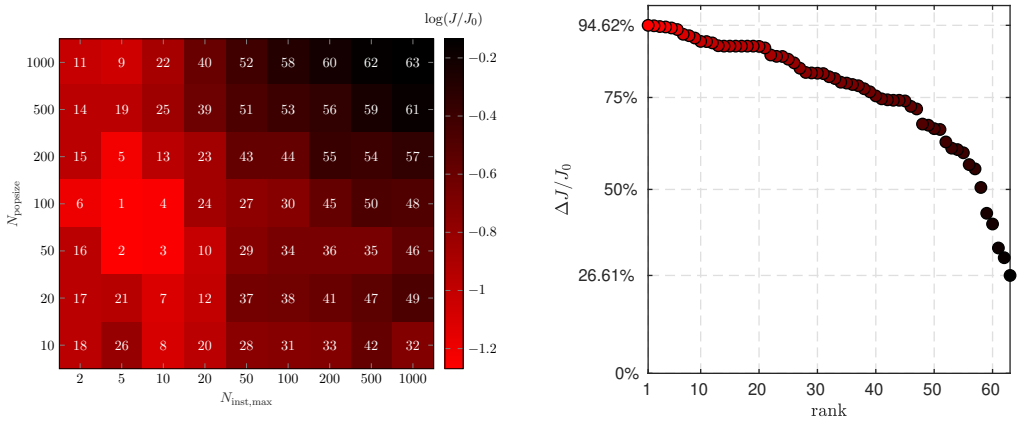
not the best one, discarding relevant structures and leading the population into a local minimum. This effect is also referred in biology as *genetic drift*.

Figure 2.23 shows that small population sizes and moderate number of instructions tend to build better solutions. Around 10% of the realizations manage to build more efficient solutions than the linear optimal control. The best control law among all realizations is:

$$b_{F_1,\text{best}} = \cfrac{(a_2 - a_1)^2}{\cfrac{(a_2 - a_1)}{\cfrac{-0.69096}{(a_2 - a_1)} - 5(a_2 - a_1)} - 6(a_2 - a_1)} - 3(a_2 - a_1)$$

and is depicted in figure 2.24. This complex expression, nesting divisions, has been found with the combination $(N_{\text{inst,max}}, N_{\text{popsize}}) = (20, 200)$ and reduces the cost by $\Delta J/J_0 = 95.50\%$. Even though a limit a 200 instructions was needed to find it, we note that the matrix of this control law has in fact 23 rows. Figure 2.24 shows that the convergence of the oscillator towards the fixed point is not as fast as previous solution, indeed, we can still distinguish oscillations after the second period. However, this solution seems to favor less intense actuation and thus reducing the J_b component of the cost function. Indeed, the cost of the optimal linear optimal control is as follows: $J_{a\text{opt}}/J_0 = 2.16\%,\ Ja_{\text{opt}}/J_0 = 3.14\%$ and the cost of $b_{F_1,\text{best}}$ is: $J_a/J_0 = 1.76\%,\ Ja/J_0 = 2.74\%$

In conclusion, a low population size is more likely to build a better/more complex solution due to the large number of generations but the performance of the median realization drops.



(a) (b)

Figure 2.22: (a) Performance of LGPC regarding population size and maximum number of instructions for the function library $F_1 = \{+, -, \times, \div\}$. The combination of parameters are ranked following the performance of their median realization. The combination ranked first is $(N_{\text{inst,max}}, N_{\text{popsize}}) = (5, 100)$. (b) Cost reduction of the median realization of the different combinations. The horizontal axis represent the rank of the configuration described in (a). (a) and (b) share the same color code.

2.3.2 $F_2 = F_1 \cup \{\exp, \tanh, \sin, \cos, \log\}$

We now add nonlinear functions in the function library, allowing oscillations with cos and sin, exponential and logarithm growth with exp and log and saturation with tanh. As we allow more complex expressions, we can expect to find complex and more efficient solutions than the linear optimal control. Figure 2.25 show similar tendencies for the median realization as for the F_1 library. We note however that the performances have dropped in general. This can be explained by the fact, as we augment the function library, the search space becomes bigger and more individuals are needed to explore more control law structures. The combination $(N_{\text{inst,max}}, N_{\text{popsize}}) = (5, 100)$ stands out from the other combinations as the best one with a reduction of $\Delta J/J_0 = 93.40\%$ of the unforced cost.

Figure 2.26, shows also similar trends as for the F_1 function library. Small populations are favored for matrices with more than 10 instructions. When a small number of instructions is allowed, diversity with a larger population size is preferred.

The best configuration for the F_2 library is $(N_{\text{inst,max}}, N_{\text{popsize}}) = (5, 100)$ as it is ranked first in terms of median cost and also it is one of the configurations to have the most realizations that outperformed the linear optimal control.

However the all time best control law has been found by the combination $(N_{\text{inst,max}}, N_{\text{popsize}}) = (20, 50)$ and reduces the cost by $\Delta J/J_0 = 97.08\%$ with a matrix containing 28 rows. We will not show the control law as it is exceedingly complex. It comprises 215 instances of cos, 217 instances of $\div$, 216 instances of

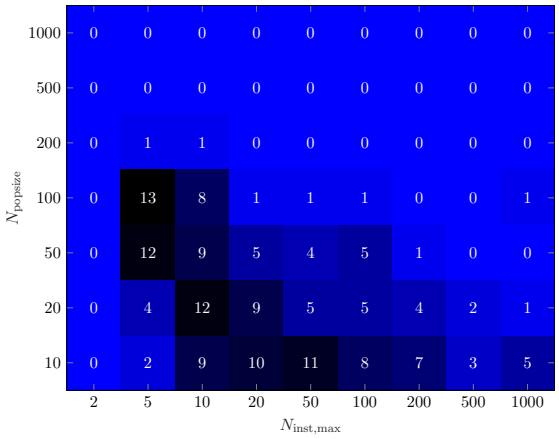


Figure 2.23: Number of realizations that managed to build a control law more efficient than the linear optimal control for the function library $F_1 = \{+, -, \times, \div\}$. The color intensity indicates the number of individuals.

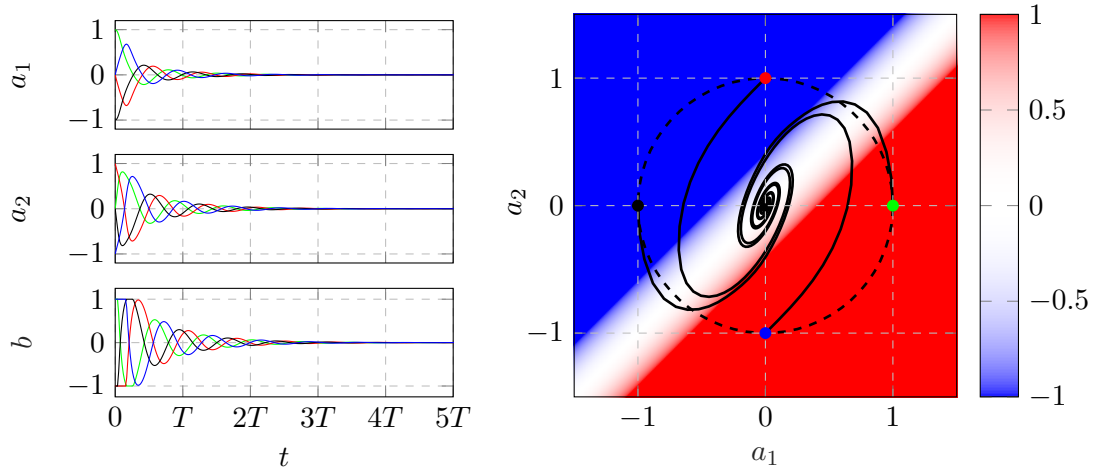
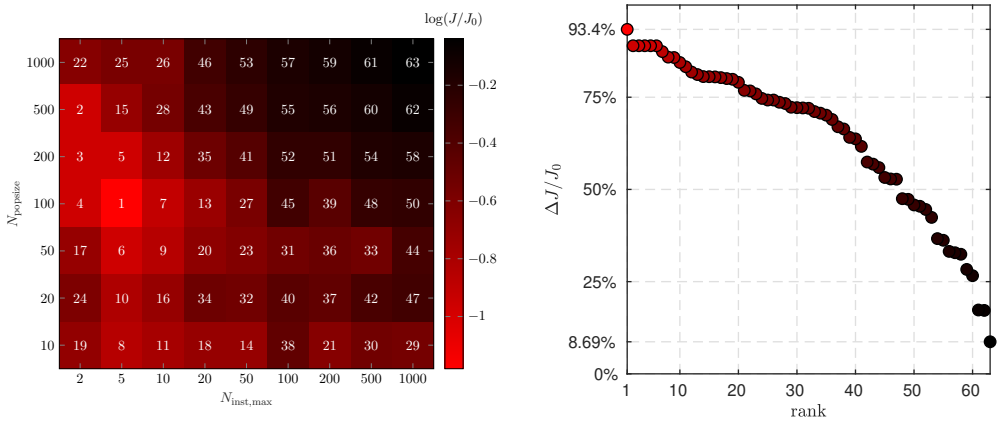


Figure 2.24: Visualization of the controlled Landau oscillator with the best control law found for the function library $F_1 = \{+, -, \times, \div\}$.

$(a_1 - a_2)$ and two times the same constant. Following the number of occurrences of the operators, we can assume that the same pattern has been repeated and nested to shape the control law. The control is depicted in figure 2.27. The external regions show a chaotic behaviour with strong actuation, however as the sign of the actuation changes intermittently, we can assume that those regions are equivalent to no actuation, indeed, we note that when leaving the chaotic control region, the system is at near the periodic orbit. The control activates once the system is in the region where actuation is the more efficient. We also note that the system converges quickly towards the fixed point. Thus, the control comprises fast convergence with lowest actuation. This control may reveal another way of controlling the system by forcing only in the regions where it matters, allowing then to limit the energy expense.

The success of LGPC is that it managed to detect and exploit the most recep-



(a) (b)

Figure 2.25: (a) Performance of LGPC regarding population size and maximum number of instructions for the function library $F_2 = F_1 \cup \{\exp, \tanh, \sin, \cos, \log\}$. The combination of parameters are ranked following the performance of their median realization. The combination ranked first is $(N_{\text{inst,max}}, N_{\text{popsize}}) = (5, 100)$ (b) Cost reduction of the median realization of the different combinations. The horizontal axis represent the rank of the configuration described in (a). (a) and (b) share the same color code.

tive state of the oscillator to efficiently control it with minimum actuation energy. The most difficult task for LGPC may have been to build the transient part of the control, to guide the system towards the state of interest. An absence of control may have been possible but building such control is maybe even more difficult. The enabler for such control was the introduction of nonlinear functions. We can assume that augmenting the function library with more nonlinear functions, such as inequality signs ($>$ or $\leq$) and boolean operators, can help the learning of better control laws. However, one must select the function library carefully since a too diverse library lowers the general performance.

2.4 Learning acceleration

In this last section, we describe some accelerators that manage to increase the learning rate of LGPC. The main idea of these accelerators is to avoid redundant evaluations, meaning to prevent the evaluation of the same control laws. First, we need to detect equivalent control laws. However, this is not an easy task as simplification of two mathematical expressions is not guaranteed to give the same results because of the commutative operators. Moreover, another complication is the protection of the $\div$ and log operators. Because of the protection, these functions present a discontinuous behaviour near 0 that prevents from simplification. Thus, we propose another way to detect equivalent expressions. To detect if two control laws are similar, we evaluate them on 1000 random sample inputs and we

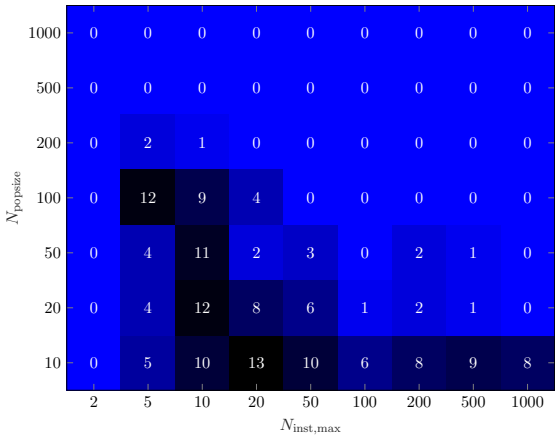


Figure 2.26: Number of realizations that managed to build a control law more efficient than the linear optimal control for the function library $F_2 = F_1 \cup \{\exp, \tanh, \sin, \cos, \log\}$. The color intensity indicates the number of individuals.

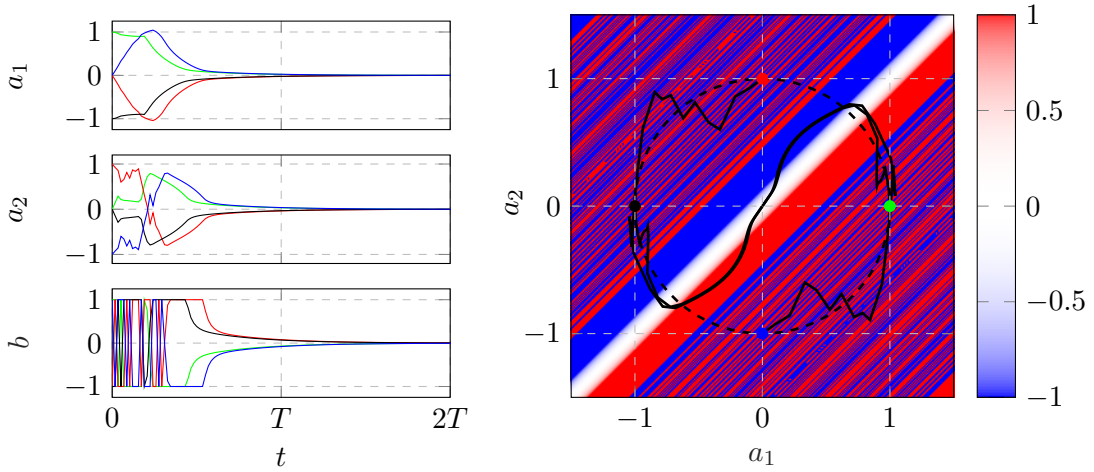


Figure 2.27: Visualization of the controlled Landau oscillator with the best control law found for the function library $F_2 = F_1 \cup \{\exp, \tanh, \sin, \cos, \log\}$.

compare the control outputs. These methods allows to detect mathematical equivalent control laws even though their expressions are very different. Indeed, thanks to such test, control laws that are beyond the actuation thresholds will also be ruled out. For example, the control laws $b = 1.01$ and $b = 10^23$ will be considered as the same control laws. Moreover, the random samples are taken in the range of the sensors signals making the comparison between the control laws even more meaningful. In practice, each time a new control law is built, we evaluate it over the 1000 random sample inputs and store the result in the database. Each new control law is then compared to the database to verify if it has not been already evaluated.

Equivalent control laws can then be filterer out in two ways:

in the population: if a new individual is equivalent to an individual already present in the population, then it is discarded and a new individual is built

(randomly if it is the first generation or with one of the genetic operators).

in the database: if a new individual is equivalent to a previous individual in the current or past generations, then it is discarded and a new individual is built.

Of course the second option includes the first one. Those rules do not apply for individuals generated thanks to replication and elitism as their role is to emulate memory through the generations. Those filtering assures that the population of individuals is always moving towards unexplored regions of the control landscape and backward steps are not possible.

These techniques have been used in Cornejo Maceda *et al.* (2019) and allowed a acceleration of the learning rate by at least a factor 3 for the net drag reduction of the fluidic pinball.

In conclusion, in this chapter, we described the LGPC algorithm and applied it to stabilize a simple, yet fundamental dynamical system, the damped Landau oscillator. We showed that the genetic operators allow the exploration of the control law landscape to find the minima of the problem and the exploitation of the explored control laws to build even better solutions. Through an extensive parametric study of the LGPC algorithm, we revealed that it is the interaction between crossover and mutation that allows an effective learning. We also drew some rule of thumbs to choose the population size, the maximum number of instructions and the function library. Too extreme values for the population size and maximum number of instructions give poor performances and a balance between population size and maximum number of instructions is required to be able to build complex control laws. For a single input problem such as the stabilization of the Landau oscillator, a population of 100 individuals and a number of instructions between 5 and 10 are appropriate for a good median performance. Finally, a rich function library allows to build complex control laws and discover new strategies of control. However, for a large number of functions the overall performance drops. A solution may be to increase the total number of evaluations; a population of 100 individuals evolving through 50 generations may bring enough diversity and combinatorial power to explore effectively such large search spaces. In the following the operator probabilities $(P_c, P_m, P_r) = (0.6, 0.3, 0.1)$ has been chosen, as it is the best in terms of median performance and most number of realization with better results than the linear optimal control. Finally, the source code of our LGPC implementation is freely available online under the name `xMLC` at `https://github.com/gycm134/xMLC`.

Chapter 3

Drag reduction on the fluidic pinball

In this chapter, we aim to reduce the net drag power of the fluidic pinball thanks to the linear genetic programming control methodology presented in chapter 2. For this study, we explore three different search spaces: first we look for a multi-frequency forcing controller, second we look for feedback control laws and finally we investigate an hybrid search space comprising periodic functions and sensor signals. Thus, in section 3.1 we present the fluidic pinball and the optimization problem. In section 3.2, we detail an open-loop control study of the fluidic pinball for a control reference. Lastly, we apply LGPC to minimize the net drag power with multi-frequency forcing (section 3.3), feedback control (section 3.4) and a search space allowing both strategies (section 3.5).

Contents

3.1 The fluidic pinball—A benchmark flow control problem

In this section, we describe the fluid system studied for the control optimization—the fluidic pinball. First we present the fluidic pinball configuration and the unsteady 2D Navier-Stokes solver in section 3.1.1, then the unforced flow spatio-temporal dynamics in section 3.1.2 and finally the control problem for the fluidic

pinball in section 3.1.3. This section is largely inspired from section 2 of Cornejo Maceda *et al.* (2020).

3.1.1 Configuration and numerical solver

The test case is a two-dimensional uniform flow past a cluster of three cylinders of same diameter D. The centre of the cylinders form an equilateral triangle pointing upstream. The flow is controlled by the independent rotation of the cylinders along their axis. The rotation of the cylinders enables the steering of incoming fluid particles, like a pinball machine. Thus, we refer this configuration as the fluidic pinball. In our study, we choose the side length of the equilateral triangle equal to be $1.5D$. Various side lengths have been explored numerically in (Chen *et al.*, 2020), revealing a myriad of interesting regimes.

The flow is described in a Cartesian coordinate system, where the origin is located midway between the two rearward cylinders. The x-axis is parallel to the stream-wise direction. The y-axis is orthogonal to the cylinder axis. The velocity field is denoted by $\boldsymbol{u} = (u, v)$ and the pressure field by p. Here, u and v are, respectively, the stream-wise and transverse components of the velocity. We consider a Newtonian fluid of constant density ρ and kinematic viscosity ν. For the direct numerical simulation, the unsteady incompressible viscous Navier-Stokes equations are non-dimensionalized with cylinder diameter D, the incoming velocity U_∞ and the fluid density ρ. The corresponding Reynolds number is $\mathrm{Re}_D = U_\infty D/\nu$. Throughout this study, only $\mathrm{Re}_D = 100$ is considered.

The computational domain Ω is a rectangle bounded by $[-6, 20] \times [-6, 6]$ excluding the interior of the cylinders:

$$\Omega = \{[x, y]^\intercal \in \mathcal{R}^2 \colon [x, y]^\intercal \in [-6, 20] \times [-6, 6] \wedge (x - x_i)^2 + (y - y_i)^2 \geq 1/4, i = 1, 2, 3\}.$$

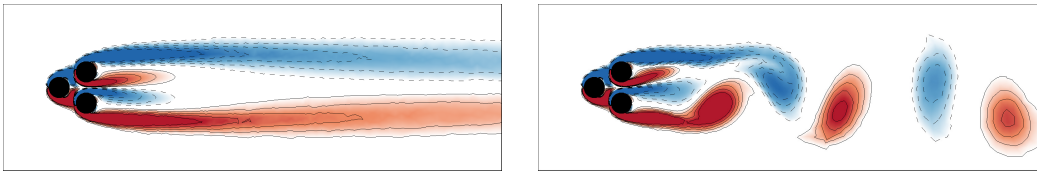
Here, $[x_i, y_i]^\intercal$ with $i = 1, 2, 3$, are the coordinates of the cylinder centres, starting from the front cylinder and numbered in mathematically positive direction,

$$\begin{aligned} x_1 &= -3/2\cos(30^\circ) & y_1 &= 0, \\ x_2 &= 0 & y_2 &= -3/4, \\ x_3 &= 0 & y_3 &= 3/4. \end{aligned}$$

The computational domain Ω is discretized on an unstructured grid comprising 4225 triangles and 8633 nodes. The grid is optimized to provide a balance between computation speed and accuracy. Grid independence of the direct Navier-Stokes solutions has been established by Deng *et al.* (2020).

The boundary conditions for the inflow, upper and lower boundaries are $U_\infty = \boldsymbol{e}_x$ while a stress-free condition is assumed for the outflow boundary. The control of the fluidic pinball is carried out by the rotation of the cylinders. A non-slip condition is adopted on the cylinders: the flow adopts the circumferential velocities of the front, bottom and top cylinder specified by $b_1 = U_F$, $b_2 = U_B$ and $b_3 = U_T$. The actuation command comprises these velocities, $\boldsymbol{b} = [b_1, b_2, b_3]^\intercal$. A positive (negative) value of the actuation command corresponds to counter-clockwise (clockwise) rotation of the cylinders along their axis. The numerical

integration of the Navier-Stokes equations is carried by an in-house solver using a fully implicit Finite-Element Method (Noack *et al.*, 2003, 2016). The method is third order accurate in time and space.



(a) Symmetric steady solution. (b) Unforced flow at $t = 400$.

Figure 3.1: Vorticity fields for the unforced fluidic pinball at $\mathrm{Re}_D = 100$. Blue (red) regions bounded by dashed lines represent negative (positive) vorticity. Darker regions indicate higher values of vorticity magnitude.

The initial condition for the numerical simulations is the symmetric steady solution. The symmetrical steady solution is computed with a Newton-Raphson method on the steady Navier-Stokes. An initial short, small rotation of the front cylinder is used to kick-start the transient to natural vortex shedding in the first period (Deng *et al.*, 2020). The transient regime lasts around 400 convective time units. Figure 3.1 shows the vorticity field for the symmetric steady solution and the natural unforced flow after 400 convective units. The snapshot at $t = 400$ in figure 3.1b will be the initial condition for all the following simulations.

3.1.2 Unforced reference

The fluidic pinball is a geometrically simple configuration that comprises key features of real-life flows such as successive bifurcations and frequency crosstalk between modes. Deng *et al.* (2020) shows that the unforced fluidic pinball undergoes successive bifurcations with increasing Reynolds number before reaching a chaotic regime. The first Hopf bifurcation at Reynolds number $\mathrm{Re} \approx 18$ breaks the symmetry in the flow and initiates the von Kármán vortex shedding. The second bifurcation at Reynolds number $\mathrm{Re} \approx 68$ is of pitchfork type and gives rise to a transverse deflection of jet-like flow appearing between the two rearward cylinders. The bi-stability of the jet deflection has been reported by Deng *et al.* (2020). At a Reynolds number $\mathrm{Re} = 100$ the jet deflection is rapid and occurs before the vortex shedding is fully established. Figure 3.2a shows an increase of the lift coefficient C_L before oscillations set in and the lift coefficient converges against a periodic oscillation around a slightly reduced mean value. Those bifurcations are a consequence of multiple instabilities present in the flow: there are two shear instabilities, on the top and bottom cylinder and a jet bi-stability originating from the gap between the two back cylinders. The shear-layer instabilities synchronize to a von Kármán vortex shedding.

Figure 3.2 illustrates the dynamics of the unforced flow from the unstable steady symmetric solution to the post-transient periodic flow. The phase portrait in figure 3.2b and the power spectral density (PSD) in figure 3.2d show a periodic regime with frequency $f_0 = 0.116$ and its harmonic. Figure 3.2a shows that the

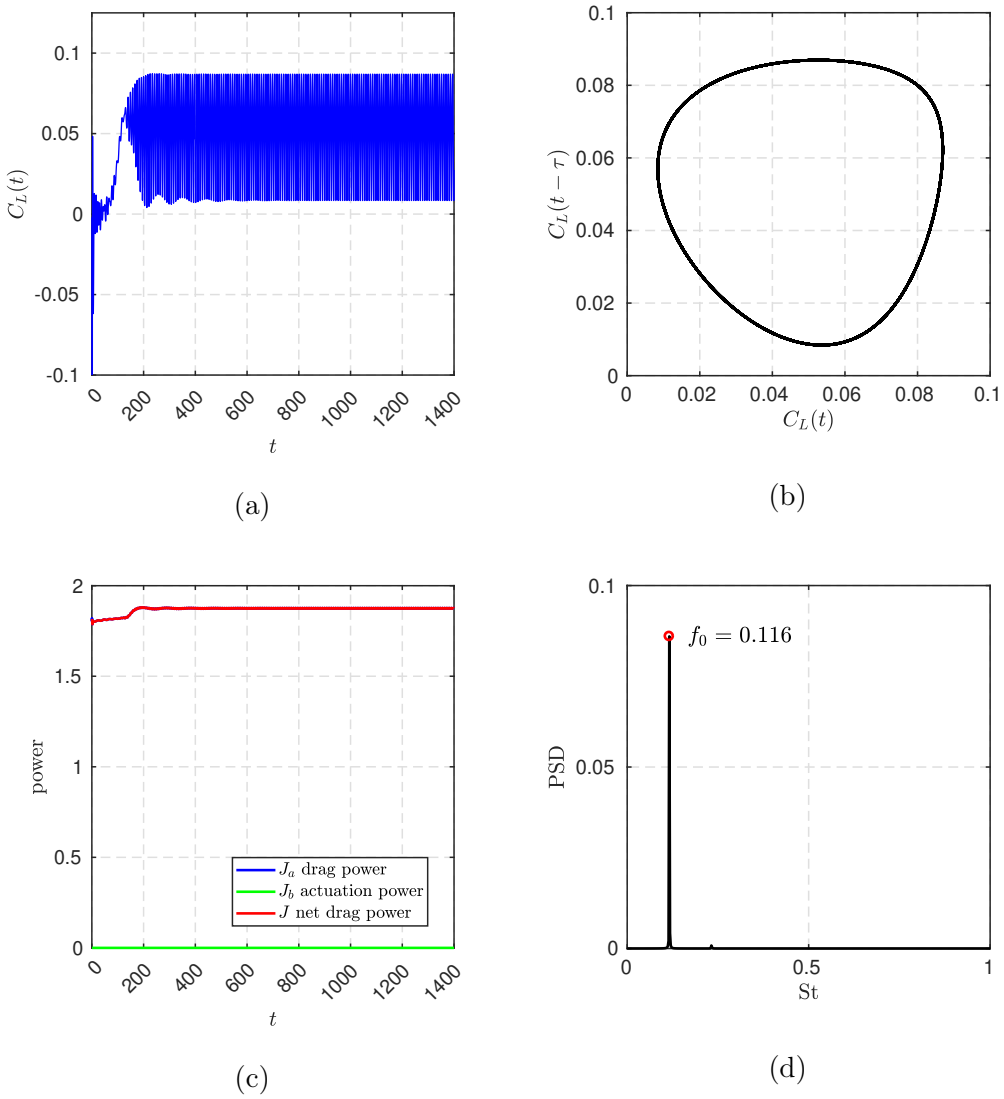
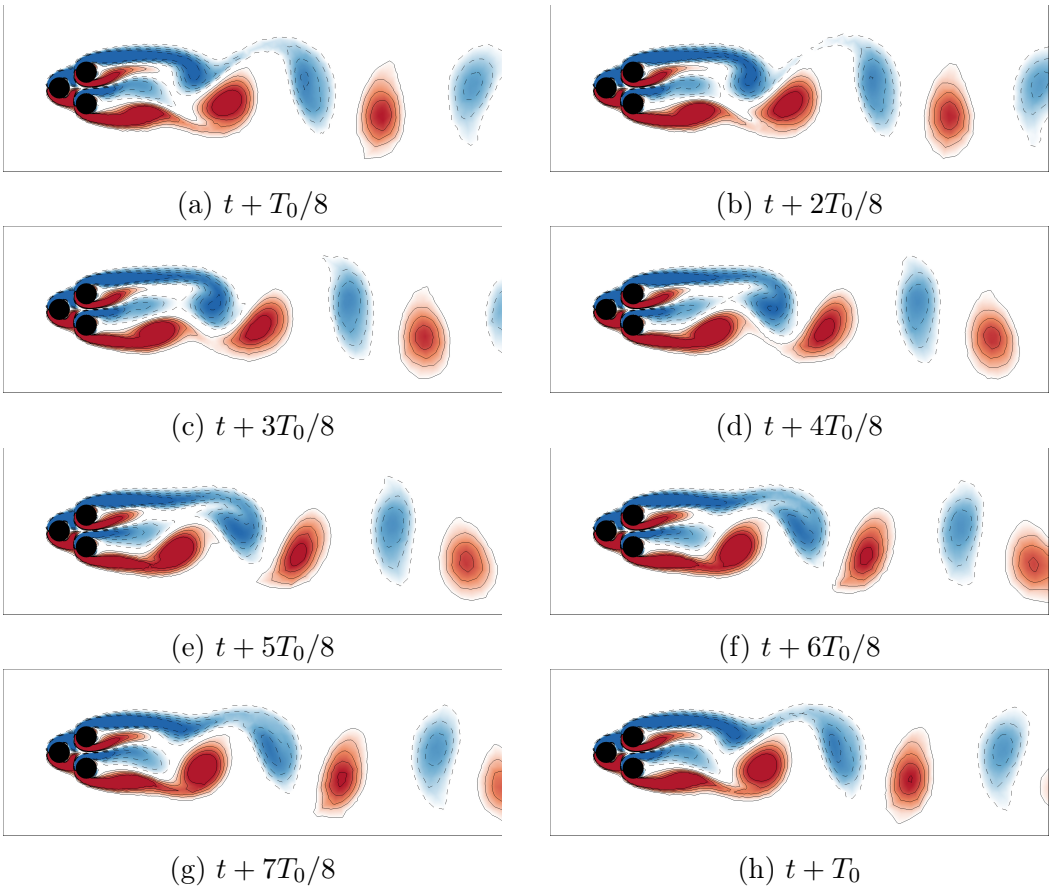


Figure 3.2: Characteristics of the unforced natural flow starting from the steady solution ($t = 0$). The transient spans until $t \approx 400$. (a) Time evolution of the lift coefficient C_L, (b) phase portrait, (c) time evolution of the drag power J_a (blue), actuation power J_b (green) and net drag power J (red) and (d) Power Spectral Density (PSD) showing the natural frequency $f_0 = 0.116$. The phase portrait is computed during the post-transient regime $t \in [900, 1400]$ and the PSD is computed over the last 1000 convective time units, $t \in [400, 1400]$.

mean value of the lift coefficient C_L is not null. This is due to the deflection of the jet behind the two rearward cylinders during the post-transient regime. During this regime, the deflection of the jet stays on one side as it is illustrated in figure 3.3a-3.3h over one period and in figure 3.3j in the mean field. This deflection explains the lift coefficient C_L asymmetry. Indeed, the upward oriented jet increases the pressure on the lower part of the top cylinder leading to an increase of the lift coefficient. In figure 3.2a, the initial downward spike on the



(a) $t + T_0/8$ (b) $t + 2T_0/8$

(c) $t + 3T_0/8$ (d) $t + 4T_0/8$

(e) $t + 5T_0/8$ (f) $t + 6T_0/8$

(g) $t + 7T_0/8$ (h) $t + T_0$

Figure 3.3: Vorticity fields of the unforced flow. (a)-(f) Time evolution of the vorticity field in the last period of the simulation. The color code is the same as figure 3.1. T_0 is the natural period associated to the natural frequency f_0.

lift coefficient is due to the initial kick. The unforced natural flow is our reference simulation for future comparisons.

Thanks to the rotation of the cylinders, the fluidic pinball is capable of reproducing six actuation mechanisms inspired from wake stabilization literature and exploiting distinct physics. Examples of those mechanisms can be found in Ishar *et al.* (2019). First, the wake can be stabilized by shaping the wake region more aerodynamically—also called fluidic boat tailing. The shear layers are vectored towards the centre region with passive devices, like vanes (Flügel, 1930) or active control through Coanda blowing (Geropp, 1995; Geropp & Odenthal, 2000; Barros *et al.*, 2016). In the case of the fluidic pinball, we can mimic this effect by a counter-rotating rearward cylinders which accelerates the boundary layers and delays separation. This fluidic boat tailing is typically associated with significant drag reduction. Second, the two rearward cylinders can also rotate oppositely ejecting a fluid jet on the centreline. Thus, interaction between the upper and lower shear layer is suppressed, preventing the development of a von Kármán vortex in the vicinity of the cylinders. Such base bleeding mechanisms has a similar physical effect as a splitter plate behind a bluff body and has been proved to be an effective means for wake stabilization (Wood, 1964; Bearman, 1967).

Third, phasor control can be performed by estimating the oscillation phase and feeding it back with a phase shift and gain (Protas, 2004). Fourth, unified rotation of the three cylinders in the same direction gives rise to higher velocities, and thus larger vorticity, on one side at the expense of the other side, destroying the vortex shedding. This effect relates to the Magnus effect and stagnation point control (Seifert, 2012). Fifth, high-frequency forcing can be effected by symmetric periodic oscillation of the rearward cylinders. With a vigorous cylinder rotation (Thiria *et al.*, 2006), the upper and lower shear layers are re-energized, reducing the transverse wake profile gradients and thus the instability of the flow. Thus, the effective eddy viscosity in the von Kármán vortices increases, adding a damping effect. Sixth and finally, a symmetrical forcing at a lower frequency than the natural vortex shedding may stabilize the wake (Pastoor *et al.*, 2008). This is due to the mismatch between the anti-symmetric vortex shedding and the forced symmetric dynamics whose clock-work is distinctly out of sync with the shedding period. High- and low-frequency forcing lead to frequency crosstalk between actuation and vortex shedding over the mean flows, as described by low-dimensional generalized mean-field model (Luchtenburg *et al.*, 2009).

We confirm therefore that the fluidic pinball is an interesting Multiple-Input Multiple-Output (MIMO) control benchmark. The configuration exhibits well-known wake stabilization mechanisms in physics. From a dynamical perspective, nonlinear frequency crosstalk can easily be enforced. In addition, even long-term simulations can easily be performed on a laptop within an hour.

3.1.3 Control objective and optimization problem

Several control objectives related to the suppression or reduction of undesired forces can be considered for the fluidic pinball. We can increase the recirculation bubble length, reduce lift fluctuations or even mitigate the total fluctuation energy.

In this study, we aim to reduce the net drag power at $\mathrm{Re}_D = 100$. The associated objectives are J_a, the drag power and J_b, the actuation power. The cost J_a is defined as the temporal average of the drag power of the controlled flow field:

$$J_a = \frac{1}{T_{ev}} \int_{t_0}^{t_0+T_{ev}} j_a(t)\, \mathrm{d}t \tag{3.1}$$

with the instantaneous cost function

$$j_a(t) = \boldsymbol{F}_x(t) \cdot \boldsymbol{U}_\infty \tag{3.2}$$

where $\boldsymbol{F}_x$ is the drag and $\boldsymbol{U}_\infty$ is the incoming velocity. The control is activated at $t_0 = 400$ convective time units after the starting kick on the steady solution. Thus, we have a fully established post-transient regime. The cost function is evaluated until $T_{\mathrm{ev}} = 525$ convective time units. Thus, the time average is effected over 125 convective time units which corresponds to more than 10 periods T_0 of the unforced flow.

J_b is naturally chosen as a measurement of the actuation energy investment. Evidently, a low actuation energy is desirable. The actuation power is computed

as the power of the torque applied by the fluid on the cylinders. J_b is the time-averaged actuation power over $T_{\text{ev}} = 125$ time units:

$$J_b(\boldsymbol{b}) = \frac{1}{T_{ev}} \int_{t_0}^{t_0+T_{\text{ev}}} \sum_{i=1}^{3} \mathcal{P}_{\text{act,i}} \, \mathrm{d}t \tag{3.3}$$

where $\mathcal{P}_{\text{act,i}}$ is the actuation power supplied integrated over cylinder i:

$$\mathcal{P}_{\text{act,i}} = - \oiint b_i F^{\theta}_{s,i} \, \mathrm{d}s$$

where $\left(F^{\theta}_{s,i}\mathrm{d}s\right)$ is the azimuthal component of the local fluid forces applied to cylinder i. The negative sign denotes that the power is supplied and not received by the cylinders.

Thus, the cost function employed for the optimization is $J = J_a + \gamma J_b$. γ is the penalization parameter. It allows to balance the terms of the cost function. In this study as we aim to reduce the net drag power, we set $\gamma = 1$ so both components J_a and J_b have the same weight.

The instantaneous values of J_a and J_b are plotted in figure 3.2c. Naturally, for the unforced flow, the actuation power is null, and the cost function is only the drag power. We note that the drag power takes around 300 convective units to stabilize. The cost of the post-transient regime is $J_0 = 1.87$. J_0 serves as a reference for future comparisons. The cost of the steady flow 3.1a is $J_{\text{steady}} = 1.80$ which is lower than J_0 but still high. Therefore, we can assume that stabilizing the symmetric steady solution may not be the best strategy to reduce the net drag power.

