Simulating the evolution and control of dynamical systems

HARVARD UNIVERSITY

Graduate School of Arts and Sciences

DISSERTATION ACCEPTANCE CERTIFICATE

The undersigned, appointed by the

Harvard John A. Paulson School of Engineering and Applied Sciences
have examined a dissertation entitled:

"Simulating the evolution and control of dynamical systems"

presented by: Shruti Mishra

Signature Chris H. Rycroft

Typed name: Professor C. Rycroft

Signature L Mahadevan

Typed name: Professor L. Mahadevan

Signature Katia Bertoldi

Typed name: Professor K. Bertoldi

October 8, 2020

Simulating the evolution and control of dynamical systems

A DISSERTATION PRESENTED BY

SHRUTI MISHRA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN THE SUBJECT OF

APPLIED MATHEMATICS

HARVARD UNIVERSITY

CAMBRIDGE, MA, USA

OCTOBER 2020

Simulating the evolution and control of dynamical systems

Abstract

This dissertation presents a series of investigations, into the evolution and control of dynamical systems, which are crucially made possible by the tools of numerical simulation. Our first study is described in Chapter 2, which considers the passive fluid dynamical system of a liquid drop impacting on a solid surface, in the presence of an intervening layer of air. Towards understanding the spatiotemporal evolution of this system, we develop a computational model that captures the essential dynamics, enabling us to carry out simulations that resolve the discrepancy between existing theory and experimental observations. Chapters 3–5 consider active dynamical systems, where we use reinforcement learning to develop control policies for agents acting to navigate their respective environments. In Chapter 3, we consider the locomotion of a segmented crawler, with minimal descriptions for its neuronal system, musculature and passive body mechanics, interacting with an external environment via friction. Embedding this system in a reinforcement learning algorithm, we recover the biological gait of peristaltic motion that ubiquitous in nature, both in the whole bodies of animals, as they move, and in their body parts, as they execute bodily functions, such as the digestion of food. In Chapter 4, we modify a state-of-the-art reinforcement learning algorithm to incorporate structure in the mechanics of a simulated quadruped, in the form of symmetry about the saggital plane. In this study, we observe a speed up in the learning of control policies via reinforcement learning when the stream of experiences is similar to that of a realistic embodied robot. Finally, in Chapter 5, we use multi-objective reinforcement learning to incorporate pre-existing knowledge of how to execute certain tasks while learning control policies for new tasks. We observe that the speed up in the learning of policies that can be achieved naturally depends on the relationship between the pre-existing policies and success metrics for the new task. We also observe that the learned control policies, when armed with pre-existing knowledge, bear closer resemblance to the pre-existing policies than would otherwise be the case.

Contents

Abstract iii

Acknowledgements vii

1 Introduction 1

References 3

2 Computing the viscous effect in early-time drop impact dynamics 5

Abstract 5

2.1 Introduction 6

2.2 Model 8

2.2.1 Physical problem and parameter regime 8

2.2.2 Mathematical model 10

2.2.3 Liquid domain and boundary conditions 11

2.2.4 Choice of domain 14

2.3 Numerical implementation 15

2.3.1 Projection method 15

2.3.2 Implementation of boundary conditions: ghost layers 17

2.3.3 Solution to the gas layer equations 17

2.3.4 Code implementation and parameter choices 19

2.4 Results and discussion 19

2.4.1 Initial dynamics and comparison with potential flow theory 19

2.4.2 Effect of liquid viscosity on drop profile 21

2.4.3 The role of surface tension 29

2.5 Conclusion 33

References 35

3 Coordinated crawling via reinforcement learning 40

Abstract 40

3.1 Introduction 41
3.2 Mathematical model of a crawler 42
3.2.1 Body mechanics 43
3.2.2 Neuromuscular dynamics 44
3.2.3 Body–substrate frictional interaction 45
3.2.4 Scaling and parameter choices 46
3.3 Reinforcement learning (RL) strategy 46
3.3.1 Formulation of observable state, action and reward 47
3.3.2 Experimental results: regularized and unregularized gaits 48
3.4 Discussion 51
References 52

4 Augmenting learning using symmetry in a biologically-inspired domain 55
Abstract 55
4.1 Introduction 56
4.2 Related work 57
4.3 Domain and tasks 57
4.4 Symmetric policy 58
4.5 Proposed algorithm 59
4.5.1 Policy evaluation 60
4.5.2 Policy improvement 60
4.6 Experiments 60
4.7 Discussion 61
References 62

5 Skill composition using multi-objective policy optimization 64
Abstract 64
5.1 Introduction 65
5.2 Scope 65
5.3 Literature on skill composition 66
5.4 Method 68
5.4.1 Teacher policies 68
5.4.2 Types of composition 68
5.4.3 Domains and tasks 69

5.5 Experiments 69
5.5.1 Humanoid domain, using a stand teacher 69
5.5.2 Humanoid domain, using stand and walk teachers 72
5.5.3 Humanoid domain, demarcating the observation space 73
5.5.4 Point mass domain 74
5.6 Discussion and future work 75
5.6.1 Principled selection of teachers 76
5.6.2 Policy composition in action-space 77
References 77

6 Overall summary and future directions 81
6.1 Leveraging structure in the evolution and control of dynamical systems 81
6.2 Role of numerical simulations 82

A Supplementary material: Computing the viscous effect in early-time drop impact dynamics 84
A.1 Calculations for the governing equation in the gas layer 84
A.1.1 Derivation of the gas layer pressure update equation 84
A.1.2 Numerical solution of the gas layer pressure 85
A.2 Tests of the simulation performance 86
A.3 Tests of the simulation accuracy 87
A.4 Additional model fits for lift-off time 89
References 94

B Supplementary material: Coordinated crawling via reinforcement learning 95
B.1 Parameter values 95
B.2 Unlearned gait 96
B.3 Videos 96

Acknowledgements

The research presented in the individual chapters of this dissertation has been done in collaboration with co-authors, and I have included references to publications in the main text. The responsibility for mistakes and oversights in these chapters is mine, and not of the co-authors, or of the groups of people I acknowledge, or sloppily forget to, here.

My ability to undertake and carry out the work in this thesis has been informed largely by my experiences before my tenure as a PhD student. Consequently, starting at *a* beginning, I'd like to thank my grandparents for encouraging me to be curious, guiding me to perform Turing tests (a concept I next encountered about two decades later), and gifting me with several encyclopaedias. I am thankful to my parents for their effort in enabling us to have and seek a good education, and dream freely, in a country and world where this is a privilege, more so for girls.

I have been taught by several wonderful teachers at several schools. In particular, I am grateful to the teachers, staff, students, banyan trees and bougainvillea arch(es?), for creating the environment of St. Mary's School, Pune. The values I was exposed to here provide guidance as I navigate this life, which includes this thesis. I am grateful to teachers at the Ellen Wilkinson School for Girls, in particular Mr. Hibbert and teachers in the Mathematics department, for allowing me the flexibility to carve my own path forward in my short time there, and advocating for me as I tried to do so.

I spent my first few years in higher education at the Department of Mechanical Engineering, Imperial College London. I enjoyed, and was challenged by, my experiences at Imperial. I am grateful to several members of faculty, in particular Prof. Balint, Prof. Hardalupas, Prof. Martinez-Botas, Dr. Marquis and Dr. Zaki, for being gracious with their time and for their guidance, and to my friends for their company. The confidence I gained in my technical abilities and

decision-making here has served me during periods of doubt.

My time as a PhD student was made interesting by several people, plants and animals. During varying periods of time spent at Harvard University, the Massachusetts Institute of Technology, the Kavli Institute for Theoretical Physics, and DeepMind, I have had the privilege of being exposed to novel research ideas, and witness a few of them flower. I am grateful to the many faculty, scientists, engineers, staff, and fellow students who made these experiences not only possible, but also intellectually engaging and fun. I thank the faculty members who served on my qualifying exam and thesis committees, Prof. Bertoldi, Prof. Mahadevan, Prof. Rubinstein and Prof. Rycroft, for feedback on research and on earlier versions of my dissertation. I am thankful to have had the opportunity to learn from Dr. Abdolmaleki and Prof. Precup, whose expertise, guidance and humanity made my two internships at DeepMind Montréal a uniquely cherished research experience.

I am grateful to my friends and family for their company, kindness, advice, support, whimsy . . . and more recently, for staying in touch through the stay-at-home measures during the Covid-19 pandemic.

Chapter 1

Introduction

A dynamical system is one that can be described by a set of state variables, $\mathbf{x} = \{x_1, \ldots, x_N\}$, that evolve in time. In science and engineering applications, these state variables $\{x_i\}, \forall i \in [1, \ldots, N]$, can typically be described in terms of a set of differential equations [8],

$$\dot{x}_i = f_i(x_1, \ldots, x_N), \qquad \forall i \in [1, \ldots, N], \tag{1.1}$$

where $f_i(x_1, \ldots, x_N)$ is a function that defines the temporal evolution of x_i. Equation 1.1 takes a very general form, and consequently the framework of dynamical systems is used to describe phenomena in a wide range of fields. Some examples of systems of interest and a possible set of their corresponding state variables are listed in Table 1.1.

Table 1.1: Examples of some dynamical systems and their possible state variables.

Dynamical system	**State variables**
Celestial mechanics	Positions and velocities of stars and planets, e.g., [6]
Chemical reactions	Concentrations of ions in a solution and heat flux from an external source, e.g., [4]
Financial markets	Prices of financial assets and relevant geopolitical events, e.g., [1]
Fluid mechanics	Pressure and velocity fields in a fluid medium, e.g., [2]
Population dynamics	Populations of species in an ecosystem, e.g., [7]

The solution to a dynamical system involves determining the value or functional form of the set of state variables $\mathbf{x}(t)$, where t is time. Initial work in the integration of a dynamical system through time was done using analytical methods, e.g., for the computation of the movement of a planet around a star using Newton's law of gravitation. The creation of simulation tools using computers allows for the resolution of dynamical systems that cannot be evolved using purely analytical methods. This is necessary for systems with a large number of interacting state variables, or systems with nonlinear terms $f_i(x_1, \ldots, x_N)$ in Equation 1.1, known as nonlinear dynamical systems.

This thesis is made up of four studies, all of which fall under the broad theme of simulating the evolution and control of a set of dynamical systems. The work in this thesis relies on advances in algorithms for simulating nonlinear dynamical systems in a wide variety of fields. Each of the subsequent chapters describes its particular dynamical system or class of dynamical systems.

Chapter 2 considers an autonomous dynamical system, where the evolution rule, i.e., $f_i(x_1, \ldots, x_N)$ in Equation 1.1, is represented using a mathematical and computational model, and the dynamical system evolves according to this model. Chapters 3–5 consider non-autonomous dynamical systems, where we as designers seek to manipulate the evolution of the dynamical system(s) in particular ways. Constrained by these design choices, the goal in Chapters 3–5 is to seek appropriate evolution rules for the system, i.e., manipulate $f_i(x_1, \ldots, x_N)$ in Equation 1.1 to meet some objective(s). In the subsequent paragraphs, we describe the specific systems in each of the chapters and the motivation for studying these.

Chapter 2 considers the passive fluid dynamical system of a drop impacting on a solid surface, and interacting with the surrounding air as it does so. Our study is motivated by a discrepancy between theoretical predictions which rely on simplification of the system [5], and experimental observations [3] made possible by recent advances in the spatiotemporal resolution at which images of the drop can be taken. We use simulation to study this system, are able to simulate processes beyond what is theoretically tractable, and recover experimental results. Simulation allows us control over physical parameters, allowing us to turn on and off certain mathematical terms to give insight into the relevant physical phenomena. We are also able to probe field variables that are not experimentally accessible, towards understanding the physics of this system.

Chapter 3 considers a neuromechanical system, inspired by the biology of soft-

bodied animals that move forward by crawling [12]. Our aim is to understand the control of the system – can a coordinated locomotion pattern be learned using artificial learning algorithms, and how does any resultant gait compare with observed biological gaits? In this chapter, we use a simple one-dimensional model for the mechanics of the body and of the neurons, and interaction with the environment is via friction. Combining the minimal model for the neuromechanics of the system with a simple reinforcement learning algorithm [9], we capture a method for controlling neuronal excitations as a function of proprioceptive feedback that imitates the biologically observed peristaltic gait. We also observe a different type of gait which results in a higher speed, but is less robust to noise in perception.

Chapters 4 and 5 consider simulated robotics systems executing locomotion tasks, inspired by the biomechanical systems of animals. In both of these chapters, our aim is to develop control policies for learning patterns of motion for moving through the environment at predefined speeds and in preferred directions. Our broad goal is to understand how agents might use knowledge about structure in their environment as part of their learning algorithms. To enable this, we provide the agents developing control policies via reinforcement learning with some domain knowledge. Using information about structure has the ability to speed up learning, as well as regularize the resultant control policies away from idiosyncrasies that are known to develop in artificial learning systems. We use locomotion tasks [10], developed using a physics simulation engine [11], to represent relevant parts of the locomotion dynamics. Working with simulated robotics systems allows us to alter and probe the learning algorithms in ways that are much harder, if not currently impossible, in animals.

In Chapter 4, we choose a biologically-inspired domain of a simulated quadruped [10]. We inform the agents about symmetry in their domain, and observe a speed up in the learning of control policies when they are in a regime where the experienced datastream arrives at a limited rate. In Chapter 5, we inform the agents of sub-tasks that bear resemblance to the overall task they are learning to solve. Here, we also observe a speed up in the learning of control policies, as well as a regularization of the behaviour towards the behaviours used to execute the sub-tasks. The broader context of these studies is: If embodied agents are able to use their experiences to derive invariances and structure in their environment, how does that speed up or shape the learning?

References

[1] Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3):637–654, 1973.

[2] Peter Constantin and Ciprian Foias. *Navier–Stokes equations*. University of Chicago Press, 1988.

[3] John M Kolinski, L Mahadevan, and Shmuel M Rubinstein. Lift-off instability during the impact of a drop on a solid surface. *Physical Review Letters*, 112(13):134501, 2014.

[4] Keith J Laidler. The development of the Arrhenius equation. *Journal of Chemical Education*, 61(6):494, 1984.

[5] Madhav Mani, Shreyas Mandre, and Michael P Brenner. Events before droplet splashing on a solid surface. *Journal of Fluid Mechanics*, 647:163–185, 2010.

[6] Henri Poincaré. *New methods of celestial mechanics*, volume 13. Springer Science & Business Media, 1992.

[7] Tomoo Royama. *Analytical population dynamics*, volume 10. Springer Science & Business Media, 2012.

[8] Steven H Strogatz. *Nonlinear Dynamics and Chaos*. Perseus Books, 1994.

[9] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT press, 2 edition, 2018.

[10] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. DeepMind control suite. Technical report, January 2018.

[11] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

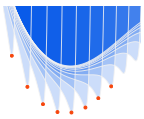
[12] Edwin Royden Trueman. *Locomotion of soft-bodied animals*. Edward Arnold, 1975.

Chapter 2

Computing the viscous effect in early-time drop impact dynamics

This chapter has been published as **Mishra, S., Rubinstein, S. M. and Rycroft, C. H., 2021. Computing the viscous effect in early-time drop impact dynamics.** [arXiv: 2108.11497]

Abstract



The impact of a liquid drop on a solid surface involves many intertwined physical effects, and is influenced by drop velocity, surface tension, ambient pressure and liquid viscosity, among others. Experiments by Kolinski et al. [24] show that the liquid–air interface begins to deviate away from the solid surface even before contact. They found that the lift-off of the interface starts at a critical time that scales with the square root of the kinematic viscosity of the liquid. To understand this, we study the approach of a liquid drop towards a solid surface in the presence of an intervening gas layer. We take a numerical approach to solve the Navier–Stokes equations for the liquid, coupled to the compressible lubrication equations for the gas, in two dimensions. With this approach, we recover the experimentally captured early time effect of liquid viscosity on the drop impact, but our results show that lift-off time and liquid kinematic viscosity have a more complex dependence than the square root scaling relationship. We also predict the effect of interfacial tension at the liquid–gas interface on the drop impact, showing that it mediates the lift-off behavior.

2.1 Introduction

The dynamics of a falling liquid drop as it impacts on a substrate depend on the initial conditions and physical properties of the liquid, the surrounding media and the substrate. Depending on the relevant physical parameters, the liquid drop may splash, spread or bounce upon impact with a solid substrate. The diversity of patterns produced by liquid drops impacting on a solid substrate in the presence of ambient air were first captured by Worthington [39], even before the invention of flash photography, through observations of patterns under the light of a spark produced by a break in an electric circuit when the drop hits the substrate. Worthington's drawings of these patterns sparked many investigations into the physical phenomena associated with drop impact, many of which are summarized in several review articles [34, 41, 20].

Physical parameters that are known to affect the dynamics of a drop upon impact with a substrate include the initial size and velocity of the drop [32], viscosity of the liquid [32], pressure and viscosity of the surrounding medium [40], interfacial tension between the liquid and its surrounding medium [35, 32], and the physical properties of the substrate [33, 34, 41, 20]. Experiments by Xu et al. [40] show that in the impact of a liquid drop on a solid substrate, ambient air pressure determines whether the drop eventually splashes, characterized by the ejection of a number of small drops into the air, or forms a thin film and spreads smoothly onto the surface. This suggests a role of the surrounding gaseous medium in determining the impact dynamics even before the drop makes contact with the solid substrate, motivating investigation into the role of an intervening gaseous layer in determining dynamics of a liquid drop as it approaches a solid substrate, prior to contact of the drop with the substrate. The series of theoretical investigations by Mandre, Mani and Brenner [30, 31, 29] considers an inviscid liquid drop falling towards a solid surface in the presence of an ideal gas, and predicts that the drop deforms due to a buildup of pressure in the gas trapped underneath the drop, prior to its contact with the solid substrate. Since these studies assume the fluid flow in the drop is inviscid, the velocity field is given by a potential flow. This substantially simplifies the analysis, allowing the dynamics to modeled using values of relevant field variables exclusively at the liquid–gas interface, without explicitly computing the fluid flow in the bulk of the drop. A schematic of the deformation of the drop is shown in Figures 2.1(a) and 2.1(b).

Experimental studies confirm the theoretical predictions of drops deforming on a layer of air between the liquid drop and solid substrate, prior to contact

between the drop and the substrate, and show measurements of the air layer underneath the impacting drop [25, 38, 16]. However, subsequent experimental work by Kolinski et al. [24] shows the effect of liquid viscosity on the evolution of the liquid–air interface even before contact with the substrate, which is not captured by the potential flow description of the evolution of liquid dynamics in previous theoretical work [30, 31, 29].

In this work, we are concerned with the dynamics of a liquid drop as it approaches a rigid, non-porous, solid surface, in the presence of an ambient gas. We modify the physical model of Mandre, Mani and Brenner by incorporating the viscosity of the liquid. Due to the additional viscous term in the governing equations for the liquid, the coupled interaction of the liquid and gas becomes theoretically intractable, and it is necessary to solve numerically for the fluid flow in the bulk of drop. Previous computational work on drop impact has been concerned with the dynamics of spreading [17] and splashing [21, 9], whereas we focus on the dynamics of the drop in its approach towards a solid surface, prior to contact.

We solve the incompressible Navier–Stokes equations in the drop, using a modern implementation of Chorin's projection method [11, 12] that incorporates improvements from Almgren, Bell, Collela and coworkers [6, 14, 5]. Our numerical model allows us to resolve the flow field and pressure in the drop, and examine the effects of many different physical parameters in ways (e.g., viscosity, surface tension) that would be difficult to do in experiment. We validate our model using theoretical results [30, 31, 29]. Using our model, we are able to recapitulate the square-root scaling with liquid viscosity of the lift-off time observed by Kolinski et al. [24]. However, our results show that the precise relationship between lift-off time and liquid viscosity is more complicated. We explore the dependence of lift-off time on surface tension, initial drop velocity, and drop radius.

In the following sections, we describe the physical model and parameter regime, followed by approximations made in the simulation domain. We then describe the results from our simulations, along with comparisons with previous theoretical and experimental studies. Our numerical results are consistent with theoretical calculations [30] as well as experimental results [24]. Our results predict the effect of viscosity and interfacial tension on the dynamics of drop impact.

2.2 Model

In this section, we first explain the physical set-up and specify the parameter regime considered in the rest of this work. Based on the physical set-up and parameter regime, we then describe the mathematical model. The mathematical model largely derives from the previous theoretical work of Mandre et al. [30, 31, 29]. We specify the predictions made by this model, experimentally-observed deviations from these predictions, and then the mathematical model used in our work. We then use this mathematical model to derive appropriate boundary conditions for the flow in the drop.

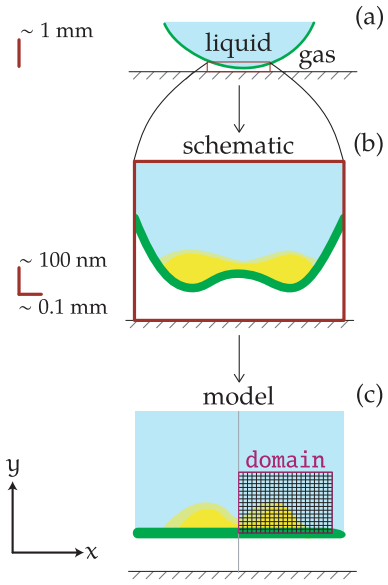


Figure 2.1: **Schematic of the physical and computational model.** (a) Relative locations of the falling drop, intervening gas layer and solid substrate. (b) A schematic zoomed into the region of interest, showing an exaggerated view of a deformed liquid–gas interface (green), and a schematic of the region where viscosity is important. (c) Control volume and grid for discretization constructed in the fluid domain.

2.2.1 Physical problem and parameter regime

The physical set-up is a drop of liquid falling towards a flat, solid surface in the presence of a surrounding gas. As the liquid drop falls towards the solid surface, it interacts with the surrounding gas. We are interested in modeling and simulating the coupled dynamics of the liquid and the gas before the liquid makes contact with the solid surface.

Figure 2.1(a) shows a schematic of the physical problem, indicating the relative locations of the liquid drop, the surrounding gas and the solid surface. The drop is initially spherical, with radius R, falling towards a horizontal solid surface at uniform vertical velocity V_0. The gas is initially at uniform pressure P_0. We specify the values of initial conditions and physical properties considered for the liquid and gas in Table 2.1. We use the subscripts l and g to denote the liquid drop and the gas, respectively, and 0 to denote the initial conditions.

Table 2.1: **Relevant initial conditions and physical parameters**, and their corresponding parameter regimes, which informs the mathematical model. The subscripts l and g denote the liquid drop and the gas, respectively, and 0 denotes initial conditions.

Quantity		Parameter range	Units
P_0	Initial gas pressure	$O(10^5)$	Pa
R	Initial drop radius	$O(10^{-3})$	m
V_0	Initial drop velocity	$O(10^{-1}$–$10^0)$	m/s
γ	Heat capacity ratio, gas	$O(1)$	–
ρ_g	Density of gas	$O(10^0)$	kg/m^3
ρ_l	Density of liquid	$O(10^3)$	kg/m^3
ν_g	Kinematic viscosity of gas	$O(10^{-5})$	m^2/s
ν_l	Kinematic viscosity of liquid	$O(10^{-6}$–$10^{-4})$	m^2/s

In the parameter regime specified in Table 2.1, previous theoretical work [30, 31, 29] argues that the relevant Reynolds number, based on the initial velocity V_0 and drop radius R, is $O\left(10^2\right)$, allowing for an inviscid consideration of the liquid. We refer to the mathematical model as the *potential flow model*, and the theoretical predictions arising from this mathematical model as the *potential flow theory*.

The potential flow theory predicts that, as the drop falls towards the solid surface, a build up of gas pressure underneath the drop causes the drop to deform in the middle, developing a *dimple*. The drop then spreads over a thin layer of gas that separates it from the substrate. The potential flow theory predicts scaling laws for the height of the gas layer. Subsequent experimental observations of the gas underneath the drop [25, 38, 16] show the existence of this dimple and the shape of the drop profile, similar to the predictions of the potential flow theory.

Experimental observations made by Kolinski et al. [24] in part of this parameter regime show that the falling drop first develops a dimple in the middle of the drop, as schematized by the liquid–gas interface shown in green in Figure 2.1(b).

This is consistent with the potential flow theory [30, 31, 29]. Subsequently in time, and prior to contact of the liquid–gas interface with the solid substrate, the shape of the interface depends on the kinematic viscosity of the liquid. Specifically, there is a critical time, τ, at which the minimum height of the drop from the substrate stops decreasing, and the *leading edge* of the drop begins to move away from the surface. This time, τ, is measured relative to the time when the drop would have made contact with the substrate if it were not deforming due to interactions with an intervening gas. Kolinski et al. [24] observe that $\tau \propto {\nu_l}^{1/2}$.

As a consequence of the discrepancy between the observations of Kolinski et al. [24], who definitively show the effect of liquid viscosity on the shape of the liquid–gas interface well before contact between the liquid and the substrate, and the potential flow theory, we are specifically interested in capturing the role of liquid viscosity, on these coupled dynamics. Based on the observations of Kolinski et al. [24], we modify the potential flow model to consider a liquid described by the full Navier–Stokes equations, rather than an inviscid approximation.

2.2.2 Mathematical model

The dynamics of an initially spherical drop are naturally described in terms of three-dimensional spherical polar co-ordinates, and those of the gas layer in terms of three-dimensional cylindrical polar co-ordinates. In this co-ordinate system, from the potential flow theory [30, 31, 29] and the experimental observations of air layer profiles under an impacting drop [25, 16, 38, 24, 23, 22], we use the simplifying approximation of no azimuthal variation in the flow field of the drop or the gas.

As in the potential flow model, we further simplify our consideration by approximating the drop as initially being an infinite cylinder, with the long axis perpendicular to the substrate, rather than spherical. With these assumptions and approximations, we can describe the dynamics of the liquid and gas using two-dimensional Cartesian co-ordinates (x, y). The initial height, $h(x, t = T_0)$, of the liquid–gas interface is described by a parabolic profile as

$$h(x, t = T_0) = H_0 + \frac{x^2}{2R}, \tag{2.1}$$

where H_0 is the initial height of the centre of the interface, i.e., at $x = 0$.

We model the flow in the liquid using the incompressible Navier–Stokes equa-

tions. The continuity equation for the liquid is

$$\nabla \cdot \mathbf{u}_l = 0, \tag{2.2}$$

and the momentum equation for the liquid is

$$\mathbf{u}_{l,t} + \mathbf{u}_l \cdot \nabla \mathbf{u}_l = -\frac{\nabla p_l}{\rho_l} + \nu_l \nabla^2 \mathbf{u}_l, \tag{2.3}$$

where $\mathbf{u}_l = (u_x, u_y)$, p_l and ρ_l are the velocity, pressure, and density of the liquid, respectively.

Based on the potential flow theory [30, 31, 29] and subsequent experimental observations [16, 38, 24], the gas layer is slender, with the horizontal length scale, L_x being much greater than the vertical length scale, L_y, allowing for an approximation of one-dimensional flow in the gas. The height, $h = h(x,t)$, of the gas layer is thus described by the compressible lubrication equation,

$$12\mu_g \left(\rho_g h\right)_t = \left(\rho_g h^3 p_{g,x}\right)_x, \tag{2.4}$$

and the equation of state is described by an adiabatic assumption,

$$\frac{p_g}{P_0} = \left(\frac{\rho_g}{\rho_0}\right)^\gamma. \tag{2.5}$$

The coupling of flows in the liquid and gas at their interface is done using Gibbs' condition for pressure,

$$p_l = p_g + \sigma h_{xx}, \tag{2.6}$$

and equality of shear stress in the liquid and gas at the liquid–gas interface is

$$\tau_l(x,h) = \tau_g(x,h). \tag{2.7}$$

Far enough away from the deforming effects of the intervening gas layer on the liquid drop, the pressure in the gas is the ambient pressure, $\lim_{x\to\pm\infty} p_g(x,t) = P_0$.

Given that the initial and boundary conditions are symmetric about $x = 0$, we modify our mathematical model to incorporate symmetry boundary conditions about $x = 0$. We do so by applying the boundary conditions $(u, v_x) = (0,0)$ in the liquid domain and $p_{g,x} = 0$ and $h_x = 0$ in the gas layer.

2.2.3 Liquid domain and boundary conditions

Having specified the physical problem in Section 2.2.1 and the mathematical model, with initial and boundary conditions, in Section 2.2.2, we now turn to

a description of the domain and boundary conditions that specify the flow in the liquid in a computationally tractable manner. The assumptions that lead to numerical approximations in this section are specified in Sections 2.2.1 and 2.2.2. We describe them here as needed.

2.2.3.1 *Rectangular domain with mass flux*

Following from the previous section, the liquid drop is described by a two-dimensional flow field. As the drop interacts with the gas layer, its shape changes. The drop begins to deform close to the middle of the interface, at $x = 0$. With this observation, we choose a region of interest, as shown in Figure 2.1(b), that stretches outwards from $x = 0$ and upwards from $y = h(x, t)$.

We make an additional approximation that the flow in the liquid at $y = h(x, t)$ can be approximated by the flow at $y = 0$. We can remove the spatial dependence because the liquid–gas interface is slender. We simulate the control volume with and without a temporal dependence $y = h(t)$, and do not observe a significant change in results. With these approximations, we are able to choose a rectangular, inertial control volume, stretching outwards from $x = 0$ and upwards from $y = 0$, whose size is chosen based on the relevant physical scales (Section 2.2.4). With the use of symmetry in the domain, as described in Section 2.2.2, we need to model only half of the control volume. We choose to model the right half, leading to the computational domain shown in Figure 2.1(c). In subsequent sections, we describe the boundary conditions at each of the boundaries in the computational domain.