In order to minimize the net drag power, the flow is forced by the rotation of the three cylinders. The actuation command $\boldsymbol{b} = [b_1, b_2, b_3]^\intercal$ is determined by the control law $\boldsymbol{K}$. This control law may operate open-loop or closed-loop with flow input. Considered open-loop actuations are steady or harmonic oscillation around a vanishing mean. Considered feedback includes velocity sensor signals in the wake. Thus, in the most general formulation, the control law reads the equation described in section 1.2.2: $\boldsymbol{b}(t) = \boldsymbol{K}(\boldsymbol{h}(t), \boldsymbol{s}(t))$ with $\boldsymbol{h}(t)$ and $\boldsymbol{s}(t)$ being vectors comprising respectively time dependent harmonic functions and sensor signals. The sensor signals include the instantaneous velocity signals as well as three recorded values over one period as elaborated in the result section 3.4. In the following, N_{b} represents the number of actuators, N_{h} for the number of time-dependent functions and N_{s} for the number of sensor signals.

3.2 Symmetric steady actuation for net drag reduction

First, we carry out an open-loop parametric study to assess the effect of control on the drag power. To achieve an exhaustive parametric study is a costly task as we need to operate in 3-dimensional parameter space. That is why we restrict the

search to the subspace of symmetric actuations: the front cylinder does not rotate and the two back cylinders rotate at the same speed but in opposite directions:

$$\begin{aligned} b_1 &= 0, \\ b_2 &= -b_3. \end{aligned}$$

Thus, we explore the effect of only one parameter b_2. b_2 is defined so that when it is positive, the flow is vectored towards the centreline—boat tailing configuration—and when it is negative, the inner flow is accelerated—base bleeding configuration. Figure 3.4 shows the evolution of J_a, J_b and $J = J_a + J_b$ as a function of b_2. Solely considering J_a, boat tailing is the best strategy to reduce the drag power. Indeed, drag power decreases monotonously with increasing b_2. For a strong actuation, $b_2 > 3$, J_a even becomes negative and the fluidic pinball becomes a jet. Base bleeding, on the other hand, is not a viable strategy to reduce the drag power. There is a local minimum around $b_2 = -4$ but its associated drag power is still higher than the natural unforced flow. When adding the actuation power, there is only one minimum for the cost function J, around $b_2 = 1$. Its associated cost is $J_{\mathrm{BT}}/J_0 = 0.77$. A more detailed analysis of the different controlled regimes is given in section 4.2.2

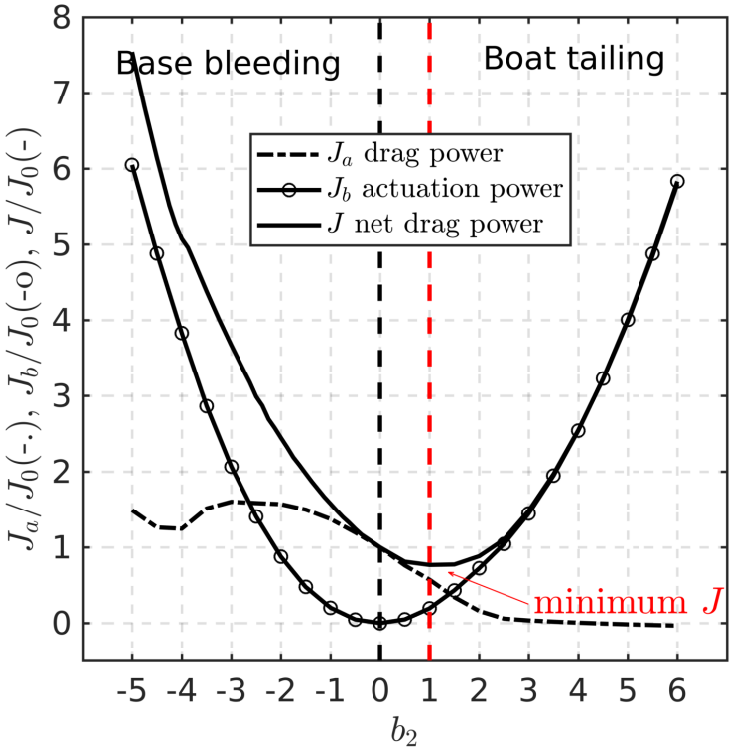


Figure 3.4: Cost function evolution regarding actuation intensity for symmetric constant actuation.

Figure 3.5d shows the characteristics of the controlled flow with the best boat

tailing solution. We note that the regime is purely harmonic with an increase of the main frequency from $f_0 = 0.116$, for the unforced flow, to $f_1 = 0.128$. We also remark a symmetrization of the lift coefficient C_L alongside a significant increase of the oscillations amplitude. The loss of the mean value results in a symmetrization of the flow. Indeed, figure 3.6 shows that the near jet completely disappeared. We also observe a considerable reduction of the recirculation bubble, as a consequence, the base pressure behind the back cylinders increases and the total drag decreases.

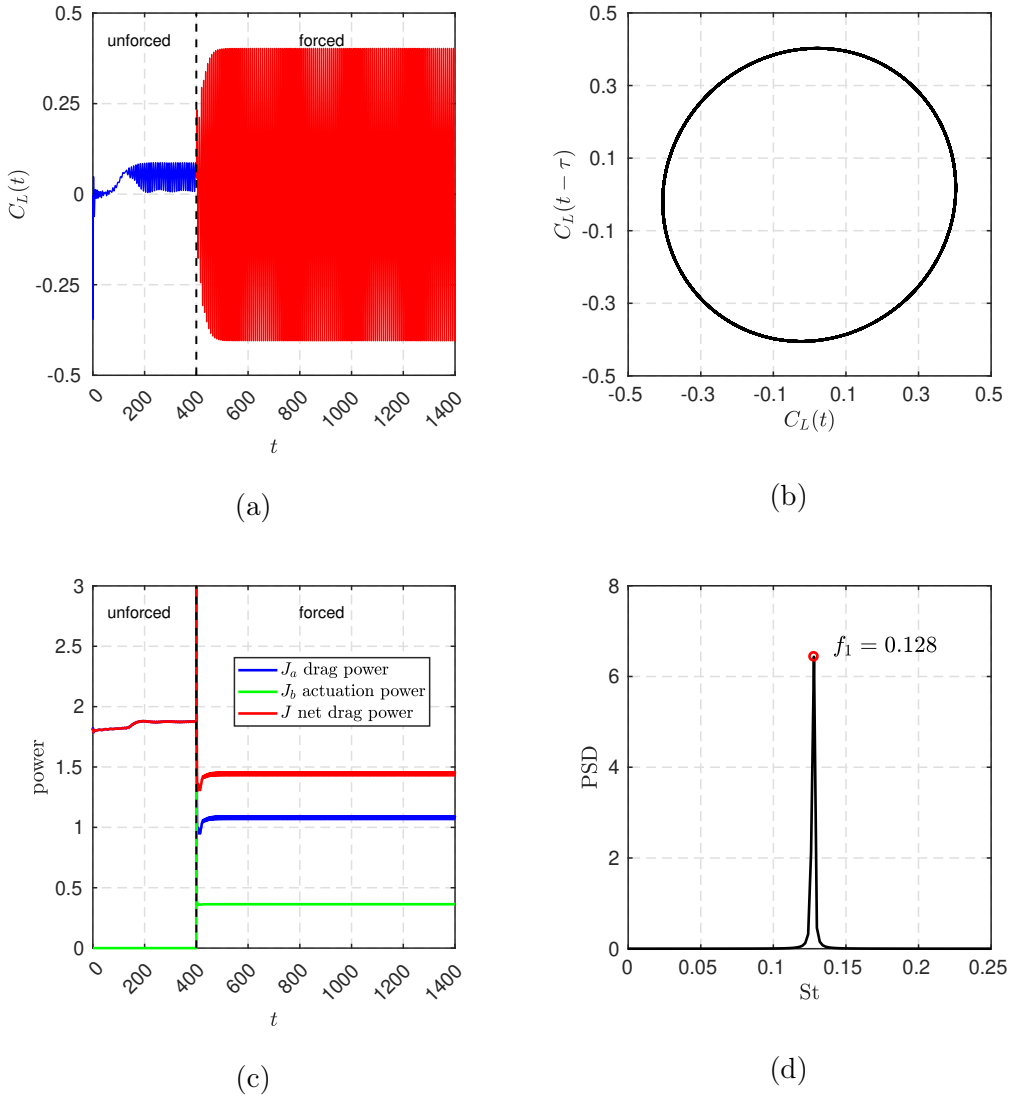
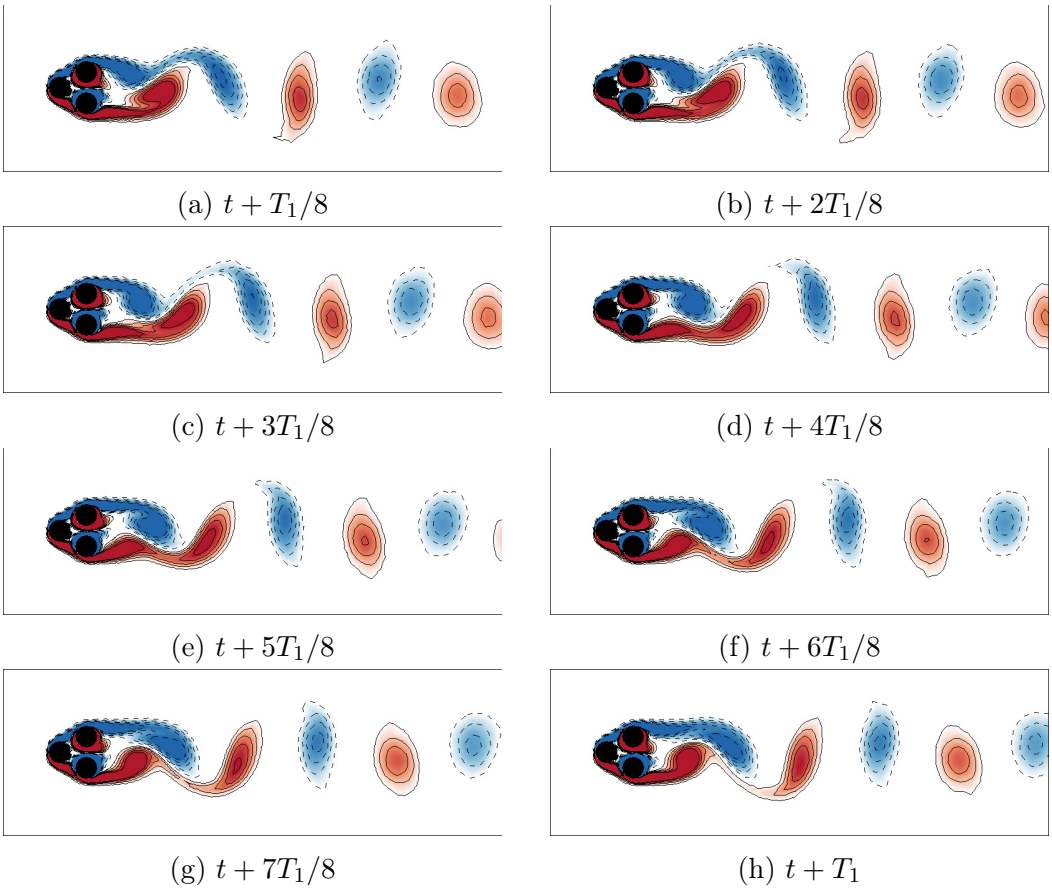


Figure 3.5: Characteristics of the best boat tailing solution starting from the steady solution ($t = 0$). The transient spans until $t \approx 400$. (a) Time evolution of the lift coefficient C_L, (b) phase portrait, (c) time evolution of the drag power J_a (blue), actuation power J_b (green) and net drag power J (red) and (d) Power Spectral Density (PSD) showing the frequency $f_1 = 0.128$.



(a) $t + T_1/8$ (b) $t + 2T_1/8$

(c) $t + 3T_1/8$ (d) $t + 4T_1/8$

(e) $t + 5T_1/8$ (f) $t + 6T_1/8$

(g) $t + 7T_1/8$ (h) $t + T_1$

Figure 3.6: Vorticity fields of the flow controlled by the best boat tailing solution. (a)-(f) Time evolution of the vorticity field in the last period of the 1000 time units simulation. The color code is the same as figure 3.1. T_1 is the period associated to the frequency f_1.

Thanks to a 1-dimensional parametric study, we highlighted the benefit of boat tailing to reduce the drag power. Of course, a control based on the three cylinders may certainly improve the results. This is explored in a model-free framework thanks to LGPC in the next sections. The exploration of a the 3-dimensional parameter space is enabled thanks to a novel machine learning algorithm, the explorative gradient method (EGM) (Li *et al.*, 2021). EGM is thoroughly explained in section 4.1.2.

3.3 Multi-frequency optimization

In this section, we explore the space of open-loop controllers, in particular periodic control laws. In order to do so, we consider cosine functions as inputs for the control laws. Thus, equation 1.1 becomes: $\boldsymbol{b}(t) = \boldsymbol{K}(\boldsymbol{h}(t))$. To enrich our search space and to avoid resonance effects, we choose eight periodic functions whose

frequencies are incommensurable with the natural frequency f_0:

$$\begin{aligned} h_1 &= \cos(2\pi\Phi^{-4}f_0t), & h_5 &= \cos(2\pi\Phi^{1}f_0t),\\ h_2 &= \cos(2\pi\Phi^{-3}f_0t), & h_6 &= \cos(2\pi\Phi^{2}f_0t),\\ h_3 &= \cos(2\pi\Phi^{-2}f_0t), & h_7 &= \cos(2\pi\Phi^{3}f_0t),\\ h_4 &= \cos(2\pi\Phi^{-1}f_0t), & h_8 &= \cos(2\pi\Phi^{4}f_0t). \end{aligned}$$

The golden ration Φ assures that the periodic function is incommensurable with the natural frequency f_0, i.e. the natural frequency cannot be reconstructed thanks to the algebraic operators $(+, -, \times, \div)$.

The rest of the LGPC parameters are summarized in table 3.1. We choose the operators probability $(P_c, P_m, P_r) = (0.6, 0.3, 0.1)$ as explained in 2.2.3. As we optimize three controllers, we decided to increase the number of maximum instructions to 50. To build complex control laws, we employ the function library $F_2 = \{+, -, \times, \div, \exp, \tanh, \sin, \cos, \log\}$. Also, since we have eight inputs for the control laws, we need to increase the number of variable registers to include an instance of all inputs. Thus, we also increase the number of constants.

parameter	description	value
	function library	$F_2 = \{+, -, \times, \div,$ $\exp, \tanh, \sin, \cos, \log\}$
$\boldsymbol{h}$	controller inputs	h_i, $i = -4, .., -1, 1, ..4$
N_{vr}	number of variable registers	$8 + 3 = 11$
N_{cr}	number of constant registers	10
$N_{\text{inst,max}}$	max number of instructions	50
N_{popsize}	population size	100
N_G	number of generations	10
N_{tour}	tournament size	7
N_e	elitism	1
P_c	crossover probability	0.6
P_m	mutation probability	0.3
P_r	replication probability	0.1

Table 3.1: LGPC parameters for multi-frequency forcing optimization.

We ran the optimization with a population of 100 individuals evolving through 10 generations. The learning process is illustrated in figure 3.7. Most of the learning is done during the Monte Carlo step, indeed after 100 random evaluations the cost is equal to $J/J_0 = 0.80$. Only small improvements are made until the fourth generation where a big jump manages to reduce the cost lower than the boat tailing solution. From there, only small improvements are achieved.

The best control law $\boldsymbol{b}^{\text{MF}}$ reads:

$$\begin{aligned} b_1^{\text{MF}} &= -0.13732,\\ b_2^{\text{MF}} &= 0.982511,\\ b_3^{\text{MF}} &= -1.1979,\\ J_{\text{MF}}/J_0 &= 0.7476 \end{aligned} \tag{3.4}$$

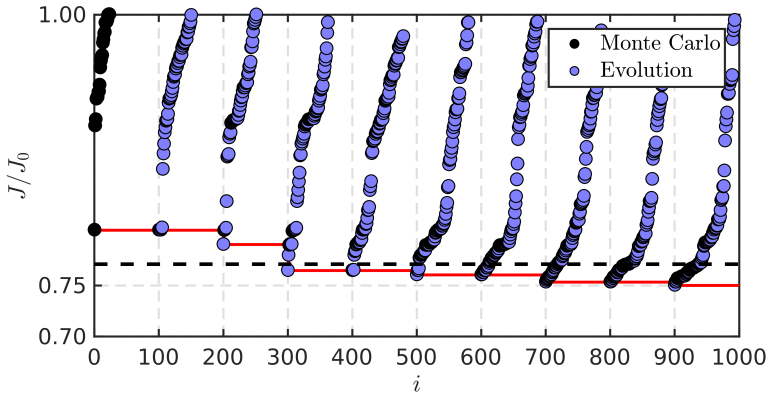


Figure 3.7: Distribution of the costs during the multi-frequency forcing LGPC optimization process. Each dot represents the cost J_a/J_0 of one individual. The color of the dots represents how the individuals have been generated. Black dots denote the individuals which are randomly generated (Monte Carlo). Blue dots refer to individuals which are generated from a genetic operator. The individuals from each generation have been sorted following their costs. The red line shows the evolution of the best cost. The dashed horizontal line corresponds to the best symmetric solution $b_2 = 1$ with a cost of $J_{\mathrm{BT}}/J_0 = 0.77$. The vertical axis is in log scale and has been truncated to help the visualization of the best individuals.

We note that the control is steady and does not contain any h_i, suggesting that periodic forcing is not a viable solution to reduce the net drag power. Indeed, periodic forcing must increase the gradient of the azimuthal speed, thus increasing the torque and actuation power. $\boldsymbol{b}^{\mathrm{MF}}$ resembles a boat tailing configuration with a slight asymmetry as the bottom cylinders rotates faster than the top one and the front cylinder also presents a small rotation. We suspect that this asymmetry is typical of pitchfork bifurcated flows as it was also reported by Raibaudo *et al.* (2019).

The characteristics of the flow controlled by $\boldsymbol{b}^{\mathrm{MF}}$ are shown in figure 3.8. As for the best boat tailing solution, the controlled flow with $\boldsymbol{b}^{\mathrm{MF}}$ is purely harmonic with a slightly lower frequency $f_2 = 0.126$. The amplitude of the oscillations of the lift coefficient has also increased. However, the slight asymmetry in the control results in a significant increase of the mean value of the lift coefficient.

The mean value variation of the lift coefficient is, nonetheless, hardly visible on the snapshots in figure 3.9. As the actuation is close to the best boat tailing control, the controlled flows are also similar. By squinting ones eye, we notice that there is a small region, at the lower-back part of the front cylinder, with intense vorticity. We can assume that this small vortex increased the local pressure and thus shifts the mean value of the lift positively.

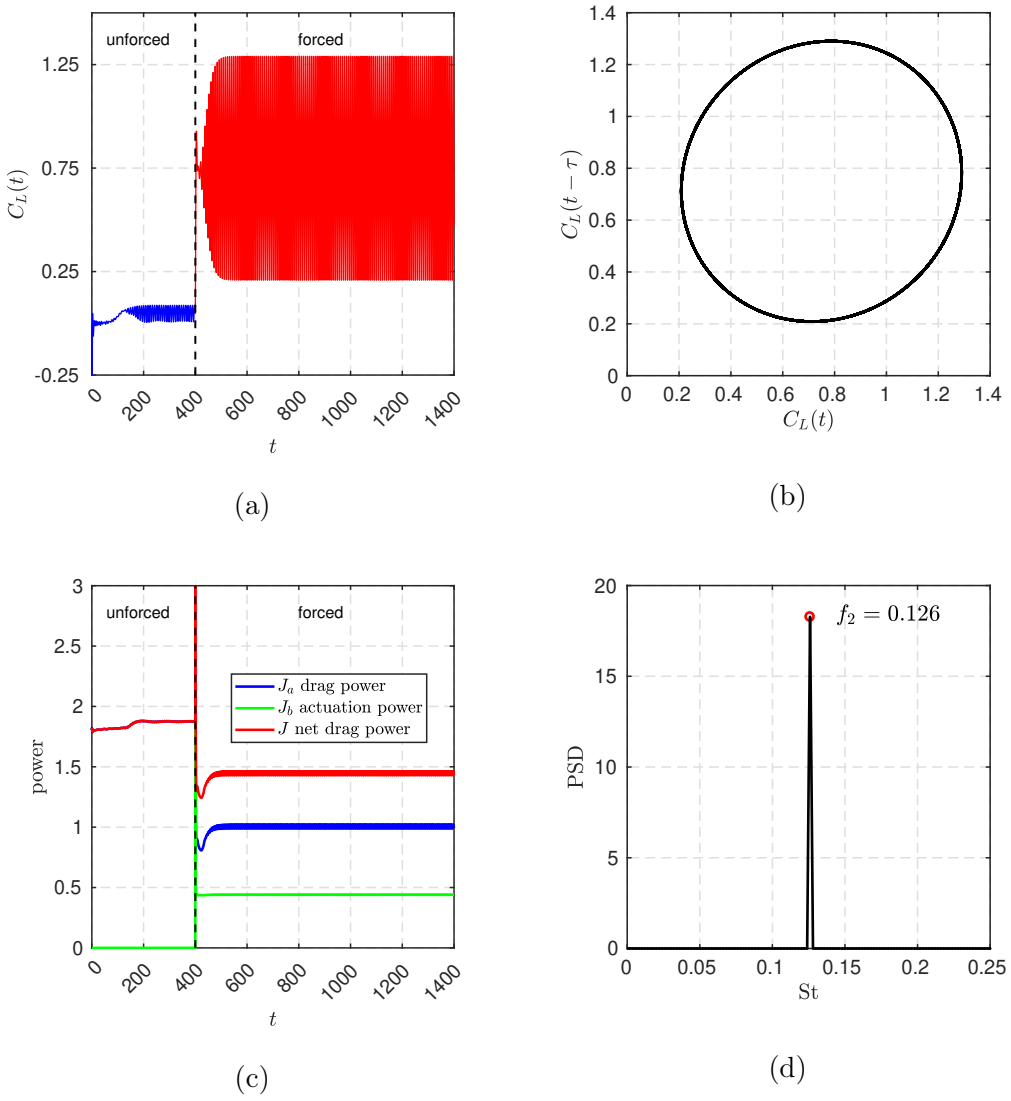
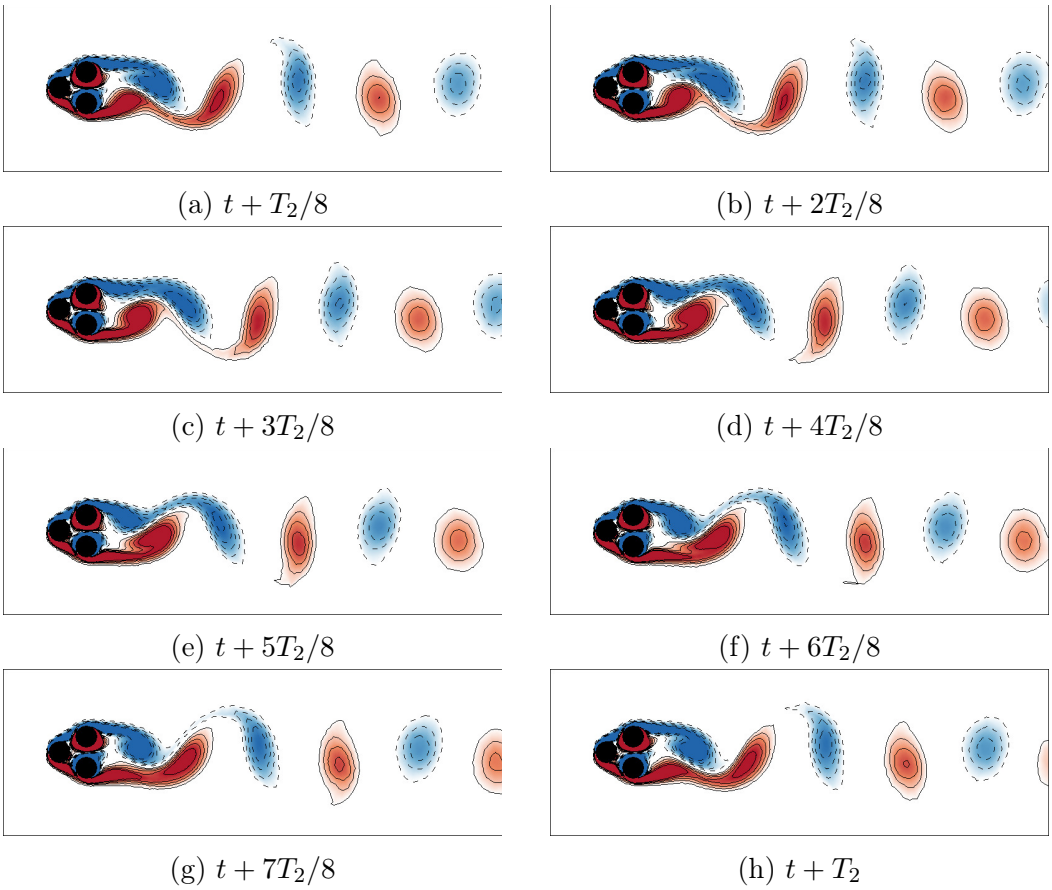


Figure 3.8: Characteristics of the flow controlled by $\boldsymbol{b}^{\mathrm{MF}}$ starting from the steady solution ($t = 0$). The transient spans until $t \approx 400$. (a) Time evolution of the lift coefficient C_L, (b) phase portrait, (c) time evolution of the drag power J_a (blue), actuation power J_b (green) and net drag power J (red) and (d) Power Spectral Density (PSD) showing the frequency $f_2 = 0.126$.

3.4 Feedback control law optimization

In this section, we allow feedback control laws by adding sensor signals as inputs. Thus, equation 1.1 becomes: $\boldsymbol{b}(t) = \boldsymbol{K}(\boldsymbol{s}(t))$. We choose a grid of nine sensor downstream measuring either x or y velocity components. The coordinates of the sensors are $x = 5,\ 6.5,\ 8$ and $y = 1.25,\ 0,\ -1.25$. The six exterior sensors are u sensors while v sensors are chosen for the ones on the symmetry line $y = 0$. The information of sensors is summarized in table 3.2. Moreover, in order to take into account the convective nature of the flow, we add time-delayed sensors as inputs of



(a) $t + T_2/8$ (b) $t + 2T_2/8$

(c) $t + 3T_2/8$ (d) $t + 4T_2/8$

(e) $t + 5T_2/8$ (f) $t + 6T_2/8$

(g) $t + 7T_2/8$ (h) $t + T_2$

Figure 3.9: Vorticity fields of the flow controlled by the best boat tailing solution. (a)-(f) Time evolution of the vorticity field in the last period of the 1000 time units simulation. The color code is the same as figure 3.1. T_2 is the period associated to the frequency f_2.

the control laws. The delays are a quarter, half and three-quarters of the unforced natural period, yielding following additional lifted sensor signals:

$$s_{i+9}(t) = s_i(t - T_0/4), \quad s_{i+18}(t) = s_i(t - T_0/2), \quad s_{i+27}(t) = s_i(t - 3T_0/4).$$

Hence, the dimension of the sensor vector $\boldsymbol{s}$ is $9 \times 4 = 36$ and $X \subset \mathbb{R}^{36}$. The time-delayed sensors are also include as control inputs to mimic ARMAX-based control (Hervé *et al.*, 2012).

The same LGPC parameters as for section 3.3 have been chosen. The number of variable constants increased as we now have 36 sensor signals.

Figure 3.10 shows cost of the individuals during the optimization process. We note the same learning trend as for multi-frequency forcing optimization. However in this case, the big jump appeared sooner, directly at the second generation. From there, only small improvements are achieved. We note also that the Monte Carlo step is less efficient as less individuals have reached a performance lower than the unforced one. Moreover the best individual of the first generation has a cost of $J/J_0 = 0.84$ which is higher than in the multi-frequency forcing case. This can

sensor	x-coordinate	y-coordinate	velocity component
s_1	5	1.25	u
s_2	6.5	1.25	u
s_3	8	1.25	u
s_4	5	0	v
s_5	6.5	0	v
s_6	8	0	v
s_7	5	-1.25	u
s_8	6.5	-1.25	u
s_9	8	-1.25	u

Table 3.2: Summary of sensor information.

parameter	description	value
	function library	$F_2 = \{+, -, \times, \div, \exp, \tanh, \sin, \cos, \log\}$
$\boldsymbol{s}$	controller inputs	$s_i(t)$, $i = 1, .., 36$
N_{vr}	number of variable registers	$36 + 3 = 39$
N_{cr}	number of constant registers	10
$N_{\text{inst,max}}$	max number of instructions	50
N_{popsize}	population size	100
N_G	number of generations	10
N_{tour}	tournament size	7
N_e	elitism	1
P_c	crossover probability	0.6
P_m	mutation probability	0.3
P_r	replication probability	0.1

Table 3.3: LGPC parameters for feedback control optimization.

be explained by the fact that as there are more inputs, the search space becomes larger thus the drop in performance of Monte Carlo.

The expression of the best control law b^{FB} is:

$$\begin{aligned} b_1^{\text{FB}}(t) &= \log(s_9(t - T_0/2)), \\ b_2^{\text{FB}}(t) &= \exp(0.18549\ \frac{s_1(t - 4T_0/3)}{s_9(t)}), \\ b_3^{\text{FB}}(t) &= -1.12724, \\ J_{\text{FB}}/J_0 &= 0.7451. \end{aligned} \tag{3.5}$$

LGPC managed to successfully combined sensors signals, delayed sensor signals and nonlinear function to build a controller b^{FB} that reduces even further the cost function compared to the b^{MF} control law. The actuation command resulting from b^{FB} is plotted in figure 3.11. We note that the control resembles, again, the boat tailing solution, however this strategy is augmented by a phasor control for

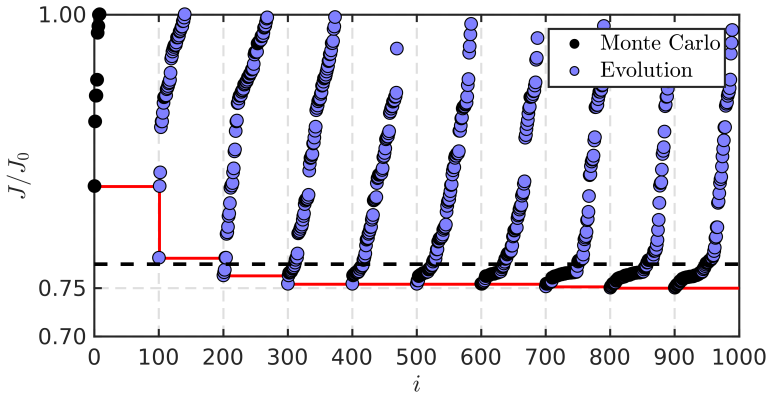


Figure 3.10: Same as figure 3.7 but for the feedback optimization LGPC.

the front and bottom cylinder, meaning that the control of the front and bottom cylinder is directly related to the oscillatory dynamics of the flow (Brunton & Noack, 2015). A spectral analysis shows that the main frequency of b_2 and b_3 are both $f_3 = 0.112$, suggesting that it is a direct feedback.

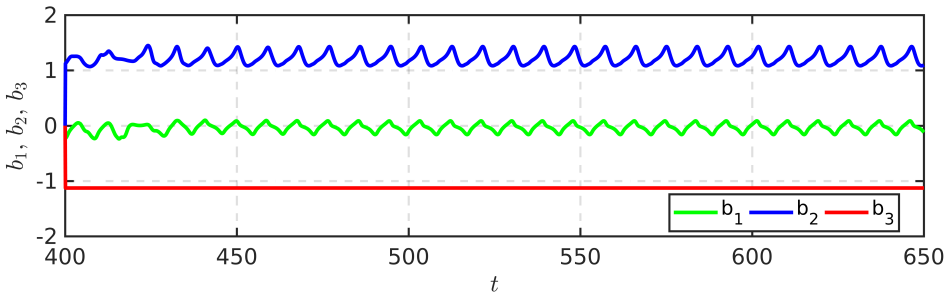


Figure 3.11: Time series of the actuation command for the best feedback control law found with LGPC.

Figure 3.12 shows the characteristics of flow controlled by b^{FB}. The controlled flow is purely harmonic according to figure 3.12d, but the phase portrait, figure 3.12b, is slightly deformed due to the third harmonic (not shown in the figure), even though its amplitude is imperceptible. The amplitude oscillations of the lift coefficient also increased and the mean value is not null.

The kinematics of the flow, figure 3.13, show that the near jet disappeared and the length of the recirculation decreased like the two last control strategies. However, we note that the recirculation bubble is a bit less reduced. Also, the vortices stay attached longer before shedding. This is especially true for the bottom part where the positive vortex stretches unusually, see figure 3.13c, 3.13d, 3.13e and 3.13f. The intensity of the positive vortices is also lesser than the previous control. This can be explained by the re-energization of the shear layers, especially the bottom one, due to the periodic component of the forcing, like Protas (2004). The rotation of the front cylinder has been reported in other studies, such as Cornejo Maceda *et al.* (2019), but its effect is not yet fully understood.

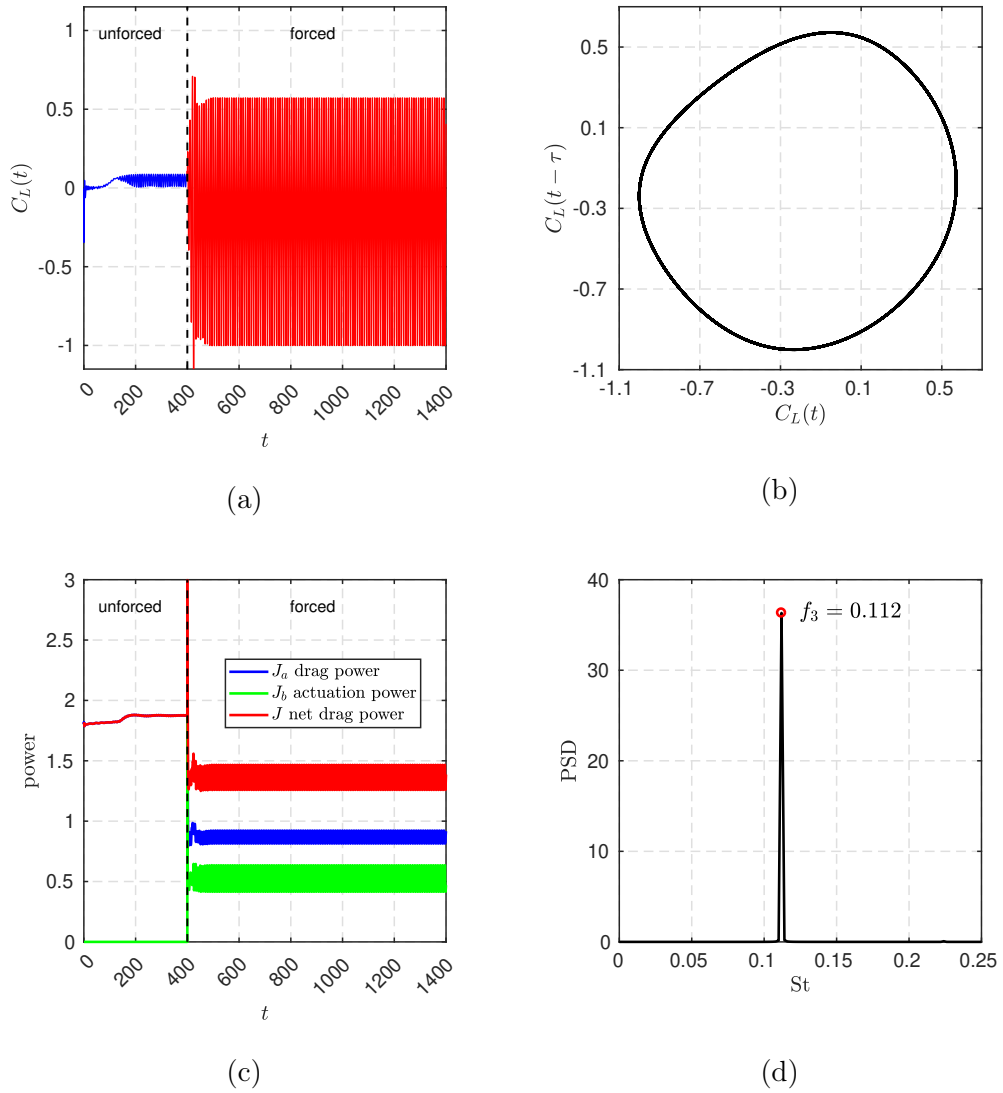
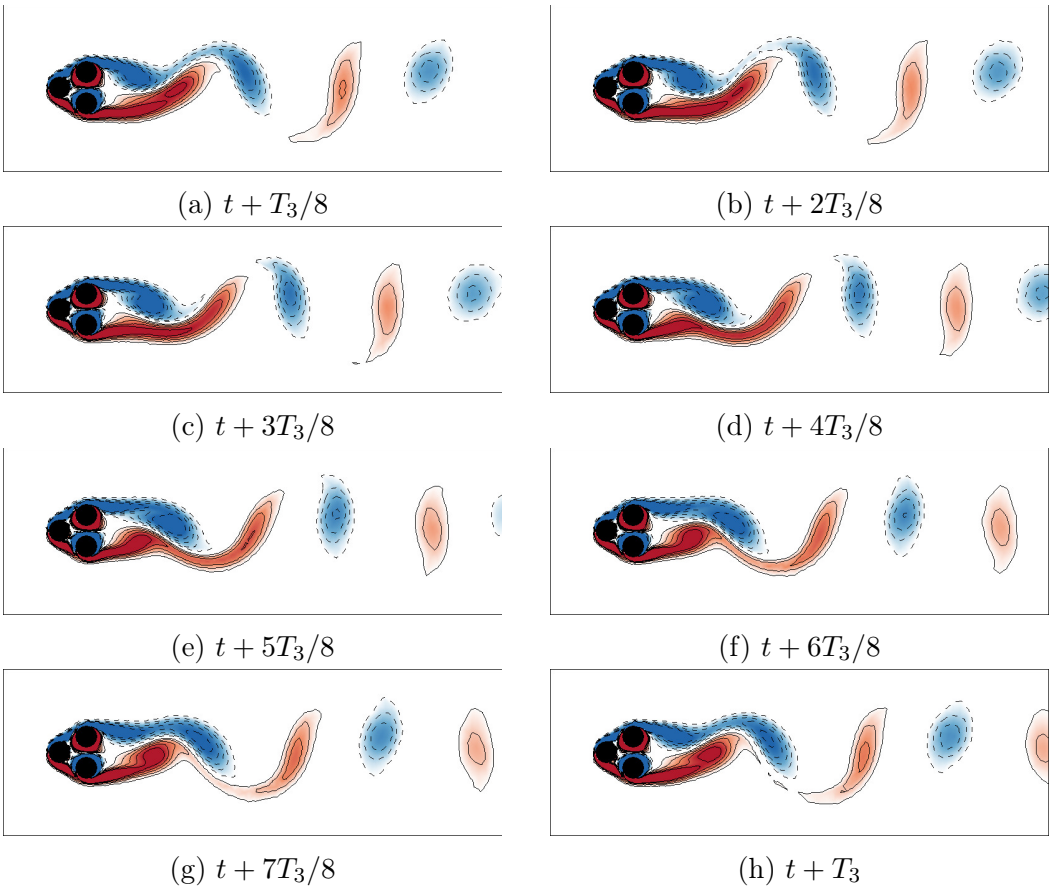


Figure 3.12: Characteristics of the flow controlled by $\boldsymbol{b}^{\mathrm{FB}}$ starting from the steady solution ($t = 0$). The transient spans until $t \approx 400$. (a) Time evolution of the lift coefficient C_L, (b) phase portrait, (c) time evolution of the drag power J_a (blue), actuation power J_b (green) and net drag power J (red) and (d) Power Spectral Density (PSD) showing the frequency $f_3 = 0.112$.