2.2.3.2 *Treatment of the viscous term and bottom boundary condition*

Initially, the drop is moving at uniform vertical velocity, meaning that all spatial velocity gradients are zero, and $\nabla^2 \mathbf{u}_l = 0$. Deformation of the drop begins at $x = 0$. Our model consequently assumes that the shear stresses are small away from this region where the drop has deformed. In a two-dimensional flow, vorticity is introduced only through shear stresses at the boundary. The vorticity therefore spreads into the interior of the drop from the deformed interface. We thus model the drop as having a region close to $(x, y) = (0, 0)$, where viscosity is important, while far away from this region, $\nu_l \nabla^2 \mathbf{u}_l$ makes a negligible contribution to Equation 2.3. A schematic of the region where viscosity is important in the deformed drop is shown by yellow in Figure 2.1(b) and (c). Consequently, we model the domain as having one boundary where

viscous effects are important, which is the bottom boundary.

At the bottom boundary, we apply the boundary conditions specified in Equations 2.6 and 2.7. Equation 2.7 is enforced through a condition on the gradient of the horizontal velocity,

$$u_{l,y}(x,0) = \frac{\mu_g}{\mu_l} u_{g,y}(x,0). \tag{2.8}$$

2.2.3.3 *Boundaries in the interior of the drop: top and right boundary conditions*

According to our approximation of a rectangular domain, the bottom boundary of the liquid interacts with the gas layer. The other boundaries are chosen to be in the interior of the drop, away from the effect of vorticity from the bottom boundary, and can be treated as inviscid. With this approximation, at these boundaries, denoted as *interior boundaries*, we simplify the momentum equation for the liquid, Equation 2.3, by noting that the viscous term, $\nu_l \nabla^2 \mathbf{u}_l$, is relatively small.

Additionally, the velocity gradients in the liquid are initially zero. Away from the effects of the gas layer, the nonlinear advective term in the momentum equation, $\mathbf{u}_l \cdot \boldsymbol{\nabla} \mathbf{u}_l$ in Equation 2.3, will also be small. With these approximations, we obtain a simplified momentum equation for the liquid at these boundaries,

$$\mathbf{u}_{l,t} = -\frac{\boldsymbol{\nabla} p_l}{\rho_l}. \tag{2.9}$$

We use this simplified momentum equation to obtain velocity boundary conditions at the interior boundaries. Taking the divergence of Equation 2.9 and noting that the flow in the liquid is incompressible, we obtain $\nabla^2 p_l = 0$. Knowing the pressure at the bottom boundary, and approximating the liquid as a semi-infinite domain, we get an analytical expression for the pressure at the interior boundaries,

$$p_l(x,y) = \frac{1}{\pi} \int_{-\infty}^{\infty} p_l(s,0) \frac{y}{(x-s)^2 + y^2} \, ds. \tag{2.10}$$

Assuming that the domain is larger than the region where the pressure from the gas is non-zero, we numerically integrate over a finite length of the domain boundary to get the pressure at a particular (x, y) location along the interior boundaries. Given the pressure at the boundary, we integrate Equation 2.9 in time to obtain velocity boundary conditions at the interior boundaries. The size of the region where vorticity is significant determines the size of the domain, and the locations of the interior boundaries, which are calibrated in simulation.

2.2.4 Choice of domain

We simulate the coupled dynamics of the liquid and gas over the time interval $0 \leq t \leq t^{end}$, for a simulation domain of $0 \leq x \leq L$ and $0 \leq y \leq \beta L$, where β is the aspect ratio, for the liquid, and a domain of $0 \leq x \leq L$ for the gas. To set the size of the domain and duration of the simulation, we consider the initial dynamics as the drop falls toward the surface, and compare to the theoretical results of the potential flow theory [30]. We use the scaling results of the potential flow theory for the centre of the gas layer, $h(x = 0, t)$. The scaling analyses predict the height, H^*, at which the pressure buildup in the gas layer causes the drop to undergo significant deformation and develop a stagnation point at $x = 0$, forming a dimple.

According to potential flow theory, the compressibility of the flow in the gas is determined by the parameter

$$\epsilon = \frac{P_0 R St^{4/3}}{\mu_g V}, \tag{2.11}$$

which is the ratio of the initial gas pressure to the pressure that would have built up in the gas were treated as incompressible.

The results of Kolinski et al. [24], showing the relationship between ν_l and the drop profile, correspond to $\epsilon^{-1} < 1$, meaning that the flow in the gas can be considered to be incompressible. While our simulations model the full incompressible flow in the gas, we use this approximation to set the initial conditions. In this regime, we have the estimate

$$H^* = R St^{2/3}, \tag{2.12}$$

where $St = \mu_g/(\rho_l V R)$ is defined as the inverse of a modified Stokes number.[1] This scaling result is strictly valid for inviscid flow, $\nu_l = 0$, $Re \to \infty$, and no surface tension at the liquid–gas interface, $\sigma = 0$, $We \to \infty$. While our simulations incorporate liquid viscosity and surface tension, Equation 2.12 still provides a good estimate for the height at which the stagnation point forms, and is useful measure of the physical scales of interest. We therefore set the initial height to be a multiple of H^*, so that

$$H_0 = \tilde{H}_0 H^* = \tilde{H}_0 R St^{2/3}, \tag{2.13}$$

for a dimensionless constant $\tilde{H}_0$. Similarly, we set $t_{end} = \tilde{t}_{end} R St^{2/3}/V$, and based on the parabolic shape of the initial profile we choose $L = \tilde{L} R St^{1/3}$, where $\tilde{L}$ and $\tilde{t}_{end}$ are dimensionless constants.

[1] The inverse of the modified Stokes number, $St = \mu_g/(\rho_l V R)$, has properties of two different media, where the usual Stokes number is defined for a single medium.

2.3 Numerical implementation

In this section, we describe the numerical implementation of the mathematical model described in Section 2.2.2. Throughout the liquid domain, we compute and record the liquid velocity, $\mathbf{u}_l$, and pressure, p_l. In the gas layer, we compute and track the height, h, and pressure, p_g. A reader may skip the remainder of section without loss of continuity.

The flow-field variables ϕ are computed at discretized time steps, with simulation time $t = n\Delta t$, where Δt is the time step discretization used in the simulation, and n is an integer counter. Given values of field variables $\phi^{(n)} = \phi(t = n\Delta t)$, the goal of the simulation is to compute the field variables $\phi^{(n+1)}$. This section describes the numerical method for doing so, for the liquid flow-field variables, $\mathbf{u}_l$ and p_l, and gas flow-field variables, h and p_g.

2.3.1 Projection method

The core of the algorithm is based on Chorin's projection method [11, 12], with further developments in the computational techniques behind the fluid simulation method [6, 4]. In particular, the work of Almgren et al. [5] introduces the approximate projection method discretized using the finite-element method (FEM). The numerical methods are described in detail by Sussman et al. [37], Yu et al. [42, 43], and Rycroft et al. [36]. Here, we sketch the main features of these methods.

We solve for $\mathbf{u}_l$ and pressure p_l in the liquid domain by solving the momentum equation for the liquid, Equation 2.3, subject to incompressibility of the liquid, as specified in Equation 2.2. In describing the method for the solution of the field variables in the liquid domain, for the remainder of this section, we drop the subscript l.

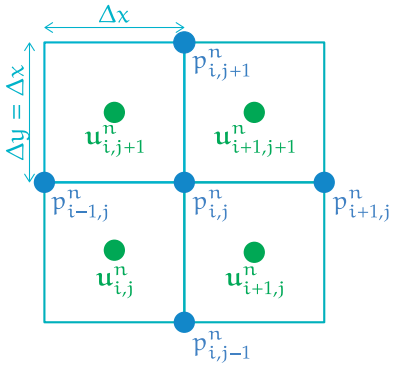
The field variables are defined on a two-dimensional grid, discretized into rectangular cells of size Δx by Δy, as shown in Figure 2.2(a). The velocities, $\mathbf{u}^n$, are located in the cell-centers, and the pressures, p^n, are located at the cell corners. To advance from step n to $n + 1$, we first compute an intermediate velocity, $\mathbf{u}^*$, according to

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} = -[\mathbf{u} \cdot \nabla \mathbf{u}]^{n+1/2} + \frac{\nu}{2}\left(\nabla^2 \mathbf{u}^n + \nabla^2 \mathbf{u}^*\right). \tag{2.14}$$

Here, the viscous stress terms, $\nu\nabla^2\mathbf{u}$ and $\nu\nabla^2\mathbf{u}^*$, are evaluated using a standard

five-point finite-difference stencil. Due to the presence of the $\mathbf{u}^*$ on the right hand side of Equation 2.14, it must be solved implicitly. We use a multigrid method to solve the resulting linear system.

(a) `discretization of flow in the liquid`



(b) `discretization of flow in the gas`

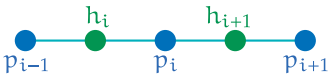


Figure 2.2: **Discretization of the field variables.** (a) The two-dimensional computational grid for the liquid. (b) The one-dimensional computational grid for the gas.

The advective term $[\mathbf{u} \cdot \nabla \mathbf{u}]^{n+1/2}$ is evaluated at the half-timestep $n + 1/2$ using the second-order Godunov upwinding scheme of Colella [14]. This is performed by extrapolating the velocity in each cell to midpoints of the four edges using a first-order Taylor expansion. Doing this requires that the gradient of the velocity is first evaluated at each cell center, which is done using the fourth-order monotonicity-limited scheme of Colella [13]. This scheme uses a five-point stencil in each coordinate, therefore requiring information from two grid points in each orthogonal direction.

After this procedure, each edge has two velocities, from the two adjacent cells. A Godunov upwinding procedure is then used to select one based on the direction of the velocity. An intermediate marker-and-cell (MAC) projection is then used to adjust the velocities to satisfy a discrete zero divergence criterion, so that the net mass flow out of each cell is zero [7, 37]. After this, the advective term can be accurately evaluated using centered finite differences of the upwinded edge-based velocities [37, 42, 43, 36].

The velocity at step $n + 1$ can be calculated from the intermediate velocity by

evaluating the pressure gradient term,

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^*}{\Delta t} = -\frac{1}{\rho}\nabla p^{n+1}, \tag{2.15}$$

but this requires knowledge of the pressure field p^{n+1}, and there is no explicit update equation for this field in the incompressible limit. To proceed, we take divergence of Equation 2.15 and enforce that $\nabla \cdot \mathbf{u}^{n+1} = 0$, according to Equation 2.2. Hence,

$$\nabla \cdot \left[\Delta t \left\{ \frac{1}{\rho}\nabla p^{n+1} \right\} \right] = \nabla \cdot \mathbf{u}^*, \tag{2.16}$$

which is a Poisson equation for the pressure that can be solved. Once p^{n+1} is known, Equation 2.15 can be used to evaluate $\mathbf{u}^{n+1}$ and complete the timestep from n to $n+1$. We solve Equation 2.16 using the FEM discretization of Almgren et al. [5]. Here, each pressure value at a cell corner has a corresponding bilinear hat function over the four neighboring grid cells [42, 43, 36]. Once the pressure is computed, the gradient in Equation 2.15 is evaluated using centered finite differences of the pressure field values.

2.3.2 Implementation of boundary conditions: ghost layers

The boundary conditions are implemented by means of ghost layers. Around the boundaries of the two-dimensional liquid domain, there are two layers of additional grid points. Two layers are required in order to evaluate the fourth-order monotonicity-limited derivatives arising from the advective term. The boundary conditions are specified in these layers of grid points. These conditions are then used to compute the necessary gradients in Equation 2.14.

The velocity boundary conditions at the top and right boundaries of the liquid domain, corresponding to the interior of the drop, and described in Section 2.2.3.3, are obtained through substituting the expression for pressure given by Equation 2.10 in the expression for velocity at these boundaries, Equation 2.9, and then using Simpson's rule to numerically integrate the pressure along these boundaries.

2.3.3 Solution to the gas layer equations

The field variables h and p_g for the gas layer are discretized in accordance with the discretization for the field variables for the liquid. Specifically, the pressure

field, p_g, is stored at grid points in the same locations along the horizontal axis as the pressure field in the liquid, as shown in Figure 2.2. The height, h, is stored at grid points in the same locations along the horizontal axis as the velocity field, $\mathbf{u}_l$, in the liquid.

As shown in Appendix A.1.1, substituting the equation of state, Equation 2.5, into the lubrication equation, Equation 2.4, yields

$$12\frac{\mu}{\gamma}hp_t + 12\mu h_t p = \frac{1}{\gamma}h^3 p_x p_x + 3h^2 h_x p p_x + h^3 p p_{xx}, \tag{2.17}$$

where the subscript, g, on the pressure has been dropped. We solve for this equation as follows:

1. p is known at the time step $t^{(n)} = n\,\Delta t$
2. h, h_t, and therefore h_x are known at time step $t^{(n)} = n\,\Delta t$
3. solve for p at time step $t^{(n+1)} = (n+1)\,\Delta t$
4. use p as a boundary condition for the flow in the liquid, to obtain $v = h_t$ at time step $t^{(n+1)}$

To numerically solve Equation 2.17, we discretize the spatial derivatives using second-order centered finite differences. To avoid a timestep restriction, we make use of the backward Euler method for discretizing the temporal derivative. Hence we obtain the discretized form,

$$\begin{aligned} 12\frac{\mu}{\gamma}\bar{h}\left(\frac{p_i^{n+1} - p_i^n}{\Delta t}\right) + 12\mu\bar{h}_t p_i^{n+1} = \frac{1}{\gamma}\bar{h}^3\left(\frac{p_{i+1}^{n+1} - p_{i-1}^{n+1}}{2\Delta x}\right)^2 \\ + 3\bar{h}^2\bar{h}_x p_i^{n+1}\left(\frac{p_{i+1}^{n+1} - p_{i-1}^{n+1}}{2\Delta x}\right) + \bar{h}^3 p_i^{n+1}\left(\frac{p_{i+1}^{n+1} - 2p_i^{n+1} + p_{i-1}^{n+1}}{\Delta x^2}\right), \end{aligned} \tag{2.18}$$

that can be solved for the gas pressure, p_i^{n+1}. In Equation 2.18, the terms $\bar{h}$, $\bar{h}_t$, and $\bar{h}_x$ must be evaluated at the location of p_i^{n+1}. Since the height field is staggered with respect to the pressure field, this is done via linear interpolation and centered differencing, so that

$$\bar{h} = \frac{h_i + h_{i+1}}{2}, \qquad \bar{h}_t = \frac{h_{t,i} + h_{t,i+1}}{2}, \qquad \bar{h}_x = \frac{h_{i+1} - h_i}{\Delta x}. \tag{2.19}$$

Due to the products of pressure terms on the right hand side of Equation 2.18, it is a nonlinear system of equations. We solve this using the Newton–Raphson method as described in Appendix A.1.2.

2.3.4 Code implementation and parameter choices

The simulations are performed using a custom code written in C++, using the OpenMP library for multithreading [1]. The code makes use of the Templated Geometric Multigrid (TGMG) library for solving the linear systems that arise when solving for the fluid [2]. Each timestep requires four linear system solves: (1) to apply the MAC projection, (2) to apply the approximate FEM projection, and (3,4) to solve for the x and y velocity components when treating the viscous stress term implicitly. The code accepts a text configuration file as input, which sets all of the physical parameters, and describes the computational domain. The code and sample text configuration files are available on GitHub [3]. The simulations are performed on a $M \times N$ grid for the liquid domain, and grid of length M for the gas layer. We choose the grid spacings to be equal, so that $\Delta x = \Delta y$. This implies that the aspect ratio, β, is equal to N/M.

Since the second derivative terms in both the liquid domain and the gas layer are handled implicitly, and the surface tension term in Equation 2.6 is small, there is no timestep restriction in the simulation that scales like Δx^2 [19]. We therefore choose a candidate timestep to satisfy $\Delta t = \zeta \Delta x$ for a dimensionless constant ζ, based on satisfying Courant–Friedrichs–Lewy conditions [15] for the advective terms in the liquid domain and gas layer.

The simulation outputs N_f frames over the duration of the simulation. Thus the time between frames is $t_f = t_{end}/N_f$. In general, an integer multiple of candidate timesteps will not exactly match t_f, so that $c\Delta t < t_f < (c+1)\Delta t$. Because of this, the actual timestep is slightly adjusted to $\Delta t' = t_f/(c+1)$ and $c+1$ timesteps are taken between frames. Several examples demonstrating the performance of the code are provided in Appendix A.2.

2.4 Results and discussion

2.4.1 Initial dynamics and comparison with potential flow theory

To validate our approach, we first simulate the initial dynamics and we compare with the scaling results of potential flow theory that were introduced in Section 2.2.4 for the height of the stagnation point H^* [30, 31, 29]. We use the baseline physical parameters given in Table 2.2. Since we only want to resolve

initial deceleration of the drop, and not the formation of the thin gas layer, we use the computational parameters in the third column of Table 2.3, which feature a relatively coarse numerical grid of size 2048×256. To compute H^*, we examine the sequence of height profiles from the simulation and find the time, t^*, when $h_t(0, t^*) = 0$, from which $H^* = h(0, t^*)$.

Table 2.2: **Baseline choices for the physical parameters used in the simulations.** These parameters are used throughout this chapter, with modifications to them noted in the text.

Quantity		Value	Units
P_0	Initial gas pressure	10^5	Pa
R	Initial drop radius	1.5×10^{-3}	m
V	Initial drop velocity	0.45	m/s
γ	Heat capacity ratio, gas	1.4	–
σ	Interfacial surface tension	0.072	N/m
ρ_g	Density, gas	1.2754	kg/m^3
ρ_l	Density, liquid	997.96	kg/m^3
ν_g	Kinematic viscosity, gas	1.506×10^{-5}	m^2/s

Table 2.3: **Baseline choices for the simulation parameters**, which are non-dimensional, except for ν_l. The first set of parameters are for computing the initial dynamics and value of H^* in Section 2.4.1. The second two sets are for the lift-off calculations in all other sections. The procedure for connecting the non-dimensional variables $\tilde{L}$, $\tilde{H}_0$, and $\tilde{t}_{end}$ to physical values is described in Section 2.2.4.

Quantity		Initial dynamics	Lift-off dynamics	
ν_l	Liquid viscosity, mm^2/s	all values	≤ 20	> 20
(M, N)	Grid dimensions	$(2048, 256)$	$(5120, 960)$	$(5120, 960)$
β	Aspect ratio	$1/8$	$16/3$	$16/3$
ζ	Timestep multiplier	8×10^{-3}	8×10^{-3}	8×10^{-3}
$\tilde{L}$	Domain width	30	30	45
$\tilde{H}_0$	Initial drop height	15	15	15
$\tilde{t}_{end}$	Duration	50	50	100
N_f	Number of frames	250	250	250

As described in Section 2.2.4, at low initial velocities the flow in the gas layer can be treated as incompressible, and the scaling result $H^* = RSt^{2/3}$ can be derived. Mani et al. [31] extend this analysis to look at higher initial velocities, where

compressibility of the gas becomes important. This happens at a height of

$$H_* = R(\mu_g V/RP_0)^{1/2}, \tag{2.20}$$

where the subscript, $*$, is used to distinguish from the stagnation height. By modeling the subsequent drop deceleration below H_* assuming gas compressibility, Mani et al. derive the result

$$H^*_{comp} = RSt^{2/3}\epsilon^{(2-\gamma)/(2\gamma-1)} \tag{2.21}$$

for the stagnation height.

We first performed sequences of simulations using low initial velocites over the range from $V = 0.15\,m/s$ to $V = 1.35\,m/s$, using four different liquid viscosities from $\nu_l = 10\,mm^2/s$ to $\nu_l = 300\,mm^2/s$. We also ran two sets of simulations with the surface tension set to half and zero its baseline value. Figure 2.3(a) shows a plot of $H^*/(RSt^{2/3})$ as a function of ϵ^{-1}, with these six sets of data points in the left half of the domain where $\epsilon^{-1} < 1$ and gas compressibility is not important. We see that $H^*/(RSt^{2/3})$ is approximately constant and is in agreement with Equation 2.12. To examine the trends more clearly, Figure 2.3(b) shows a zoomed-in plot of the same data. As expected, the best agreement is achieved for the smallest value of ν_l and for zero surface tension. At $\epsilon^{-1} < 1$, the values of $H^*/(RSt^{2/3})$ are slightly lower for the cases of larger ν_l and σ. This hints at the limits of the potential flow theory predictions [30, 31].

We also performed two sequences of simulations with higher initial velocities from $V = 1\,m/s$ to $V = 18\,m/s$ using $\nu_l = 10\,mm^2/s$, to examine the compressible regime. However, we note from Equation 2.13 that our usual choice for initial height scales according to $V^{-2/3}$, whereas from Equation 2.20, the height where compressibility is important scales according to $V^{1/2}$. To fully capture the compressible effects, the drop must start higher than H_*, and thus the usual initialization procedure is problematic for large V. For these simulations we therefore set H_0 at a fixed value based on using $V = 2\,m/s$ in Equation 2.13. We ran two sequences of simulations using the usual choice of $\gamma = 1.4$, and another with $\gamma = 1$; as shown in Figure 2.3, these results are in very good agreement with the scalings of $\epsilon^{1/3}$ and ϵ, respectively, that are expected from Equation 2.21.

2.4.2 Effect of liquid viscosity on drop profile

With the initial dynamics validated, we now turn attention to simulating the continued spreading of the liquid drop on a layer of gas, and the effect of

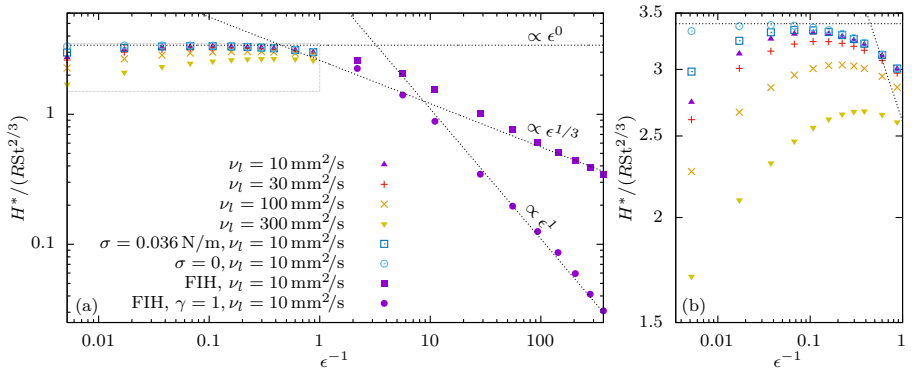


Figure 2.3: **A comparison of our simulation results with the potential flow theory.** (a) Plot of rescaled initial stagnation height $H^*/(RSt^{2/3})$ as a function of the inverse of the dimensionless compressibility parameter $\epsilon = P_0 RSt^{4/3}/\mu_g V$, over a range of values for the liquid viscosity, ν_l, surface tension, σ, and gas parameter, γ. We use the baseline parameters from Table 2.2, unless specified otherwise. By default, the drop starts from a height H_0 scaled according to Equation 2.13. For $\nu_l = 10\,\mathrm{mm}^2/\mathrm{s}$, to account for compressibility effects, data from two simulation sequences that use a fixed initial height (FIH) based on substituting $V = 2\,\mathrm{m/s}$ into Equation 2.13 are shown. (b) Zoomed-in plot of the same data, showing the region bounded by the dotted gray rectangle in panel (a).

viscosity on the evolution of the liquid–gas interface. We use the same baseline physical parameters given in Table 2.2, but since we must now accurately resolve the gas layer as it becomes very thin, we increase the simulation resolution as shown in the fourth and fifth columns of Table 2.3. Furthermore, since the deviation of the interface from the solid surface happens later for high viscosities [24], we use a larger domain and longer duration when $\nu_l > 20\,\mathrm{mm}^2/\mathrm{s}$, as indicated in the fifth column of Table 2.3. We begin by using the baseline initial drop velocity of $V = 0.45\,\mathrm{m/s}$ to match the experiments of Kolinski et al. [24].

Figure 2.4 shows the height profiles for three different simulations using liquid viscosities of $\nu_l = 10\,\mathrm{mm}^2/\mathrm{s}$, $\nu_l = 32\,\mathrm{mm}^2/\mathrm{s}$, and $\nu_l = 100\,\mathrm{mm}^2/\mathrm{s}$. Panels (a), (b), and (c) show a large-scale view, where the initial parabolic profile approaches the surface, begins to decelerate and deforms to create the dimple. After the drop continues to fall, a thin layer of gas from $x \gtrapprox 200\,\mu\mathrm{m}$ is created and spreads outward. In all three simulations, the height profiles have a front that sweeps outward as more liquid approaches the surface.

Figure 2.5 shows snapshots of the pressure and vorticity in the liquid domain for the simulation with with $\nu_l = 10\,\mathrm{mm}^2/\mathrm{s}$. At $t = 24.02\,\mu\mathrm{s}$, the drop is still approaching the surface. The pressure builds up close to $x = 0$. Since the liquid near $x = 0$ is decelerated faster, this creates a region of negative vorticity over

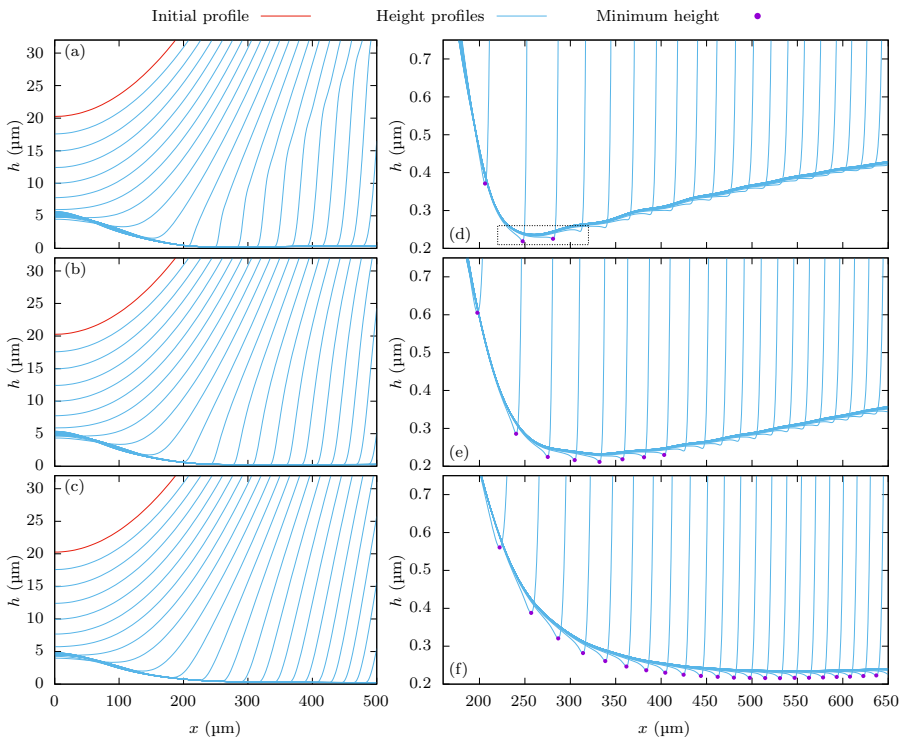


Figure 2.4: **Profiles of the height** $h(x, t)$ **of the gas layer.** The profiles are spaced 6.004 µs apart, corresponding to an integer multiple of the frame output interval t_f described in Section 2.3.4, for liquid viscosities of (a) $\nu_l = 10\,\mathrm{mm}^2/\mathrm{s}$, (b) $\nu_l = 32\,\mathrm{mm}^2/\mathrm{s}$, and (c) $\nu_l = 100\,\mathrm{mm}^2/\mathrm{s}$. Panels (d), (e), and (f) show the same data as (a), (b), and (c), respectively, but with a smaller range of h to highlight the lift-off behavior. For each profile, the global minimum, which follows the leading tip, is also plotted on the curves; once the global minimum is no longer at the leading tip, it is no longer plotted. The dashed box in panel (d) marks a further zoomed-in region shown in Figure 2.6.

$0 \lessapprox x \lessapprox 250\,\mu\mathrm{m}$. By $t = 72.05\,\mu\mathrm{s}$, the thin layer of gas has formed, and the position of the front from Figure 2.4(a) is marked with a small triangle. There is a region of large positive pressure behind the front, a small region of negative pressure ahead of it. The contour of zero vorticity follows the front as it moves outward to $t = 96.07\,\mu\mathrm{s}$.

Figure 2.4(d–f) shows zoomed-in plots of the height profiles in the thin layer for the three simulations. Behind the front, the height profiles align on top of each other, and trace out relatively stable envelopes, appearing as thick blue lines in Figure 2.4(d–f). In all three cases the envelopes initially slope downward before curving upward, indicating the deviation of the liquid–gas interface away from the solid surface, prior to contact between the solid and the liquid. This is the

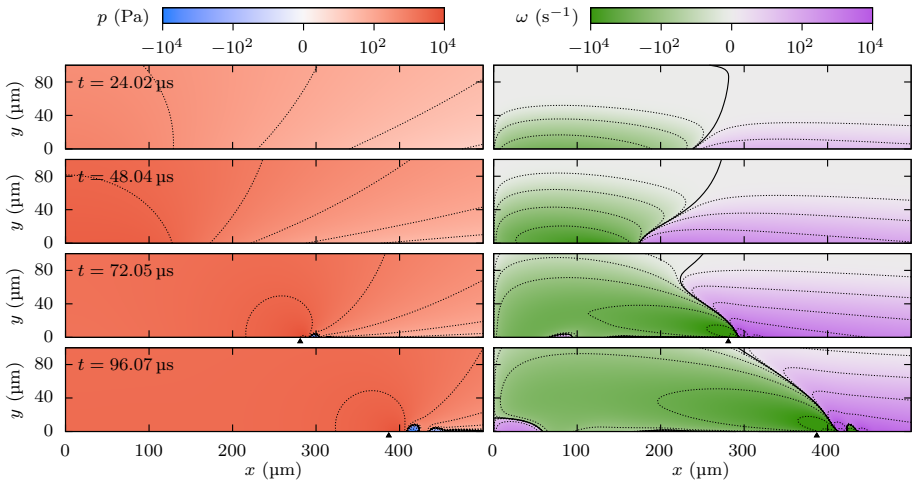


Figure 2.5: **Snapshots of pressure, p, (left) and vorticity, $\omega = [\nabla \times \mathbf{u}]_3$ (right), in a portion of the liquid domain** for a simulation with liquid viscosity $\nu_l = 10\,\mathrm{mm}^2/\mathrm{s}$. The field values exhibit large variations in scale and also switch sign, so a nonlinear mapping $f(\alpha) = \frac{1}{\log 10} \sinh^{-1} \frac{\alpha}{2}$ is used to create the color maps. The thick black lines indicate the zero contour. The thin black lines show contours for $\pm 10^{n/2}$ Pa for pressure and $\pm 10^n\ \mathrm{s}^{-1}$ for vorticity, where $n \in \mathbb{N}_0$. For $t = 72.05\,\mu\mathrm{s}$ and $t = 96.07\,\mu\mathrm{s}$ the position of the front (given by the local minimum of the profiles in Figure 2.4(a,d)) is marked with a triangle.

lift-off phenomenon [24]. Lift-off occurs more quickly for lower viscosities, and the envelope slopes upward faster. These results are in close agreement with the experimental results of Kolinski et al. [24].