3.5 General control law optimization: multi-frequency and feedback control

In this last section, we run a hybrid optimization allowing both multi-frequency forcing and feedback control. We have seen in section 3.3, that periodic forcing has not been selected to control the fluidic pinball, this may be related to the difficulty to build the proper frequency for control. By adding flow information, we can expect LGPC to build an open-loop periodic forcing modulated by a sensor signal



(a) $t + T_3/8$ (b) $t + 2T_3/8$

(c) $t + 3T_3/8$ (d) $t + 4T_3/8$

(e) $t + 5T_3/8$ (f) $t + 6T_3/8$

(g) $t + 7T_3/8$ (h) $t + T_3$

Figure 3.13: Vorticity fields of the flow controlled by the feedback control law derived by LGPC (a)-(f) Time evolution of the vorticity field in the last period of the 1000 time units simulation. The color code is the same as figure 3.1. T_3 is the period associated to the frequency f_3.

and thus enable a richer control. Such approach has been successfully employed to reduce the recirculation bubble of a back-ward facing step at Reynolds number $Re = 31500$ in Chovet *et al.* (2017). Then, we allow $\boldsymbol{s}$ and $\boldsymbol{h}$ as inputs of the controller and equation 1.1 becomes: $\boldsymbol{b}(t) = \boldsymbol{K}(\boldsymbol{s}(t), \boldsymbol{h}(t))$. The LGPC parameters are summarized in table 3.4.

Figure 3.14 illustrates the learning process for the hybrid optimization. First, Monte Carlo optimization struggles to find a good individual and after 100 random evaluations, the cost of the best individual is only $J/J_0 = 0.92$. As detailed in section 3.4, this can be explained by the adding of extra inputs, which enlarges the search space. Contrary to the multi-frequency optimization and feedback control optimization, we note that significant and regular improvements are made at each generation until reaching a plateau at $J_{\mathrm{HB}} = 0.7363$ after seven generations. For all generations the distribution of the individuals looks linear as opposed to the two previous optimizations where there was an accumulation of good individuals in the final generations. This may be explained by the fact that as new and more efficient individuals are built at each generation, the is still a lot of diversity in the population. When the learning is slowed down, then the best individual takes

parameter	description	value
	function library	$F_2 = \{+, -, \times, \div,$ $\exp, \tanh, \sin, \cos, \log\}$
$\boldsymbol{s}$, $\boldsymbol{h}$	controller inputs	$s_i(t)$, $i = 1..36$ $h_i(t)$, $i = -4, .., -1, 1, ..4$
N_{vr}	number of variable registers	$8 + 36 + 3 = 47$
N_{cr}	number of constant registers	10
$N_{\text{inst,max}}$	max number of instructions	50
N_{popsize}	population size	100
N_G	number of generations	10
N_{tour}	tournament size	7
N_e	elitism	1
P_c	crossover probability	0.6
P_m	mutation probability	0.3
P_r	replication probability	0.1

Table 3.4: LGPC parameters for general optimization including multi-frequency forcing and feedback control.

over the population thanks to replication. From there, we enter a phase of fine tuning of the control law with only small improvements and thus an accumulation of good individuals in the generation. Such behaviour is likely to be observed if more generations were computed.

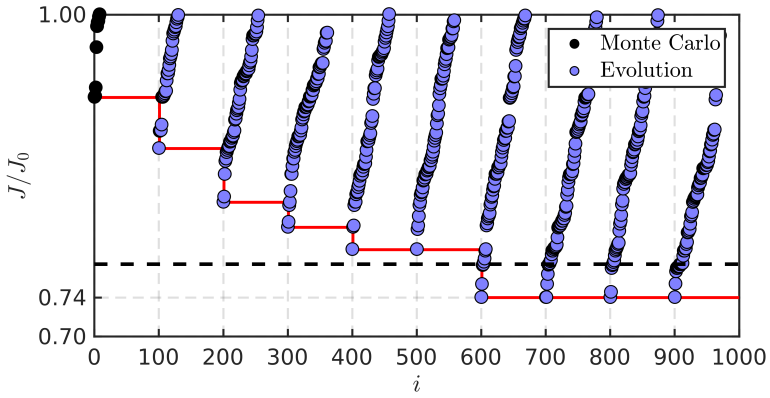


Figure 3.14: Same as figure 3.7 but for the hybrid optimization LGPC.

The final control law b^{HB} reads:

$$\begin{aligned} b_1^{\text{HB}}(t) &= \cos(\cos(s_8(t - T_0/2)) + 0.88123), \\ b_2^{\text{HB}}(t) &= \cos(\cos(s_8(t - T_0/2))), \\ b_3^{\text{HB}}(t) &= -0.36574 - s_2(t), \\ J_{\text{HB}}/J_0 &= 0.7363 \end{aligned} \tag{3.6}$$

The control built includes sensor information, a nonlinear function, cos, but no open-loop periodic function. The three components contain feedback information.

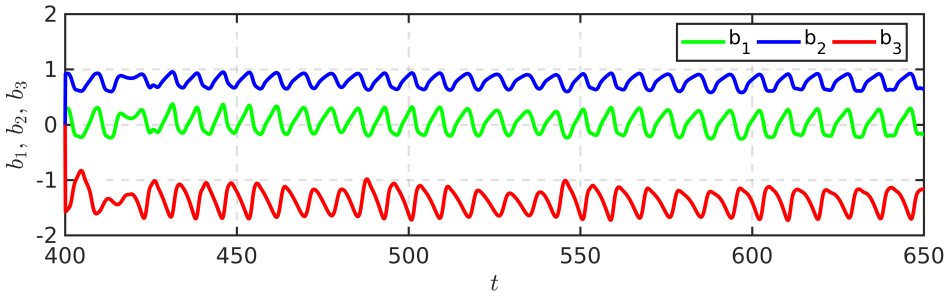


Figure 3.15: Time series of the actuation command for the best feedback control law found with LGPC.

The time series of this control are plotted in figure 3.15. The mean values of all controllers are in line with a boat tailing configuration, however there is a non-negligible oscillatory component for all actuations. The three cylinders are in direct feedback as the dominant frequency of the actuation commands are $f_4 = 0.108$, the main frequency of the flow. So far, b^{HB} is the control that reduces the most the cost function with $J_{\mathrm{HB}}/J_0 = 0.7363$. However, in figure 3.16a and 3.16c, we notice that the transient extends until ≈ 700 convective time units. The cost of the controlled flow computed on the post-transient regime is $J/J_0 = 0.7369$, showing that the extended transient only brings a negligible improvement. We note that taking into account a longer time-window may enable solutions with long transients.

We notice in figure 3.16a that the oscillations of the lift coefficient increased alongside with its mean value. Yet, this asymmetry is not obvious in the figure 3.17. Controlled with b^{HB}, the flow is similar to the previous feedback control. We note, nevertheless, that the recirculation bubble is slightly larger. As for b^{FB}, the vortices stay attached longer thanks to the closed-loop periodic forcing. We notice in particular that the positive vortex is shed after a longer time interval than the negative vortex, indeed we notice a larger distance between a positive vortex and the previous negative vortex downstream than between a negative vortex and the previous positive vortex downstream.

3.6 Conclusion

In this chapter, we apply the LGPC methodology previously described to minimize the net drag power of the fluidic pinball. First, a parametric study on the subspace of symmetric steady forcing supports that the boat tailing configuration appears as a key strategy to reduce the drag power. Then three search spaces are explored, first we allow for multi-frequency forcing, then we optimize a feedback control law, and finally, we allow both strategies for a hybrid optimization. All three optimizations built control laws that include a boat tailing-like structure and discard the open-loop periodic functions when they are in the function library. However, we notice that an asymmetry in the boat tailing is systematically favored. Some improvement can be achieved with the addition of sensor

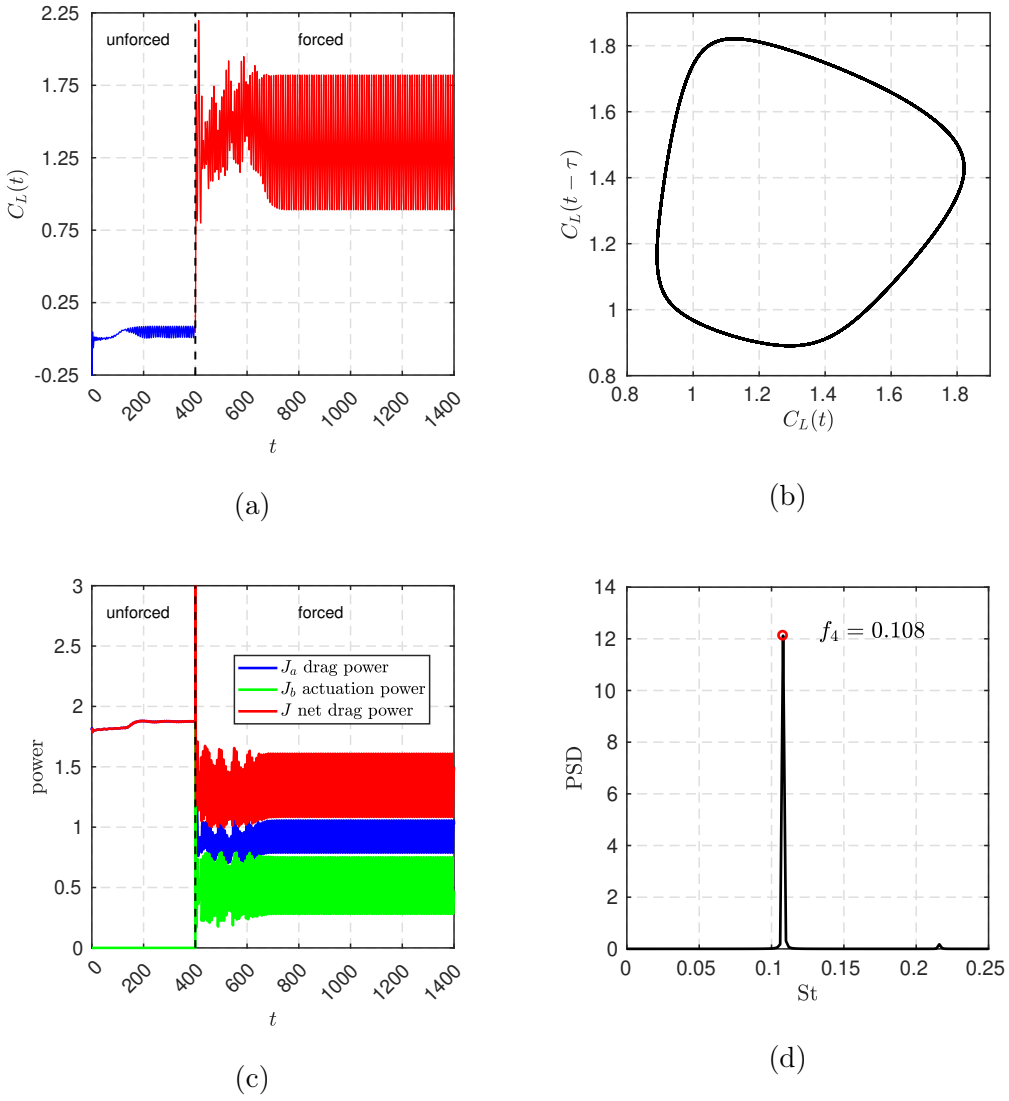
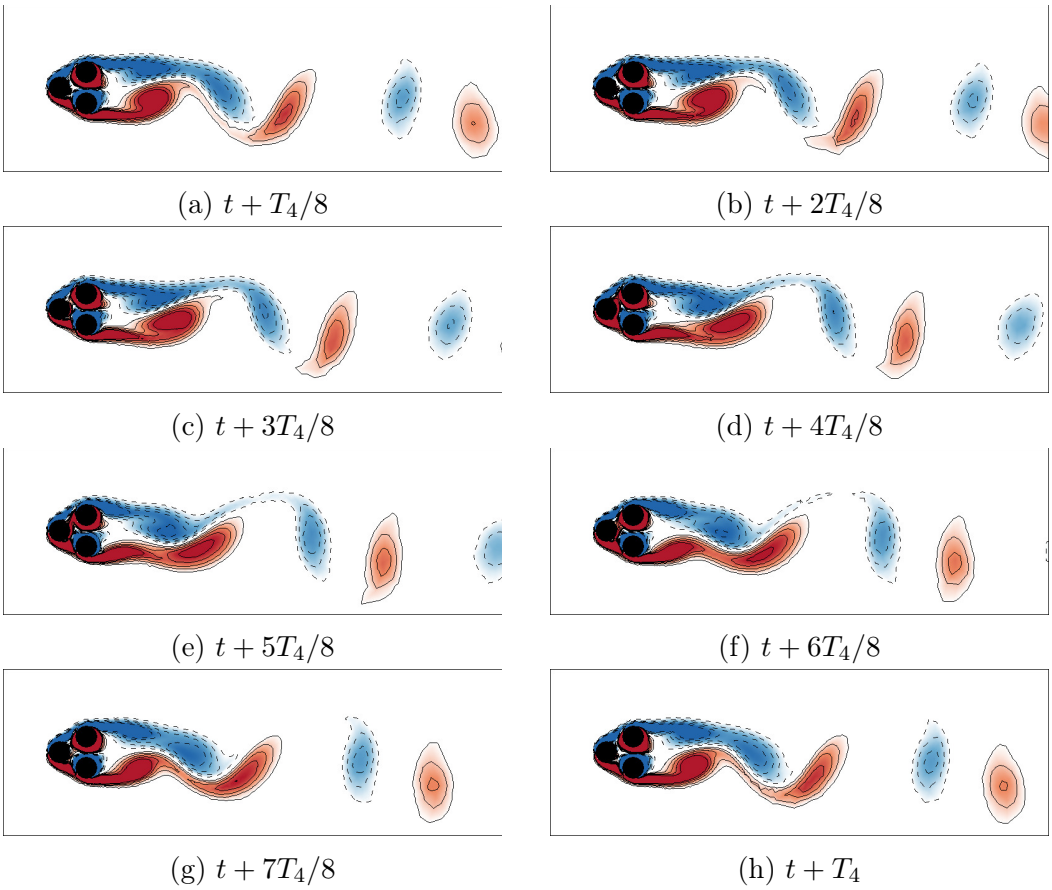


Figure 3.16: Characteristics of the flow controlled by $\boldsymbol{b}^{\mathrm{HB}}$ starting from the steady solution ($t = 0$). The transient spans until $t \approx 400$. (a) Time evolution of the lift coefficient C_L, (b) phase portrait, (c) time evolution of the drag power J_a (blue), actuation power J_b (green) and net drag power J (red) and (d) Power Spectral Density (PSD) showing the frequency $f_4 = 0.108$ and its first harmonic.

information, reducing the cost from $J_{\mathrm{MF}}/J_0 = 0.7476$ to $J_{\mathrm{HB}}/J_0 = 0.7363$. The costs of all runs are summarized in table 3.5. Thus, in less than 1000 evaluations, LGPC managed to build a control combining asymmetric boat tailing and phasor control to reduce the net drag power in a model-free and with very few knowledge a priori. LGPC rediscovers, in particular, that to delay the vortex shedding, one can re-energize the shear layer with periodic forcing and that vectoring the flow towards the centreline helps to increase the base pressure. The hybrid control built with LGPC achieves the most net drag reduction so far.

In this part, we unveiled the forces at play in the learning process of genetic



(a) $t + T_4/8$ (b) $t + 2T_4/8$

(c) $t + 3T_4/8$ (d) $t + 4T_4/8$

(e) $t + 5T_4/8$ (f) $t + 6T_4/8$

(g) $t + 7T_4/8$ (h) $t + T_4$

Figure 3.17: Vorticity fields of the flow controlled by b^{HB}. (a)-(f) Time evolution of the vorticity field in the last period of the 1000 time units simulation. The color code is the same as figure 3.1. T_4 is the period associated to the frequency f_4.

programming and applied to the reduction of the net drag power of the fluidic pinball. The parametric study of LGPC, carried out in chapter 2, revealed the importance of key meta-parameters. Such analysis serves as a guide to select adequate parameters for future LGPC studies. Applied to the fluidic pinball, LGPC successfully managed to build control laws in different search spaces in less than 1000 evaluations, revealing key actuation mechanisms, without any prior knowledge, and combining them for further performance: the best control comprises a combination of asymmetric boat tailing and phasor control.

The next part will focus on accelerating the learning process and building better control laws thanks to our new algorithm the gradient-based machine learning control.

search space	control law	J/J_0	J_a/J_0	J_b/J_0
unforced natural	-	1	1	0
symmetric steady	$b_1 = 0$, $b_2 = -b_3 = \text{cst}$	0.7652	0.5695	**0.1956**
multi-frequency forcing	$\boldsymbol{b}(t) = K(\boldsymbol{h}(t))$	0.7476	0.5109	0.2368
feedback control	$\boldsymbol{b}(t) = K(\boldsymbol{s}(t))$	0.7451	**0.4744**	0.2706
general control	$\boldsymbol{b}(t) = K(\boldsymbol{s}(t), \boldsymbol{h}(t))$	**0.7363**	0.4845	0.2518

Table 3.5: Summary of the performances for the best solutions of each type of optimization. The bold values are the best for each cost.

Part II

Faster wake control with genetic programming and gradient-based methods.

Chapter 4

Explorative gradient-method

The second part of this thesis focuses on gradient-based machine learning control methods. In this chapter, we further explore the exploration-exploitation optimization principles, presented in chapter 2, with a new algorithm, the explorative gradient method (EGM) introduced in Li *et al.* (2021). EGM is a parametric optimization algorithm for a moderate number of parameters. EGM combines the exploration power of latin hypercube sampling and the convergence rate and efficiency of downhill simplex for exploitation. The algorithm is described in detail as it shares key characteristics with the gradient-enriched machine learning control (gMLC) algorithm studied (chapter 5). Firstly, we discuss the principles of exploration and exploitation and describe the EGM algorithm in section 4.1. Secondly, we detail the cost function and carry out a parametric study on the space of steady symmetric actuations in section 4.2. Finally, the EGM is employed to stabilize the fluidic pinball in two different search spaces: a 2-dimensional space of symmetric periodic actuations (section 4.3) and a 3-dimensional space by allowing the independent rotation of the three cylinders (section 4.4). This chapter is adapted from Cornejo Maceda *et al.* (2020).

Contents

4.1 Control optimization framework

In this section, we describe the optimization principles behind EGM and gMLC and describe the EGM algorithm in detail.

4.1.1 Optimization principles—Exploration versus exploitation

EGM and gMLC both combine the advantages of exploitation and exploration. In this new context, exploitation is based on a downhill simplex method with the best performing of all tested control laws. The goal is to 'slide down' the best identified minimum.

Exploration is performed with another algorithm using all previously tested individuals. The goal is to find potentially new and better minima, ideally the global minimum. The method for exploration depends on the search space. For a low-dimensional parameter space, a space-filling version of the Latin Hypercube Sampling (LHS) guarantees optimal geometric coverage of the search space. For a high-dimensional function space, genetic programming is found to be efficient.

EGM and gMLC start with an initial set of individuals to be evaluated. Then, exploitive and explorative phases iterate until a convergence criterion is reached. The iteration hedges against several worst-case scenarios. The control landscape may have only a single minimum accessible from any other point by steepest descend. In this case, exploration is often inefficient, although it might help in avoiding slow marches through long shallow valleys (Li *et al.*, 2021). The control landscape may also have many minima accessible by gradient-based searches. In this case, exploitation is likely to incrementally improve performance in suboptimal minima and the search strategy should have a significant investment in exploration. The minima of the control landscape may also have narrow basins of attractions for gradient-based iterations and extended plateaus. This is another scenario where iteration between exploitation and exploration is advised.

Many optimizers balance exploration and exploitation and gradually shift from the former to the latter. This strategy sounds reasoning but is not a good hedge against the described worst-case scenarios where almost all exploitative or almost all explorative algorithms are doomed to fail.

Note that the chances of exploration landing close to a new better minimum are small. Yet, the explorative phases may find new basins of attractions for successful gradient-based descends. This is another argument for the alternating execution of exploration and exploitation.

Finally, we note that the proposed explorative-exploitive schemes allows that both kinds of iterations may be adjusted to the control landscape. For instance, LHS in a high-dimensional search space will initially explore only the boundary and may better be replaced by Monte Carlo or a genetic algorithm. We refer to

Li *et al.* (2021) for a thorough comparison of EGM and five common optimizers. In this chapter, we focus on EGM, whereas gMLC is detailed in chapter 5.

4.1.2 Parameter optimization with the explorative gradient method (EGM)

The Explorative Gradient Method (EGM) optimizes N_p parameters $\boldsymbol{b} = \left[b_1, \ldots, b_{N_p}\right]^\intercal$ with respect to cost $J(\boldsymbol{b})$ and comprises exploration and exploitation phases. In the context of parameter optimization, we do not differentiate between the control law $\boldsymbol{K} = \text{const}$ and the associated actuation command $\boldsymbol{b} = \boldsymbol{K}$. The search space, or actuation domain, is a compact subset $\mathcal{B}$ of $\mathbb{R}^{N_p}$, typically defined by upper and lower bounds for each parameter. The exploration phase is based on a space-filling variant of Latin hypercube sampling (LHS) (McKay *et al.*, 1979) whereas the exploitation phase is carried out by Nelder-Mead's *downhill simplex* (Nelder & Mead, 1965).

The first $N_p + 1$ initial individuals $\boldsymbol{b}_m$, $m = 1, \ldots, N_p + 1$ define the first 'amoeba' of the downhill simplex method. The first individual $\boldsymbol{b}_1$ is typically placed at the centre of $\mathcal{B}$. The N_p remaining vertices are slightly displaced along the b_m axes. In other words, $\boldsymbol{b}_m = \boldsymbol{b}_1 + h_m \boldsymbol{e}_{m-1}$ for $m = 2, \ldots, N_p + 1$. Here, $\boldsymbol{e}_m := \left[\delta_{m,1}, ..., \delta_{m,N_p}\right]^\intercal$ is the unit vector in the mth direction and h_m is the corresponding step size. The increment h_m is chosen to be small compared to the range of the corresponding dimension.

The exploitation phase employs the downhill simplex method. This method is robust and widely used for data-driven optimization in low and moderate-dimensional search spaces. No local gradients are required. The new individual is a linear combination of the simplex individuals and follows a geometric reasoning. The vertex with the worst performance is replaced by a point reflected at the centroid of the opposite side of the simplex. This step leads to a mirror-symmetric version of the simplex where the new vertex has the best performance if the cost function depends linearly on the input. Subsequent operations, like expansion, single contraction and global shrinking ensure that iterations exploit a favorable downhill behaviour and avoid getting stuck by nonlinearities. The downhill simplex algorithm is detailed in the *Exploitation phase* of algorithm 1.

The explorative phase of EGM is inspired by the LHS method. LHS aims to fill the complete domain $\mathcal{B}$ optimally. The pre-defined number m of individuals maximizes the minimum distance of its neighbours:

$$\left\{\boldsymbol{b}_1^{\text{LHS}}, \ldots, \boldsymbol{b}_m^{\text{LHS}}\right\} := \underset{\boldsymbol{b}_1,\ldots,\boldsymbol{b}_m \in \mathcal{B}}{\arg\max} \quad \min_{\substack{i \in \{1,\ldots,m-1\},\\ j \in \{i+1,\ldots,m\}}} \left\|\boldsymbol{b}_i - \boldsymbol{b}_j\right\|.$$

Here, $\|\cdot\|$ denotes the Euclidean norm. The number of individuals has to be determined in advance and cannot be augmented. This static feature is incompatible with the iterative EGM algorithm. Thus, we resort to a recursive 'greedy' version. Let $\boldsymbol{b}_1^\bullet$ be the first individual. Then, $\boldsymbol{b}_2^\bullet$ maximizes the distance from $\boldsymbol{b}_1^\bullet$,

$$\boldsymbol{b}_2^\bullet := \underset{\boldsymbol{b} \in \mathcal{B}}{\arg\max} \|\boldsymbol{b} - \boldsymbol{b}_1^\bullet\|.$$

The mth individual maximizes the minimum distance to all previous individuals,

$$\boldsymbol{b}_m^\bullet := \arg\max_{\boldsymbol{b}\in\mathcal{B}} \min_{i\in\{1,\dots,m-1\}} \|\boldsymbol{b} - \boldsymbol{b}_i^\bullet\|.$$

This recursive definition allows adding explorative phases from any given set of individuals.

Exploitation and exploration are iteratively continued until the stopping criterion is reached. The stopping criterion may be a total number of cost function evaluations, a threshold value for performance improvement. The *Explorative Gradient Method* (EGM) phases are summarized in algorithm 1.

4.2 Control objective—Flow stabilization

In this section, we present the control problem associated to the stabilization of the fluidic pinball. The control is performed following the same conditions as chapter chapter 3.

4.2.1 Cost function

In this study, we aim to stabilize the unstable steady symmetric Navier-Stokes solution at $\mathrm{Re}_D = 100$. The associated objectives are J_a, quantifying the closeness to the symmetric steady solution and J_b, the actuation power. The cost J_a is defined as the temporal average of the residual fluctuation energy of the actuated flow field $\boldsymbol{u_b}$ with respect to the symmetric steady flow $\boldsymbol{u}_s$:

$$J_a = \frac{1}{T_{ev}} \int_{t_0}^{t_0+T_{ev}} j_a(t)\,\mathrm{d}t \tag{4.1}$$

with the instantaneous cost function

$$j_a(t) = \|\boldsymbol{u_b}(t) - \boldsymbol{u}_s\|_\Omega^2 \tag{4.2}$$

based on the L_2-norm

$$\|\boldsymbol{u}\|_\Omega = \sqrt{\iint_\Omega u^2 + v^2\,\mathrm{d}\boldsymbol{x}}. \tag{4.3}$$

The control is activated at $t_0 = 400$ convective time units after the starting kick on the steady solution. Thus, we have a fully established post-transient regime. The cost function is evaluated until $T_{\mathrm{ev}} = 1400$ convective time units. Thus, the time average is effected over 1000 convective time units to make sure that the transient regime has far less weight as compared to the actuated regime. Yet, a faster stabilizing response to actuation is clearly desirable and factors positively into the cost.

J_b is naturally chosen as a measurement of the actuation energy investment. J_b has already been described in section 3.1 and we only recall its expression here:

$$J_b(\boldsymbol{b}) = \frac{1}{T_{ev}} \int_{t_0}^{t_0+T_{\mathrm{ev}}} \sum_{i=1}^{3} \mathcal{P}_{\mathrm{act,i}}\,\mathrm{d}t$$

Algorithm 1: Explorative Gradient Method

Result: $\boldsymbol{b}^*$, the best individual
Initialize the $N_p + 1$ individuals of the dataset $\mathcal{B}_I$;
Test all the individuals;
Build the simplex $\mathcal{S}$ by taking the $N_p + 1$ best individuals;
while *Stopping criterion is not reached* **do**
  Exploration phase—Latin hypercube sampling
    Select $\boldsymbol{b}^{\text{LHS}}$ by solving:

$$\boldsymbol{b}^{\text{LHS}} := \arg\max_{\boldsymbol{b}\in\mathcal{B}} \quad \min_{\boldsymbol{b}_i\in\mathcal{B}_I} \|\boldsymbol{b} - \boldsymbol{b}_i\|$$

    Test $\boldsymbol{b}^{\text{LHS}}$;
    Augment dataset: $\mathcal{B}_I := \mathcal{B}_I \cup \left\{\boldsymbol{b}^{\text{LHS}}\right\}$;
    Update simplex: replace the worst individual of $\mathcal{S}$ by $\boldsymbol{b}^{\text{LHS}}$ if b^{LHS} performs better;
  end
  Exploitation phase—Downhill simplex
    Sort and relabel $\mathcal{S}$ such as: $J_1^S \leq J_2^S \leq \ldots \leq J_{N_p+1}^S$;
    Compute the centroid $\boldsymbol{c} = \frac{1}{N_p}\sum_{i=1}^{N_p} \boldsymbol{b}_i$ of $\mathcal{S}$ excluding $\boldsymbol{b}_{N_p+1}$;
    Reflection: compute and test $\boldsymbol{b}_r := \boldsymbol{c} + (\boldsymbol{c} - \boldsymbol{b}_{N_p+1})$;
    if $J_1^S < J_r^S < J_{N_p+1}^S$ **then**
      Update simplex: $\boldsymbol{b}_{N_p+1} := \boldsymbol{b}_r$;
    else if $J_r^S < J_1^S$ **then**
      Expansion: compute and test $\boldsymbol{b}_e := \boldsymbol{c} + 2\ (\boldsymbol{c} - \boldsymbol{b}_{N_p+1})$;
      Update simplex: $\boldsymbol{b}_{N_p+1} := \min\{\boldsymbol{b}_r, \boldsymbol{b}_e\}$;
    else if $J_{N_p+1}^S \leq J_r^S$ **then**
      Contraction: compute and test $\boldsymbol{b}_c := 1/2\ (\boldsymbol{c} + \boldsymbol{b}_{N_p+1})$;
      if $J_c^S < J_{N_p+1}^S$ **then**
        Update simplex: $\boldsymbol{b}_{N_p+1} := \boldsymbol{b}_c$;
      else
        Shrink: compute and test
        $\boldsymbol{b}_{s,i} := 1/2\ (\boldsymbol{b}_1 + \boldsymbol{b}_i), i = 2, \ldots, N_p + 1$;
        Update simplex: $\boldsymbol{b}_i := \boldsymbol{b}_{s,i}, i = 2, \ldots, N_p + 1$;
      end
    end
    Augment dataset: add all the new individuals to $\mathcal{B}_I$;
  end
end

where $T_{\text{ev}} = 1000$ time units in this case and $\mathcal{P}_{\text{act,i}}$ is the actuation power supplied to cylinder i.

In this study, optimization is based on the cost function $J = J_a$ and the actuation investment J_b is evaluated separately. We refrain from a cost function J which employs the objective function J_a and penalizes the actuation investment J_b with suitable weight γ, i.e., $J = J_a + \gamma J_b$. The procedure has two reasons. First, the distance between two flows and actuation energy belong to two different worlds, kinematics and dynamics. Any choice of the penalization parameter γ will be subjective and implicate a sensitivity discussion. Second, the complete stabilization of the steady solution would lead to a vanishing actuation $\boldsymbol{b} \equiv 0$ and thus vanishing energy J_b. Thus, the optimization problem without actuation energy can be expected to be well-posed.

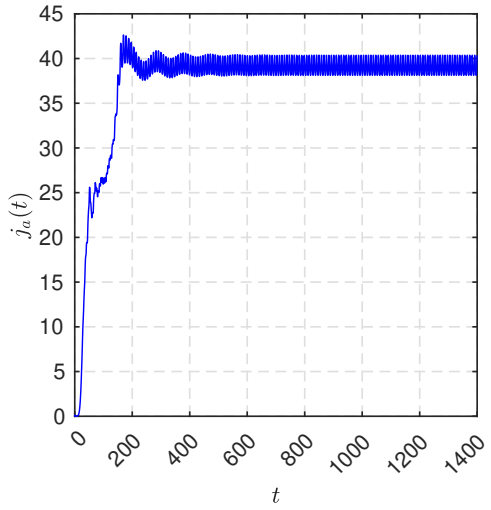


Figure 4.1: Time evolution of the instantaneous cost function j_a for the unforced natural flow.

The instantaneous cost function j_a of the unforced flow is shown in figure 4.1. We notice a slight overshoot around $t = 200$ before converging to a post-transient fluctuating regime. The post-transient regime shows the expected periodic behaviour from von Kármán vortex shedding. The cost averaged over 1000 convective time units is $J_0 = 39.08$ and serves as reference to actuation success.

For more informations on the optimization problem and the unforced flow of the fluidic pinball, we refer to section 3.1.

4.2.2 Symmetric steady actuation for wake stabilization

This section describes the behaviour of the fluidic pinball under a symmetric steady actuation. In this configuration, only the two rearward cylinders rotate at equal but opposite rotation speeds, $b_2 = -b_3$. When b_2 is positive, the rearward cylinders accelerate the outer boundary layers and suck near-wake fluid upstream. This

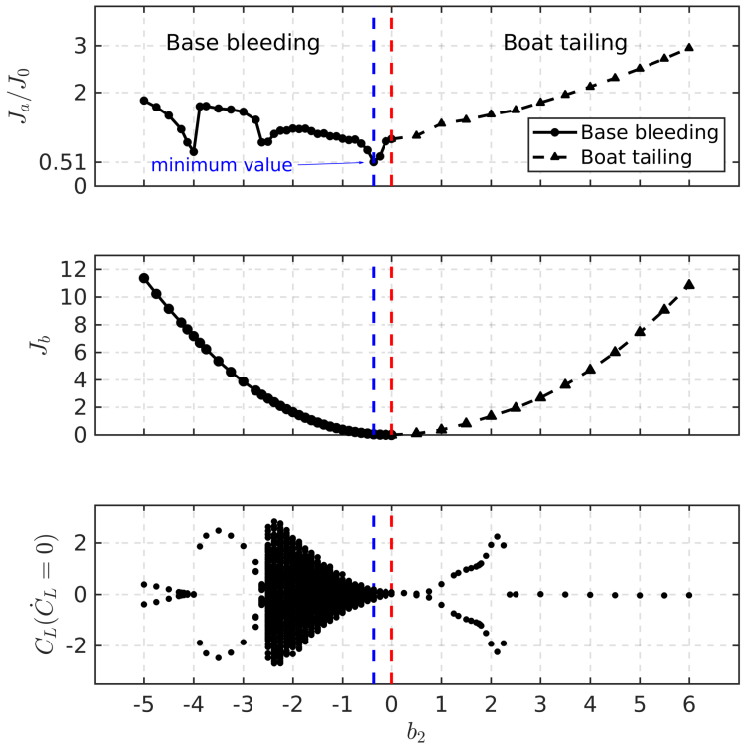


Figure 4.2: Parametric study for symmetric steady forcing. $b_2 = -b_3$ is the velocity of the bottom cylinder. The normalized distance to the steady solution J_a/J_0 (top) and the actuation power J_b (middle) are plotted as a function of b_2. The bifurcation diagram (bottom) comprises all local maximum and minimum lift values. The vertical red dashed line corresponds to $b_2 = 0$ and separates the base bleeding and the boat tailing configurations. The global minimum of J_a/J_0 is reached at $b_2 = -0.375$, as indicated by a vertical blue dashed line.

forcing delays separation, mimics Coanda forcing and leads to a fluidic boat tailing. When b_2 is negative, the cylinders eject fluid in the near wake like in base bleed and oppose the outer boundary-layer velocities. Figure 4.2 shows the evolution of J_a/J_0 (top), J_b (middle) and the bifurcation diagram (bottom) as a function of b_2.

We limited our study to $b_2 \in [-5, 6]$. The trends are resolved with a discretization step of 0.25 and a finer resolution in the ranges $[-2.5, 0]$ and $[1, 2]$. For each parameter, the cost J_a and actuation power J_b have been computed over 1000 convective time units. The bifurcation diagram has been built by detecting the extrema of the lift coefficient over the last 600 convective time units. The bifurcation diagram reveals five regimes:

Regime $b_2 < -4$: the lift amplitude decreases to zero and the cost decreases to the first minimum.

Regime $-4 < b_2 < -2.5$: the extremal lift values increase and decrease to zero again. The cost approaches another local minimum near $b \approx -2.5$.

Regime $-2.54 < b_2 < 0$**:** a period doubling cascade is observed for decreasing b_2 leading to a chaotic regime. At $b_2 \approx 0.375$, the cost assumes it global minimum with residual fluctuation of the lift coefficient.