We now quantify the lift-off time and position. While the envelopes in Figure 2.4(d–f) are relatively well defined, the height profiles shift slightly in over time, make it difficult to precisely define the moment of lift-off. However, close inspection of Figure 2.4(d–f) shows that in all cases, the height profiles have a leading tip that dips slightly below the envelope that forms. We found this to be a well-defined feature that occurs universally across all of our simulations. The tip position can be determined as the global minimum of the height profile at each time, and provides a way to precisely define when lift-off occurs, at the moment when the leading tip starts to rise.

Figure 2.6(a) shows a further zoomed-in plot of the height profiles during lift-off for the case of $\nu_l = 10\,\mathrm{mm}^2/\mathrm{s}$. Once the leading tip has risen sufficiently, then it is no longer the global minimum. However, since lift-off has always occurred by the time the leading tip is rising, this does not cause any difficulty in identifying the lift-off time. It is natural to consider whether the leading tip is a real feature or a numerical artifact that emerges from discretization error

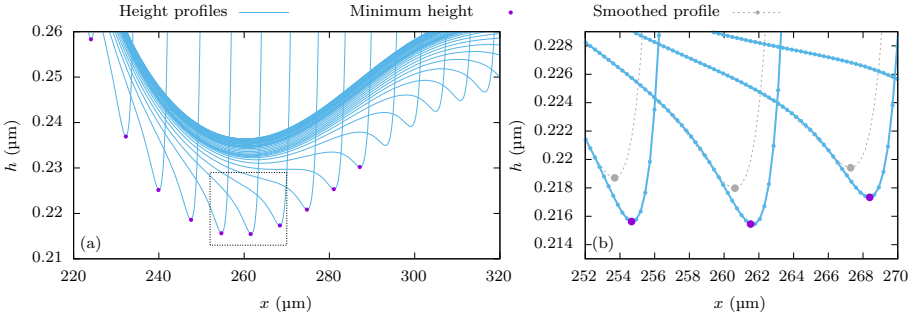


Figure 2.6: **Zoomed-in plots of the height of the gas layer** at intervals spaced 1.2009 µs apart for liquid viscosity $\nu_l = 10\,\mathrm{mm}^2/\mathrm{s}$. Panel (a) shows the region marked by the dashed box in Figure 2.4(d). For each profile, the global minimum, which follows the leading tip, is also plotted on the curves; once the global minimum is no longer at the leading tip, it is no longer plotted. Panel (b) shows the region marked by the dashed box in panel (a). In panel (b), the small blue circles indicate the computational grid. The gray dashed lines show the profiles with Gaussian smoothing applied, and the gray circles show the global minima of the smoothed lines.

and limited resolution. To address this, Figure 2.6(b) shows further zoomed-in region, depicting the leading tip in detail. Here, the blue circles show individual simulation grid points. The grid spacing is substantially smaller than the width of the leading tip, indicating that it is a real feature. Further numerical tests of accuracy are provided in Appendix A.3.

The width of the tip is governed by surface tension. While many of our simulations use the baseline value of $\sigma = 0.072\,\mathrm{N/m}$ from Table 2.2, we also consider reduced surface tension where the tip becomes sharper. In this case, there can be numerical difficulties with in identifying the minimum due to per-gridpoint variations the height profile. To create a scheme for identifying the minimum across all simulations, we therefore smooth the height profile using a Gaussian kernel with length scale 1.2 µm. This results in a minimal alteration in the leading tip position (Figure 2.6(b)), but improves robustness when analyzing the simulations. Numerically, the global minimum is found by identifying the local minima of the cubic interpolant of the smoothed profile, and selecting the smallest one.

Figure 2.7 shows the trajectories of the leading tip for a range of liquid viscosities from $\nu_l = 2.5\,\mathrm{mm}^2/\mathrm{s}$ to $\nu_l = 160\,\mathrm{mm}^2/\mathrm{s}$. The moment of lift-off is indicated on each trajectory, by identifying the global minimum of each curve. Two regimes are visible. For $\nu_l \lessapprox 20\,\mathrm{mm}^2/\mathrm{s}$, increasing viscosity results in the trajectory reaching a lower value of h, and the lift-off position moving slightly outward.

For $\nu_l \gtrapprox 20\,\mathrm{mm}^2/\mathrm{s}$, increasing viscosity results in the trajectory reaching a higher value of h, and the lift-off position moves outward substantially. Small undulations in the trajectories are visible for $\nu_l \gtrapprox 20\,\mathrm{mm}^2/\mathrm{s}$, which arise due to capillary waves in the height profile. This can have a substantial effect on the measured lift-off time, depending on which undulation achieves the global minimum. For example, there is a large difference between $\nu_l = 20\,\mathrm{mm}^2/\mathrm{s}$ and $\nu_l = 25\,\mathrm{mm}^2/\mathrm{s}$.

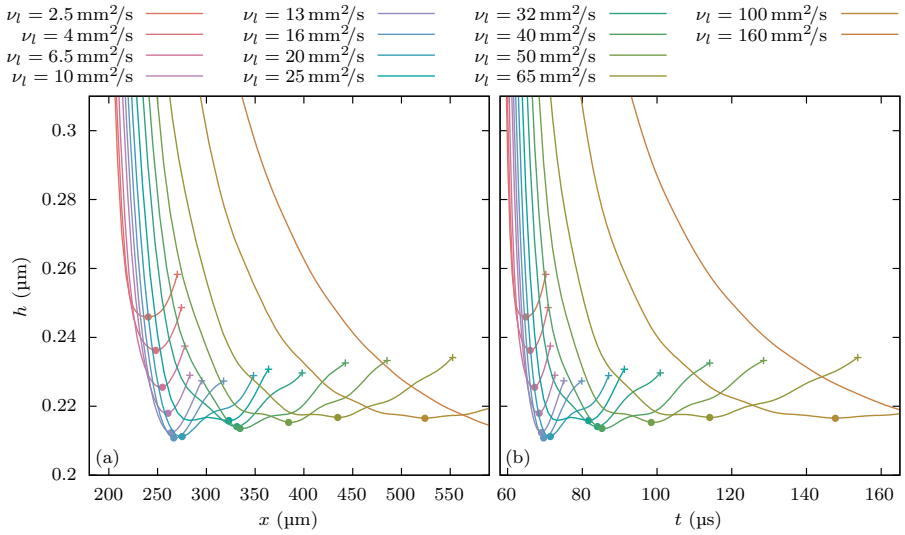


Figure 2.7: **Trajectories of the global minimum of the drop profile** in (a) the (x, h) plane, and (b) the (t, h) plane for the baseline parameters with initial drop velocity $V = 0.45\,\mathrm{m/s}$, for a range of different liquid viscosities. For each trajectory, the filled circle indicates where the global minimum reaches its lowest point, which we define as when lift-off occurs. Each cross indicates when the global minimum no longer marks the leading tip.

In Figure 2.8, we show trajectories for the case when the initial drop velocity is doubled to $V = 0.9\,\mathrm{m/s}$. The scale of h is smaller in the plot, but the relative amplitude of the capillary waves is increased. As a result, they have a noticeable effect of lift-off times at lower viscosities, with a large difference in lift-off time between $\nu_l = 10\,\mathrm{mm}^2/\mathrm{s}$ and $\nu_l = 13\,\mathrm{mm}^2/\mathrm{s}$.

We now compare to the experimental finding of Kolinski et al. [24] that the lift-off time is proportional to $\nu_l^{1/2}$. The lift-off time, τ, in this previous work is defined relative to a time origin of when the drop would reach $h = 0$ in the absence of an intervening layer of air. Here, since the initial height, h_0, and velocity, V, are known, we compute the time origin as $t_0 = h_0/V$. By contrast, Kolinski et al. [24] were not able to directly view h_0, since the drop can only be observed once it enters an evanescent field of height $h_{ev} = 1\,\mu\mathrm{m}$. Instead,

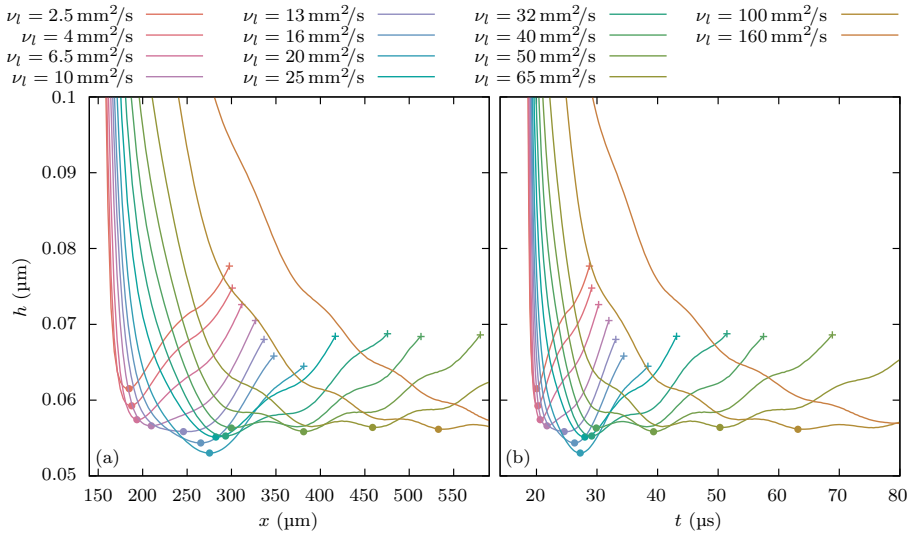


Figure 2.8: **Trajectories of the global minimum of the drop profile** in (a) the (x, h) plane, and (b) the (t, h) plane for the baseline parameters with initial drop velocity $V = 0.9\,\mathrm{m/s}$, for a range of different liquid viscosities. For each trajectory, the filled circle indicates where the global minimum reaches its lowest point, which we define as when lift-off occurs. Each cross indicates when the global minimum no longer marks the leading tip.

they measured the first position and time (h', r', t') when the drop enters the evanescent field, and assume that the drop is undeformed at this position, so that $h' = h_0 - Vt' + r'^2/(2R^2)$. Hence h_0 can be calculated, and therefore the time origin is $t_0^{\mathrm{alt}} = h_0/V - t' + r'^2/(2R^2V)$.

Figure 2.9 shows the lift-off times τ as a function of viscosity for seven different values of initial velocity V. The data points from Kolinski et al. [24] are also plotted. Even though the experiments of Kolinski et al. are performed in a three-dimensional axisymmetric configuration, there is good quantitative agreement with the two-dimensional simulation results.

Overall, the results are consistent with the $\nu_l^{1/2}$ scaling result, but the additional precision provided by the simulations reveals a more complicated relationship between ν_l and τ. For each value of V, the data points exhibit some fluctuations, which are due to the undulations visible in Figure 2.7 where the global minimum defining the lift-off time may abruptly change. While a $\nu_l^{1/2}$ scaling appears consistent with the high impact velocities where $V \gtrapprox 0.75\,\mathrm{m/s}$, the data for low impact velocities looks better fit to ν_l^{η}, where there are two different values of η with a change at $\nu_l \approx 20\,\mathrm{mm}^2/\mathrm{s}$. This is also consistent with the two different types of behavior observed in Figure 2.7.

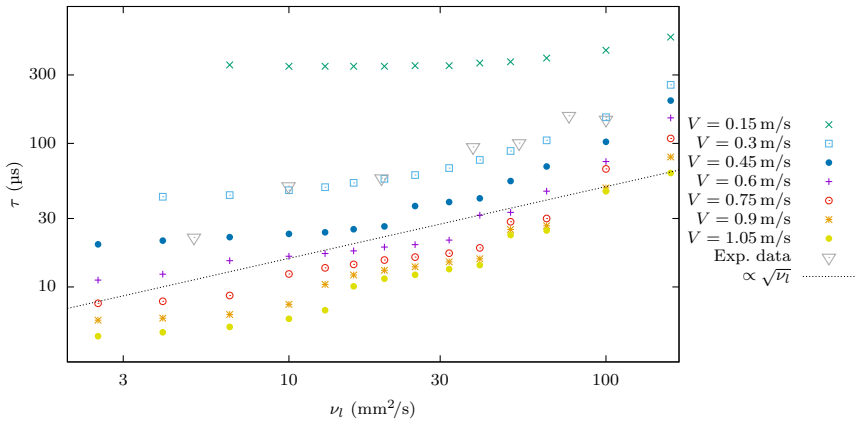


Figure 2.9: **Lift-off time τ as a function of liquid viscosity ν_l, for a range of initial drop velocities V.** The experimental data is taken from Figure 4(a) of the paper by Kolinski et al. [24], which used $V = 0.45$ m/s. Three data points are omitted from the plot for low V and low ν_l due to physical difficulties with running the simulation; see Section 2.4.3.

The slope of data points for small values of ν_l in Figure 2.9 is strongly affected by the choice of time origin, and may affect the conclusions about the relevant exponents. To investigate this further, we replotted the data using t_0^{alt} as the time origin; this resulted in small shift upwards of the data (since $t_0^{alt} < t_0$ in all cases), but did not affect the overall patterns. As a third approach, we defined a time origin t_0^{dat} directly from the data, by finding the best fit to the model $\tau_{dat} = t - t_0^{dat} = \gamma(\nu_l/\nu_{sc})^\alpha$ for the three free parameters $(t_0^{dat}, \gamma, \alpha)$. Here $\nu_{sc} = 20\,\text{mm}^2/\text{s}$ is chosen as an arbitrary viscosity scale. Specifically, for the data points $(\nu_{l,k}, t_k)$, we minimized the residual

$$S(t_0^{dat}, \gamma, \alpha) = \frac{1}{2}\sum_k \left(\log\gamma + \alpha \log\frac{\nu_{l,k}}{\nu_{sc}} - \log(t_k - t_0^{dat})\right)^2. \tag{2.22}$$

We used the L-BFGS-B algorithm for bound-constrained nonlinear optimization [10], and enforced $t_0^{dat} < 0.999 t_{min}$ and $\alpha > 0$, where $t_{min} = \min_k\{t_k\}$. We started the minimization using multiple initial guesses with $t_0^{dat} \in [0.7t_{min}, 0.9t_{min}]$ and $\alpha \in [0.4, 1.2]$. For each pair (t_0^{dat}, α) the value of γ is set so that the bracketed expression in Equation 2.22 is zero for the $\nu_l = 20\,\text{mm}^2/\text{s}$ data point, so that the initial guess should be close to the minimum of S.