Regime $0 < b_2 < 2.375$**:** the cost and the extremal lift values monotonically increase.

Regime $2.375 < b_2$**:** the Coanda forcing completely stabilizes a symmetric steady solution. The cost increases with the rotation speed.

Interestingly, the boat tailing discontinuity at $b_2 = 2.375$ does not appear in the graph of the cost function J_a/J_0. This continuity, even in the derivative, corresponds to a continuous passage from a periodic symmetrical solution to a stationary solution which is itself symmetrical. As the value of the cost function indicates, this stationary solution is quite far from the unforced symmetric steady solution. The global minimum of $J_a/J_0 = 0.51$ is reached near $b_2 = -0.375$, i.e., for a base bleeding configuration, corresponding to a small actuation power $J_b = 0.0490$, roughly 0.1% of the J_0.

The characteristics of the best base bleeding solution leading closest to the symmetric steady solution are depicted in figure 4.3. In figure 4.3a, the lift coefficient is displayed for the unforced transient (blue curve) and the forced flow (red curve). The unforced flow terminates in an asymmetric shedding with positive lift values. After the start of forcing, the lift coefficient oscillates vigorously around its vanishing mean value. This forced statistical symmetry is corroborated by the oscillating jet in figures 4.4a-4.4h. Base bleed increases the velocity of the rearward jet compared to the unforced flow. This jet instability mitigates the Coanda effect on the bottom and top cylinder, i.e., the jet neither stays long at either side.

The vortex shedding persists similar to the unforced flow. However, the dominant frequency is increase from $f_0 = 0.116$ to $f_5 = 0.132$. The instantaneous cost function j_a in figure 4.3c shows an unsteady non-periodic behaviour, reaching intermittently low levels. The broad spectral peak in figure 4.3d is a characteristic of a chaotic regime. The phase portrait in figure 4.3b corroborates the non-periodic oscillatory behaviour. The mean field in figure 4.4j, shows that actuated mean jet is symmetric unlike the mean field of the unforced flow. Moreover, the shear-layer on the upper and lower sides extend farther downstream as compared to the unforced state.

This parametric study reveals that base bleeding is the best symmetric steady forcing strategy to bring the flow close to the symmetric steady solution. However, even though the cost J_a/J_0 is almost halved, the best base bleeding control fails to stabilize the flow.

4.3 Periodic forcing optimization

In this section, we aim to stabilize the symmetric steady solution thanks to a symmetric periodic forcing. In this case, the two back cylinders oscillate in opposite directions whereas the front cylinder stays still. The control ansatz is the

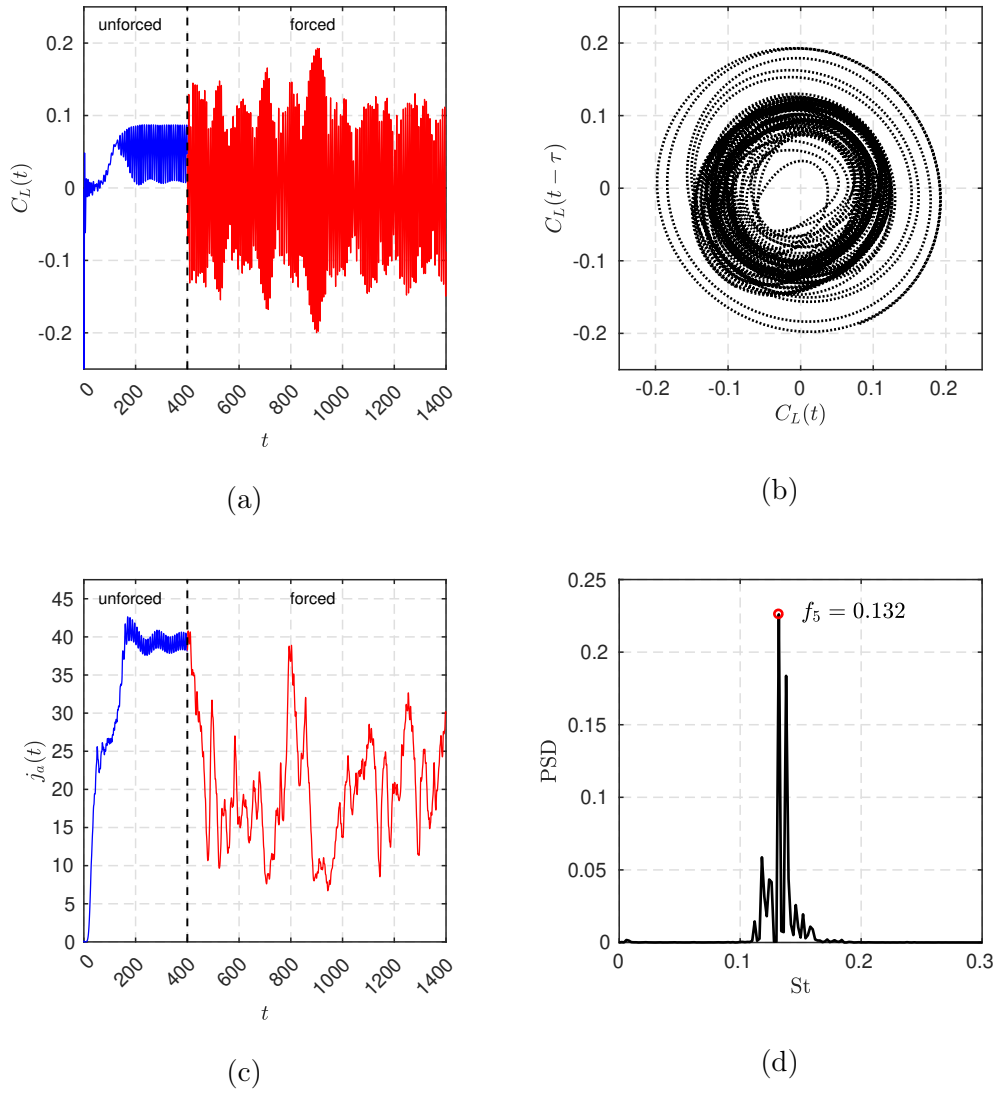
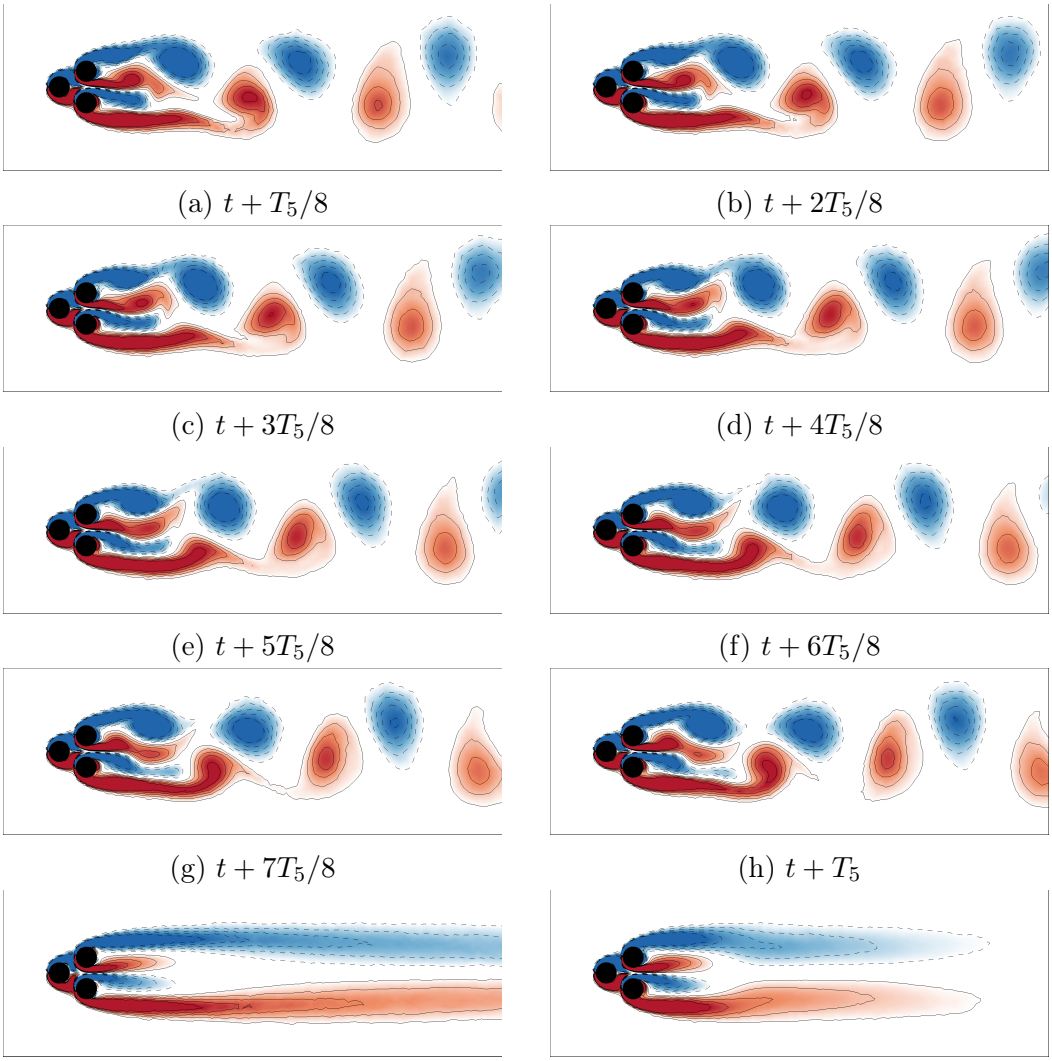


Figure 4.3: Characteristics of the best base bleeding solution. (a) Time evolution of the lift coefficient C_L, (b) phase portrait (c) time evolution of instantaneous cost function j_a and (d) Power Spectral Density (PSD) showing a broad spectral peak at $f_5 = 0.132$. The control starts at $t = 400$. The unforced phase is depicted in blue and the forced one in red. The phase portrait is computed over $t \in [900, 1400]$ and the PSD is computed on the forced regime $t \in [400, 1400]$.

following:

$$\begin{aligned} b_1 = &\quad 0 \\ b_2 = &\quad B\cos(2\pi F t) \\ b_2 = &\quad -B\cos(2\pi F t) \end{aligned}$$

with B, the amplitude of the oscillations, and F, the frequency, being the two parameters to optimize. The search space is limited to $[B, F/f_0]^\intercal \in [0, 5] \times [0, 10]$ as higher amplitudes and frequencies would be beyond our solver capabilities. This two-dimensional search space is explored with EGM. The contour plot in



(a) $t + T_5/8$ (b) $t + 2T_5/8$

(c) $t + 3T_5/8$ (d) $t + 4T_5/8$

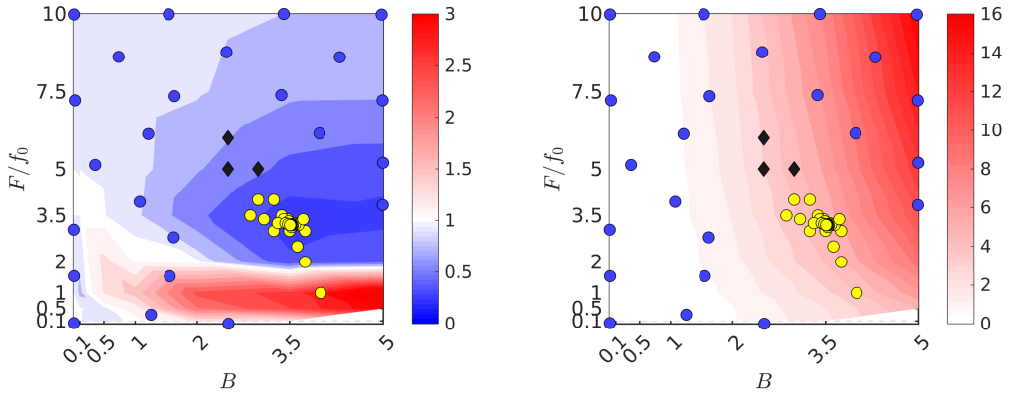
(e) $t + 5T_5/8$ (f) $t + 6T_5/8$

(g) $t + 7T_5/8$ (h) $t + T_5$

(i) Symmetric steady solution (j) Mean field

Figure 4.4: Vorticity fields of the best base bleeding solution. (a)-(f) Time evolution of the vorticity field throughout the last period of the 1400 convective time units, (i) the objective symmetric steady solution and (j) the mean field of the forced flow. The color code is the same as figure 3.1. T_5 is the period associated to the main frequency f_5 of the forced flow. The mean field has been computed by averaging 100 periods.

figure 4.5 depicts the search space based on J_a/J_0 and J_b. The contour plot has been produced thanks to simulations for $B \in \{0.1,\ 0.5,\ 1,\ 2,\ 3.5,\ 5\}$ and $F/f_0 \in \{0.1,\ 0.5,\ 1,\ 2,\ 3.5,\ 5,\ 7.5,\ 10\}$. The steps are finer for low frequencies and low amplitudes. The individuals have been evaluated over 250 convective time units. We notice that there is only one minimum on the plane, close to $[B, F/f_0]^\intercal = [3.51, 3.19]^\intercal$. Also, forcing at frequencies close to the natural frequency resonates with the flow and drastically increases the distance to the steady solution for high amplitudes. For J_b, the contour map expectedly displays high values at high frequencies and large amplitudes. The three initial control laws for EGM are the



(a) J_a/J_0. (b) J_b.

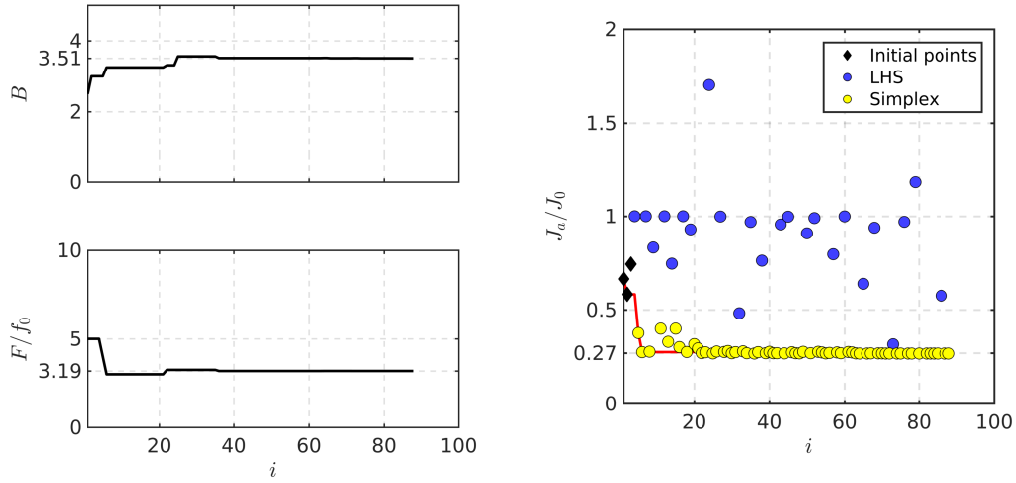
Figure 4.5: Contour plot for J_a/J_0 (a) and J_b (b) as a function of the amplitude B and the normalized frequency F/f_0. For (a), blue (red) regions denote good (bad) performances while white regions correspond to costs that are equivalent to the natural flow. For (b), the color code describes the actuation energy. The symbols represent the individuals tested with EGM: black diamonds for the initial conditions, blue solid circles for exploration phases and yellow solid circles for the exploitation phases. For the legend, refer to figure 4.9.

centre of the box and increments of $1/5$ of the box size in each direction: $[2.5, 5]^\intercal$, $[3, 5]^\intercal$, $[2.5, 6]^\intercal$. As expected, the LHS steps (in blue) spread rather evenly in the domain whereas the simplex steps (in yellow) quickly descend into the global minimum.

Figure 4.6 shows the progression of the best individual throughout the evaluations. The EGM optimization process converges after few tests as J_a/J_0, the amplitude and the frequency reach asymptotic values, without any significant improvement afterwards. The parameters of the best symmetric periodic forcing are denoted by the superscript 'EGM' and read

$$B^{\mathrm{EGM}} = 3.51,$$
$$F^{\mathrm{EGM}}/f_0 = 3.19.$$

The proximity between the initial values and the aimed minimum certainly accelerates the observed convergences. Figure 4.7 shows the evolution of the lift coefficient, the phase portrait, the power spectral density and the instantaneous cost function j_a for the controlled flow. The lift coefficient presents rather symmetric low amplitude oscillations, see figure 4.7a. This goes along with the flow symmetry in figure 4.8a-4.8h. The oscillations are explained by the remaining vortex shedding on both, the upper and lower, side of the fluidic pinball. Even though the far field is close to the symmetric steady solution, this periodic solution changes radically the near field profile. The jet is completely flattened, see figure 4.8c, 4.8d and 4.8e. Moreover, the vorticity around the cylinders is more intense compared to the initial steady solution. This difference is present in the final mean value j_a in figure 4.7c and is responsible for the high actuation power



(a) (b)

Figure 4.6: Evolution of (a) the amplitude B and the normalized frequency F/f_0 and (b) the reduced cost J_a/J_0 as a function of the number of evaluations i, for the EGM optimization process. The red line in (b) shows the evolution of the best cost. The evaluation time is 250 convective time units.

expense, $J_b = 5.2799$. The phase portrait shows a periodic regime, though deformed by the harmonics. The mean frequency $f_6 = 0.398$ is slightly lower than the forcing frequency $F^{\mathrm{EGM}} = 0.37$ and much lower than the natural frequency, showing that it is not just a simple frequency locking, but a nonlinear frequency crosstalk. The non-centred phase portrait indicates that there is still an asymmetry in the flow, that may be a residual effect of the grid's asymmetry. The mean field in figure 4.8j is similar to the symmetric steady solution, however the jet completely vanished. In addition, the distance between the upper and lower vorticity branches is wider compared to the symmetric steady solution.

4.4 General non-symmetric steady actuation

In this section we aim to stabilize the symmetric steady solution by commanding the three cylinders with constant actuation without symmetry constraint. This three-dimensional parameter space is explored with the explorative gradient method. The symmetry along the x-axis of the fluidic pinball allows us to reduce our search space and to explore only positive values of b_1. A coarse initial parametric study carried on b_1, b_2 and b_3 by steps of unity indicates that the global minimum of J_a/J_0 should be near $[b_1, b_2, b_3]^\intercal = [1, 0, 0]^\intercal$. Thus, we limit our research to the actuation domain $\mathcal{B} = [0, 2] \times [-2, 2] \times [-2, 2]$. The limitation of b_1 to positive values exploits the mirror symmetry of the configuration. Figure 4.9 (bottom) depicts the cost function in the actuation domain $\mathcal{B}$. Three planes ($b_1 = \mathrm{const}$) are computed by interpolating parameters on a coarse grid. The individuals computed with EGM are all shown in the 3D space. The four ini-

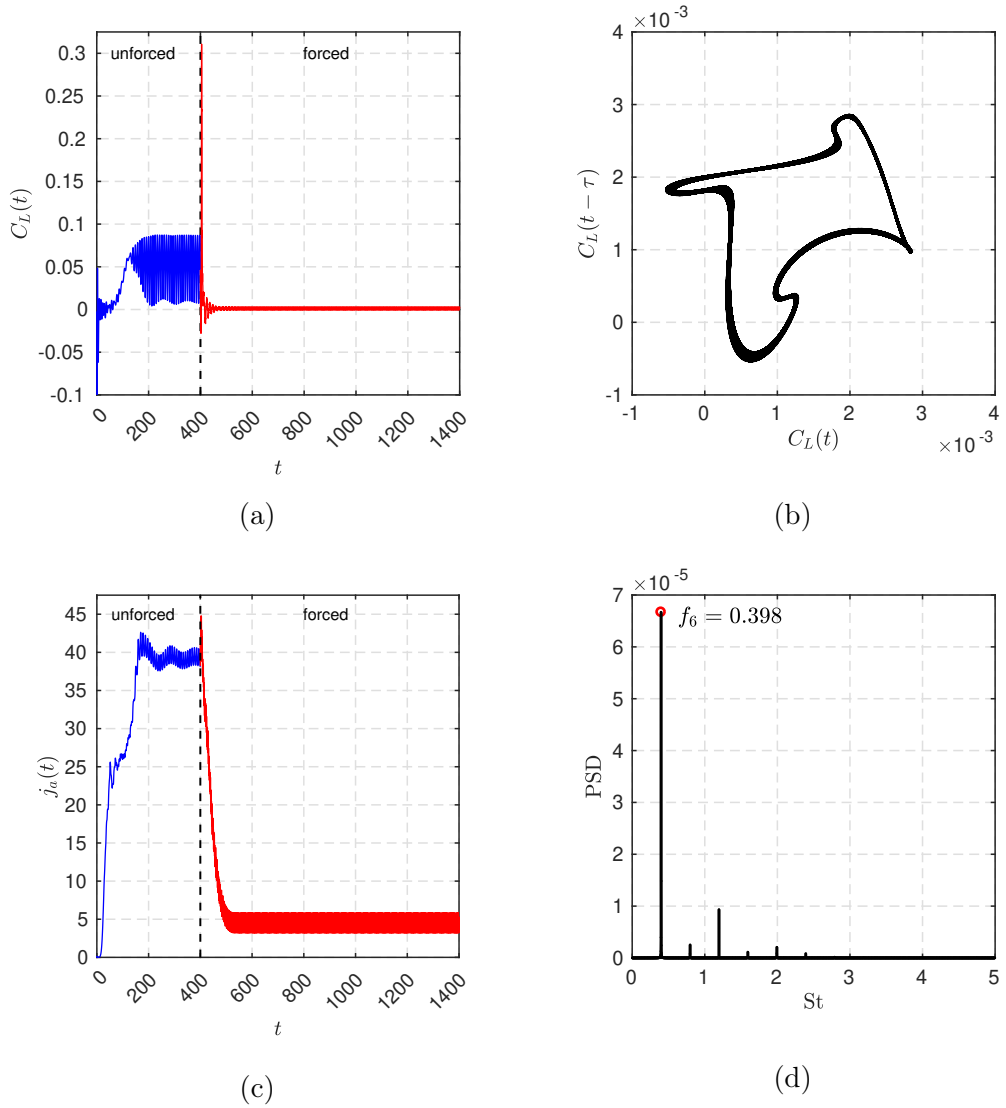
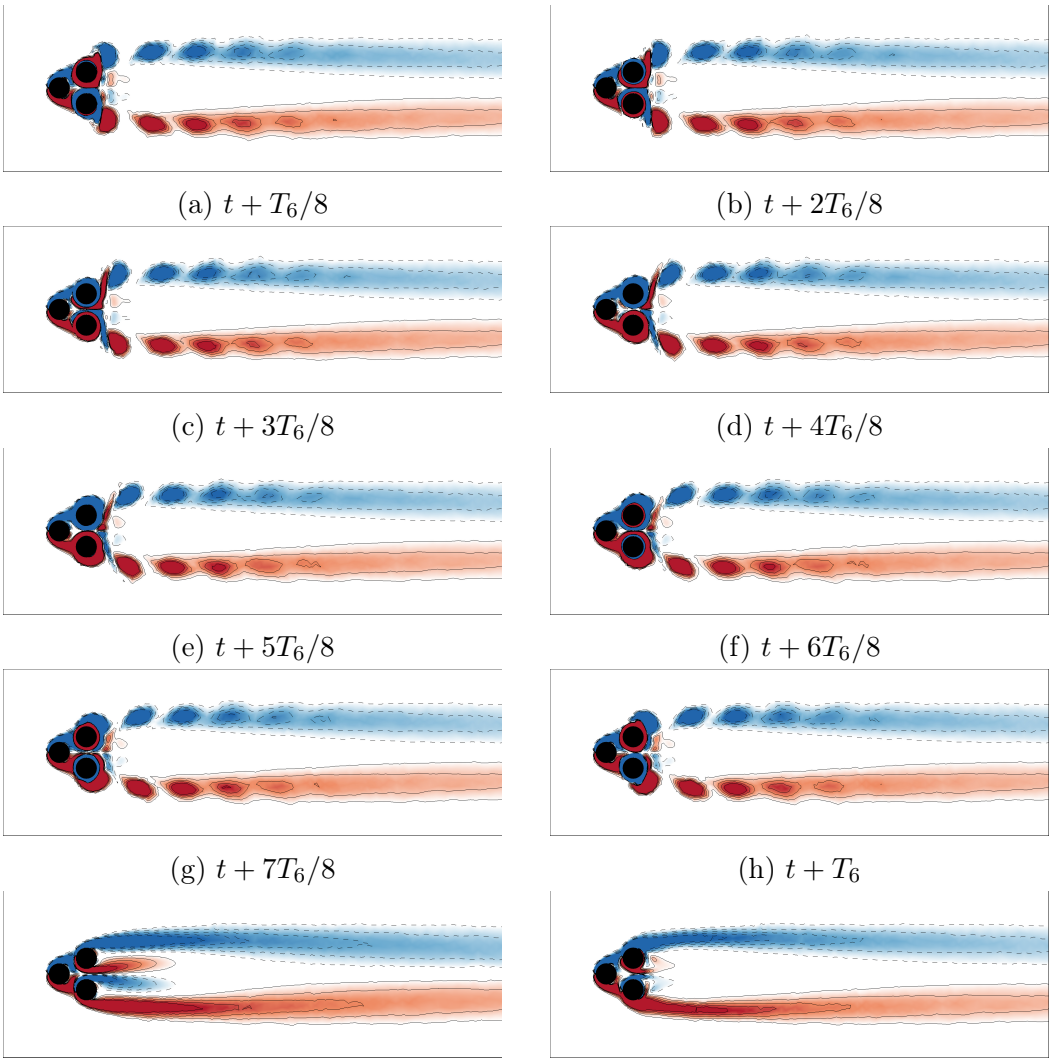


Figure 4.7: Characteristics of the best periodic forcing found with EGM. (a) Time evolution of the lift coefficient C_L, (b) phase portrait (c) time evolution of instantaneous cost function j_a and (d) Power Spectral Density (PSD) showing the main frequency $f_6 = 0.398$ of the forced flow and six harmonics. The control starts at $t = 400$. The unforced phase is depicted in blue and the forced one in red. The phase portrait and the PSD are computed over $t \in [900, 1400]$, during the post-transient regime.

tial control laws (black diamonds) for EGM are the centre of the box and shifted points from this centre. The shift is 10% of the box size in positive coordinate direction. Thus, the four initial control laws are: $[1, 0, 0]^\intercal$, $[1.2, 0, 0]^\intercal$, $[1, 0.4, 0]^\intercal$ and $[1, 0, 0.4]^\intercal$. The exploration phase is then performed in $\mathcal{B}$. For algorithmic reasons, the explorative points are chosen from 1 million points obtained from a space filling LHS. On one hand, we notice that the exploration phases (LHS in blue) focus on the boundary of the search space. This is consistent with the



(a) $t + T_6/8$ (b) $t + 2T_6/8$

(c) $t + 3T_6/8$ (d) $t + 4T_6/8$

(e) $t + 5T_6/8$ (f) $t + 6T_6/8$

(g) $t + 7T_6/8$ (h) $t + T_6$

(i) Symmetric steady solution (j) Mean field

Figure 4.8: Vorticity fields of the best periodic forcing found with EGM. (a)-(f) Time evolution of the vorticity field throughout the last period of the 1400 convective time units, (i) the objective symmetric steady solution and (j) the mean field of the forced flow. The color code is the same as figure 3.1. T_6 is the period associated to the frequency f_6. The mean field is computed by averaging 200 periods.

definition of LHS, as the furthest initial individuals are on the boundary of the box. On the other hand, the exploitation phases (simplex in yellow) stay in the same neighbourhood near the initial individuals, crawling along the local gradient to find the minimum.

Figure 4.10 shows the progression of the best control laws throughout the evaluations after 25 iterations of the exploration/exploitation process. The progression is plotted according to the number of cost function evaluations counted with the dummy index i. Figure 4.10a depicts the progression of the best control law after each downhill simplex step. We notice that a plateau is reached after 50 evalua-

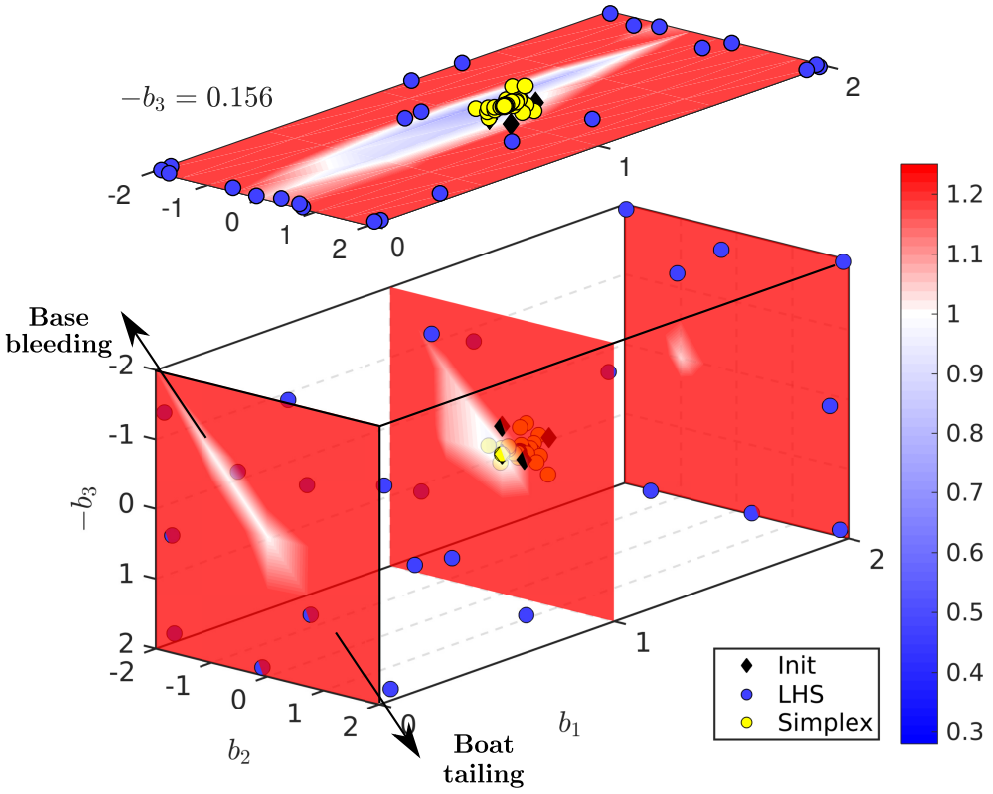


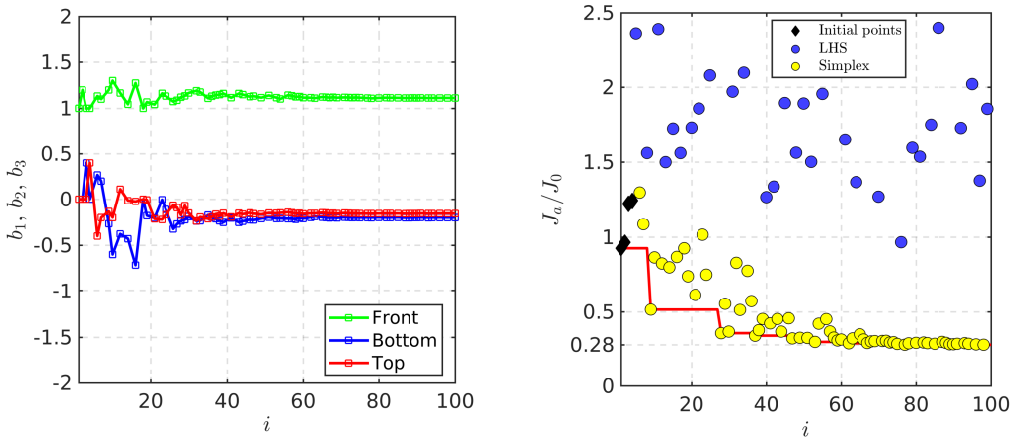
Figure 4.9: Contour map of J_a/J_0 at the optimal plane $b_3 = b^{\text{EGM}} = -0.156$ found with EGM (top) and at different levels of b_1: $b_1 = 0$, $b_1 = 1$, $b_1 = 2$ (bottom). The colormap denotes white for $J_a/J_0 = 1$, blue for better performances and red for worse performances. The planes are shown with 75% transparency. The four initial conditions $[1, 0, 0]^\intercal$, $[1.2, 0, 0]^\intercal$, $[1, 0.4, 0]^\intercal$ and $[0, 0, 0.4]^\intercal$ are represented by black diamonds. Blue dots are the control laws build with the exploration phases and yellow dots are the individuals build with the exploitation phases. All the individuals have been projected on the plane $b_3 = -0.156$. The arrows, on plane $b_1 = 0$, depict the base bleeding/boat tailing diagonal studied in section 4.2.2. A parametric study shows that the minimum is close to $[b_1, b_2, b_3]^\intercal = [1, 0, 0]^\intercal$ whose associated cost is J_a/J_0=0.93.

tions and there are only small variations afterwards. The final control law after 100 evaluations reads

$$\left[b_1^{\text{EGM}}, b_2^{\text{EGM}}, b_3^{\text{EGM}}\right]^\intercal = [1.11207, -0.20025, -0.15588]^\intercal \quad \text{with} \quad J_a = 10.85 \tag{4.4}$$

From visualizations of the control landscape of J_a in figure 4.9, we can safely infer that (4.4) describes the global minimum of our search space. Figure 4.10b shows convergence after 70 evaluations. Thereafter, the downhill simplex iterations show negligible improvements. In the whole EGM optimization, the exploration appears to be ineffective as the initial individuals are close to the minimum. An EGM run with different initial individuals ($[1, 0, 0]^\intercal$, $[1.5, 0, 0]^\intercal$, $[1, 1, 0]^\intercal$ and $[1, 0, 1]^\intercal$, corresponding to a 25% of the box size shift) have been tested. After a

few iterations, this new run started sliding down towards same minimum. This can be explained by the fact that the neighbourhood around the minimum is smooth enough for a downhill slide of the exploitation individuals.



(a) (b)

Figure 4.10: Evolution of b_1, b_2 and b_3 (left) for each new simplex step indicated by the scattered squares and J_a/J_0 (right) according to the number of evaluations i for the EGM optimization process. The red line on (b) shows the evolution of the best cost. The color code of the dots on (b) is the same as figure 4.9. The best control law is $[b_1^{\mathrm{EGM}}, b_2^{\mathrm{EGM}}, b_3^{\mathrm{EGM}}]^\intercal = [1.11207, -0.20025, -0.15588]^\intercal$ with $J_a/J_0 = 0.28$.

The control law (4.4) shows that the front cylinder rotates almost five times faster than the two other cylinders and in opposite directions. The asymmetry in the control law corresponds to the asymmetry in the lift coefficient in figure 4.11a, where the mean value is close to -0.7. The flow asymmetry can be visualized in the mean field (figure 4.12j). The vorticity in the vicinity of the cylinder is directly related to the actuation; thus the upward deflection near the front cylinder is explained by its fast rotation, around 1.1 times the incoming velocity. In addition, the tip of the positive vorticity lobe in the jet is slightly deflected downwards. Figure 4.12a-4.12h shows that EGM control (4.4) enables a jet fluctuation around vanishing mean, like the best base bleeding solution. Moreover, the phase portrait and the PSD in figure 4.11 reveal that the flow is purely harmonic. The main frequency $f_7 = 0.140$ is close to the main frequency $f_5 = 0.132$ of the base bleeding solution. Contrary to the best base bleeding solution, the instantaneous cost function j_a stays at low levels with a mean value around 9. The associated normalized cost is $J_a/J_0 = 0.28$. It is worth noting that, even though the control law $[b_1, b_2, b_3]^\intercal = [1, 0, 0]^\intercal$ is close to the best one found with EGM, its cost, $J_a/J_0 = 0.93$, is much higher. Moreover, the coarse description of the optimal plane $b_3 = b^{\mathrm{EGM}} = -0.15588$, in figure 4.9 (top), does not show any minimum a priori. This reveals large gradients in the control landscape, near the EGM solution, where a small change in the control amplitude can drastically change the

associated cost J_a/J_0.

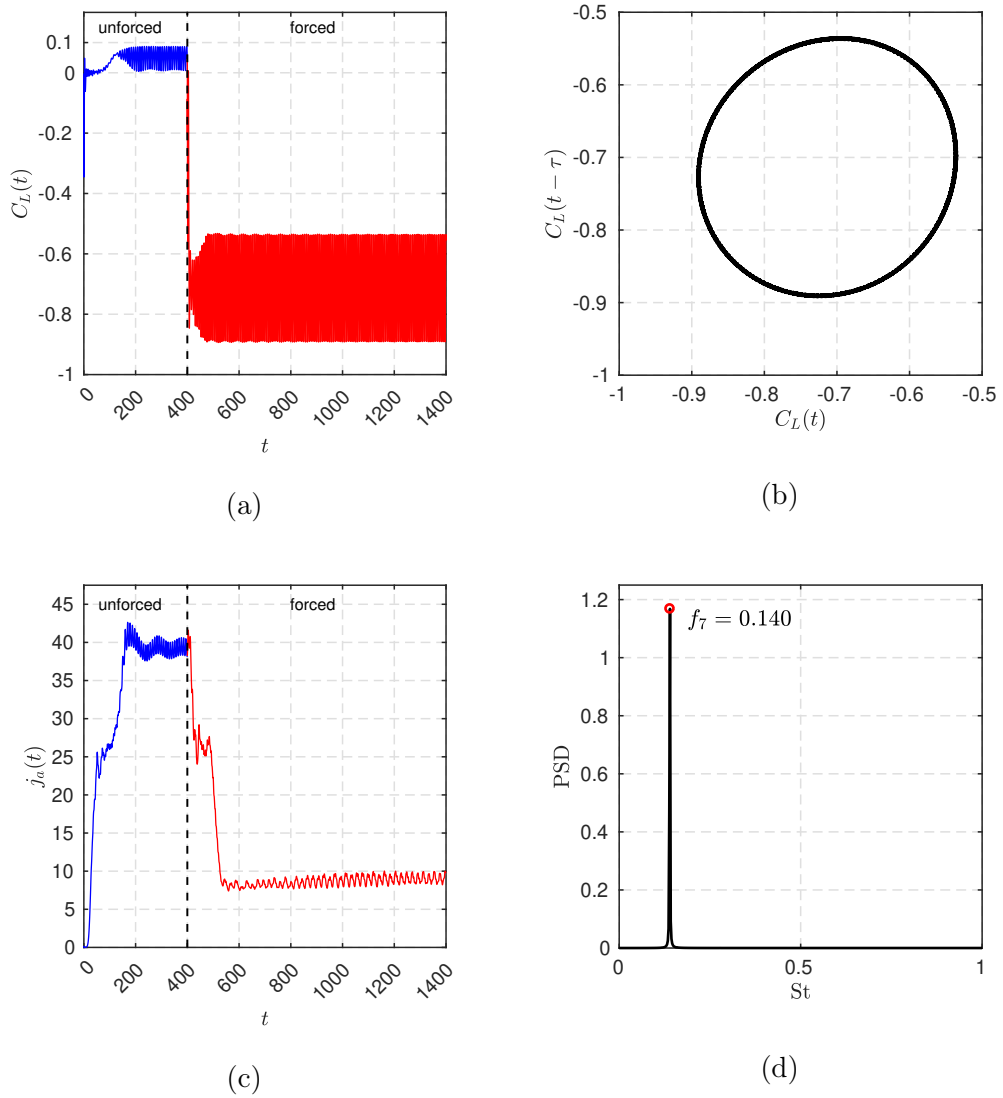
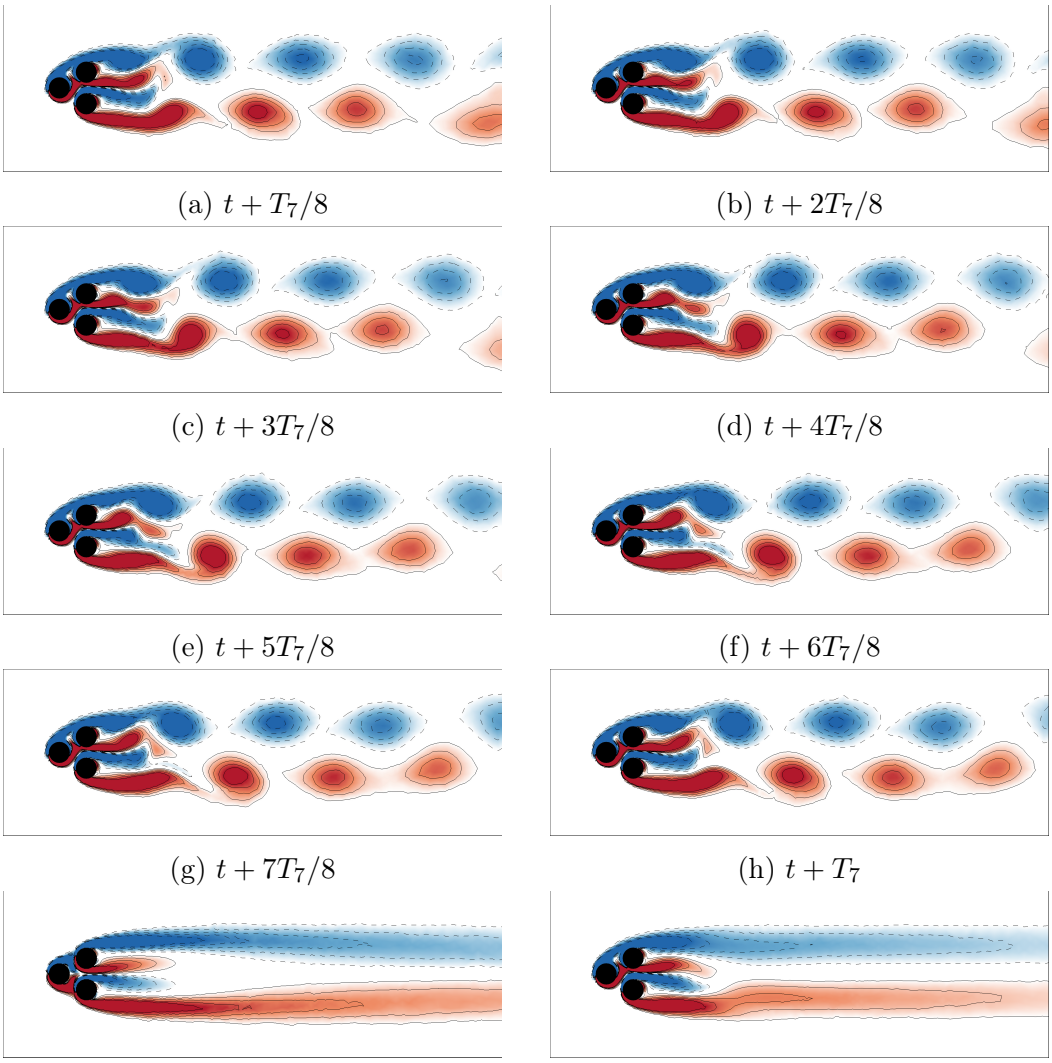


Figure 4.11: Characteristics of the best steady actuation found by EGM. (a) Time evolution of the lift coefficient C_L, (b) phase portrait (c) time evolution of instantaneous cost function j_a and (d) Power Spectral Density (PSD) showing the only frequency $f_7 = 0.140$ of the forced flow. The control starts at $t = 400$. The unforced phase is depicted in blue and the forced one in red. The phase portrait and the PSD are computed over $t \in [900, 1400]$ the post-transient regime.

In addition to the less deflected jet, we notice in figure 4.12 that the vortex shedding differs from the symmetric steady solution leading to a more symmetric flow. There are now two vortex streets of the shear layers, one on the upper side and one on the lower side of the flow. These shear layer dynamics hardly interact in the whole domain. Indeed, we notice that the distance between two consecutive vortices increases significantly only before leaving the computational domain which goes along with a slightly upward deflection of the wake. This



(a) $t + T_7/8$ (b) $t + 2T_7/8$

(c) $t + 3T_7/8$ (d) $t + 4T_7/8$

(e) $t + 5T_7/8$ (f) $t + 6T_7/8$

(g) $t + 7T_7/8$ (h) $t + T_7$

(i) Symmetric steady solution (j) Mean Field

Figure 4.12: Vorticity fields of the best steady actuation found with EGM. (a)-(f) Time evolution of the vorticity field throughout the last period of the 1400 convective time units, (i) the objective symmetric steady solution and (j) the mean field of the forced flow. The color code is the same as figure 3.1. T_7 is the period associated to the frequency f_7 of the forced flow. The mean field has computed by averaging over 100 periods.

results in extended vorticity branches in the mean field (figure 4.12j) but with a lower vorticity level compared to the symmetric steady solution.

In conclusion, we explore the stabilization of the fluidic pinball in different search space. First, we carried out a study on steady symmetric actuation. We show that, in the case of stabilization, it is the base bleeding that is favored. We then explored a two-dimensional and three-dimensional search spaces thanks to the explorative gradient method. As expected, exploring a richer search space improved the stabilization of the flow. Stabilization of the flow with periodic forcing is possible but at the cost of a high actuation power expense and a deformation

of the near jet. Surprisingly, an asymmetric forcing managed to bring partial symmetry to the flow and reduces the cost function even further compared to the best base bleeding solution. Experimentally, the optimization of the steady fluid pinball actuation also lead to asymmetric forcing Raibaudo *et al.* (2019). The explorative gradient method managed to converge to the global minimum in less than 50 evaluations. The exploration phases had a lesser impact during the optimization process as we initiated the algorithm close to the global minimum. We can expect the exploration phases to play a major role for more complex search space, comprising several minima. The exploration and exploitation principle illustrated with EGM inspired the gradient-enriched machine learning method for accelerated learning. This new method is the subject of the next chapter.

Chapter 5

Gradient-enriched machine learning control

In this chapter, we present the gradient-enriched machine learning control (gMLC) algorithm to solve challenging nonlinear non-convex function optimization problems. Gradient-enriched MLC combines exploration of the search landscape with evolutionary methods and fast convergence thanks to gradient-based methods. In section 5.1, we detail the exploration and exploitation capabilities of gMLC by describing the algorithm. Then, in section 5.2, we demonstrate gMLC by optimizing a feedback control law to stabilize the fluidic pinball. In section 5.3, we compare the performances of LGPC and gMLC in terms of convergence rate, final solution and repeatability. Finally, we end the chapter with a conclusion recalling all the algorithms employed to stabilize the fluidic pinball (section 5.4). sections 5.1, 5.2 and 5.3.1 are largely inspired by Cornejo Maceda *et al.* (2020).

Contents

5.1 Multiple-input multiple-output control optimization with gradient-enriched machine learning control (gMLC)

In this chapter, we cure a challenge of linear genetic programming control—the suboptimal exploitation of gradient information. Starting point is machine learning control (MLC) (Duriez *et al.*, 2016) based on linear genetic programming

(LGP). MLC or LGPC optimizes a control law without assuming a polynomial or other structure of the mapping from input to output. The only assumption is that the law can be expressed by a finite number of mathematical operations with a finite memory, i.e., is computable. The optimization process relies on a stochastic recombination of the control laws, also called *evolution*. LGPC has been amazingly efficient in outperforming existing optimal control laws—often with surprising frequency crosstalk mechanisms—in dozens of experiments (Noack, 2019). LGPC demonstrates a good exploration of actuation mechanisms but a slow convergence to an optimum, despite an increasing testing of redundant similar control laws.

The proposed gradient-enriched MLC departs in two aspects from LGPC. First, the concept of evolution from generation to generation is not adopted. The genetic operations include *all* tested individuals. One can argue that the neglection of previous generations might imply loss of important information. Second, the exploitation is accelerated by downhill subplex iteration (Rowan, 1990). The best $k+1$ individuals are chosen to define a k-dimensional subspace and a downhill simplex algorithm optimizes the control law in this subspace.

Both LGPC and gMLC share the same matrix representation of the control laws used in LGP (Brameier & Banzhaf, 2006) and described in chapter 2.

The algorithm begins with a Monte Carlo initialization of N_{MC} individuals, i.e., the indices of the matrix. The cost of these randomly generated functions are evaluated in the plant. The number of individuals N_{MC} needs to balance exploration and cost. Too few individuals may lead to descend in a suboptimal local minimum. Too many individuals may lead to unnecessary inefficient testing, as Monte Carlo sampling is purely explorative.

Once the initial individuals are evaluated, an exploration phase is carried out. New individuals are generated thanks to crossover and mutation operations. Thus, this phase is also referred as evolution phase. These operations are performed on the matrix representation of the individuals. As for MLC, crossover combines two individuals by exchanging lines in their matrix representation, whereas mutation randomly replaces values of some lines by new ones. In this approach, we no longer consider a population but the database of all the individuals evaluated so far. Thus, we no longer need the replication and elitism operators of MLC. This choice is justified by the fact that we want to learn as much as possible from what we already know and avoid reevaluating individuals. To perform the crossover and mutation operation, individuals are selected from the database thanks to a tournament selection. A tournament selection of size 7 for a population of 100 individuals is used in Duriez *et al.* (2016). That means that for a population of 100 individuals, 7 individuals are selected randomly and among the 7, the best one is chosen for the crossover or mutation operation. For gMLC, as the individuals are selected among all the evaluated individuals, the tournament size is properly scaled at each call to preserve the 7/100 ratio between the tournament size and the size of the database. The crossover and mutation operation are repeated randomly following P_c, the crossover probability, and P_m, the mutation probability, until N_G individuals are generated. The probabilities P_c and P_m are such as $P_m + P_c = 1$.

Once the evolution phase is achieved, N_G new individuals are generated thanks

to downhill subplex iterations. Being in an infinite dimension function space, Nelder-Mead's downhill simplex is impractical as an exploitation tool. Thus, we propose a variant of downhill simplex inspired by Rowan (1990), commonly called *downhill subplex*. Just as downhill simplex, the strength of this approach is to exploit local gradients to explore the search space. In the original approach of Rowan (1990), downhill simplex is applied to several orthogonal subspaces. However, in order to limit the number of cost function evaluations, we apply downhill simplex to only one subspace. This subspace is initialized by selecting N_{sub} individuals. Two ways to build the subspace after the Monte Carlo process are listed below:

- **Choose only the best individual**: select the best N_{sub} individuals evaluated so-far in the whole database.
- **Individuals near a minimum**: select the best individual evaluated so-far and the $N_{\text{sub}} - 1$ individuals closest to the best one.

The first approach has the benefit to comprise several minima candidates, whereas the second one is bound to lead to a minimum in the neighbourhood of the best individual and relies on a given metric. Once the subspace is built, the next steps are similar to the downhill simplex method. As subplex and simplex are essentially the same algorithm applied to different spaces, we will not designate them differently.

Following the situation, downhill subplex may call 1 (only reflection), 2 (expansion or single contraction) or $N_{\text{sub}}+1$ (shrink) times the cost function. Several iterations of downhill subplex are repeated until at least N_G individuals are generated. In this study, the same number of individuals generated with the evolution phase and the downhill subplex phase is chosen to balance exploration and exploitation.

Once the stopping criterion is reached, the most efficient individual in the database is given back. Otherwise, we restart a new cycle by generating new individuals with a new evolution phase, combining and modifying individuals derived by evolution and downhill subplex. However, the individuals built thanks to downhill subplex are linear combination of the original N_{sub} individuals. These new individuals do not have a matrix representation which is necessary to generate new individuals with genetic operators in the exploitation phase. To overcome this problem, we introduce a new phase to compute a matrix representation for the linearly combined control laws. The matrix representation is computed by solving a regression problem of the first kind, similar to a function fitting problem, for all the linearly combined control laws. First, each control law $\boldsymbol{K}_i$ is evaluated on randomly sampled inputs $\boldsymbol{s}_{\text{rand}}$, just as in section 2.4. The resulting output $\boldsymbol{K}_i(\boldsymbol{s}_{\text{rand}})$ is used to solve a secondary optimization problem:

$$\boldsymbol{K}_{\boldsymbol{M}}^{*} = \arg\min_{\boldsymbol{K}_{\boldsymbol{M}}} \|(\boldsymbol{K}_{\boldsymbol{M}}(\boldsymbol{s}_{\text{rand}}) - \boldsymbol{K}_i(\boldsymbol{s}_{\text{rand}}))\|^2 \tag{5.1}$$

where $\|\cdot\|$ denotes the Euclidean norm. This optimization problem is a function fitting problem that we solve with linear genetic programming. The LGP parameters are the same used for the gMLC so the computed individuals are compatible

with the ones in the database. The best fitting control law $\boldsymbol{K}_M^*$ has then a matrix representation and is used as a substitute for the original linear combination of control laws. The substitutes are then employed for the evolution phase even though they may not be perfect substitutes of the original control laws. Indeed, following the stopping criterion and population size of the secondary LGP optimization, the control law substitutes may not be able to reproduce all the characteristics of the linearly combined control laws. An accurate but costly representation may not be needed as the control laws will be recombined afterwards. Moreover, the introduction of some error may be beneficial to improve the exploration phase and enrich our database.

Once the matrix representations are computed, a new cycle may begin with a new evolution phase. In this phase, if any individual has a better performance than the N_{sub} individuals in the simplex, then the least performing individuals among the N_{sub} individuals are replaced. Thus, each evolution phase replaces elements in the simplex, allowing exploration beyond the initial subspace. Then, the optimization continues with the exploitation phase on the updated N_{sub} individuals.

Algorithm 2: Gradient-enriched Machine Learning Control

```
Result: K*, the best individual
Monte Carlo initialization: generate N_MC individuals;
Test all the individuals;
Build the subplex S by taking the N_sub best individuals;
while Stopping criterion is not reached do
    Exploration phase—Evolution
        Generate and test N_G individuals from all the individuals evaluated
         so far thanks to crossover and mutation;
        Update subplex S: choose the N_sub best indviduals among the new
         N_G individuals and the N_sub subplex individuals;
    end
    Exploitation phase—Downhill subplex
        while The number of subplex individuals generated < N_G do
            Perform a downhill subplex iteration in the subspace spanned by
             linear combinations of N_sub subplex control laws
             (Downhill simplex method like in algorithm 1);
        end
        Reconstruction phase—Linear genetic programming
        Compute a matrix representation for each new downhill subplex
         individual (replace linearly combined individuals by matrices using
         LGP);
    end
end
```

Figure 5.1 illustrates the initialization, exploration and exploitation of gMLC. The exploration is based on LGP. Also, the exploitation requires LGP. In the

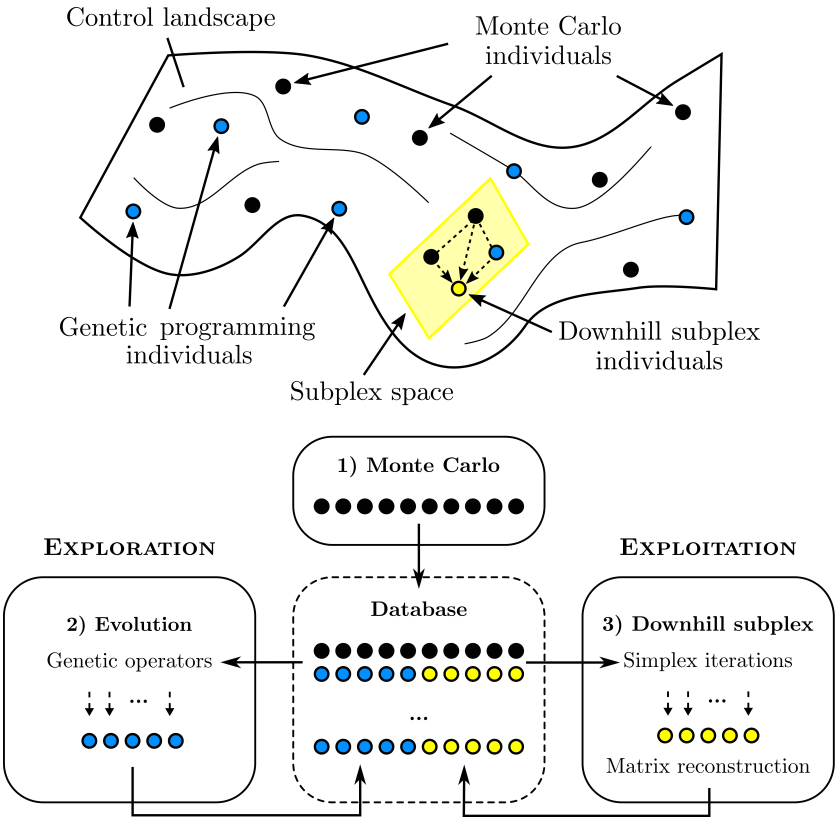


Figure 5.1: Schematic of the gradient-enriched MLC algorithm (bottom) and distribution of individuals in the search space (top) . First 1), Monte Carlo initialization performs a first coarse exploration of the search space. Second 2), further exploration is performed thanks to genetic programming. Individuals are selected in the whole dataset and combined thanks to genetic operators to generate new individuals (blue dots). Then the datase is augmented with the new individuals. Third 3), exploitation focuses on a subspace (represented in yellow) of finite dimension where downhill simplex iterations builds new individuals by linear combination (yellow dots). A matrix representation is computed for the downhill subplex individuals thanks to linear genetic programming, allowing the downhill subplex individuals to be included in the database.

downhill simplex method, the individuals are linear combinations of the subplex basis and are finally approximated as matrices. This process is repeated until the stopping criterion is reached. The *Gradient-enriched Machine Learning Control (gMLC)* is summarized by pseudo code in algorithm 2. And the source code of our gMLC implementation is freely available online under the name `gMLC` at `https://github.com/gycm134/gMLC`.

Finally, figure 5.2 summarizes the exploration and exploitation phases for EGM and gMLC and highlights their similarities.

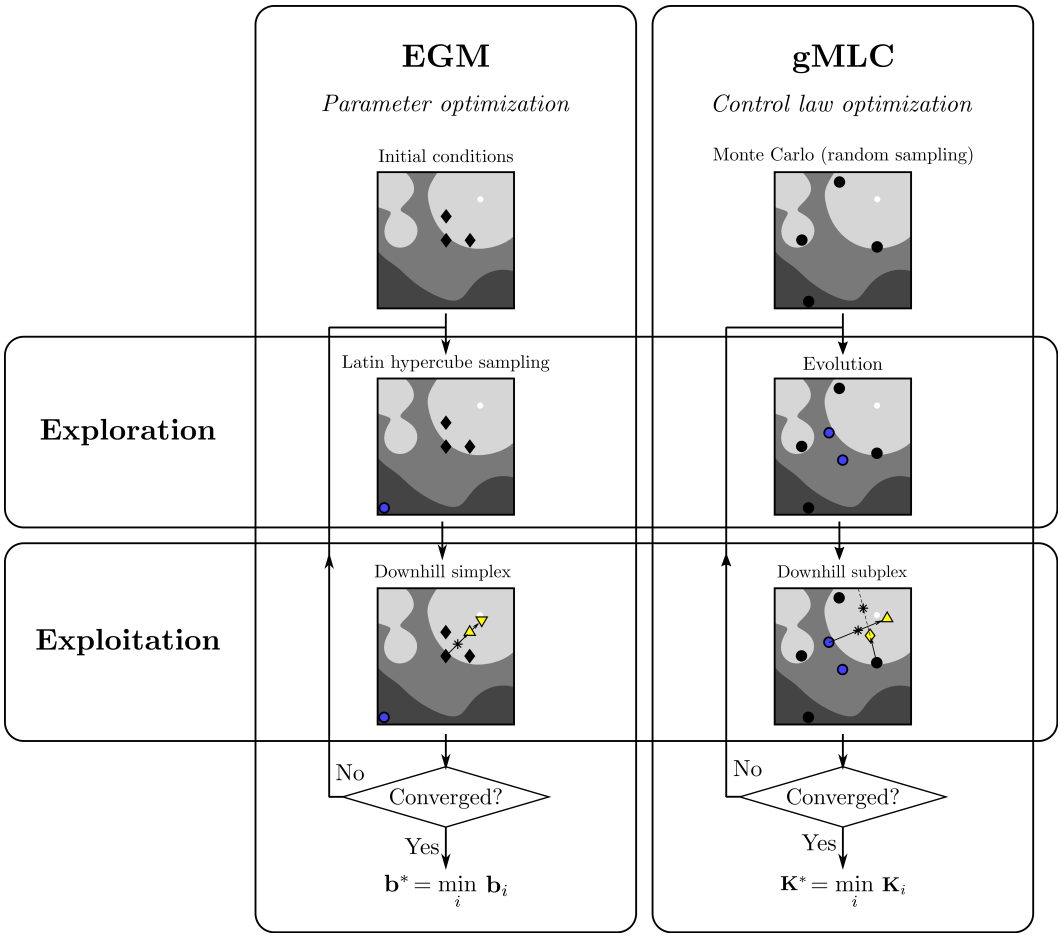


Figure 5.2: Summary of the explorative gradient method (EGM) (left column) and gradient-enriched machine learning control (gMLC) (right column). The level plots are a schematic representation of the control landscape. Darker regions depict poor performances and light regions depict good performances. Three minima are shown, two on the top left and one in the top right (the global one). The map represents an affine space (of finite dimension) for EGM and a Hilbert function space for gMLC. The initialization step is depicted with black diamonds for EGM and black dots for gMLC. The individuals generated thanks to an exploration phase are represented by blue dots. Exploration is carried out with LHS for EGM and evolution with genetic operators (crossover and mutation) for gMLC. The individuals generated thanks to an exploitation phase are represented in yellow. For EGM, downhill simplex steps are carried out. One iteration of downhill simplex is depicted: the reflected individual (yellow triangle) and the expanded individual (reversed yellow triangle); the star is the centroid of the two best black diamonds. For gMLC, the simplex steps are carried out in a subspace (downhill subplex) of finite dimension. Two distinct simplex steps are depicted: first, a reflection step (yellow triangle) with the two best black dots and the best blue dot; then a contraction step (yellow diamond) with the same black dots and the newly evaluated yellow triangle. The stars are the centroids for each step. This process is repeated until the stopping criterion is reached. In this figure, only one iteration of the loop is depicted. The reconstruction phase is not depicted for the sake of clarity.

5.2 Feedback control optimization

In this section, we optimize a feedback control law, again, to stabilize the unforced symmetric steady solution. The feedback is provided by 9 velocity signals in the wake as discussed in section 3.1.3.

We do not include time dependent functions as inputs of the control laws since we aim to stabilize the flow towards the steady solution so an open-loop strategy is not pursued.

As detailed in section 4.3, periodic forcing can stabilize the flow but at the expanse of high actuation power, thus we refrain ourselves to include them as control inputs. Thus, $N_\mathrm{b} = 3$, $N_\mathrm{s} = 36$ and $N_\mathrm{h} = 0$. The control laws are then built from 9 basic operations ($+$, $-$, $\times$, $\div$, cos, sin, tanh, exp and log), 36 sensors signals $\boldsymbol{s}_{i=1..36}$ and 10 constants. The control laws are restricted to the range $[-5, 5]$ to avoid excessive actuation. The basic operations $\div$ and log are protected in order to be defined on $\mathbb{R}$ in its entirety. The cost function has been computed over 1000 convective time units, so that the post-transient regime is fully established and the transient phase has a lesser weight.

For the implementation of the gMLC algorithm on the fluidic pinball, we start with a Monte Carlo step of $N_\mathrm{MC} = 100$ individuals, the crossover probability and mutation probability are both set at $P_c = P_m = 0.5$. Indeed, as the evolution phase is mostly an explorative phase, the mutation probability is increased, from 0.3, in previous studies, to 0.5, to improve the exploration capability. Moreover, even though crossover is an exploitative operator, it is likely to find new minima thanks to recombinations of radically different control laws. That is why, the crossover and mutation probabilities are both set to 0.5. The dimension of the subspace is set to $N_\mathrm{sub} = 10$, so it is large enough to explore a rich subspace but not too large to avoid a slowdown in the optimization process. Evidently with a higher dimension subspace the control law can be more finely tuned. To assure that the subplex step effectively goes down the local minimum, we choose to evaluate $N_\mathrm{p} = 50$ individuals during the exploitation phase. Test runs with $N_\mathrm{p} = 5$ have been carried out and showed that the learning process was slower. We believe one reason is that each exploration phase changes systematically the subspace, which makes it difficult for the subplex to improve effectively in only a few steps, thus, subplex has almost no benefit in the early phases. Table 5.1 summarizes all the parameters for gMLC. The secondary optimization problem (equation 5.1) used to build a matrix representation for the control laws, is solved with LGP. To speed up the computation, we choose to solve the secondary optimization problem with 100 individuals evolving through 10 generations. Finally, our implementation is enhanced by a screening of the individuals to avoid reevaluating individuals that have different mathematical expressions but are numerically equivalent, just as LGPC and described in section 2.4. This screening is used only in steps where the individuals are generated stochastically, meaning in the Monte Carlo step and in the exploration phases. This improvement is also used in LGP to solve the secondary optimization problem.

Figure 5.3 presents the learning process of gMLC for the stabilization of the fluidic pinball. We notice that the first exploration phase, individuals $i = 101, \ldots, 150$,

parameter	description	value
N_b	number of actuators	3
N_s	number of control inputs	9 sensors $\times$ 4 delays $=$ 36 sensor signals
N_h	number of time-dependent functions	0
N_vr	number of variable registers	$36 + 3 = 39$
N_cr	number of constant registers	10
$N_\mathrm{inst,max}$	max number of instructions	50
N_MC	Monte Carlo individuals	100
N_sub	subspace size	10
P_c	crossover probability	0.5
P_m	mutation probability	0.5
N_p	number of individuals per phase	50
	function library	$+$, $-$, $\times$, $\div$, cos, sin, tanh, exp, log

Table 5.1: gMLC parameters for the fluidic pinball.

already improved the best cost compared to the Monte Carlo phase. The following exploitation, individuals $i = 151, \ldots, 200$, present a steep descent, improving the best solution even further. During this phase, we notice a clear trend for the cost of the new individuals. This trend indicates that the simplex is going down towards a minimum. But this descent is interrupted by the next exploration phase. Individuals $i = 201, \ldots, 250$, greatly improve the best solution. Particularly, two individuals have a much lower cost that the ones in the simplex, suggesting that a new minima have been found. The next exploitation phase with individuals $i = 251, \ldots, 300$ brings no improvement. The high cost in the exploitation steps following the exploration phases is explained by the fact that as we are exploring new minima, shrink steps must be performed to bring the simplex towards the new minima; and the shrink steps replaces all individuals in the simplex except the best one. As we are leaving one minimum for another one, the intermediate values can be arbitrarily high until the simplex reached the neighbourhood of the new minimum. The next exploration phase with individuals $i = 301, \ldots, 350$ also give good individuals that have been included in the simplex. After 350 evaluations, the only improvements are performed by exploitation phases. Even if the best cost keeps decreasing slowly, the improvements are small, indicating that we are close to the minimum. Once we reach a plateau, further improvement can only be performed if an exploration phase finds an individual close to a better minimum. That is why after 800 individuals we performed only exploration phases.

The final control law build with gMLC reads

$$\begin{aligned} b_1^{\text{gMLC}} &= = -0.0004\sin(\cos(s_{30})) - 0.0034(s_6 + s_{22}) - 0.0033(\log(s_{11})) - 0.0305(s_3) \\ &\quad -0.0098(s_{16} + s_{15}) + 0.0055 s_{35}(s_{16} + 0.31016) - 0.0091(s_3 - s_{23}) \\ &\quad +0.9206\tanh(s_{16}) - 0.1238\cos(s_{31}) + 0.1907, \\ b_2^{\text{gMLC}} &= -0.0459(\log(\log(s_{31}))) - 0.1946, \\ b_3^{\text{gMLC}} &= -0.0004(0.841471 s_{34} - s_{36}) - 0.0043\log(s_9) - 0.0022(s_{25} - s_{16}) \\ &\quad -0.0098(\cos(s_3) - s_{16}) + 0.9206\log(\tanh(\exp(s_2))) - 0.0295 \\ J_a &= 7.82. \end{aligned} \tag{5.2}$$

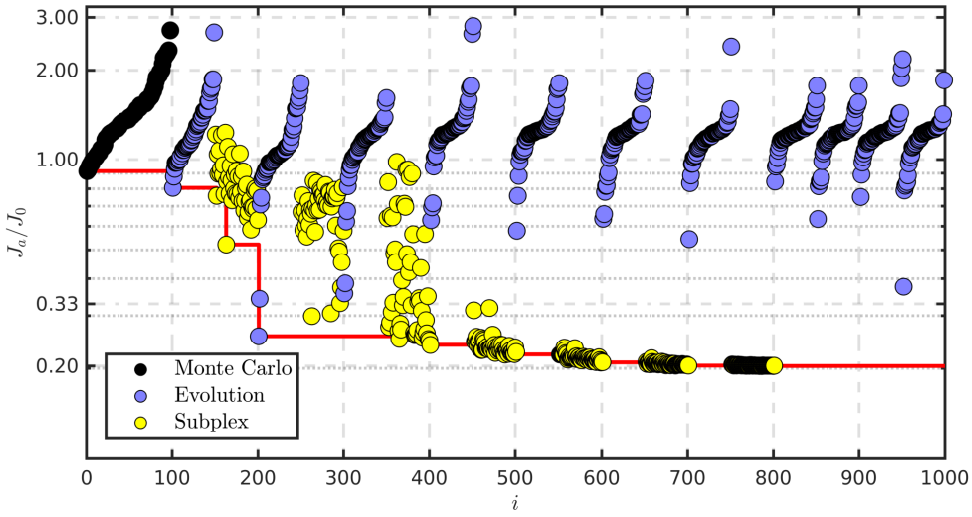


Figure 5.3: Distribution of the costs during the gMLC optimization process. Each dot represents the cost J_a/J_0 of one individual. The color of the dots represents how the individuals have been generated. Black dots denote the individuals which are randomly generated (Monte Carlo). Blue dots refer to individuals which are generated from a genetic operator (exploration). And yellow dots correspond to individuals arising from the subplex method (exploitation). The individuals from the Monte Carlo step and the exploration phase have been sorted following their costs. The red line shows the evolution of the best cost. The vertical axis is in log scale.

Figure 5.4 presents the characteristics of the flow controlled by the best control law $\boldsymbol{K}^{\text{gMLC}}$ built with gMLC. This control law is detailed later specially in table 5.2. In figure 5.4a, we can see that even if the resulting lift coefficient is still asymmetric, the mean value (around -0.1) is closer to 0 as compared to the EGM solution. The PSD in figure 5.4d shows a dominant frequency at $f_8 = 0.144$ and one of its higher harmonics. A small peak can be seen for $f_9 \approx 0.016$. The nonlinear interaction between the frequencies $f_8 = 0.144$ and $f_9 = 0.016$ gives rise to another small peak at $f_{10} = 0.160$. The phase portrait in figure 5.4b reveals drifts in pronounced oscillations due to the low frequency modulation. The presence of the dominant frequency $f_8 = 0.144$ and its harmonic in the spectrum is consistent

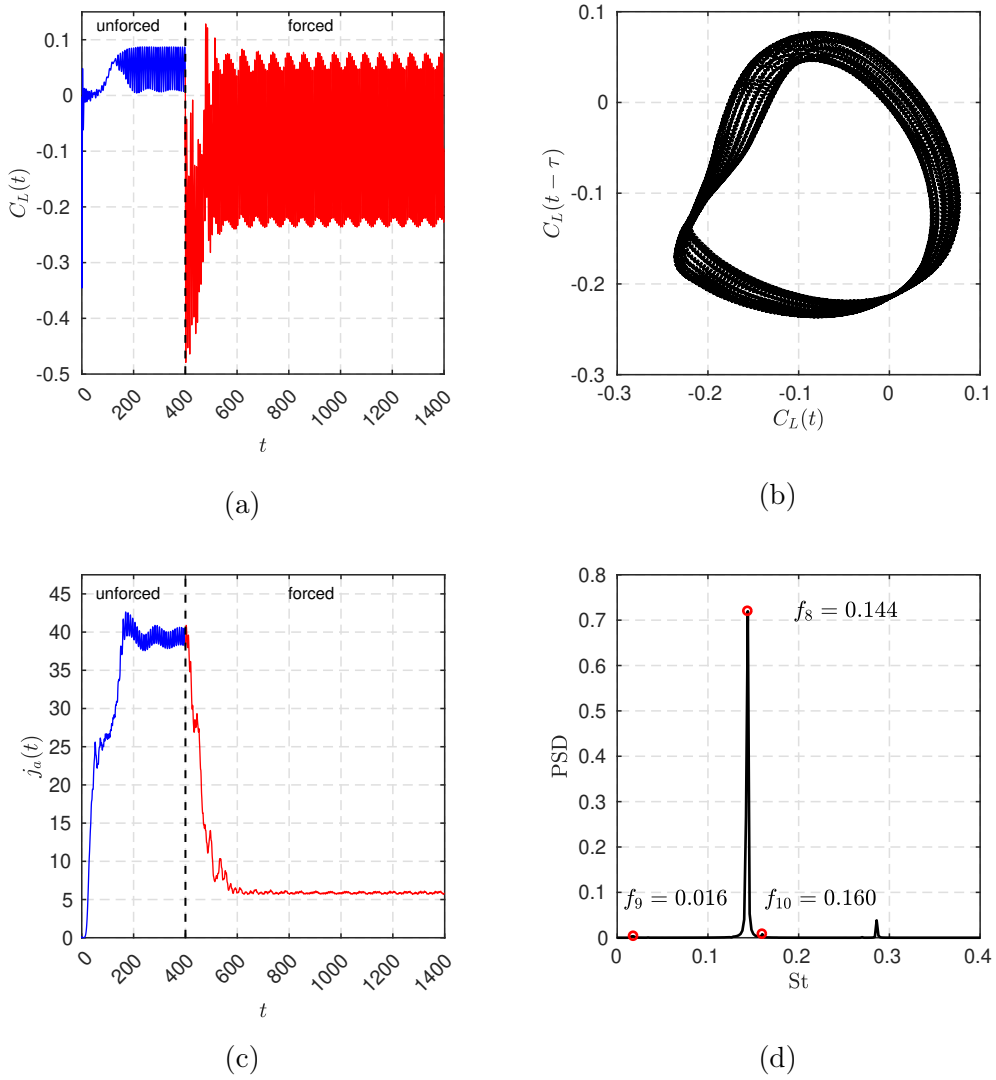


Figure 5.4: Characteristics of the flow controlled by the best feedback control law found with gMLC. (a) Time evolution of the lift coefficient C_L, (b) phase portrait, (c) time evolution of instantaneous cost function j_a and (d) Power Spectral Density (PSD) showing the frequency $f_8 = 0.144$ of the forced flow, one of its harmonics and two low-power frequencies $f_9 = 0.016$ and $f_{10} = 0.160$. The control starts at $t = 400$. The unforced phase is depicted in blue and the forced one in red. The phase portrait and the PSD are computed over $t \in [900, 1400]$, during the post-transient regime.

with the periodic behaviour of the flow. The $f_9 = 0.016$ peak is responsible for the width of a predominant limit-cycle dynamics in the phase portrait.