Figure 2.10 shows a replotting of the data relative to this time origin. The data for $V = 0.15$ m/s is omitted from this plot since the lift-off times are non-monotonic for the smallest values of ν_l. For each other value of V, the minimization procedure converges to a single unique solution for all initial guesses. With this definition of time origin, all of the data is more consistent with a linear scaling

relationship as opposed to the square root relationship in Figure 2.9. Panels (b), (c), and (d) show the values of the fitted parameters, and demonstrate that $\alpha \in [1.02, 1.28]$ in all cases. The average residual over the six different values of V is $\bar{S} = 0.2536$. If the exponent is constrained to $\alpha = \frac{1}{2}$ to match the exponent of Kolinski et al. [24] then the average residual increases to $\bar{S} = 0.3855$ and the data points exhibit an upward curve a log–log plot that systematically deviates from a power law. Constraining the exponent to $\alpha = 1$ results in $\bar{S} = 0.2805$ and a better fit to the model that is similar to the three-parameter fit shown in Figure 2.10. See Appendix A.4 for additional discussion and parameter fits. These results highlight that any theoretical analysis of the relationship between τ and ν_l would need to take into account the sensitivity of defining the time origin.

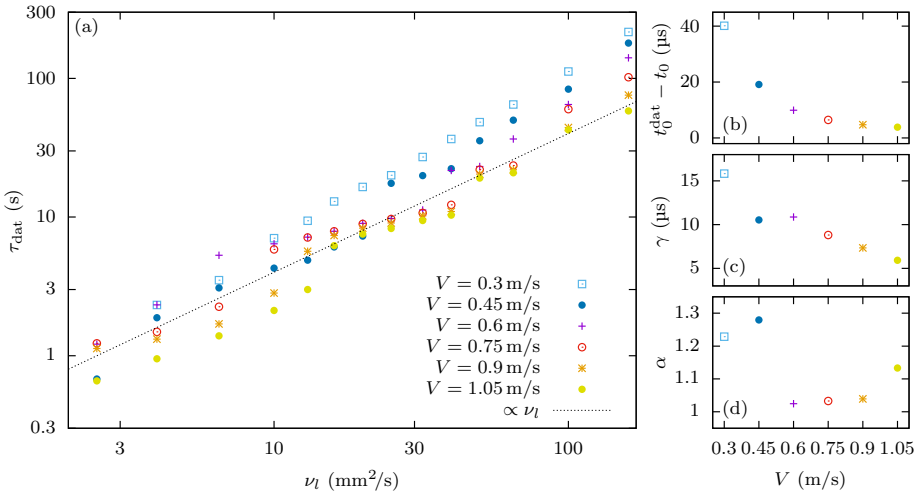


Figure 2.10: (a) **Lift-off time, τ_{dat}, as a function of liquid viscosity, ν_l**, over a range of values for the initial drop velocity, V, using the alternative time origin definition based on minimizing the three parameter residual function in Equation 2.22. (b–d) best fits of the parameters $t_0^{dat} - t_0$, γ, and α for different values of V.

Figure 2.11(a) shows lift-off times (relative to t_0) for three different drop radii: the original value of $R = 1.5$ mm, as well as half and double this value. Increasing the radius increases the lift-off time, similar to the effect of lowering velocity in Figure 2.9.

2.4.3 The role of surface tension

The simulations allow us to change the surface tension in ways that would be difficult to do experimentally, to investigate the importance of this physical

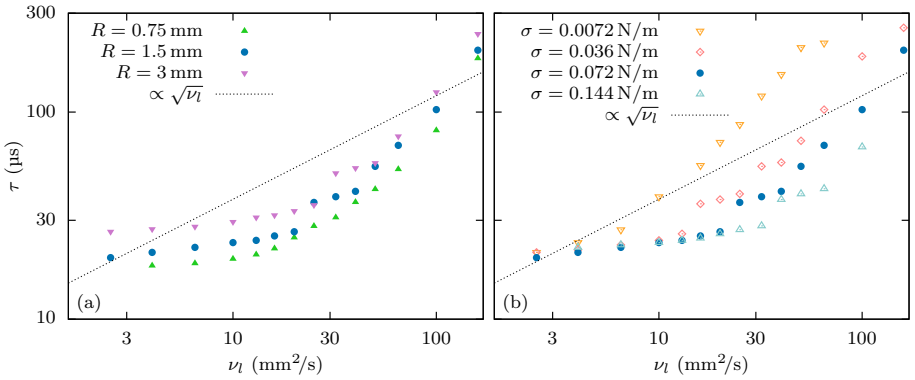


Figure 2.11: **Lift-off time as a function of liquid viscosity**, for a range of (a) different drop radii, and (b) different surface tension values. The data point for $(\nu_l, R) = (2.5\,\mathrm{mm}^2/\mathrm{s}, 0.75\,\mathrm{mm})$ is omitted due to physical difficulties with running the simulation (Section 2.4.3). The data points for $\sigma = 0.0072\,\mathrm{N/m}$ and $\nu_l \geq 100\,\mathrm{mm}^2/\mathrm{s}$ are omitted because lift-off does not occur over the simulation duration.

effect. Changing the surface tension allows us to suppress or accentuate the capillary waves in the liquid–gas interface, as shown in Figure 2.7, to investigate the effect of surface tension on the phenomenon of lift-off. Following the same procedure as in Section 2.4.2, we calculated the lift-off times for $\sigma = 0.0072\,\mathrm{N/m}$, $\sigma = 0.036\,\mathrm{N/m}$ and the $\sigma = 0.144\,\mathrm{N/m}$, corresponding to a tenth, half, and double the usual surface tension values, respectively. Figure 2.11(b) shows that as surface tension is reduced, the lift-off times increase markedly. For the case of $\sigma = 0.0072\,\mathrm{N/m}$ lift-off is eliminated for large values of ν_l, with the leading tip continuing to decrease over the course of the simulation.

Figure 2.11(b) strongly suggests that surface tension is important in creating lift-off. Reducing from $\sigma = 0.036\,\mathrm{N/m}$ to $\sigma = 0.0072\,\mathrm{N/m}$ almost doubles the lift-off times in most cases, and one can ask whether lift-off will be completely eliminated in the limit as surface tension vanishes. To examine this, we ran a sequence of simulations with $\sigma = 0$, with several representative examples for $\nu_l = 10\,\mathrm{mm}^2/\mathrm{s}$, $\nu_l = 32\,\mathrm{mm}^2/\mathrm{s}$, and $\nu_l = 100\,\mathrm{mm}^2/\mathrm{s}$ shown in Figure 2.12. This is a difficult limit to probe in our simulations, since as discussed for Figure 2.6, surface tension regularizes the leading tip. Figure 2.13 shows close-ups of the profiles for $\nu_l = 10\,\mathrm{mm}^2/\mathrm{s}$, indicating that leading tip becomes as sharp as a single grid point, and the minima of the profiles can fluctuate non-monotonically depending on exactly how the tip aligns with the computational grid. However, in this case the profile smoothing procedure introduced in Section 2.4.2 is sufficient to extract smooth leading tip trajectories.

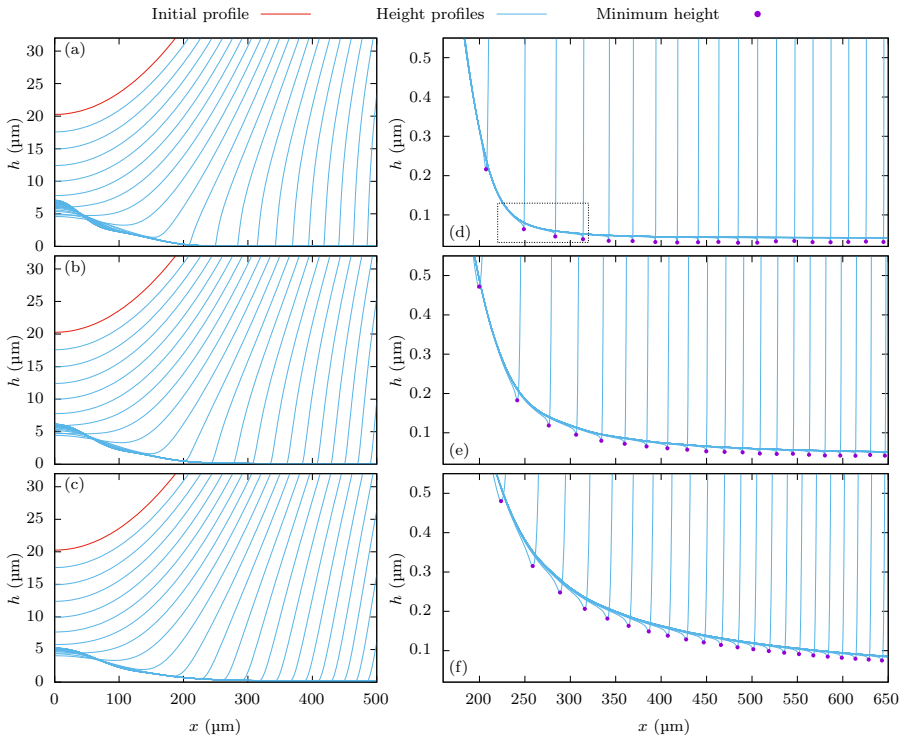


Figure 2.12: **Profiles of the height of the gas layer** at intervals spaced 6.004 µs apart for liquid viscosities of (a) $\nu_l = 10\,\mathrm{mm}^2/\mathrm{s}$, (b) $\nu_l = 32\,\mathrm{mm}^2/\mathrm{s}$, and (c) $\nu_l = 100\,\mathrm{mm}^2/\mathrm{s}$ using zero surface tension. Panels (d), (e), and (f) show the same data as (a), (b), and (c), respectively, but with a smaller range of h to highlight that no lift-off occurs in this case. For each profile, the global minimum, which follows the leading tip, is also plotted on the curves. The dashed box in panel (d) marks a further zoomed-in region shown in Figure 2.13.

The very sharp leading tip causes another difficulty in the simulations: as the tip is advected across the computational grid, there will be an effective numerical diffusion, which will act as though a small surface tension has been imposed. This is an important issue since Figure 2.11(b) already confirms that small surface tensions can considerably alter the behavior. To test this, we compared simulations with the baseline parameters to those on a finer grids (Appendix A.3). Unlike the case for finite surface tension, the simulations on a finer grids are noticeably different, with the profiles reaching lower heights and the leading tip not curving up as rapidly. Since liquid viscosity also regularizes the evolution of the height profile, these discrepancies are more significant when ν_l is small.

We calculated the trajectories of the leading tip for a range of values for liquid

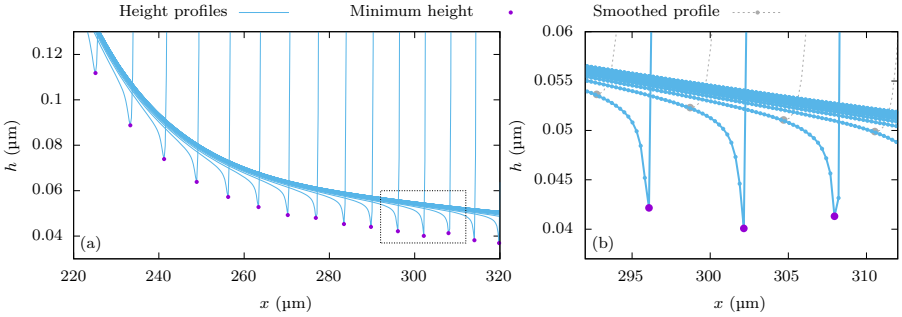


Figure 2.13: **Zoomed-in plots of the height of the gas layer** at intervals spaced 1.2009 µs apart for liquid viscosities of $\nu_l = 10\,\mathrm{mm}^2/\mathrm{s}$ using zero surface tension. Panel (a) shows the region marked by the dashed box in Figure 2.12(d). For each profile, the global minimum, which follows the leading tip, is also plotted on the curves. Panel (b) shows the region marked by the dashed box in panel (a). In panel (b), the small blue circles indicate the computational grid. The gray dashed lines show the profiles with Gaussian smoothing applied, and the gray circles show the global minima of the smoothed lines.

viscosity, ν_l. For $13\,\mathrm{mm}^2/\mathrm{s} < \nu_l \leq 40\,\mathrm{mm}^2/\mathrm{s}$ we switched to a larger computational grid of size 8192×1536, and for $\nu_l \leq 13\,\mathrm{mm}^2/\mathrm{s}$ we switched to a very large grid of size 12288×2304. For $\nu_l \leq 44\,\mathrm{mm}^2/\mathrm{s}$ we were not able to simulate on a large enough grid to adequately resolve the numerical diffusion, and hence results are omitted. Figure 2.14 shows the trajectories in both the (x, h) and (t, h) planes. In all cases, the leading tips continue to decrease. While our results only examine one particular set of physical parameters (from Table 2.2) our results strongly suggest that surface tension is required for lift-off to occur.

We also found a regime where the effect of surface tension can qualitatively affect the lift-off behavior. Figure 2.15(a) shows the height profiles for a simulation with the baseline value of surface tension of $\sigma = 0.072\,\mathrm{N/m}$ in the regime of low initial velocity, $V = 0.3\,\mathrm{m/s}$ and low viscosity, $\nu_l = 6.5\,\mathrm{mm}^2/\mathrm{s}$. In this regime, prominent capillary waves are generated outside the thin gas layer, which grow larger as time progresses. Figure 2.15(b) shows a zoomed-in plot of the profiles in the thin gas layer. Lift-off appears to occur at $x \approx 280\,\mu\mathrm{m}$ and the leading tip starts to rise. However the influence of the capillary wave causes the tip to move downward again, ultimately dipping below the previous minimum height. It is possible that the tip may move upward again at a later point, but we were unable to track the behavior further. The sharp features visible in Figure 2.15(a) (e.g., at $(x, h) \approx (700\,\mu\mathrm{m}, 100\,\mu\mathrm{m})$) cause the simulation to terminate early, since the Newton–Raphson iterations can no longer be solved to an acceptable level of accuracy.

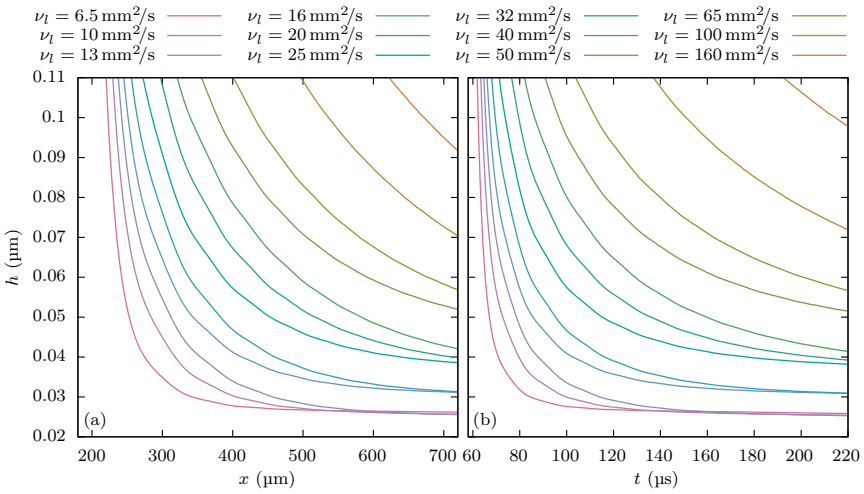


Figure 2.14: **Trajectories of the global minimum of the drop profile** in (a) the (x, h) plane, and (b) the (t, h) plane for a range of different liquid viscosities, using the baseline parameters and zero surface tension. Simulations with $13\,\mathrm{mm}^2/\mathrm{s} < \nu_l \leq 40\,\mathrm{mm}^2/\mathrm{s}$ use a grid of size 8192×1536, and simulations with $\nu_l \leq 13\,\mathrm{mm}^2/\mathrm{s}$ use a grid of size 12288×2304.

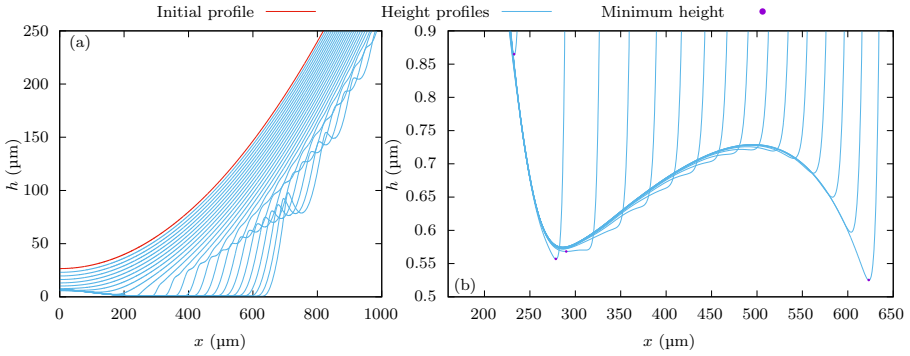


Figure 2.15: (a) **Profiles of the height of the gas layer** at intervals spaced 11.80 µs apart for liquid viscosity of $\nu_l = 6.5\,\mathrm{mm}^2/\mathrm{s}$ and initial drop velocity of $V = 0.3\,\mathrm{m/s}$, showing the development of capillary waves. (b) Zoomed-in plot of the same data. The global minima of the curves are plotted when they indicate the leading tip.

2.5 Conclusion

We investigated the viscous effects in the early stages of drop impact on a surface. We coupled a Navier–Stokes solver to model flow in the interior of the drop with a partial differential equation to model the pressure and height in the thin gas layer between the drop and the surface. We demonstrated that our simulations

are consistent with previous work using potential flow theory where flow in the liquid is assumed incompressible [30, 31, 29]. However, our simulations allow us to go beyond this previous work and investigate viscous effects. We showed that at low values of the initial drop velocity, liquid viscosity plays a weak role in the deceleration of the drop, and the height, H^*, at which it reaches a stagnation point.

Using our simulations, we are able to recreate the lift-off phenomenon that was experimentally reported by Kolinski et al. [24]. We have therefore demonstrated that with the reduced model described in Section 2.2, our simulations are able to capture lift-off. Our numerical results for the lift-off time τ are consistent with the $\nu_l^{1/2}$ scaling relationship found by Kolinski et al. [24]. However, the additional precision afforded by the simulations indicates that the precise relationship between τ and ν_l is more complex, due to the effects of capillary waves and the sensitivity of the results to the definition of the time origin. The simulation allows us to probe conditions that would be difficult to observe experimentally. Using this capability, our results provide strong evidence that surface tension is necessary for lift-off to occur.

There are a number of possible next steps. The simulations provide a detailed view of the lift-off phenomenon, and all aspects (e.g., stress, velocity, vorticity) can be calculated. The results may therefore guide theoretical analyses of lift-off, and may allow scaling relationships similar to those presented by Mandre et al. [30, 29] and Mani et al. [31] to be derived. As part of our study, we have released a complete high-performance open source code that can examine lift-off across a wide range of configurations.

We opted to use a fixed-grid simulation for simplicity in coupling the liquid domain and gas layer together, and as described in Appendices A.2 and A.3 we are able to obtain accurate simulation results in a reasonable timeframe. However, it is likely that the behavior at the bottom of the liquid domain, close to the liquid–gas interface and near the leading tip, dominates. Because of this, the simulation is a good candidate for use with mesh refinement [8, 18], where the liquid–gas interface and the region around the leading tip could use a finer mesh. This could result in substantial computational savings, although the liquid–gas coupling would become more complicated and numerical errors would be harder to quantify.

The simulation could also be generalized to use cylindrical axisymmetric coordinates. The overall numerical approach would stay the same, but additional radial factors would have to be incorporated throughout the simulation. The

Navier–Stokes solver that we employ has already been demonstrated to work in axisymmetric coordinates [42, 43], but the routines that involve the gas layer would require modifications. For example, the kernel in Equation 2.10 would need to be modified. While the current simulation is already in good agreement with the experimental results of Kolinski et al. [24], an axisymmetric solver would allow for a near-perfect comparison. This would, for example, help elucidate further the precise relationship between lift-off time τ and liquid viscosity ν_l.

Experiments by Thoroddsen and coworkers have used interferometry to examine the evolution of the gas layer across a full two-dimensional surface [26]. Their results show a number of interesting effects beyond the scope of our current model, such as a breakage of rotational symmetry and the formation of ruptures in the air film [26, 28]. They have also examined the case of nano-rough surfaces [27]. To connect with this work, our model could be generalized to a full three-dimensional simulation of the liquid and two-dimensional simulation of the gas-layer. This would be substantially more computationally challenging, and would be a good candidate for using parallel computing and adaptive mesh refinement [44].

References

[1] OpenMP website. http://openmp.org/.

[2] TGMG: A templated multigrid library in C++. https://github.com/chr1shr/tgmg.

[3] VDropImpact: A simulation of viscous effects in early-time drop impact dynamics. https://github.com/chr1shr/vdropimpact.

[4] Ann S Almgren, John B Bell, Phillip Colella, Louis H Howell, and Michael L Welcome. A conservative adaptive projection method for the variable density incompressible Navier–Stokes equations. *Journal of Computational Physics*, 142(1):1–46, 1998.

[5] Ann S Almgren, John B. Bell, and William G Szymczak. A numerical method for the incompressible Navier–Stokes equations based on an approximate projection. *SIAM Journal on Scientific Computing*, 17(2):358–369, 1996.

[6] John B Bell, Phillip Colella, and Harland M Glaz. A second-order projection method for the incompressible Navier–Stokes equations. *Journal of Computational Physics*, 85(2):257–283, 1989.

[7] John B Bell, Louis H Howell, and Phillip Colella. An efficient second-order projection method for viscous incompressible flow. In *10th Computational Fluid Dynamics Conference*, 1991.

[8] Marsha J Berger and Joseph Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics*, 53(3):484–512, 1984.

[9] Arnout MP Boelens and Juan J de Pablo. Simulations of splashing high and low viscosity droplets. *Physics of Fluids*, 30(7):072106, 2018.

[10] Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

[11] Alexandre J Chorin. A numerical method for solving incompressible viscous flow problems. *Journal of Computational Physics*, 2(1):12–26, 1967.

[12] Alexandre J Chorin. Numerical solution of the Navier–Stokes equations. *Mathematics of Computation*, 22(104):745–762, 1968.

[13] Phillip Colella. A direct Eulerian MUSCL scheme for gas dynamics. *SIAM Journal on Scientific and Statistical Computing*, 6(1):104–107, 1985.

[14] Phillip Colella. Multidimensional upwind methods for hyperbolic conservation laws. *Journal of Computational Physics*, 87(1):171–200, 1990.

[15] Richard Courant, Kurt Friedrichs, and Hans Lewy. On the partial difference equations of mathematical physics. *IBM Journal of Research and Development*, 11(2):215–234, March 1967.

[16] Jolet de Ruiter, Jung Min Oh, Dirk van den Ende, and Frieder Mugele. Dynamics of collapse of air films in drop impact. *Physical Review Letters*, 108(7):074505, 2012.

[17] Jens Eggers, Marco A. Fontelos, Christophe Josserand, and Stéphane Zaleski. Drop dynamics after impact on a solid wall: Theory and simulations. *Physics of Fluids*, 22(6):062101, 2010.

[18] Arthur Guittet, Maxime Theillard, and Frédéric Gibou. A stable projection method for the incompressible navier–stokes equations on arbitrary geometries and adaptive quad/octrees. *Journal of Computational Physics*, 292:215–238, 2015.

[19] Michael T Heath. *Scientific Computing: An Introductory Survery*. McGraw-Hill, 2002.

[20] Christophe Josserand and Sigurdur T Thoroddsen. Drop impact on a solid surface. *Annual Review of Fluid Mechanics*, 48:365–391, 2016.

[21] Christophe Josserand and Stéphane Zaleski. Droplet splashing on a thin liquid film. *Physics of Fluids*, 15(6):1650–1657, 2003.

[22] John M Kolinski, Ramin Kaviani, Dylan Hade, and Shmuel M Rubinstein. Surfing the capillary wave: wetting dynamics beneath an impacting drop. *Physical Review Fluids*, 4(12):123605, 2019.

[23] John M Kolinski, L. Mahadevan, and Shmuel M Rubinstein. Drops can bounce from perfectly hydrophilic surfaces. *EPL (Europhysics Letters)*, 108(2):24001, 2014.

[24] John M Kolinski, L Mahadevan, and Shmuel M Rubinstein. Lift-off instability during the impact of a drop on a solid surface. *Physical Review Letters*, 112(13):134501, 2014.

[25] John M Kolinski, Shmuel M Rubinstein, Shreyas Mandre, Michael P Brenner, David A Weitz, and L Mahadevan. Skating on a film of air: drops impacting on a surface. *Physical Review Letters*, 108(7):074503, 2012.

[26] Kenneth Langley, Er Qiang Li, and Sigurdur T Thoroddsen. Impact of ultra-viscous drops: air-film gliding and extreme wetting. *Journal of Fluid Mechanics*, 813:647–666, 2017.

[27] Kenneth R Langley, Er Qiang Li, Ivan U Vakarelski, and Sigurdur T Thoroddsen. The air entrapment under a drop impacting on a nano-rough surface. *Soft Matter*, 14:7586–7596, 2018.

[28] Er Qiang Li, Kenneth R Langley, Yuan Si Tian, Peter D Hicks, and Sigurdur T Thoroddsen. Double contact during drop impact on a solid under reduced air pressure. *Physical Review Letters*, 119:214502, Nov 2017.

[29] Shreyas Mandre and Michael P Brenner. The mechanism of a splash on a dry solid surface. *Journal of Fluid Mechanics*, 690:148–172, 2012.

[30] Shreyas Mandre, Madhav Mani, and Michael P Brenner. Precursors to splashing of liquid droplets on a solid surface. *Physical Review Letters*, 102(13):134502, 2009.

[31] Madhav Mani, Shreyas Mandre, and Michael P Brenner. Events before droplet splashing on a solid surface. *Journal of Fluid Mechanics*, 647:163–185, 2010.

[32] M Pasandideh-Fard, Y M Qiao, Sanjeev Chandra, and Javad Mostaghimi. Capillary effects during droplet impact on a solid surface. *Physics of Fluids*, 8(3):650–659, 1996.

[33] Kai Range and François Feuillebois. Influence of surface roughness on liquid drop impact. *Journal of Colloid and Interface Science*, 203(1):16–30, 1998.

[34] Martin Rein. Phenomena of liquid drop impact on solid and liquid surfaces. *Fluid Dynamics Research*, 12(2):61, 1993.

[35] R Rioboo, C Bauthier, J Conti, M Voue, and J De Coninck. Experimental investigation of splash and crown formation during single drop impact on wetted surfaces. *Experiments in Fluids*, 35(6):648–652, 2003.

[36] Chris H Rycroft, Chen-Hung Wu, Yue Yu, and Ken Kamrin. Reference map technique for incompressible fluid–structure interaction. *Journal of Fluid Mechanics*, 898:A9, 2020.

[37] Mark Sussman, Ann S Almgren, John B Bell, Phillip Colella, Louis H. Howell, and Michael L Welcome. An adaptive level set approach for incompressible two-phase flows. *Journal of Computational Physics*, 148(1):81–124, 1999.

[38] Roeland CA van der Veen, Tuan Tran, Detlef Lohse, and Chao Sun. Direct measurements of air layer profiles under impacting droplets using high-speed color interferometry. *Physical Review E*, 85(2):026315, 2012.

[39] Arthur Mason Worthington. XXVIII. On the forms assumed by drops of liquids falling vertically on a horizontal plate. *Proceedings of the Royal Society of London*, 25(171–178):261–272, 1877.

[40] Lei Xu, Wendy W Zhang, and Sidney R Nagel. Drop splashing on a dry smooth surface. *Physical Review Letters*, 94(18):184505, 2005.

[41] Alexander L Yarin. Drop impact dynamics: splashing, spreading, receding, bouncing. . . . *Annual Review of Fluid Mechanics*, 38:159–192, 2006.

[42] Jiun-Der Yu, Shinri Sakai, and James A Sethian. A coupled level set projection method applied to ink jet simulation. *Interfaces and Free Boundaries*, 5(4):459–482, 2003.

[43] Jiun-Der Yu, Shinri Sakai, and James A Sethian. Two-phase viscoelastic jetting. *Journal of Computational Physics*, 220(2):568–585, 2007.

[44] Weiqun Zhang, Ann Almgren, Vince Beckner, John Bell, Johannes Blaschke, Cy Chan, Marcus Day, Brian Friesen, Kevin Gott, Daniel Graves, Max P Katz, Andrew Myers, Tan Nguyen, Andrew Nonaka, Michele Rosso, Samuel

Williams, and Michael Zingale. AMReX: a framework for block-structured adaptive mesh refinement. *Journal of Open Source Software*, 4(37):1370, 2019.