The evolution of the instantaneous cost function j_a in figure 5.4c shows a plateau after 200 convective time units, reaching an even lower level (around 6), compared to the EGM solution (around 9). The associated cost $J_a/J_0 = 0.20$ is better than the EGM solution at $J_a/J_0 = 0.28$.

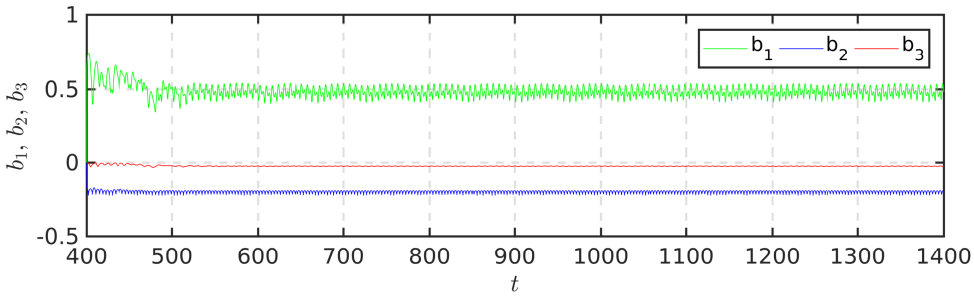


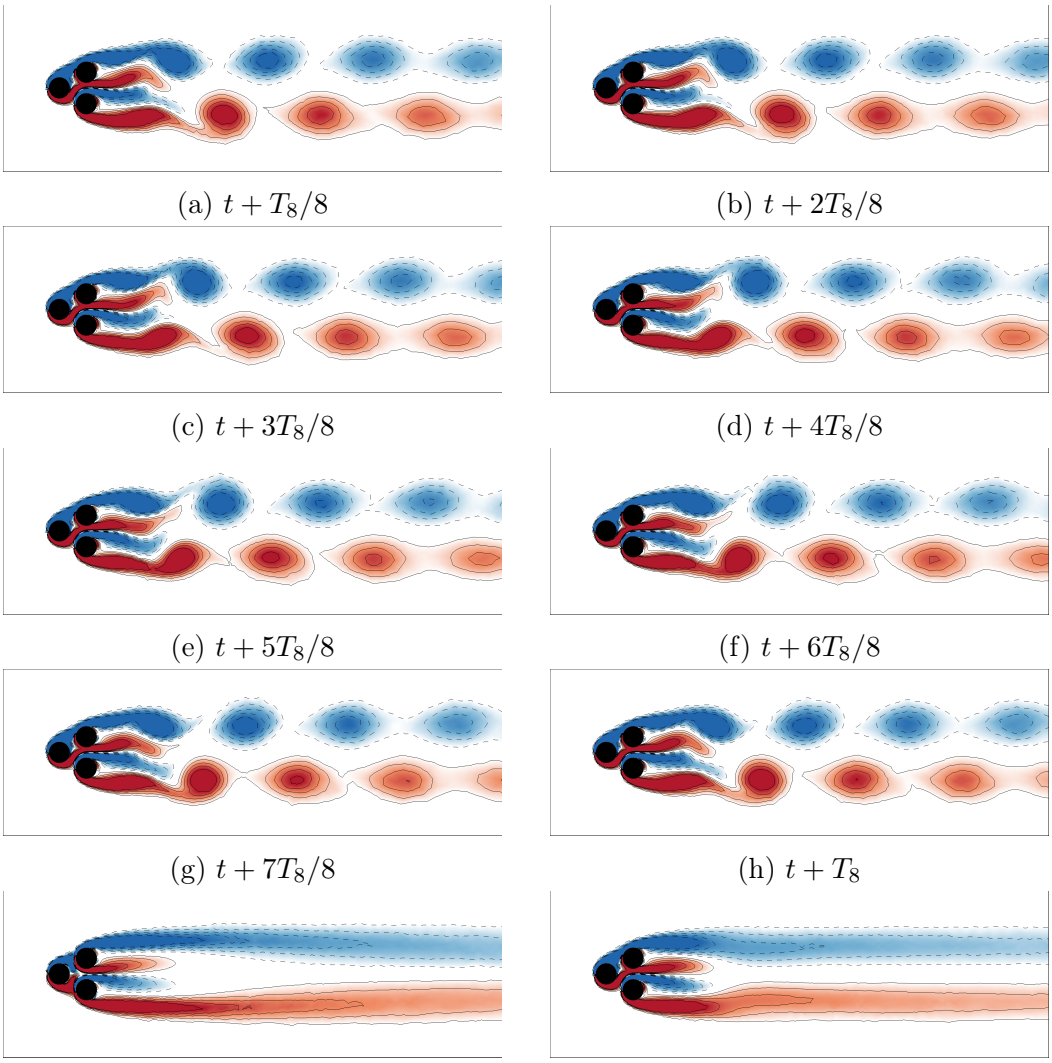
Figure 5.5: Time series of the actuation command for the best feedback control law found with gMLC.

#	b_1	b_2	b_3	weight	J_a/J_0
1	$\sin(\cos(s_{30}))$	0	$0.841471 s_{34} - s_{36}$	-4×10^{-4}	0.91
2	$s_6 + s_{22}$	0	0	-3.4×10^{-3}	0.94
3	0	0	$\log(s_9)$	-4.3×10^{-3}	0.97
4	0	0	$s_{25} - s_{16}$	-2.2×10^{-3}	0.97
5	$\log(s_{11})$	0	0	-3.3×10^{-3}	0.95
6	s_3	0	0	-3.05×10^{-2}	0.92
7	$s_{16} + s_{15}$	0	$\cos(s_3) - s_{16}$	-9.8×10^{-3}	0.97
8	$s_{35}(s_{16} + 0.31016)$	0	0	5.5×10^{-3}	0.80
9	$s_3 - s_{23}$	0	0	-9.1×10^{-3}	0.88
10	1	$\log(\log(s_{31}))$	0	-4.59×10^{-2}	0.93
11	$\mathbf{\tanh(s_{16})}$	$\mathbf{-0.187071}$	$\mathbf{\log(\tanh(\exp(s_2)))}$	$\mathbf{9.206 \times 10^{-1}}$	**0.26**
12	0.540302	-0.144304	-0.0144074	6.87×10^{-2}	0.34
13	$\mathbf{\cos(s_{31})}$	$\mathbf{-0.144304}$	$\mathbf{-0.0144074}$	$\mathbf{-1.238 \times 10^{-1}}$	**0.36**
14	**0.949948**	$\mathbf{-0.144304}$	$\mathbf{-0.0144074}$	$\mathbf{2.100 \times 10^{-1}}$	**0.39**

Table 5.2: Summary of the 14 control laws composing $\boldsymbol{K}^{\mathrm{gMLC}}$ described in equation (5.2). For each control law, we present b_1, b_2, b_3, the associated weight and the reduced cost J_a/J_0. The three best control laws are #11, #13 and #14.

cylinder	mean value	main frequency	peak-to-peak amplitude
front (b_1, green)	0.48	$2f_8$	0.12
bottom (b_2, blue)	-0.19	$2f_8$	0.03
top (b_3, red)	-0.02	f_8	< 0.01

Table 5.3: Summary of control law information. The frequencies and peak-to-peak amplitude have been computed on the post-transient regime.



(a) $t + T_8/8$ (b) $t + 2T_8/8$

(c) $t + 3T_8/8$ (d) $t + 4T_8/8$

(e) $t + 5T_8/8$ (f) $t + 6T_8/8$

(g) $t + 7T_8/8$ (h) $t + T_8$

(i) Symmetric steady solution (j) Mean field

Figure 5.6: Vorticity fields of the best feedback control found with gMLC. (a)-(f) Time evolution of the vorticity field throughout the last period of the 1400 convective time units, (i) the objective symmetric steady solution and (j) the mean field of the forced flow. The color code is the same as figure 3.1. T_8 is the period associated to the frequency f_8. The mean field has been computed by averaging 100 periods.

Table 5.2, 5.3 and figure 5.5 give more details on the control law $\boldsymbol{K}^{\mathrm{gMLC}}$ built with gMLC. Firstly, we can see that even though the simplex comprises $N_{\mathrm{sub}} = 10$ individuals, subplex build the control law $\boldsymbol{K}^{\mathrm{gMLC}}$ by linearly combining 14 control laws. Indeed, after a few iterations of simplex, all the individuals are eventually a linear combination of the initial individuals forming simplex. When a new individual is introduced in the basis thanks to the exploration phase, the exploitation phase will combine the remaining individuals with the new one, making the next individual a linear combination of more than 10 individuals. It is important to note that even after the introduction of new individuals with the exploration

phase, the subspace to explore changes but the dimension remains. In this case, with $N_{\text{sub}} = 10$, the dimension of the subspace is 9. The repetition of this process builds each time more complex control laws. Thus, in table 5.2, individuals $i = 11, 12, 13, 14$ have been introduced thanks to exploration phases. The control laws with the strongest weights are $i = 11, 13$ and 14, whereas the weight associated with the other control laws are at least one order of magnitude lower. Control law $i = 11$ is also the one with the lowest cost $J_a/J_0 = 0.26$. $\boldsymbol{K}^{\text{gMLC}}$ is then mainly based on $i = 11$ and corrected by the remaining control laws. This indicates that on the last phase of the learning, it is the minimum in the neighbourhood of $i = 11$ that has been found.

Moreover, table 5.2 shows that all three control components b_1^{gMLC}, b_2^{gMLC} and b_3^{gMLC} of the gMLC control law include sensor information. However, figure 5.5 shows that the actuation command associated with $\boldsymbol{K}^{\text{gMLC}}$ for the two rearward cylinders (b_2 and b_3) are nearly constant. This is partially due to the low weights associated to the control laws with sensor signals. We can also assume that the sensor signals cancel each other, leading to such low peak-to-peak amplitudes. Table 5.3 shows the characteristics of the actuation command during the post-transient regime. A spectral analysis shows that the main frequency of the actuation command for the front and bottom cylinder are twice the main frequency of the flow f_8, revealing that the actuation is a function of the flow. Thus, gMLC managed to build a combination between asymmetric steady forcing and feedback control. Finally, like EGM, the best solution found is asymmetric but with lower amplitudes. Consequently, the associated actuation power is lower compared to general steady actuation found with EGM: $J_b = 0.2018$ for the general steady actuation and $J_b = 0.0391$ for the feedback control law found with gMLC.

The controlled flow is depicted over one period in figure 5.6a-5.6h. First, we notice that the jet fluctuates around a vanishing mean, as for the EGM actuation. Also, the vortex shedding of the upper and lower shear layers hardly interact. Compared to the EGM solution, the stability of the wake is improved as the two Kelvin-Helmholtz vortices keep their transverse distance to the symmetry line until the very end of the computational domain. This is explained by the re-energization of the shear layers thanks to the vigorous rotation of the front cylinder at twice the main frequency f_8 of the controlled flow, like Protas (2004). The mean field, in figure 5.6j, is similar to the symmetric steady solution. Indeed, we notice that the vorticity regions extend to the end of the computation domain, like the symmetric steady solution. Also, like for the best general steady actuation, the region near the cylinders is non-symmetrical due to the action. However, contrary to the symmetric steady solution, the mean field of the feedback control has a narrower region between the vorticity regions upstream and a wider region downstream.

As expected, gMLC manages to find a new solution that surpasses the best general steady actuation found with EGM. Surprisingly, gMLC built a non-trivial solution, combining asymmetric steady forcing and feedback control for the front cylinder. Interestingly, gMLC composed a control law that forces the flow at twice the main frequency. In addition, compared to the best general steady actuation, the actuation power is significantly reduced. Lastly, the learning process of gMLC

exploited both the evolution phases and the simplex steps to rapidly build better solutions. Thanks to the evolution phases, new minima have been successfully found and thanks to the simplex steps, the solutions have been improved even more. The progress of the subplex steps show that local gradient information can be exploited in a subspace of an infinite dimension space to minimize a cost function. Building on this success, we believe that gradient-enriched MLC will greatly accelerate the optimization of control laws for MIMO control as compared to the linear genetic programming control.

5.3 Comparison between LGPC and gMLC

5.3.1 Performance comparison on the fluidic pinball

In this section, we compare the performances of a linear genetic programming control (LGPC), based on linear genetic programming control (Li *et al.*, 2018; Zhou *et al.*, 2020) and presented in chapter 2 and the proposed gradient-enriched MLC (gMLC) variant for the stabilization of the fluidic pinball. During the evolution process, the better-performing individuals are selected with larger probability to build new individuals thanks to the genetic operators. The best individuals are selected thanks to a tournament selection method. As in Duriez *et al.* (2016), we choose a tournament selection of size 7 for 100 individuals. A genetic operation is chosen randomly following given probabilities: the crossover probability P_c, the mutation probability P_m and the replication probability P_r. We recall that the probabilities add up to unity $P_c + P_m + P_r = 1$. The set of parameters $[P_c, P_m, P_r]^\intercal = [0.6, 0.3, 0.1]^\intercal$ have been chosen for LGPC, see chapter 3.

The meta-parameters of the linear programming control laws are the same for LGPC and gMLC, including the mathematical operations, number of constants, number of registers, as well as inputs and outputs (see table 5.1).

The cost function is evaluated over 1000 convective time units, both in LGPC and gMLC.

The LGPC and gMLC algorithms are compared over 1000 evaluations. For LGPC, a population of 100 individuals is chosen to evolve over 10 generations. For a fair comparison, LGPC and gMLC share the same initial Monte Carlo generation, comprising the first 100 randomly generated individuals. Figure 5.7 shows the distribution of the costs J_a/J_0 as a function of the evaluations. We notice that for both algorithms the first exploration phase makes great improvement. In the second generation, the best cost is 0.80 for gMLC and 0.70 for LGPC. LGPC's better performance is understandable as 100 individuals have been evaluated for the second generation whereas only 50 individuals have been evaluated for gMLC. After testing 200 individuals, gMLC surpasses LGPC thanks to the subplex steps, reaching a cost $J_a/J_0 = 0.52$. For the second evolution phase, both LGPC and gMLC perform well reaching low levels of J_a/J_0: 0.36 for LGPC and 0.26 for gMLC. Then, LGPC achieves only small progress after 900 evaluations, the cost improves from 0.36 to 0.33. The series of blue dots at $J_a/J_0 = 0.36$ from $i = 201$ to $i = 900$ represents several instances of the best individual of generation 3,

duplicated thanks to elitism. For gMLC, figure 5.3 shows that evolution phases do not bring any progress after 250 evaluations and further improvement is made thanks to the subplex steps. As described in section 5.2, the evolution phases help to enrich the simplex subspace. The subplex steps manage to reduce the cost function from 0.26 to 0.20. We notice that after 600 evaluations all new subplex individuals have the same cost. Hence, gMLC surpasses LGPC with a smaller number of evaluations and enables improvement/fine-tuning of the control laws in the final phase.

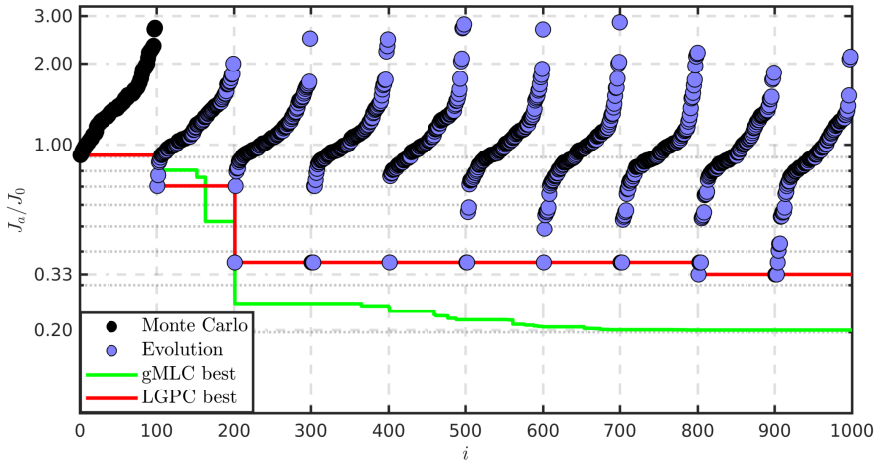


Figure 5.7: Distribution of the costs during the LGPC optimization process and evolution of the best individual for the gMLC learning process (green line). Each dot represents the cost J_a/J_0 of one individual. The color of the dots represent how the individuals have been generated. Black dots for the individuals randomly generated by a Monte Carlo process (individuals $i = 1, \ldots, 100$), blue dots for the individuals generated from a genetic operator (individuals $i = 101, \ldots, 1000$). For each generation the individuals have been sorted according to their cost. The red line shows the evolution of the best cost for the LGPC optimization process. The green curve corresponds to the gMLC optimization process. The vertical axis is in log scale.

5.3.2 Repetability study on the generealized mean-field model

In this section, we discuss the repeatability and convergence speed of the gMLC algorithm. To assess gMLC's performances, we compare the method with Monte Carlo optimization and LGPC. Running several instances of gMLC on the fluidic pinball for comparison can be expensive, so we decide to study a simpler dynamical system that comprises key nonlinear interactions. The dynamical system considered for control is the generalized mean-field model (GMFM). The GMFM is a system of four equations describing two oscillators nonlinear coupled. The first oscillator is unstable with a nonlinear damping of the growth rate, stabilizing at an limit cycle R_c. The second oscillator is stable with a frequency ten times higher

than the first oscillator. The objective is to stabilize the first oscillator by actuating on one of the components of the second oscillator. This dynamical system models the interaction between the natural vortex shedding and the shear layer oscillation for the cylinder flow (Luchtenburg *et al.*, 2009; Duriez *et al.*, 2016). The equations of the GMFM are the following:

$$\begin{bmatrix} \dot{a_1} \\ \dot{a_2} \\ \dot{a_3} \\ \dot{a_4} \end{bmatrix} = \begin{bmatrix} \sigma a_1 - a_2 \\ \sigma a_2 + a_1 \\ -0.1a_3 - 10a_4 \\ -0.1a_4 + 10a_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ K(a_1, a_2, a_3, a_4) \end{bmatrix} \tag{5.3}$$

with

$$\sigma = 0.1 - {a_1}^2 - {a_2}^2 - {a_3}^2 - {a_4}^2 \tag{5.4}$$

A full-state control is performed by adding a forcing term b on the fourth equation of the system. The cost function J comprises the mean distance to the fixed point J_a and a penalization term for the actuation J_b:

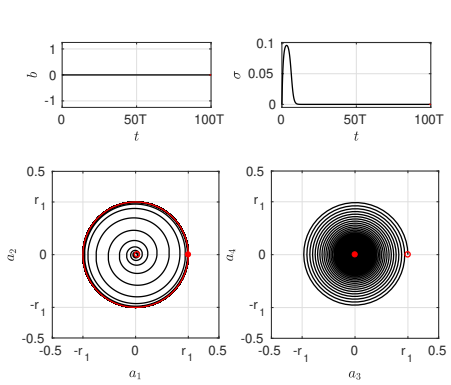
$$\begin{aligned} J =& J_a + \gamma J_b \\ J_a =& \frac{1}{T_{max}} \int_0^{T_{max}} a_1^2(t) + a_2^2(t)\, \mathrm{d}t \\ J_b =& \frac{1}{T_{max}} \int_0^{T_{max}} b^2(t)\, \mathrm{d}t \end{aligned} \tag{5.5}$$

with $b(t) = K(a_1(t), a_2(t), a_3(t), a_4(t))$. The penalization parameter γ is taken equal to 0.01 as in Duriez *et al.* (2016). The cost function is computed for $T_{max} = 100T_1$ where $T_1 = 2\pi$ is the period of the first oscillator. We note $R_c = \sqrt{0.1}$ the radius of the limit cycle. The initial condition is set to $(a_1, a_2, a_3, a_4) = (R_c, 0, 0, 0)$, in this case the first cylinder is on its limit cycle and the second cylinder is at the fixed point (0,0). The dynamics of the GMFM without control is described in figure 5.8a. Figure 5.8b shows the control of the GMFM with an open-loop control law $K = cos(10t)$. This control illustrates the coupling between the two cylinders as the first one is stabilized by destabilizing the first one. Any linearizion of the equations, leads to a decoupling of the oscillators and rendering any control hopeless. Thus, the nonlinear coupling is a key enabler for controlling the GMFM. We benchmark the Monte Carlo, LGPC and gMLC algorithms with the GMFM. Each algorithm has been tested one hundred times. The stopping criterion for all of them is the total number of individuals evaluated, fixed at $N_i = 1000$ individualss. For one realization, first a Monte Carlo step of 100 individuals is performed. Then those 100 individuals are completed with Monte Carlo, LGPC or gMLC. The parameters for each algorithm is summarized in table 5.4.

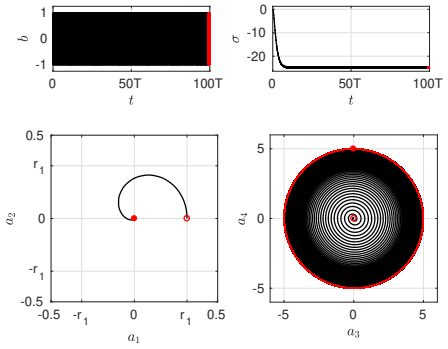
To assess the speed of each algorithm, we introduce a metric to measure the learning speed: $N_{95\%}$.

$$N_{95\%} = \underset{\mathbf{N}\in[1,1000]}{\arg\min}\ [J_N - J_{final} - 0.05(J_0 - J_{\text{final}})]$$

with J_0 being the unforced cost and J_{final} being the final value that is the best cost of the realization. $N_{95\%}$ represents the number of the individual that reaches 95%



(a) Unforced dynamics for the GMFM with $[0.01, 0, R_c, 0]$ as initial conditions.



(b) Dynamics of the GMFM controlled with the open-loop control law $K = cos(10t)$. The initial conditions taken are $[R_c, 0, 0, 0]$.

Figure 5.8: Unforced and energy-based open-loop control. For each case, we display the actuation command b, the growth rate σ and the two phase portrait. The red circle depicts the initial condition and the red dot the final state after one hundred periods.

of the final value. $N_{95\%}$ measure how fast we reach 95% of the final value.This measure is not enough to assess the performance of a realization, as a quick convergence can mean getting stuck in a local minimum. Thus, figure 5.9 displays the $N_{95\%}$ measure of one hundred realizations of each algorithm against the cost of their best individual. The x-axis is learning time and the y-axis is the cost. Thus, we take into account both the convergence speed and the final solution.

Figure 5.9a, reveals that all three algorithms, Monte Carlo, LGPC and gMLC, converges rather quickly as most of the realizations have a $N_{95\%}$ value less than 400 individuals, except isolated cases. Moreover, the distribution of costs of the best individuals shows that most of Monte Carlo realizations are above $J/J_0 = 0.02$ whereas LGPC and gMLC are below that limit. Moreover, the density of costs, in figure 5.9b, reveals that there is a higher concentration of individuals in the lower left corner of the graph for gMLC compared to LGPC. This shows that gMLC is the algorithm that converges the fastest and towards better individuals.

5.4 Conclusion

We have stabilized the wake behind a fluidic pinball with three independent cylinder rotations in successively larger search spaces for control laws. Figure 5.10 summarizes the corresponding performances quantified by the average distance between the controlled flow and the steady symmetric solution. First, steady symmetric forcing is employed. A base bleed with a cylinder rotation of 28% of the oncoming velocity leads to a flow which is 49% closer to the symmetric solution than the unforced attractor. Other studies also report about a stabilizing effect of base bleed on bluff body wakes (Wood, 1964; Bearman, 1967). In contrast,

parameter	description	value
	function library	$+$, $-$, $\times$, $\div$, sin, tanh, exp, log
N_b	number of controllers	1
$\boldsymbol{s}$	control law inputs	a_1, a_2, a_3, a_4
N_vr	number of variable registers	1+4=5
N_cr	number of constant registers	10
$N_\mathrm{inst,max}$	max number of instructions	50
N_MC	Monte Carlo individuals	100
	LGPC parameters	
P_c	crossover probability	0.6
P_m	mutation probability	0.3
P_r	mutation probability	0.1
N_e	elitism number	1
	gMLC parameters	
N_sub	subspace size	10
P_c	crossover probability	0.5
P_m	mutation probability	0.5
N_p	number of individuals per phase	50

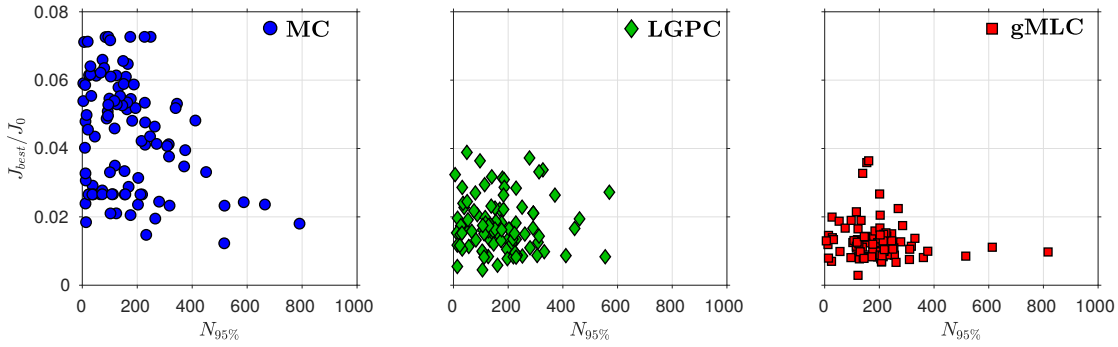
Table 5.4: Algorithm parameters for the GMFM

Coanda forcing, i.e., two symmetric cylinder rotations which accelerate the outer flow, may completely stabilize the flow. Yet, this new wake has no long vortex bubble and is further away from the symmetric steady solution than the unforced vortex shedding.

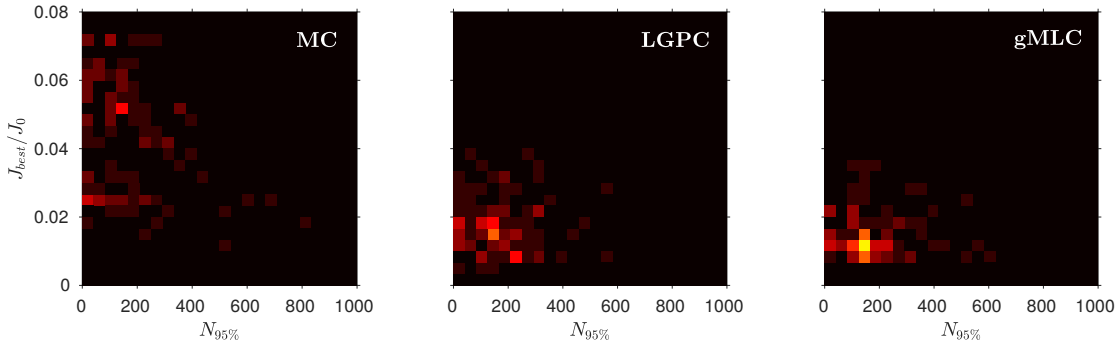
Second, a general non-symmetric actuation is optimized with the explorative gradient method. Surprisingly, an asymmetric actuation reduces the average distance between the flow and the steady target solution further to 28% of the unforced value. This asymmetric actuation leads to shear layer vortices which do not interact and thus do not form von Kármán vortices. The mean flow is slightly asymmetric, but largely mimics the elongated steady symmetric solution. The price for the better performance is a larger actuation power (see figure 5.10). Intriguingly, machine learning control also leads to distinctly asymmetric actuation in experiments (Raibaudo *et al.*, 2019) and simulations (Cornejo Maceda *et al.*, 2019) for other cost functions.

Third, a feedback actuation obtained from gradient-enriched machine learning control brings the flow even closer to the steady target solution. The associated actuation power is smaller than the previous optimized steady actuations (see figure 5.10). The actuation is a combination of asymmetric steady forcing and phasor control. The resulting flow looks similar to the optimal asymmetric steady forcing.

The feedback control does not seem to have the authority to completely stabilize the symmetric target solution like for the cylinder wake controlled by a



(a) Display of the best cost for each one of the one hundred runs according to $N_{95\%}$.



(b) Joint pdf for the one hundred realizations.

Figure 5.9: Results of the one hundred realizations for each algorithm: Monte Carlo (left), LGPC (middle) and gMLC (right). In the joint pdf plots, bright colors denote higher probability of presence. $N_{95\%}$ represents the number of individuals to reach 95% of the final value.

volume force (Gerhard *et al.*, 2003). The wake can be 'almost' stabilized for short periods of time, starting from the unforced flow. Then, new coherent structures emerge and lead to residual shear-layer shedding. This lack of complete authority for stabilization may be explained by the complexity of the dynamics. The fluidic pinball has a primary instability associated with von Kármán vortex shedding, a secondary pitchfork instability associated with the centreline jet and two Kelvin-Helmholtz instabilities of the top and bottom shear layer.

Intriguingly, symmetric high-frequency forcing can bring the flow even closer to the steady target solution but with an actuation power which is roughly two orders of magnitude larger than the previous control laws (see figure 5.10). Protas (2004) and Thiria *et al.* (2006) also find a stabilizing effect of high-frequency forcing on vortex shedding. The thickening of the shear layers by high-frequency vortices reduces the gradients and thus the instability. To summarize, machine learning control has automatically discovered well known stabilizing mechanisms, like base bleed and phasor control, but added an unexpected asymmetric forcing and combination of this open-loop actuation and phasor feedback for improved performance.

Finally, the combination of the gradient optimizer with genetic programming manage a significant improvement in terms of convergence speed and final solution.

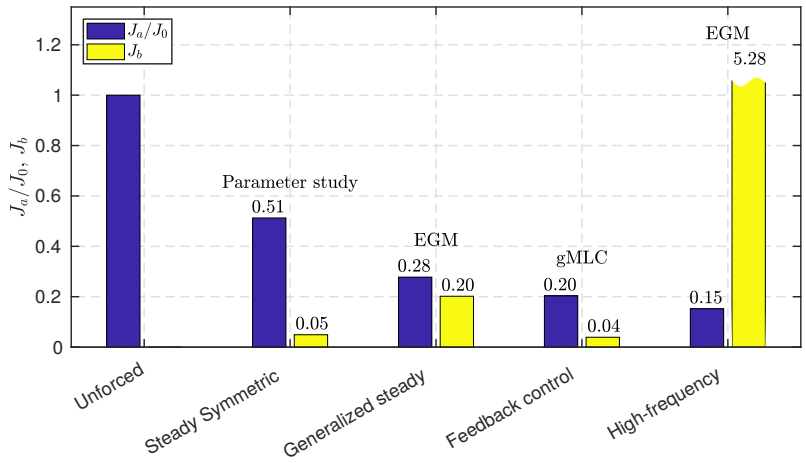


Figure 5.10: Summary of the performances for the best solutions for each method/search space couple. The relative distance to the symmetric steady solution J_a/J_0 and the actuation power J_b is represented. The costs are computed over 1000 convective time units.

This has been confirmed both on the stabilization of the fluidic pinball and the control of the generalized mean-field model. The most impressive results lies for the fluid system where after 200 evaluations, gMLC already built a better control law than 1000 evaluations of LGPC. Such speed up is a valuable feature as it allows to multiply the number of parameters tested in experiments. Moreover, the repeatability test on the GMFM assures the robustness of the benefits of gMLC compared to LGPC. Even for such simple dynamical system, gMLC performs better than LGPC in terms of speed and final solution.

In the next part, gMLC is experimentally demonstrated on the control of the open cavity.

Part III

Experimental demonstration

Chapter 6

Open cavity experiment

Now, we test the gradient-enriched machine learning control on real experimental conditions. We employ gMLC to control a single-input single-output experiment, the open cavity flow. In this chapter, we describe the experiment setup and the dynamics of the flow to be controlled. First, we give an introduction on cavity flows in section 6.1. Then, we present the wind tunnel, the means of actuation and sensing and the real-time system in section 6.2. In section 6.3, we detail the dynamics of the unforced flow and the controlled dynamics with constant actuation. In the next chapter, we present the control results both LGPC and gMLC.

Contents

6.1 Introduction

The development of the many possible machine learning methods to achieve active control of fluid flows must cope with the intrinsic complexity of the fluid-structure interaction as soon as high velocity regimes are considered. The fairly recent tendency to use artificial intelligence methods in this field of control is not the result of a fashion effect, but of the observation that deducing a law of control from the general equations of the highly nonlinear problem to be solved has found a solution only in very specific cases, mainly when a linearization (Brunton & Noack, 2015; Rowley *et al.*, 2006), formal or implicit, around the state to be

reached was conceivable. It is clear that such approach cannot answer to the demand for complex process optimizations in the fields of transport, heat and chemistry, to cite a few examples. The already very significant successes of passive control, particularly in the automotive and aviation sectors are coming up against the constraints of use and design. It is to overcome these limitations that active control, i.e. time-dependent action by means of an actuator, is considered. In case of closed loop control, the action depends on a measurement of the system state at the current or/and previous time. In this experimental work, we aim to confront the efficiency of our methodological developments in machine learning control with the test of experience on real flows. We will compare the two approaches presented for the building of the control laws: the linear genetic programming control (LGPC, chapter 2) our newly developed method, the gradient-enriched machine learning control (gMLC, chapter 5).

If, in fluid control, the ultimate goal is indeed the control of fully developed turbulence in its whole complexity, we choose here a flow configuration gathering most of the mechanisms responsible for nonlinear interactions but keeping the self-organization of the spatial structures still coherent: the flow above an open cavity for a moderate Reynolds number value ($Re_L \approx 10^4$). The dynamics of the interaction between a boundary layer and a rectangular cavity depends on 6 parameters, the ratios of the 3 spatial dimensions of the cavity, in particular the width L and depth D, the boundary layer momentum thickness θ_0 at the upstream edge, the velocity of the incoming flow U_∞ and for compressible flows the Mach number. A given cavity, open to an incoming flow, successively presents, as the incoming velocity increases, first, intra-cavitary centrifugal instabilities and second, self-sustained oscillation in the mixing layer (Rowley & Williams, 2006; Basley *et al.*, 2014; Feger *et al.*, 2019). Therefore, by playing on the 2 main control parameters, L/D and $Re_L = \frac{U_\infty L}{\nu}$, it is possible to scan a wide range of dynamics, from a single main mode to spectra with a rich number of coupled modes. This is, in addition to the practical implications, the reason for the repeated interest in this flow pattern from pioneering work to the present day (Krishnamurty, 1955; Rossiter, 1964; Gharib & Roshko, 1987). We cannot forget to mention one of the very first and remarkable closed-loop control attempts by Gharib and his co-authors on an open cavity in a water canal (Gharib *et al.*, 1985).

6.2 Experimental setup

6.2.1 Wind tunnel and cavity setup

The cavity is inserted into the rectangular cross-section duct of a small wind tunnel, 0.075 m high and 0.30 m wide. The cavity is $D = 0.05$ m deep, $S = 0.30$ m wide and $L = 0.075$ m long. Figure 6.1 illustrates the cavity configuration. Upstream of the cavity a Blasius-type boundary layer develops over a distance of 0.30 m from an elliptical edge. The residual standard deviation of velocity fluctuations is less than 1%.

An anemometer is located at the exit of the open wind tunnel vein. Measure-

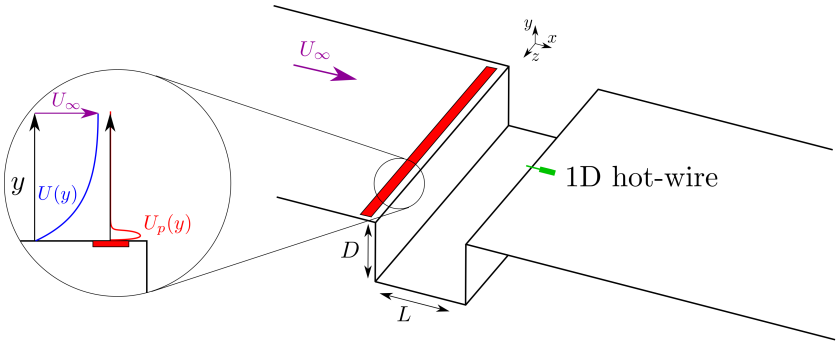


Figure 6.1: Experimental set up. The red band depicts the DBD upstream. A focus is made at the trailing edge: the Blasius boundary layer is depicted in blue and the body force effect of the plasma actuator in red.

ments show that the free stream velocity U_∞ and the velocity measured at the exit are linearly related to the rotation speed of the wind tunnel fan ω motor, thus U_∞ can be estimated from the anemometer. We set then $U_\infty = 2.11 \pm 0\,04$m/s, so that the Reynolds number is $Re_L = 1.03 \times 10^4 \pm 200$ and the momentum boundary layer thickness is estimated at $\theta_0/L = 1.05 \times 10^{-2}$ following chapter 4 of Basley (2012). Great care has been taken to calibrate the incoming speed with reference to LDV measurements and its regulation. However due to the variation in temperature ($\approx$2 °C) and very low cycle frequencies in the wind tunnel at this low operating point of around two meters per second, it is difficult to maintain the speed constant within better than 2% over the 24 hours necessary for the longest experiment sessions. The flow is within the incompressible range with a Mach number less than 10^{-2}. A more detailed description of the set-up can be found in Basley *et al.* (2013); Lusseyran *et al.* (2018).

6.2.2 Actuation and sensing

In this section we describe the DBD plasma used for actuation and the hot-wire sensor used to measure the outflow velocity.

Actuation

The actuation is carried out thanks to a non-thermal plasma actuator to locally force the boundary layer at the entrance of the cavity as depicted in red in figure 6.1. The idea of acting on a flow using a cold plasma is due to J. R. Roth who filed a patent in 1995 (Roth *et al.*, 1998). The DBD electrode consists of two conductive blades placed on either side of an insulating blade and subjected to a high alternating voltage. The lateral offset between the two blades creates the component of the electric field parallel to the plate and responsible for an ion wind in the stream-wise direction. The principle and the adjustment of the parameters for an application as a fluid actuator are detailed thoroughly in Forte *et al.* (2007); Benard *et al.* (2010).

In our experimental setup, the dielectric is made of 2 mm-thick acrylic glass

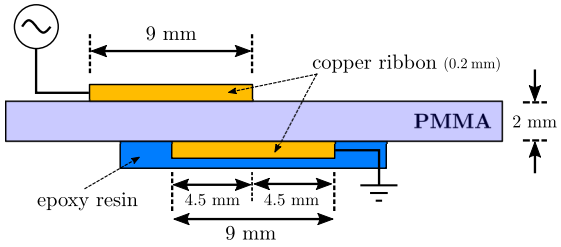


Figure 6.2: Sketch of the DBD used for the experiment.

and the electrodes are made of 9 mm wide and 26 cm long copper ribbon. Our DBD actuator is placed at $x = 4$ mm upstream to the leading edge.

The active electrode is supplied by a Trek high-voltage amplifier amplifying ($\times$3000) a carrying signal $e(t) = \hat{e}_{\max}(t)\sin(2\pi f_p t)$ created by an Agilent Function Generator. The amplitude of the carrying signal $\hat{e}_{\max}$ is modulated by a command signal y, such as $\hat{e}_{\max}(t) = (y(t) + 5)\ /\ 10e_{\max}$, where $y \in [-5, 5]$. f_p being the carrying frequency and $e_{\max}$ is the maximum amplitude. Both f_p and $e_{\max}$ are parameters of the generator. Thus, the active electrode voltage is finally given by

$$E(t) = \frac{y(t) + 5}{10} E_{\max}\sin(2\pi f_p t) \tag{6.1}$$

where $E_{\max} = 3000 \times e_{\max}$. To generate the ionic wind, $E(t)$ needs to be higher than the ignition voltage $E_0 = 3000 \times e_0$. $E_{\max}$ is set such as the electro-hydrodynamic force does not modify too much the shear layer thickness. It is defined as the maximum action level that keeps the main resonance of the cavity. The command signal $y(t)$ is sent from a dSPACE system and defined by

$$y(t) = \frac{\alpha}{2} b(t) + y_0 \tag{6.2}$$

where y_0 is an user-defined offset, α is a scaling factor and $b \in [-1, 1]$ is the actuation command. α and y_0 are chosen such as $E|_{b=-1} = E_0$ and $E|_{b=1} = E_{\max}$.

In practice, the values of e_0 and $e_{\max}$ are measured before each experiment as they are sensible to the atmospheric pressure, temperature of the room and number of hours of use of the electrode. The range of the actuation command b being independent of e_0 and $e_{\max}$ makes the control robust against their variations.

Sensing

For sensing, we use constant temperature anemometer (DANTEC hot wire prob 55P16, converter mini CTA54T30) with a single 1D hot-wire sensor 5 µm diameter and 1 mm length located slightly before the trailing edge of the cavity, as sketched in green in figure 6.1. The measured signal $E_{\text{hw}}(t)$ is converted into stream-wise velocity information u_{hw} according to King's law:

$$E_{\text{hw}}^2 = A + B u_{\text{hw}}^n \tag{6.3}$$

where the coefficients A, B and n are determined using a calibrated anemometer at different speeds. Time series recorded by the 1D hot-wire for the natural open

cavity flow is presented in figure 6.5. We do not correct the temperature drift on the hot-wire measurements as we do not have a direct measurements of the temperature.

6.2.3 Real-time system

In our experiment, the signal acquisition and control are carried out by a dSPACE real-time controller, including a DS1600 4 cores processors board and a DS2201 I/O board with a 12 bits on ± 10 volt analog-to-digital converter. The output of the hot-wire device is translated and amplified ($\times 40$) before conversion. A Simulink model is built and exploited by the ControlDesk software. The simulink model is displayed in figure 6.3. First, the signal from the hot-wire sensor is amplified and and serves as input to a control model. Then, a linear correction is applied before the nonlinear transformation by the King's law. A correction of the mean value is then performed to compensate the motor's drift. The input for the second block is then the velocity measured by the hot-wire. A list of control laws is generated and integrated directed in the model thanks to the *controlLaws* function block, the selection of the control laws is then done thanks to an external 'knob'. The output is the actuation command $b(t)$ with value in $[-1, 1]$. Then, the next block scales the actuation command to $y(t)$ so that $E|_{b=-1} = E_0$ and $E|_{b=1} = E_{\max}$. The final block is the output block that sends the scaled signal $y(t)$ to the amplitude modulation protocol of the Agilent Function Generator V5462IA. Of course, all the intermediate quantities are stored for each control law to compute their cost and carry out all the post-processing.

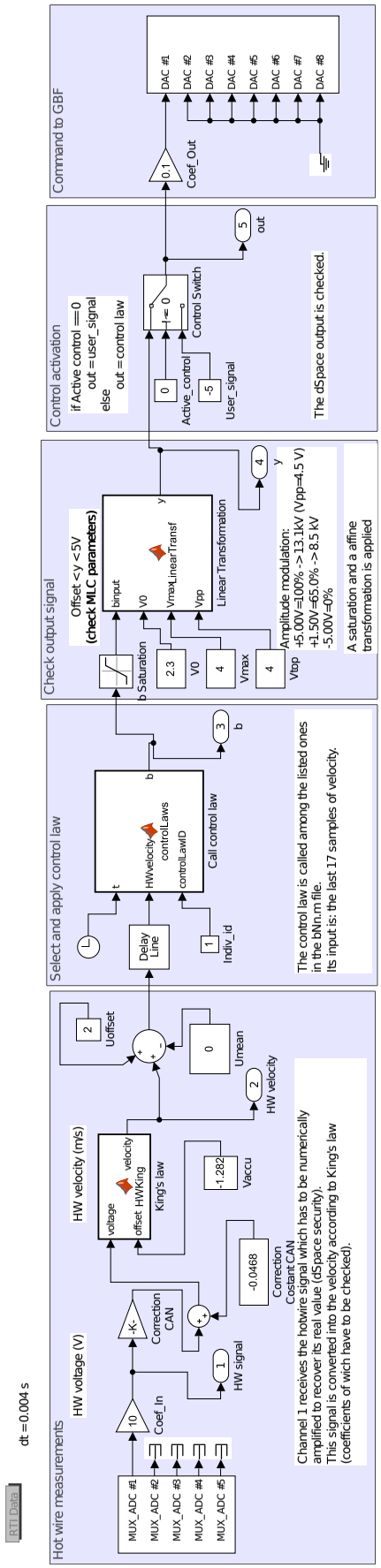


Figure 6.3: Simulink model for the control of the open cavity.

6.3 Flow dynamics

6.3.1 Natural unforced flow

As suggested in section 6.1, one can easily choose the degree of complexity of the intra-cavity dynamics by playing on the two control parameters which remain adjustable: the width and depth of the cavity. Following Basley *et al.* (2013), most main frequencies measured in the downstream shear layer align together around the lines of locked-on modes (figure 6.4). The associated Strouhal number based on L is then given by equation 6.4:

$$St_n^L\left(\frac{L}{\theta_0}\right) = \frac{f_n\ L}{U_\infty} = \frac{n-\gamma_n(L/\theta_0)}{2} \tag{6.4}$$

where the parameter $n = 1, 2, 3$ can be seen as the number of wavelengths within the cavity length and the corrective term γ_n corresponds to the deviation from the ideal phase difference $2\pi n$. The expression of γ_n being:

$$\gamma_n(L/\theta_0) = \frac{41\ n - L/\theta_0}{10\,(17-n)} \tag{6.5}$$

In this work, θ_0 was not measured directly, but deduced from an modified Blasius law $\theta_0 = \kappa\sqrt{2\nu l_x/U_\infty}$ with $\kappa = 0.3973$ instead of the classical Blasius's value $\kappa = 0.4696$, $\nu = 15.3\times10^{-6}\mathrm{m}^2\,\mathrm{s}^{-1}$. This reduction of the boundary layer thickness is due to the planing effect of the 200 µm thick DBD electrode, glued just before the leading edge.

We target a flow whose oscillations are centred on a main frequency f_a such that $8\times f_a \leq f_s$, with $f_s = 250\,\mathrm{Hz}$ being the sampling frequency. In order for this frequency f_a to be predominant but nevertheless accompanied by side-band peaks, it must be close to resonance, i.e. with a small but not zero γ_n. By choosing a cavity with an aspect ratio $R = L/D = 1.5$ we obtain for $U = 2.15\,\mathrm{m\,s}^{-1}$, a Strouhal number $St_L = fL/U_\infty$ close to 1 for n=2 and in the right frequency range. Table 6.1 summarizes the main frequencies along with their associated characteristics computed from equation 6.4 and 6.2.2eq:gamma for $\theta_0/L = 91.0$. The frequencies are also depicted in figure 6.5. The low and very low frequencies

peaks	f^-	f_a	f^+	$f_b \approx f^+ - f_a \approx f_a - f^-$
n	1	2	3	-
St_n^L	0.6563	1.0300	1.3857	-
γ_n	-0.3126	-0.0601	0.2285	-
freq. (Hz)	18.80	29.50	39.69	10.45

Table 6.1: Summary of the main frequencies f_b, f^-, f_a, f^+ for $\theta_0/L = 91.0$.

generated by nonlinearities are a source of difficulties, thus we choose to reduce their impact in this first study of control of intra-cavity dynamics. The choice of parameters (cf. table 6.1) contributes to that in two ways. Firstly, we reduce the intermittence phenomena between the main frequency f_a and its side-band

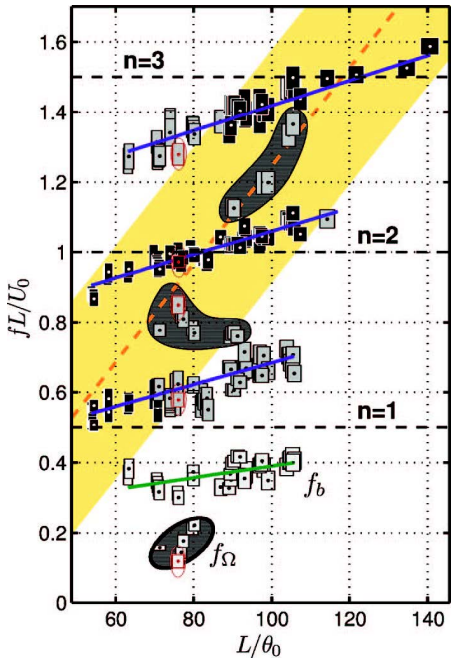


Figure 6.4: Primary Strouhal numbers based on L are displayed as functions of dimensionless cavity length L/θ_0; self-sustained oscillations frequencies (black), side-band peaks (gray) and low frequencies (white). Rectangle dimensions represent uncertainties. The shaded area (yellow) is drawn a posteriori such as to segregate self-sustained oscillation frequencies from most side-band peaks. It is delimited by $St^L = St^L_* \pm 1/3$,with $St^L_* = 0.014 \times (L/\theta_0 - 10)$ the centreline Strouhal number. Hatched regions highlight side-band frequencies departing from the general scheme. Figure and legend from Basley *et al.* (2013).

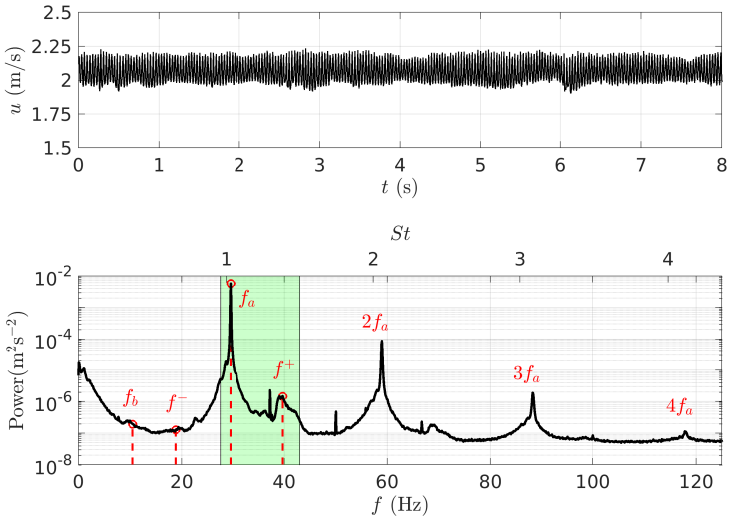


Figure 6.5: Characteristics of the unforced cavity flow. Time series of the velocity retrieved by the hot-wire downstream (top) and PSD of the signal with its main frequencies (bottom). The green region expands from $f = 27.5\,\text{Hz}$ to $f = 43\,\text{Hz}$. The frequencies displayed are theoretical values computed from equation 6.4 and 6.5.

frequencies when their amplitude become comparable, as it is the case when $|\gamma_2| \gtrsim 0.1$. And secondly, we reduce the coupling between the low frequencies originating from the transverse centrifugal instabilities and the energy supply provided by the instability of the shear layer.

To conclude this description of the cavity dynamics, let us specify that the goal we set for the control is to reduce the oscillations of the mixing layer by penalizing the amplitude of the frequencies $f_a = 29.50\,\text{Hz}$ and those connected to $f^+ = 39.69\,\text{Hz}$ in the first order of the spectrum, as indicated by the green shaded area of the figure 6.5. We note, however, that there is a small discrepancy between the measured values and the theoretical values, indeed, the measured values of f^- and f^+ seems to be higher and lower, respectively, than the theoretical ones. The measured value of f_b is then consistent with its definition. The sharp peak at frequency $50\,\text{Hz}$ corresponds to electric noise, its associated power is more than three orders of magnitude lower compared to the f_a. The other sharp frequency at $\approx 36\,\text{Hz}$ may corresponds to another external noise.

6.3.2 Actuation response

In this section, we described the response to a steady actuation at different levels. For this, we vary $\hat{e}_{\max}$ from $1.8\,\text{V}$ to $4\,\text{V}$ with a $0.2\,\text{V}$ step, such as we cross the ignition voltage associated to e_0. Each actuation is run over one minute with a five second pause between the tests. Two sets of measurements have been done, one where the values are tested increasingly and another where the values are tested decreasingly. Figure 6.6 shows the spectrum for each actuation level for

the two sets of measurements. First, we notice that, as we increase the actuation level, the spectrum flattens. The third harmonic $3f_a$ is no longer distinguishable after $V = 2.4$, the second harmonic $2f_a$ subsists until $V = 3.4$ and the first harmonic f_a disappears around $V = 3.9$. We note that along the decrease of the harmonics, the noise level increases. The frequency f^+ is visible at start, then becomes less distinguishable from $V = 2.4$ to $V = 2.7$, then it appears again, gaining magnitude until it vanishes in the noise level at $V = 3.9$. We note that the frequency f^+ slightly shifts towards higher levels, this may be explained by the increase of the momentum boundary layer thickness due of the strong actuation. We also remark a discontinuous behaviour between $V = 2.4$ and $V = 2.5$ for the ascending measurements and between $V = 2.5$ and $V = 2.6$ for the descending measurements. Indeed, at $V = 2.4$, the second harmonic is clearly visible but at $V = 2.5$, it almost flattened while a bulge appeared around f^+. This bulge progressively decreases to the benefit of the second harmonic. The ascending and the descending measurements show almost the same dynamics except for the discontinuity described. We take care to choose e_0 and $e_{\max}$, the action level thresholds such as they include the rich dynamics around the discontinuity. We note that there is also a discrepancy between the measured frequencies and the theoretical values. This kind of hysteresis underlines the complexity of the coupling between the DBD actuator and the flow, especially just above the E_0 threshold. In this voltage range, the actuator does not reduce itself to an affine law between command and action. The learning process will have to take this complexity into account.

In this chapter, we presented the experimental setup of the single-input single-output control system that is the open cavity. We described the hardware, software and the control tools. As a first study, we decided to work on a weakly nonlinear regime but that still displays complex and rich dynamics.

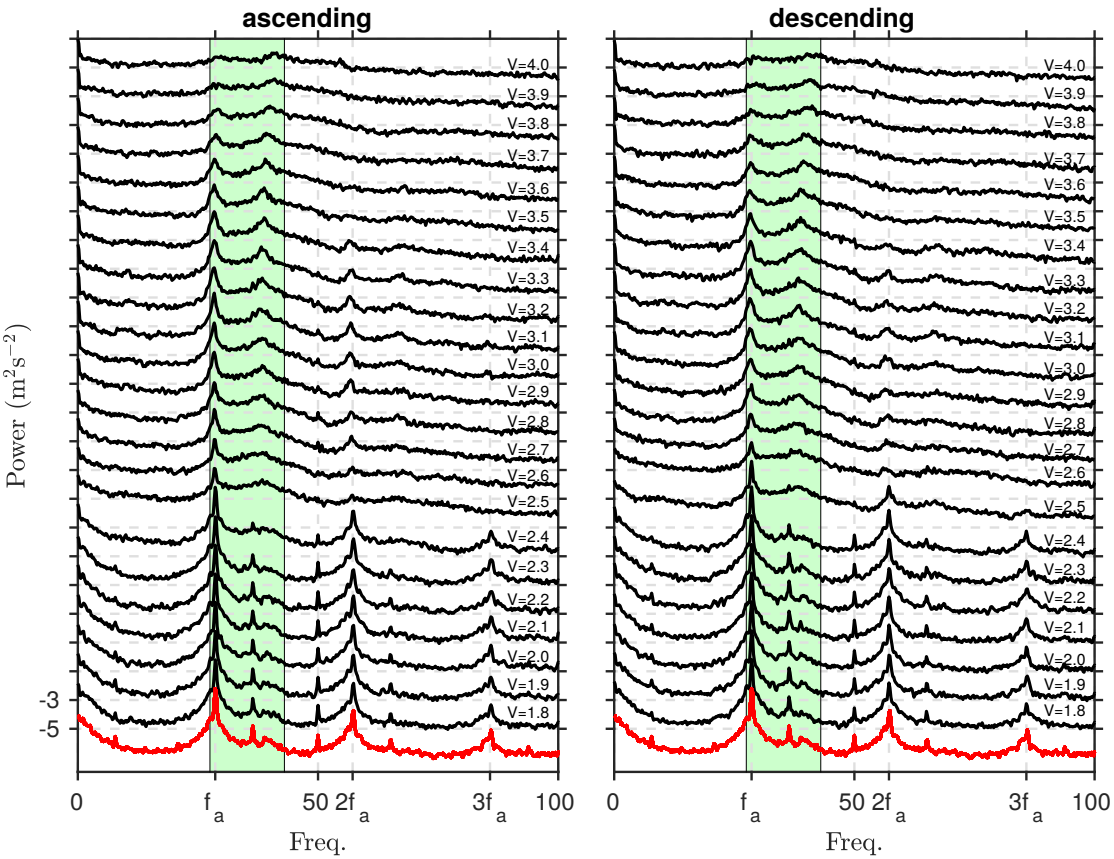


Figure 6.6: Actuation response for different steady actuation levels. The actuation levels are displayed on the right side of the plots. The vertical axis is in log scale. The spectra have been successively shifted by 10^2 from the unforced natural flow (in red) to ease the visualization.

Chapter 7

Feedback control optimization

In this chapter, we aim to control the single-input single-output cavity flow by the means a DBD situated at the trailing edge upstream. We employ our two codes: `xMLC` based on linear genetic programming control (LGPC, chapter 2) and `gMLC` based on gradient-enriched machine learning control (gMLC, chapter 5) in order to find a control law aiming to mitigate the oscillations of the mixing layer. Then, we carry out an analysis of the control law derived with each algorithm. We show, in particular, that gMLC presents a significant speed-up in learning efficient feedback control laws compared LGPC, confirming the speed-up observed for numerical simulations in chapter 5. First, in section 7.1, we present the control problem, then we detail the LGPC and gMLC results in section 7.2 and section 7.3 respectively. Finally, we conclude with a discussion on the learning process in experimental conditions in section 7.4.

Contents

7.1 Control problem

In this section, we describe the cost function associated to our control problem.

7.1.1 Cost function

As stated in chapter 6, our aim is to mitigate the oscillations of the mixing layer, for this we consider the main frequency of the spectrum in a given range as a measure of the control performance. Also, in section 6.3.2, we notice that a strong constant actuation is enough to reduce significantly the main peak of the spectrum, thus we consider an actuation penalization term for the control. The cost function then reads:

$$J(b) = J_a(b) + \gamma J_b(b) \tag{7.1}$$

Amplitude reduction

The term J_a serves to characterize the reduction of the main frequency f_a. We choose to focus our search in the range $[27.5, 43]$ comprising f_a and f_+. f^+ is also included in the detection range as it corresponds to a mode that is excited with enough steady actuation, see section 6.3.2. Thus, J_a reads:

$$J_a(b) = \frac{a_{\mathrm{PSD}}(\tilde{u}(f))}{a_{\mathrm{PSD}}(\tilde{u}_0(f))} \tag{7.2}$$

where $a_{\mathrm{PSD}}(\tilde{u}_{\mathrm{hw}}(f)) = \max_{f\in[27.5,43]} \tilde{u}(f)$ and $\tilde{u}(f)$ is the Fourier transform of the velocity u measured by the hot-wire sensor. We normalize the cost by the value of the peak for the natural unforced flow, so we have a direct measure of the reduction of the peak. The PSD is computed over $T_{\mathrm{ev}} = 40\,\mathrm{seconds}$. The evaluation time has been chosen to balance converged statistics and practicality. Indeed, a too long evaluation time would be unpractical as several hundreds of control laws are tested.

Actuation penalization

For the actuation penalization term, we base J_b on the actuation command $b(t)$ as we do not have access to the effective power supplied for the control. J_b is then computed as the square of the actuation command averaged over T_{ev} so that it is analogue to an energy. J_b is normalized by the range of the actuation, so that $J_b(b=-1) = 0$ and $J_b(b=1) = 1$.

$$J_b(b) = \frac{\langle (E - E_0)^2 \rangle_{T_{\mathrm{ev}}}}{(E_{max} - E_0)^2} = \frac{\langle (b(t)+1)^2 \rangle_{T_{\mathrm{ev}}}}{4} \tag{7.3}$$

As both cost function components J_a and J_b are normalized, we choose the penalization parameter $\gamma = 1$.

Standard deviation

Moreover to assess the performance of the control, we compute the standard deviation of the signal. T_{ev} is also chosen such as the standard deviation is sufficiently converged.

$$\widetilde{\sigma} = \frac{\sigma_{T_{\mathrm{ev}}}(u_{\mathrm{hw}})}{\sigma_{T_{\mathrm{ev}}}(u_0)} \tag{7.4}$$

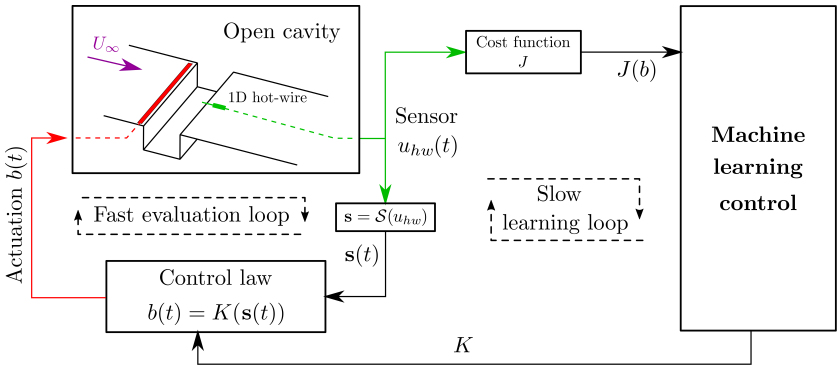


Figure 7.1: Diagram of the machine learning control learning process. $\mathcal{S}$ is the memory buffer for the velocity signal.

The standard variation is normalized by the standard variation of the natural unforced flow.

7.1.2 Control law ansatz

To control the cavity, we feedback the velocity measured downstream to the DBD upstream. We choose to take into account times-delayed signals as additional inputs. This choice is motivated by two reasons: first, by adding delayed information, we take into account the convective character of the fluid and secondly, with sensor history we allow to build ARMAX-based control such as in Hervé *et al.* (2012). Thus, the control ansatz is:

$$b(t) = K(\mathbf{s}(t)) \tag{7.5}$$

with

$$\mathbf{s}(t) = [u_{\mathrm{hw}}(t), u_{\mathrm{hw}}(t - 2T_s), ..., u_{\mathrm{hw}}(t - 20T_s)]. \tag{7.6}$$

$T_s = 1/f_s = 1/250$ s being the sampling time. We took 10 delays with a $2T_s$ step so that information of the 2.5 past periods is employed as input for the control laws. Figure 7.1 illustrates the experimental and control setup. The velocity measured by the hot-wire sensor is used both to characterize the flow thanks to the cost function and as an input for the controller. There is a fast evaluation loop for the evaluation of all the individuals and slow learning loop that updates the control laws according to the performance of the previous individuals. In the following, we present results for two machine learning control algorithms: linear genetic programming control and gradient-enriched machine learning control.

7.2 Linear genetic programming control

In this section, we present the results on controlling the open cavity flow by means of LGPC. The parameters chosen for the optimization are detailed in table 7.1.

The control optimization has been carried out for 10 generations of 100 individuals for a total of 1000 evaluations. As, we are dealing, in experiment, with a

parameter	description	value
	function library	$+$, $-$, $\times$, $\div$, sin, cos, tanh, exp, log
N_{b}	number of controllers	1
$\boldsymbol{s}$	control law inputs	$u_{\mathrm{hw}}(t)$,$u_{\mathrm{hw}}(t-2T_s)$, ..., $u_{\mathrm{hw}}(t-20T_s)$
N_{vr}	number of variable registers	14
N_{cr}	number of constant registers	10
$N_{\mathrm{inst,max}}$	max number of instructions	100
N_{MC}	Monte Carlo individuals	100
	LGPC parameters	
P_c	crossover probability	0.6
P_m	mutation probability	0.3
P_r	mutation probability	0.1
N_e	elitism number	1
	gMLC parameters	
N_{sub}	subspace size	10
P_c	crossover probability	0.5
P_m	mutation probability	0.5
N_{p}	number of individuals per phase	50

Table 7.1: LGPC and gMLC parameters to control the open cavity.

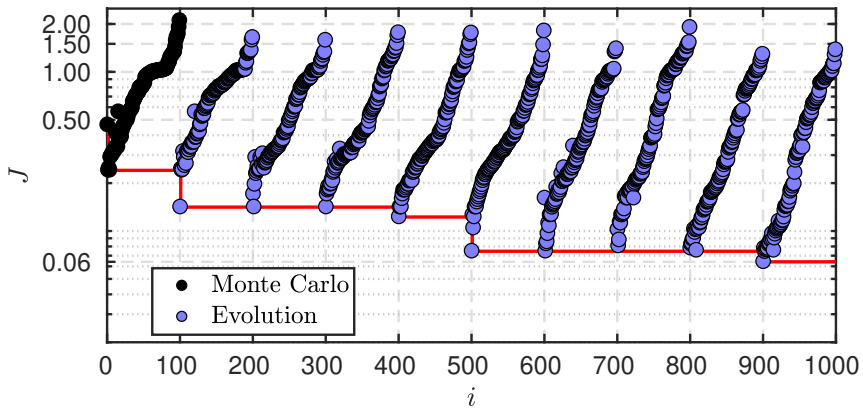


Figure 7.2: Leaning process of LGPC for the control of the open cavity. Each generation is sorted according to their cost. The vertical axis is in log scale.

noisy environment, the performance of an individual is taken as the average cost of all its past evaluations. Replicated individuals and copied thanks to elitism are then evaluated more than one time, thus their performance is more accurately assessed. However during the evolution process, only the immediate cost of the individual is used for the relative ordering of the individuals. Indeed, as we are likely to face drifts due to the temperature variations of the room, the time evolution of the response of the DBD actuator or the variations of the frequency of the fan motor, a comparison of individuals evaluated in the same conditions is preferred.

7.2.1 Controlled flow analysis

Figure 7.2 depicts the learning process for the 10 generations. We note that some dots come out of line. This is because those individuals have been evaluated several times and their prior evaluation were different from the following(s). Also, we remark that most of the learning is made in the first 6 generations. The last generation brings only a small improvement. The final cost after 1000 evaluations is: $J_{\text{LGPC}} = 0.0561$.

The associated control law is:

$$
\begin{aligned}
K_1^{\text{LGPC}} &= \log(\sin(\exp(s_4)) \times \sin(\log(0.636628 \sin(\exp(s_4))))) \\
\\
J_{\text{LGPC}} &= \mathbf{0.0561}, \\
J_a &= 0.0313, \\
J_b &= 0.0248, \\
\widetilde{\sigma}_{\text{LGPC}} &= 1.4631.
\end{aligned}
\tag{7.7}
$$

We notice that it is a nonlinear feedback control law comprising sin, log, exp and only one sensor $s_4(t) = u_{\text{hw}}(t - 8T_s)$ repeated twice. We note that both J_a and J_b have been reduced by a factor ≈ 30. However the associated standard deviation increased compared to the unforced flow. This is translated by an erratic velocity

signal in figure 7.3. The associated spectrum shows that, LGPC successfully managed to reduce the main peak f_a without exciting f^+. Nevertheless, this solution increased the general noise level of the flow compared to the unforced flow. This solution is similar to a strong steady actuation, such as $V = 4$ in fig. 6.6, but with a much lower actuation.

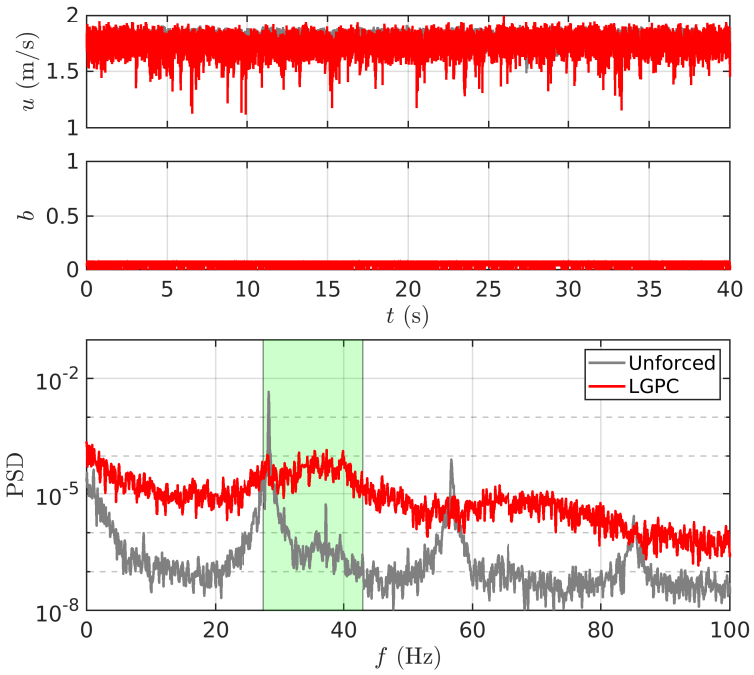


Figure 7.3: Dynamics of the open cavity controlled with LGPC.

7.3 Gradient-enriched machine learning control

Now let's consider the results of feedback control optimization with gradient-enriched machine learning control (gMLC), see chapter 5. The parameters used for this optimization are summarized in table 7.1.

7.3.1 Fast learning of feedback control laws

Figure 7.4, depicts the learning process for the gMLC optimization. Only 300 evaluations have bee carried out because of the drifts that perturb the learning process. We note that for both optimization algorithms, LGPC and gMLC, the initial Monte Carlo step manage to reduce the cost by more than 90%. The first evolution step of gMLC also brings comparable results with LGPC. Most of the improvement is done during the first exploitation step, where the simplex phase builds, after 50 evaluations, individuals whose cost is comparable to the best LGPC solution. The next evolution step do not bring any improvement and the last exploitation step only slightly reduces the cost. The final cost after 300

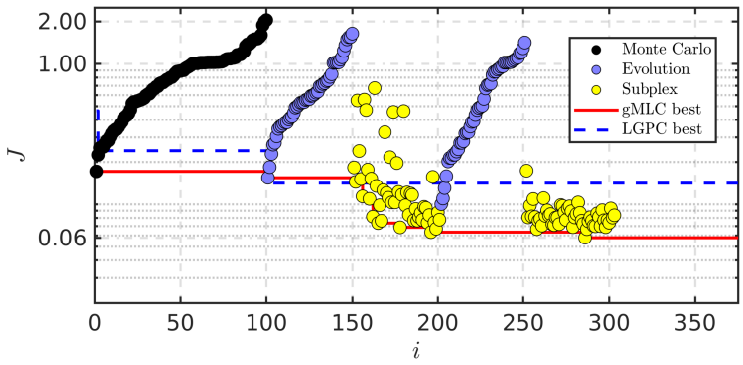


Figure 7.4: Learning process of gMLC for the control of the open cavity. The individuals are colored following the phase that generated them: blue for the exploration steps and yellow for the exploitation steps.

evaluations $J_{\mathrm{gMLC}} = 0.0578$ is slightly higher than J_{LGPC}. The expression of the best individual after simplification is:

$$K_1^{\mathrm{gMLC}} = \frac{-0.382095}{\tanh(s_6)} + \left(\frac{-0.324022}{s_1}\right) - \frac{0.107455}{\tanh(s_9)} + 0.071158,$$

$$\begin{aligned} J_{\mathrm{gMLC}} &= 0.0578, \\ J_a &= \mathbf{0.0173}, \\ J_b &= 0.0405, \\ \widetilde{\sigma}_{\mathrm{gMLC}} &= 0.8681. \end{aligned} \tag{7.8}$$

We note that K^{gMLC} allows a greater reduction of J_a compared. However, the actuation cost associated with the gMLC control is higher than the actuation cost of LGPC but still in the lower range of actions. Surprisingly, this nonlinear control law combining three sensors s_1, s_6 and s_9 reduces the standard deviation of the velocity to $\widetilde{\sigma}_{\mathrm{gMLC}} = 0.8681$.

The control law K^{gMLC} is in fact a linear combination of 10 control laws. Table 7.2 gathers all the control laws that compose b^{gMLC} along with their weights, costs and standard deviation. We note, that control law #7 managed an equivalent reduction of J_a but with a much higher actuation cost. The success of the downhill simplex step is to combine not so good individuals to build a better or that surpasses all of them. Indeed the cost of the best control law among the ten is $J = 0.15$ and K^{gMLC} improves this cost by almost a factor 3.

7.3.2 Feedback control law

The time series and PSD of the controlled flow by K^{gMLC} is depicted in fig. 7.5. We note that the time series of the measured velocity is comparable to the unforced flow unlike the flow controlled by K^{LGPC}. This solution manages to efficiently reduce the main peak f_a and its harmonics without exciting f^+, just like the LGPC solution. However, in the case of the gMLC solution, the noise level stays

#	b	weight	J	J_a	J_b	$\widetilde{\sigma}$
1	-0.286174	7.5977×10^{-2}	0.25	0.13	0.13	0.90
2	-0.361287	-1.0647×10^{-1}	0.27	0.17	0.10	0.93
3	-0.223013	-1.0111×10^{-1}	0.22	0.07	0.15	0.86
4	$\frac{-0.468662}{\tanh(s_6)}$	$\mathbf{8.1529 \times 10^{-1}}$	0.17	0.11	**0.07**	0.96
5	-0.128124	-2.8648×10^{-1}	0.25	0.06	0.19	0.87
6	-0.5174	4.821×10^{-2}	0.26	0.20	0.06	1.01
7	$\frac{-0.5174}{s_1}$	6.2625×10^{-1}	**0.15**	**0.02**	0.14	**0.79**
8	$\frac{-0.468662}{\tanh(s_9)}$	2.2928×10^{-1}	0.18	0.12	**0.07**	0.99
9	-0.174613	-5.2789×10^{-1}	0.27	0.10	0.17	0.90
10	-0.317493	2.2694×10^{-1}	0.23	0.11	0.12	0.98
	K^{gMLC}		0.06	0.02	0.04	0.87

Table 7.2: Summary of the 10 control laws composing K^{gMLC} described in equation (7.8). For each control law, we present the control law, the associated weight, J, J_a, J_b and the standard deviation. The best values for each quantity are written in bold. The values associated to the whole K^{gMLC} are also given for comparison in the last line.

low. Indeed, the spectrum of the gMLC controlled flow is close to the unforced one beyond the frequency window scanned. In fig. 7.5, the gMLC control command looks like a constant actuation at 10% of the maximum action. Thus, one can wonder if an equivalent constant action can produce the same results. Thus, to test this hypothesis, we control the flow with the same actuation command but as an open loop control. The PSD of the controlled with the open loop control equivalent is depicted in blue in figure 7.5 as 'gMLC-OL'. In the case of the gMLC-OL control, the main peak at f_a and its harmonics come back. We note in particular that the amplitude of the main peak increased almost by a factor 10 between the gMLC control and the gMLC-OL control. This shows that the small feedback, barely visible on the time series, is essential to the control of f_a. Thus, gMLC managed to build a feedback control law that effectively takes advantage of the flow information to kill the oscillations of the mixing layer without rising the noise level.

7.4 Conclusion

In conclusion, we successfully managed to control the flow in an experiment. Despite the complexity of the dynamics, the noisy environment and the various source of drift, both LGPC and gMLC managed to build relevant control laws capable to act effectively on the shear layer. The most impressive results are given by gMLC.

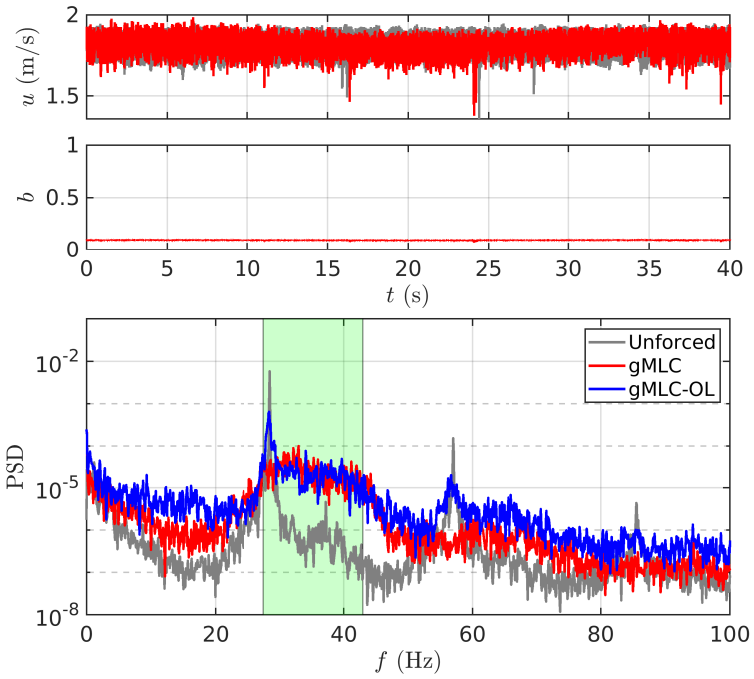


Figure 7.5: Dynamics of the open cavity controlled with the best control law derived by gMLC (in red). The control of the flow by the equivalent open loop actuation is depicted in blue, see text for more details.

Indeed, gMLC manage to build in less than 300 evaluations a feedback control law that manages to act on the main frequency of the flow. Moreover, the significant speed up of gMLC compared to LGPC, observed in chapter 5, is also observed in experimental conditions. These are not isolated results, as the speed up has been also observed in the preliminary tests. The drift of the DBD actuator resulted in slightly different solutions between the runs, but the order of magnitude of the cost reduction was the same between the different tests.

Chapter 8

Conclusion

This final section summarizes the key achievements described in this thesis and concludes with the perspectives and futures challenges of the field.

8.1 Achievements

8.1.1 Algorithmic development: fast learning of control laws with gMLC

In this work, we have pushed back the boundaries of control law optimization thanks to our new algorithm, the gradient-enriched machine learning control. We inspired ourselves from the EGM methodology (Li *et al.*, 2021) to combine genetic programming explorative power and gradient-based methods for speed. We employed our algorithms both in numerical simulations and experiments, revealing that gMLC outperforms consistently LGPC in terms of convergence speed and final solution.

Between the accelerated speed and quality of the solutions, we can safely state that we have accelerated the learning process by, at least, a factor 10, though a factor 25 can be said for some cases.

Such acceleration is valuable as it will open the access to a wide range of control experiments, including many actuators and sensors and multi-parameter testing for robustness. It will also enable control for costly experiments where long hours of training is often prohibitive.

8.1.2 Code development: `xMLC` and `gMLC`

Our algorithmic advancements come along software development. Indeed, we developed our own codes based on the literature Duriez *et al.* (2016); Brameier & Banzhaf (2006) and our methodological progress. Our two softwares `xMLC` and `gMLC` have been developed on MATLAB and are compatible with the free software GNU Octave. To understand and improve the learning process of LGPC, we carried out an extensive meta-parameter study of the genetic programming algorithm and elaborated some rules of thumb for the choice of the meta-parameters. Such

study has also been directly profitable to the success of gMLC as it contains a version of xMLC as a module for the reconstruction phase. The role of the xMLC solver and its efficiency is essential in the learning process of gMLC as it allows to integrate the simplex individuals in the database. Finally, thanks to the fruitful interaction with experimentalist collaborators such as Eliot Varon, Fan Dewei and Philipp Oswald, we designed our codes to an easy interface with real time controllers. This allowed us to distribute our code easily and facilitate its implementation in many experiments.

8.1.3 Control of the fluidic pinball in numerical simulations

Our codes have been applied to the fluidic pinball for two different problems: the reduction of net drag power and the stabilization of the flow. We carried out open-loop studies in the subspace of symmetric controllers revealing that boat tailing is the best strategy to reduce the drag power and that for flow stabilization base bleeding is preferred. Then, we allowed for a non-symmetric control with an independent rotation of the three cylinders. In those cases, LGPC, EGM and gMLC algorithms are employed to deal with such complex search spaces. EGM and LGPC showed for both control problems that an asymmetric control gives the best performances. We note also that a full stabilization of the flow is possible with high-frequency but with a high cost expense. However, for drag reduction, periodic forcing seems inefficient as LGPC systematically discarded the time-dependent functions provided. For both problems the best solutions correspond to combinations of known strategies including information of flow state thanks to velocity probes downstream:

- phasor control and asymmetric boat tailing for the net drag reduction problem;

- phasor control and asymmetric forcing for the flow stabilization problem.

8.1.4 Control of the open cavity in experiments

xMLC and gMLC have also been deployed in experimental conditions for the control of the open cavity. Despite the noisy environment and many sources of drifts, xMLC and gMLC managed to learn efficient control laws that reduce the main peak of the power spectrum, related to the oscillations of the mixing layer, by a factor 1000. The most impressive results are for the gMLC optimization, that managed a fast learning (less than 300 evaluations) of a feedback control law that reduces the target main frequency by a factor 1000 (in power) but also whole standard deviation. The feedback control law has been analyzed in detail, revealing that the control is effectively based on a feedback mechanism. Indeed, the suppression of the feedback leads to drastic loose in performance.

8.2 Future challenges

8.2.1 Performance with many sensors and actuators

We demonstrated the performances of gMLC for systems with less than O(10) actuators and sensors. One can also wonder how gMLC performances will scale with control problems including many inputs and outputs. From a practical point of view, nothing prevents us from including as many inputs and outputs as required. The stabilization of the fluidic pinball shows that increasing the number of sensor information is likely to enhance the maximum control performance. However, it is expected to also increase the total learning time. From LGPC's experience in past experiments (Ren *et al.*, 2020; Noack, 2019), we notice that there is only a low impact on the learning speed with increasing control inputs and outputs.

Also, to consider many inputs and outputs comes with the curse of dimensionality. One remedy for this, would be to augment the learning methods by exploiting the symmetries and invariants of the problem, such as Belus *et al.* (2019) for deep reinforcement learning. For LGPC, self-discovery of symmetries and invariants of the problem is unlikely, especially with high dimensional spaces. Indeed, genetic operations, crossover and mutation, work in a stochastic way and preserving symmetries of the controls laws with such nonlinear operators seems improbable. However, gradient-based methods have the benefit to linearly combine solutions, preserving symmetries and invariance among the solutions. We believe that, in this case, the continuous exploration of the search space can steer the simplex towards active subspaces and avoid the exploration of too large subspaces.

Finally, we can see two additional ways to overcome this challenge. We could artificially impose the symmetries and invariants of the problems on the individuals themselves, discarding those who do not satisfy the constraints. Another solution would be to relax the symmetry constraints by adding a specific term in the cost function. In this case, the choice of the penalization coefficient is open to discussion and would be tuned depending the need. Both approaches would, of course, be based on a prior analysis of the dynamics of the problem.

8.2.2 Control of complex flows

The scalability of the method to more complex flows, such as high Reynolds number presenting broadband turbulence, is also one question to be addressed in future works. We believe that the optimization principle that we present, balancing exploration (for discovering new basins of attraction of minima) and exploitation (for a fast descent towards the minimum) is a general scheme for non-convex optimization in high or infinite dimension spaces. In the case of gMLC, the exploration phases may be capable to discover the basin of attraction of the global minimum for more complex problems. As for the gradient-based part, our slide towards the minimum might only be stopped by erratic gradients. However, this limitation does not only apply to gMLC as it is inherently correlated to the definition of the cost function. It is likely that all model- free methods based on the same metric would face the same issues near the optimum. For more complex flows, the main

challenge lies in the physics of the problem and the definition of a cost function that is the least sensible to the random behaviour of turbulence.

Finally, recent successes on the control of turbulent jet (Zhou *et al.*, 2020) with GPC in high Reynolds number strengthen us to expect that our methodology is generalizable to complex flows. Recent experimental applications of gMLC, including mitigation of cavity oscillations, drag reduction of a generic truck model under yaw and lift increase of an airfoil under angle of attack at Reynolds number near one million, also reinforce our expectations. Performance and reproducibility of gMLC control are promising and outperform other methods, including LGPC.

LGPC and gMLC are powerful regression solvers validated in numerical simulations and experiments. They can derive complex expressions that exploit the nonlinearities of the flow. However, they may not be the most adequate algorithm for all problems. Indeed, a minimum number of evaluations is required for a converged learning and it may become impractical for complex 3D simulations or systems comprising very low frequencies. Moreover, employing LGPC or gMLC to derive smooth control laws that have a linear or affine relationship with the sensor signals may be overkill. Other methods such as cluster-based control (Nair *et al.*, 2019) or gradient augmented genetic algorithm Maehara & Shimoda (2013) may be more suited for such problems. Finally, optimization of the whole trajectory such as the stabilization of the fluidic pinball may not be ideal for systems that have low time scale responses. In this case, deep reinforcement learning approaches may be more efficient to build an optimal solution (Fan *et al.*, 2020; Rabault *et al.*, 2019). A hybridization of these techniques may result in more efficient algorithms, in the same way gMLC combines LGPC and gradient-based methods. In addition, so far, we only employed our algorithms as plug-and-play codes. Another next step to accelerate the learning process and achieve better performances would be to pre-process the feedback information to extractor key features based (or not) on the physics of the problem. For this purpose, the introduction of Morlet filters for the control of the cavity is already in progress.

8.2.3 Robustness

Lastly, we approach the problem of the robustness of the control laws. In this work, we optimized control laws only for a single Reynolds number. We believe that the question of robustness is also intrinsically linked to the cost function definition. Only a proper cost function that takes into account several conditions (such as different initial conditions or Reynolds numbers) may be general enough to regularize the control law landscape and set robustness as one the criteria of optimality. However such trainings, with varying conditions, require proper methods like multiple evaluations of the same control law (Asai *et al.*, 2019). An example of such training is proposed in Tang *et al.* (2020) for deep reinforcement learning. Therefore, the significant increase in the learning speed compared to LGPC is a decisive feature as it allows the testing of the same control at different conditions while keeping the total number of evaluation at realistic levels for experiment steadiness.

Robustness is currently explored in various experiments and will be the theme

of future publications.

Bibliography

Anderson, J. D. 2011 *Fundamentals of Aerodynamics*. McGraw-Hill.

Asai, S., Yamato, H., Sunada, Y. & Rinoie, K. 2019 Designing machine learning control law of dynamic bubble burst control plate for stall suppression. In *2019 AIAA SciTech Forum*. San Diego, CA.

Bagheri, S. & Henningson, D. S. 2011 Transition delay using control theory. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* **369** (1940), 1365–1381, arXiv: https://royalsocietypublishing.org/doi/pdf/10.1098/rsta.2010.0358.

Barros, D., Borée, J., Noack, B. R., Spohn, A. & Ruiz, T. 2016 Bluff body drag manipulation using pulsed jets and Coanda effect. *J. Fluid Mech.* **805**, 442–459.

Basley, J. 2012 An experimental investigation on waves and coherent structures in a three-dimensional open cavity flow. PhD thesis.

Basley, J., Pastur, L. R., Delprat, N. & Lusseyran, F. 2013 Space-time aspects of a three-dimensional multi-modulated open cavity flow. *Physics of Fluids* **25** (064105).

Basley, J., Pastur, L. R., Lusseyran, F., Soria, J. & Delprat, N. 2014 On the modulating effect of three-dimensional instabilities in open cavity flows. *J. Fluid Mech.* **759**, 546–578.

Bearman, PW 1967 The effect of base bleed on the flow behind a two-dimensional model with a blunt trailing edge. *Aeronautical Quarterly* **18** (03), 207–224.

Belson, B. A., Semeraro, O., Rowley, C. W. & Henningson, D. S. 2013 Feedback control of instabilities in the two-dimensional blasius boundary layer: The role of sensors and actuators. *Physics of Fluids* **25** (5), 054106, arXiv: https://doi.org/10.1063/1.4804390.

Belus, V., Rabault, J., Viquerat, J., Che, Z., Hachem, E. & Reglade, U. 2019 Exploiting locality and translational invariance to design effective deep reinforcement learning control of the 1-dimensional unstable falling liquid film. *AIP Advances* **9** (12), 125014, arXiv: https://doi.org/10.1063/1.5132378.

Benard, N., Moreau, E., Griffin, J. & Cattafesta, L. N. III 2010 Slope seeking for autonomous lift improvement by plasma surface discharge. *Experiments in Fluids* **48**, 791–808.

Benard, N., Pons-Prats, J., Periaux, J., Bugeda, G., Braud, P., Bonnet, J.P. & Moreau, E. 2016 Turbulent separated shear flow control by surface plasma actuator: experimental optimization by genetic algorithm approach. *Exp. Fluids* **57** (2), 22:1–17.

Bobusch, B. C., Woszidlo, R., Bergada, J. M., Nayeri, C. N. & Paschereit, C. O. 2013 Experimental study of the internal flow structures inside a fluidic oscillator. *Experiments in Fluids* **54** (1559).

Brameier, M. & Banzhaf, W. 2006 *Linear Genetic Programming*. Springer Science & Business Media.

Brunton, S. L. & Noack, B. R. 2015 Closed-loop turbulence control: Progress and challenges. *Appl. Mech. Rev.* **67** (5), 050801:01–48.

Brunton, S. L., Tu, J. H., Bright, I. & Kutz, J. N. 2014 Compressive sensing and low-rank libraries for classification of bifurcation regimes in nonlinear dynamical systems. *SIAM Journal on Applied Dynamical Systems* **13** (4), 1716–1732, arXiv: https://doi.org/10.1137/130949282.

Bucci, M. A., Semeraro, O., Allauzen, A., Wisniewski, G., Cordier, L. & Mathelin, L. 2019 Control of chaotic systems by deep reinforcement learning. *Proc. R. Soc. A.* **475**.

Cattafesta, L. & Shelpak, M. 2011 Actuators for active flow control. *Ann. Rev. Fluid Mech.* **43**, 247–272.

Cattafesta, L. N., Garg, S. & Shukla, D. 2001 Development of piezoelectric actuators for active flow control. *AIAA Journal* **39** (8), 1562–1568, arXiv: https://doi.org/10.2514/2.1481.

Chen, W., Ji, C., Alam, Md M., Williams, J. & Xu, D. 2020 Numerical simulations of flow past three circular cylinders in equilateral-triangular arrangements. *Journal of Fluid Mechanics* **891**, 1–44.

Choi, H., Jeon, W.-P. & Kim, J. 2008 Control of flow over a bluff body. *Ann. Rev. Fluid Mech.* **40**, 113–139.

Choi, H., Moin, P. & Kim, J. 1994 Active turbulence control for drag reduction in wall-bounded flows. *J. Fluid Mech.* **262**, 75–110.

Chovet, C., Feingesicht, M., Plumjeau, B., Lippert, M., Keirsbulck, L., Kerhervé, F., Polyakov, A., Richard, J. P., Abassi, W. & Foucaut, J. M. 2020 Sliding mode control applied to a square-back ahmed body. *European Journal of Mechanics - B/Fluids* **81**, 151 – 164.

Chovet, C., Lippert, M., Keirsbulck, L., Noack, B. R. & Foucaut, J. M. 2017 Machine learning control for experimental shear flows targeting the reduction of a recirculation bubble. *IFAC-PapersOnLine* **50** (1), 12307 – 12311, 20th IFAC World Congress.

Cornejo Maceda, G. Y., Li, Y., Lusseyran, F., Morzyński, M. & Noack, B. R. 2020 Stabilization of the fluidic pinball with gradient-based machine learning control. *J. Fluid Mech.* **(in revision, see arXiv)**.

Cornejo Maceda, G. Y., R., Noack B., Lusseyran, F., Deng, N., Pastur, L. & Morzyński, M. 2019 Artificial intelligence control applied to drag reduction of the fluidic pinball. *Proc. Appl. Math. Mech.* **19** (1), 1–2.

Cortelezzi, L., Leonard, A. & Doyle, J.C. 1994 An example of active circulation control of the unsteady separated flow past a semi-infinite plate. *J. Fluid Mech.* **260**, 127–154.

Debien, A., von Krbek, K. A. F. F., Mazellier, N., Duriez, T., Cordier, L., Noack, B. R., Abel, M. W. & Kourta, A. 2016 Closed-loop separation control over a sharp-edge ramp using genetic programming. *Exp. Fluids* **57** (3), 40:1–19.

Deng, N., Noack, B. R., Morzyński, M. & Pastur, L. R. 2020 Low-order model for successive bifurcations of the fluidic pinball. *J. Fluid Mech.* **884**, A37.

Dowling, A. P. & Morgans, A. S. 2005 Feedback control of combustion oscillations. *Annual Review of Fluid Mechanics* **37** (151–182).

Dracopoulos, D. C. 1997 *Evolutionary Learning Algorithms for Neural Adaptive Control*. London, etc.: Springer-Verlag.

Duriez, T., Brunton, S. L. & Noack, B. R. 2016 *Machine Learning Control — Taming Nonlinear Dynamics and Turbulence*, *Fluid Mechanics and Its Applications*, vol. 116. Springer-Verlag.

El Hassan, M. & Keirsbulck, L. 2017 Passive control of deep cavity shear layer flow at subsonic speed. *Canadian Journal of Physics* **95** (10), 894–899, arXiv: https://doi.org/10.1139/cjp-2016-0822.

European Commission, EU 2019 Co2 emission performance standards for cars and vans (2020 onwards). `https://ec.europa.eu/clima/policies/transport/vehicles/regulation_en`, [Online; accessed 22-December-2020].

Fan, S. L., Yang, L., Wang, Z. C., Triantafyllou, M. S. & Karniadakis, G. M. 2020 Reinforcement learning for bluff body active flow control in experiments and simulations. *Proc. Natl. Acad. Sci. USA* **117** (42), 26091–26098.

Feger, G., Lusseyran, F. & Pastur, L. 2019 Bifurcations successives de l'écoulement transverse en cavité ouverte et interaction avec les oscillations de la couche cisaillée. In *Congrès Français de Mécanique*. Brest, France.

Fisk, P. R. 1967 Models of the second kind in regression analysis. *Journal of the Royal Statistical Society. Series B (Methodological)* **29** (2), 266–281.

Flügel, G. 1930 Ergebnisse aus dem strömungsinstitut der technischen hochschule danzig. In *Jahrbuch der Schiffbautechnischen Gesellschaft*, pp. 87–113. Springer.

Forte, M., Jolibois, J., Pons, J., Moreau, E., Touchard, G. & Cazalens, M. 2007 Optimization of a dielectric barrier discharge actuator by stationary and non-stationary measurments of the induced flow velocity: application to airflow control. *Experiments in Fluids* **43**, 917–928.

Fukagata, K. & Nobuhide, K. 2003 Drag reduction in turbulent pipe flow with feedback control applied partially to wall. *Int. J. Heat Fluid Flow* **24**, 480–490.

Gautier, N., Aider, J.-L., Duriez, T., Noack, B. R., Segond, M. & Abel, M. W. 2015 Closed-loop separation control using machine learning. *J. Fluid Mech.* **770**, 424–441.

Gelbert, G., Moeck, J. P., Paschereit, C. O. & King, R. 2012 Advanced algorithms for gradient estimation in one-and two-parameter extremum seeking controllers. *J. Process Control* **22** (4), 700–709.

Gerhard, J., Pastoor, M., King, R., Noack, B. R., Dillmann, A., Morzyński, M. & Tadmor, G. 2003 Model-based control of vortex shedding using low-dimensional Galerkin models. In *33rd AIAA Fluids Conference and Exhibit*. Orlando, Florida, USA, June 23–26, 2003, paper 2003-4262.

Geropp, D. 1995 Process and device for reducing the drag in the rear region of a vehicle, for example, a road or rail vehicle or the like. United States Patent **US 5407245 A**.

Geropp, D. & Odenthal, H.-J. 2000 Drag reduction of motor vehicles by active flow control using the Coanda effect. *Exp. Fluids* **28** (1), 74–85.

Gharib, M. & Roshko, A. 1987 The effect of flow oscillations on cavity drag. *Journal of Fluid Mechanics* **177**, 501–530.

Gharib, M., Roshko, A. & Sarohia, V. 1985 Effect of flow oscillations on cavity drag and a technique for their control .

Giannetti, F. & Luchini, P. 2007 Structural sensitivity of the first instability of the cylinder wake. *Journal of Fluid Mechanics* **581**, 167–197.

Glezer, A., Amitay, M. & Honohan, A.M. 2005 Aspects of low- and high-frequency actuation for aerodynamic flow control. *AIAA Journal* **43** (7), 1501–1511.

Guyot, D., Bobusch, B., Paschereit, C. O. & Raghu, S. 2008 *Active Combustion Control Using a Fluidic Oscillator for Asymmetric Fuel Flow Modulation*, arXiv: https://arc.aiaa.org/doi/pdf/10.2514/6.2008-4956.

Hansen, N. & Ostermeier, A. 1996 Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 312–317.

Hervé, A., Sipp, D., Schmid, P. J. & Samuelides, M. 2012 A physics-based approach to flow control using system identification. *J. Fluid Mech.* **702**, 26–58.

Holland, J. H. 1975 *Adaptation in natural and artificial systems*. Ann Arbor: The University of Michigan Pres.

Ishar, R., Kaiser, E., Morzyński, M., Fernex, D., Semaan, R., Albers, M., Meysonnat, P. S., Schröder, W. & Noack, B. R. 2019 Metric for attractor overlap. *Journal of Fluid Mechanics* **874**, 720–755.

Jordan, P. & Colonius, T. 2013 Wave packets and turbulent jet noise. *Ann. Rev. Fluid Mech.* **45**, 173–195.

Juang, J.-N. & Pappa, R. S. 1985 An eigensystem realization algorithm for modal parameter identification and model reduction. *Journal of Guidance, Control, and Dynamics* **8** (5), 620–627, arXiv: https://doi.org/10.2514/3.20031.

Keirsbulck, L., El Hassan, M., Lippert, M. & Labraga, L. 2008 Control of cavity tones using a spanwise cylinder. *Canadian Journal of Physics* **86** (12), 1355–1365, arXiv: https://doi.org/10.1139/p08-086.

Koumoutsakos, P., Freund, J. & Parekh, D. 2001 Evolution strategies for automatic optimization of jet mixing. *AIAA J.* **39** (5), 967–969.

Kourta, A. & Vitale, E. 2008 Analysis and control of cavity flow. *Physics of Fluids* **20** (7), 077104, arXiv: https://doi.org/10.1063/1.2959170.

Koza, J. R. 1992 *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Boston: The MIT Press.

Koza, J. R 2010 Human-competitive results produced by genetic programming. *Genetic Programming and Evolvable Machines* **11**, 251–284.

Krishnamurty, K. 1955 Acoustic radiation from two-dimensional rectangular cutouts in aerodynamic surfaces. *Tech. Rep.* 3487.

Krstic, M., Krupadanam, A. & Jacobson, C. 1999 Self-tuning control of a nonlinear model of combustion instabilities. *IEEE Tr. Contr. Syst. Technol.* **7** (4), 424–436.

Lee, C., Kim, J., Babcock, D. & Goodman, R. 1997 Application of neural networks to turbulence control for drag reduction. *Phys. Fluids* **9** (6), 1740–1747.

Li, R., Borée, J., Noack, B. R., Cordier, L. & Harambat, F. 2019 Drag reduction mechanisms of a car model at moderate yaw by bi-frequency forcing. *Phys. Rev. Fluids* **4** (3), 034604.

Li, R., Noack, B. R., Cordier, L., Borée, J., Kaiser, E. & Harambat, F. 2018 Linear genetic programming control for strongly nonlinear dynamics with frequency crosstalk. *Archives of Mechanics* **70** (6), 505–534.

Li, Y., Cui, W., Jia, Q., Li, Q., Yang, Z., Morzyński, M. & Noack, B. R. 2021 Explorative gradient method for active drag reduction of the fluidic pinball and slanted ahmed body. *J. Fluid Mech.* **(revised) arXiv:1905.12036**.

Lorang, L. V., Podvin, B. & Le Quéré, P. 2008 Application of compact neural network for drag reduction in a turbulent channel flow at low reynolds numbers. *Physics of Fluids* **20** (4), 045104, arXiv: https://doi.org/10.1063/1.2904993.

Luchtenburg, D. M., Günther, B., Noack, B. R., King, R. & Tadmor, G. 2009 A generalized mean-field model of the natural and high-frequency actuated flow around a high-lift configuration. *Journal of Fluid Mechanics* **623**, 283–316.

Lusseyran, F., Pastur, L. & Letellier, C. 2018 Dynamical analysis of an intermittency in an open cavity flow. *Phys. Fluids* **20** (1).

Maehara, N. & Shimoda, Y. 2013 Application of the genetic algorithm and downhill simplex methods (nelder–mead methods) in the search for the optimum chiller configuration. *Applied Thermal Engineering* **61** (2), 433 – 442.

McKay, M. D., Beckman, R. J. & Conover, W. J. 1979 A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics* **21** (2), 239–245.

Mestiri, R., Ahmed-Bensoltane, A., Keirsbulck, L., Aloui, F. & Labraga, L. 2014 Active flow control at the rear end of a generic car model using steady blowing. *Journal of Applied Fluid Mechanics* **7**, 565–571.

Meyer, H. O. 2015 Applications of physics to archery. *arxiv* **arXiv:1511.02250 [physics.pop-ph]**.

Moreau, E. 2007 Airflow control by non-thermal plasma actuators. *Journal of Physics D: Applied Physics* **40** (3), 605–636.

Myose, R., Papadakis, M. & Heron, I. 1998 Gurney flap experiments on airfoils, wings, and reflection plane model. *Journal of Aircraft* **35** (2), 206–211, arXiv: https://doi.org/10.2514/2.2309.

Nair, A., Yeh, C.-A., Kaiser, E., Noack, B. R., Brunton, S. L. & Tiara, K. 2019 Cluster-based feedback control of turbulent post-stall separated flows. *J. Fluid Mech.* **875**, 345–375.

Nelder, J. A. & Mead, R. 1965 A simplex method for function minimization. *J. Comput.* **7**, 308–313.

Noack, B. R. 2019 Closed-loop turbulence control—From human to machine learning (and retour). In *Proceedings of the 4th Symposium on Fluid Structure-Sound Interactions and Control (FSSIC), Tokyo, Japan* (ed. Y. Zhou, M. Kimura, G. Peng, A. D. Lucey & L. Hung), pp. 23–32. Springer.

Noack, B. R., Afanasiev, K., Morzyński, M., Tadmor, G. & Thiele, F. 2003 A hierarchy of low-dimensional models for the transient and post-transient cylinder wake. *J. Fluid Mech.* **497**, 335–363.

Noack, B. R., Stankiewicz, W., Morzyński, M. & Schmid, P. J. 2016 Recursive dynamic mode decomposition of transient and post-transient wake flows. *J. Fluid Mech.* **809**, 843–872.

Parezanović, V., Cordier, L., Spohn, A., Duriez, T., Noack, B. R., Bonnet, J.-P., Segond, M., Abel, M. & Brunton, S. L. 2016 Frequency selection by feedback control in a turbulent shear flow. *J. Fluid Mech.* **797**, 247–283.

Paschereit, C. O., Schuermans, B. & Büche, D. 2003 Combustion process optimization using evolutionary algorithm. *ASME Turbo Expo 2003, collocated with the 2003 International Joint Power Generation Conference* **2**, 281–291.

Pastoor, M., Henning, L., Noack, B. R., King, R. & Tadmor, G. 2008 Feedback shear layer control for bluff body drag reduction. *J. Fluid Mech.* **608**, 161–196.

Pfeiffer, J. & King, R. 2012 Multivariable closed-loop flow control of drag and yaw moment for a 3d bluff body. In *6th AIAA Flow Control Conference*, pp. 1–14. Atlanta, Georgia, USA.

Plumejeau, B., Delprat, S., Keirsbulck, L., Lippert, M. & Abassi, W. 2019 Ultra-local model-based control of the square-back ahmed body wake flow. *Physics of Fluids* **31** (8), 085103, arXiv: https://doi.org/10.1063/1.5109320.

Podvin, B., Fraigneau, Y., Lusseyran, F. & Gougat, P. 2005 A reconstruction method for the flow past an open cavity. *Journal of Fluids Engineering* **128** (3), 531–540, arXiv: https://doi.org/10.1115/1.2175159.

Proctor, J. L., Brunton, S. L., Brunton, B. W. & Kutz, J. N. 2014 Exploiting sparsity and equation-free architectures in complex systems. *The European Physical Journal Special Topics* **223**, arXiv: https://doi.org/10.1140/epjst/e2014-02285-8.

Proctor, J. L., Brunton, S. L. & Kutz, J. N. 2016 Dynamic mode decomposition with control. *SIAM Journal on Applied Dynamical Systems* **15** (1), 142–161, arXiv: https://doi.org/10.1137/15M1013857.

Protas, B. 2004 Linear feedback stabilization of laminar vortex shedding based on a point vortex model. *Phys. Fluids* **16** (12), 4473–4488.

Rabault, J., Kuchta, M., Jensen, A., Réglade, U. & Cerardi, N. 2019 Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control. *J. Fluid Mech.* **865**, 281–302.

Raibaudo, C., Zhong, P., Noack, B. R. & Martinuzzi, R. J. 2019 Machine learning strategies applied to the control of a fluidic pinball. *Phys. Fluids* **32**, 015108.

Ren, F., Hu, H.-B. & Tang, H. 2020 Active flow control using machine learning: A brief review. *J. Hydrodyn.* **32** ((2)), 247–253.

Reuther, J., Jameson, A., Farmer, J., Martinelli, L. & Saunders, D. 1996 *Aerodynamic shape optimization of complex aircraft configurations via an adjoint formulation*.

Rossiter, J. 1964 Wind-tunnel experiments on the flow over rectangular cavities at subsonic and transonic speeds. , vol. 3438.

Roth, J. R., Sherman, D. M. & Wilkinson, S. P. 1998 Boundary layer flow control with a one atmosphere uniform glow discharge surface plasma. *Aerospace Sciences Meeting and Exhibit, AIAA-1998-328, 36th, January 12-15, 1998 Reno, NV* **38** (7), 1–28.

Roussopoulos, K. 1993 Feedback control of vortex shedding at low Reynolds numbers. *J. Fluid Mech.* **248**, 267–296.

Rowan, T. 1990 The subplex method for unconstrained optimization. PhD thesis, PhD thesis, Department of Computer Sciences, University of Texas.

Rowley, C. W. & Williams, D. R. 2006 Dynamics and control of high-Reynolds number flows over open cavities. *Ann. Rev. Fluid Mech.* **38**, 251–276.

Rowley, C. W., Williams, D. R., Colonius, T., Murray, R. M. & Macmynowski, D. G. 2006 Linear models for control of cavity flow oscillations. *Journal of Fluid Mechanics* **547**, 317–330.

Schoenauer, M., Sebag, M., Jouve, F., Lamy, B. & Maitournam, H. 1996 Evolutionary identification of macro-mechanical models. In *Advances in Genetic Programming II* (ed. P. Angeline & J. Kinnear), pp. 467–488. MIT Press, chap. 24.

Seifert, J. 2012 A review of the magnus effect in aeronautics. *Progress in Aerospace Sciences* **55**, 17–45.

Semaan, R., Kumar, P., Burnazzi, M., Tissot, G., Cordier, L. & Noack, B. R. 2016 Reduced-order modeling of the flow around a high-lift configuration with unsteady Coanda blowing. *J. Fluid Mech.* **800**, 71–110.

Semeraro, O., Bagheri, S., Brandt, L. & Henningson, D. S. 2011 Feedback control of three-dimensional optimal disturbances using reduced-order models. *Journal of Fluid Mechanics* **677**, 63–102.

Semeraro, O., Bagheri, S., Brandt, L. & Henningson, D. S. 2013 Transition delay in a boundary layer flow using active control. *Journal of Fluid Mechanics* **731**, 288–311.

Sipp, D., Marquet, O., Meliga, P. & Barbagallo, A. 2010 Dynamics and control of global instabilities in open-flows: A linearized approach. *Applied Mechanics Reviews* **63** (3), arXiv: https://doi.org/10.1115/1.4001478.

Sousanis, J. 2011 World vehicle population tops 1 billion units. `https://www.wardsauto.com/news-analysis/world-vehicle-population-tops-1-billion-units`, [Online; accessed 15-August-2011].

Tang, H., Rabault, J., Kuhnle, A., Wang, Y. & Wang, T. 2020 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), 053605, arXiv: https://doi.org/10.1063/5.0006492.

Thiria, B., Goujon-Durand, S. & Wesfreid, J. E. 2006 The wake of a cylinder performing rotary oscillations. *J. Fluid Mech.* **560**, 123–147.

Turing, A. M. 1950 Computing machinery and intelligence. *Mind* **49** (236), 433–460, arXiv: https://academic.oup.com/mind/article-pdf/LIX/236/433/30123314/lix-236-433.pdf.

Wahde, M. 2008 *Biologically Inspired Optimization Methods: An Introduction*. WIT Press.

White, D. R., McDermott, J., Castelli, M, Manzoni, L., Goldman, B. W., Kronberger, G., Jaśkowski, W., O'Reilly, U. & Luke, S. 2013 Better gp benchmarks: community survey results and proposals. *Genetic Programming and Evolvable Machines* **14**, 3–29.

Winkler, M., Becker, K., Doolan, C., Kameier, F. & Paschereit, C. O. 2012 Aeroacoustic effects of a cylinder-plate configuration. *AIAA Journal* **50** (7), 1614–1620, arXiv: https://doi.org/10.2514/1.J051578.

Wood, C. J. 1964 The effect of base bleed on a periodic wake. *Journal of the Royal Aeronautical Society* **68** (643), 477–482.

You, D. & Moin, P. 2008 Active control of flow separation over an airfoil using synthetic jets. *Journal of Fluids and Structures* **24** (8), 1349–1357, unsteady Separated Flows and their Control.

Zhang, P., Wang, J. & Feng, L. 2008 Review of zero-net-mass-flux jet and its application in separation flow control. *Science in China Series E: Technological Sciences* **51** (9), 1862–281X, arXiv: https://doi.org/10.1007/s11431-008-0174-x.

Zhou, Y., D., Fan, Zhang, B., , Li, R. & Noack, B. R. 2020 Artificial intelligence control of a turbulent jet. *J. Fluid Mech.* **897**, 1–46.

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences mécaniques et énergétiques, matériaux et géosciences (SMEMAG)

Titre: Contrôle par apprentissage automatique et méthodes de gradient appliqué aux écoulements cisaillés numériques et expérimentaux

Mots clés: contrôle d'écoulement, pinball fluidique, cavité ouverte, contrôle par apprentissage automatique (MLC), contrôle par programmation génétique (GPC), gradient-enriched machine learning control (gMLC).

Résumé: Nous proposons un algorithme rapide et automatisé de contrôle par apprentissage automatique enrichi de méthodes de gradients (gMLC) pour l'optimisation de lois de contrôle en boucle fermée. Notre méthodologie alterne entre l'exploration de l'espace de recherche et l'exploitation des gradients locaux, et généralise la programmation génétique (GPC) et l'Explorative Gradient Method (EGM). L'algorithme gMLC est implémenté et testé numériquement, par la stabilisation d'un système multi-entrées multi-sorties, le pinball fluidique et expérimentalement, par le contrôle de la cavité ouverte. Dans les deux cas, gMLC a construit des lois de contrôle en boucle fermée permettant les meilleures performances répertoriées. Nous démontrons aussi que les mécanismes de contrôle pour la cavité reposent effectivement sur la rétroaction à partir de la mesure de l'état. La comparaison entre gMLC et GPC est toujours à l'avantage de gMLC aussi bien en termes de vitesse de convergence que de qualité de la solution finale. Le gain en vitesse d'apprentissage est d'au moins un facteur 10, permettant d'envisager le contrôle d'expériences complexes avec, par exemple, un grand nombre d'entrées et de sorties ou des tests multi-paramètres pour assurer la robustesse de l'apprentissage. Enfin, deux codes sont mis en ligne en libre accès: `xMLC`, basé sur le contrôle par programmation génétique et `gMLC`, basé sur notre nouvel algorithme.

Title: Gradient-enriched machine learning control exemplified for shear flows in simulations and experiments

Keywords: flow control, fluidic pinball, open cavity, machine learning control (MLC), genetic programming control (GPC), gradient-enriched machine learning control (gMLC).

Abstract: As main contribution we propose a fast and automated gradient-enriched machine learning control (gMLC) algorithm to learn feedback control laws. The framework alternates between explorative and exploitive gradient- based iterations, generalizing genetic programming control (GPC) and the Explorative Gradient Method (EGM). The gMLC algorithm has been demonstrated both numerically, with the stabilization of a MIMO system, the fluidic pinball and experimentally, with the control of the open cavity. In both cases, gMLC successfully built closed-loop control laws allowing the best performances so far. We prove, in particular, that the mechanisms behind the control of the cavity rely effectively on feedback. The benchmark of gMLC with GPC on both problems, shows that gMLC outperforms GPC both in terms of convergence speed and final solution efficiency. An acceleration of at least a factor 10 between the GPC and gMLC has been achieved, allowing the control of many experiments, e.g., with a large number of inputs and outputs or multiple parameters testing for robustness. The two developed codes are both freely available online: `xMLC`, based on GPC and `gMLC`, based on our new algorithm.