Chapter 3

Coordinated crawling via reinforcement learning

This chapter has been published as **Mishra, S., van Rees, W. M. and Mahadevan, L., 2020. Coordinated crawling via reinforcement learning. Journal of the Royal Society Interface 17: 20200198** [arXiv: 2003.12845]

Abstract

Rectilinear crawling locomotion is a primitive and common mode of locomotion in slender soft-bodied animals. It requires coordinated contractions that propagate along a body that interacts frictionally with its environment. We propose a simple approach to understand how this coordination arises in a neuromechanical model of a segmented, soft-bodied crawler via an iterative process that might have both biological antecedents and technological relevance. Using a simple reinforcement learning algorithm, we show that an initial all-to-all neural coupling converges to a simple nearest-neighbour neural wiring that allows the crawler to move forward using a localized wave of contraction that is qualitatively similar to what is observed in *Drosophila melanogaster* larvae and used in many biomimetic solutions. The resulting solution is a function of how we weight gait regularization in the reward, with a trade-off between speed and robustness to proprioceptive noise. Overall, our results, which embed the brain–body–environment triad in a learning scheme, have relevance for soft robotics while shedding light on the evolution and development of locomotion.

3.1 Introduction

The locomotion of an animal is a result of coordination of its nervous system with its body and environment. Understanding coordinated motions that involve sensory feedback and proprioception requires a theoretical framework integrating the brain, body and environment [4, 20, 2]. But how do these smooth rhythmic motions arise in the first place? Experiments on locomotory dynamics in model organisms, such as the larva of *Drosophila melanogaster* [9], suggest that, early in larval morphogenesis, neurons are part of a well-connected network. During development, the pruning of neuronal connections reduces the connectivity of neurons via both biochemical and biomechanical feedback modulated by behaviour and function embodied in twitching that gradually gives way to coordinated locomotion [15, 1]. In the larva and more generally in many soft-bodied organisms, motion arises via rectilinear crawling [6, 24], wherein rhythmic contraction and relaxation of muscles create waves that propagate either forward (prograde) or backward (retrograde) along the length of the body. This induces forward locomotion when the interaction with the substrate is asymmetric, e.g., when friction in the forward direction and that in the backward direction are very different. The asymmetry in friction has both a passive and an active component: the presence of anisotropic denticles allows the body to slide more easily in one direction than another passively, while dorso-ventral muscles can partially lift the body to modulate friction actively [9]. In either case, the result is the conversion of waves of contraction to net motion of the body, which has been studied for over a century [8]. Substantial previous experimental work characterizing *D. melanogaster* crawling has highlighted the role of sensory feedback in initiating and maintaining the gait [22] and has inspired recent theoretical work on the dynamics of a segmented, soft-bodied crawler moving on a frictional surface [16, 17]. These studies have shown that minimal representations of the musculature and neural dynamics suffice to explain a number of these experimental observations that include the onset and propagation of contractile waves that lead to locomotion, and further suggest that the rhythmic gait can arise without a central pattern generator. Here, neural impulses drive the activation of muscle forces, resulting in deformation of the body, producing biomechanical strain. Proprioceptive sensing of this strain in turn drives neural impulses, thereby closing the feedback loop. The result is that the crawler moves forward by simultaneously lifting and contracting its body segments, starting from the posterior segments and moving towards the anterior end. Critically, in these and most other studies, the neural system is assumed to have a fixed, predetermined connectivity. Since the muscles, body wall and

connective tissue in the body of a *D. melanogaster* larva develop asynchronously [22], a natural question is how these subsystems are wired together for robust performance. Indeed, could the crawler use proprioceptive feedback to learn a coordinated gait for forward crawling, i.e., rewire the neuronal connections using experience-driven sensory feedback to achieve a coordinated gait, as observed experimentally [22]? To explore this, we use the framework of reinforcement learning (RL) [23]. Originally inspired by observations of how animals learn to perform certain functions, the approach has gained significant traction recently in the context of training computers in games [21], strategies for moving through a fluid [5, 19] and other domains. We frame our question in terms of the coupled dynamics of a neurophysical system for the crawler and an RL algorithm for neuronal wiring, using sensory feedback to maximize a reward associated with crawling forward.

3.2 Mathematical model of a crawler

Our mathematical model is chosen to roughly mimic a soft-bodied crawler, the larva of the fruitfly *D. melanogaster*, given that it has become a focus to understand the link between molecules, circuits, physiology and behaviour using a variety of approaches [13, 25]. The soft-bodied cylindrical larva consists of 10 segments and is about 1 mm in length and 200 μm in diameter in the first instar stage and 4 mm in length and about 800 μm in diameter in the third instar stage. Previous work introduced a coupled neurodynamical model, with a given neural and mechanical connectivity to explain how coordinated crawling can arise even without a central pattern generator [16], and later included a more detailed neuromechanical network model in a similar framework to quantify the experimentally observed importance of proprioception [17]. This sets the stage to ask how the larva might learn to coordinate crawling, given the importance of properly wiring the neuromuscular system for robust functioning of its locomotory system. Anatomically, the segments are connected at their boundaries (nodes) as shown in Figure 3.1(a). Each segment is assumed to behave like a linear viscoelastic solid, which can be actively contracted by muscles that respond to neuronal inputs as schematized in Figure 3.1(a). The firing of a segmental neuron, minimally modelled as a leaky integrator, causes muscular activation to deform the segment, which then moves if the forces overcome friction; simultaneously, the segment transmits forces to neighbouring segments, where neurons can be activated if the strain crosses a threshold. This leads to a propagating wave even in the absence of a central pattern generator.

We now turn to quantifying the three sub-systems corresponding to the body, the brain and the environment.

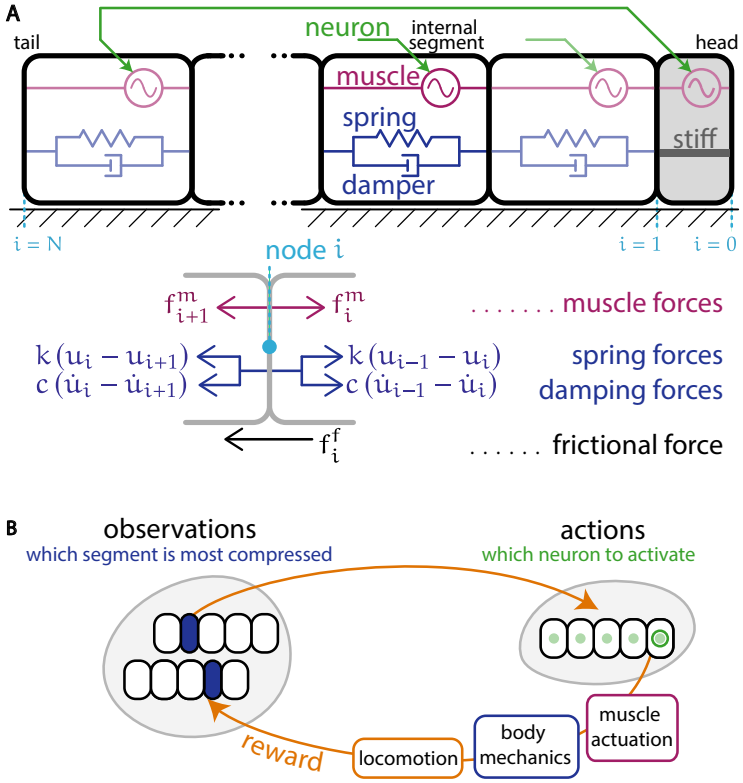


Figure 3.1: **Schematic of the crawler.** (a) Each segment of the soft-bodied crawler is represented by a spring-damper system and a muscle. Each muscle acts to stretch the segment and is driven by a single neuron. (b) Interactions between the different components of the crawler as it learns using the feedback from its environment.

3.2.1 Body mechanics

We assume that the passive viscoelastic properties of each segment can be described in terms of an elastic modulus E and viscosity μ. To construct a simple one-dimensional model for its mechanical behaviour, we integrate across the cross-section of the body, so that the segmented crawler resembles a set of active viscoelastic springs with stiffness $k = EA/L$ and damping constant $c = \mu A/L$, where A is the area of cross-section of a segment of length L; see Figure 3.1(a). The segment boundaries or nodes $i \in [0, 10]$ are mechanically characterized by their displacements $u_i(t)$, which change due to a contractile force f^m_i exerted by muscles on either side of node i and due to a frictional force f^f_i from the

external environment at node i. Ignoring the role of inertia, since the animals move slowly, force balance at node $i \in [1, 9]$ in Figure 3.1(a) implies that

$$k\,(u_{i+1} - 2u_i + u_{i-1}) + c\,(\dot{u}_{i+1} - 2\dot{u}_i + \dot{u}_{i-1}) + f_i^m - f_{i+1}^m = f_i^f. \tag{3.1}$$

The force-balance equations at the head and the tail are different from those at the internal nodes as the head and tail do not have a segment ahead of and behind them, respectively. At the head ($i = 0$),

$$k\,(u_1 - u_0) + c\,(\dot{u}_1 - \dot{u}_0) + f_0^m - f_1^m = f_0^f, \tag{3.2}$$

while at the tail ($i = N = 10$),

$$k\,(u_{N-1} - u_N) + c\,(\dot{u}_{N-1} - \dot{u}_N) + f_N^m = f_N^f. \tag{3.3}$$

To complete the formulation of the model for how the body responds, i.e., solve for $u_i(t)$, $i \in [0, 10]$, we need to specify dynamical laws for the evolution of the neural dynamics that drive the internal muscular forces f_i^m and the environmental frictional forces f_i^f.

3.2.2 Neuromuscular dynamics

In each segment, we assume that the neural dynamics follow a minimal first-order dynamical law known as the θ-model [7] to drive the activation of neuron i. Assuming the time scale of neuronal relaxation to be τ_θ, we can then write

$$\tau_\theta \frac{d\theta_i}{dt} = 1 - \cos\theta_i + (1 + \cos\theta_i)\min\left[1, I_i(t)\right]. \tag{3.4}$$

Here $I_i(t)$ is the time-dependent input to the neuron i, and the neuron fires every time $\mod(\theta - \pi, 2\pi)$ crosses zeros, so that the set of spike times t_i^s for neuron i is given as

$$\left\{t_i^s\right\} = \left\{t \middle| \mod(\theta_i(t) - \pi, 2\pi) = 0\right\}, \tag{3.5}$$

resulting in muscle actuation. The input $I_i(t)$ is then restricted to be a binary variable, with only one neuron active at a given time

$$I_i(t) = \begin{cases} 1 & \text{if } i = a \\ 0 & \text{if } i \neq a \end{cases}, \qquad a \in \{0, \ldots, N-1\} \tag{3.6}$$

Noting that experimental observations of larval crawling show that the head and the tail move together [9], we activate the tail neuron I_N every time the head neuron is activated, i.e., when $I_0 = 1$, we set $I_N = 1$.

We note that a more sophisticated neural model with a population of excitatory and inhibitory neurons that are coupled is probably more realistic as a model for the fruitfly larva [9] but yields the same qualitative results as the simple integrate and fire model above, and so we will limit ourselves to the current simple model.

In each segment, we assume that the muscle forces vary between zero and a maximum F^m_{max}. The contracting muscles respond asymmetrically to the timing of neuronal spikes with a characteristic rise time that is about half that of the exponential decay [9]. For simplicity, we use a symmetric rise and decay with a built-in temporal decay constant τ_m and a limiter to set the maximum force amplitude so that

$$\tau_m \frac{df^m_i}{dt} = -f^m_i + F^m_{max} \min\left[1, F^m_i(t)\right], \tag{3.7}$$

where

$$F^m_i(t) = \Sigma_{t^s \in \{t^s_i\}} e^{-(t-t^s)/\tau_\theta} \tag{3.8}$$

is the sum of all the forces due to the spiking neurons. This dynamical law is consistent with our previous models [16, 17]. Here, we note that it is possible for several segments to have active muscles even though only one neuron can be active at a particular time, because the muscle forces can decay much more slowly than neural activity.

3.2.3 Body–substrate frictional interaction

To complete the formulation of our model, we need to prescribe a frictional law for the interaction of the crawler with the environment. Experimental observations [9] show that the larva actively reduces friction in a contracting segment by lifting it off the substrate, and the presence of asymmetrically shaped denticles on the ventral surface make the passive friction asymmetric, so that forward motion experiences less friction than backward motion. While previous work [17] has included both these effects, in our one-dimensional model we do not distinguish between the passive and active components of the friction for simplicity. We further impose the condition that the friction force vanishes whenever $\dot{u}_i = 0$, leading to a smooth transition between the positive and negative values for forward and backward velocity. Then the friction force on node i is given by

$$f^f_i = \frac{f^f_{max}}{2}\left[(1+\eta_f)\tanh\left(\frac{\dot{u}_i - \dot{u}^0}{\varepsilon^f}\right) + (1-\eta_f)\right], \tag{3.9}$$

where η_f is the ratio of the maximum frictional force in the backward to the forward directions, ϵ^f is a smoothing parameter and $\dot{u}^0$ is a constant chosen such that $f^f(0) = 0$. All together, our mathematical model, Equations 3.1–3.9, determines the gait and locomotion of the crawler: given the neural connectivity weights and an initial neural impulse leads to an input that drives Equation 3.4; this drives Equations 3.8 and 3.9 and thence Equations 3.1–3.3.

3.2.4 Scaling and parameter choices

We scale the relevant variables in our model using the time scale of neuronal activity τ_θ, the equilibrium length of a segment L and the stiffness of a segment k. Then the dimensionless parameters corresponding to the variables presented in the mechanical model are: τ_m/τ_θ, which is the ratio of time scales for muscular and neuronal relaxation; $c\tau_\theta/k$, which is the dimensionless damping; f^f_{max}/kL, which is the scaled maximum frictional force; and F^m_{max}/kL, which is the scaled maximum muscular force. The specific values for these dimensionless parameters used throughout this work, given in Table B.1 in Appendix B, are consistent with experimental estimates for a *D. melanogaster* larva [17].

To compare our results of converged coordinated crawling shown in Figure 3.2 (see also Appendix B.3) to the experimental observations of *D. melanogaster* larva we use our simulations to extract the scaled maximum segment deformation $\Delta u/L$, the characteristic wave speed $v\ \tau_\theta/L$, and the speed of the larva $v_{crawler}\ \tau_\theta/L$. For the parameter values given in Table B.1 in Appendix B, we find that the peak contraction of a segment is 33%, yielding a contraction speed of 0.026 waves/τ_θ and a crawler forward speed of $0.0056L/\tau_\theta$. For a third instar larva [11], using the value of 1.5 waves per second and a length of 4 mm implies that τ_θ = 17 ms and $L = 4/10 = 0.4$ mm, respectively, so that the forward speed of the crawler is 0.13 mm/s. For a first instar larva [9], using a wave speed of $0.5 - 1.5$ waves per second and a length of 1 mm, we get a range of $17 - 51$ ms, which translates to a forward speed of $11 - 33$ μm/s, compared with the observed range of $45 - 120$ μm/s.

3.3 Reinforcement learning (RL) strategy

With the established physical model and parameter choices for the crawler, we turn to RL to determine the neural weights for efficient crawling. The framework of RL consists of an agent interacting with its environment, with

the aim of achieving a goal. An agent moves through different environmental observable states by taking actions. While doing so, it accumulates rewards from the environment, with the goal of taking actions that maximize its long-term rewards, calculated as a discounted sum of successive rewards. This goal is achieved by learning a mapping that links an action to its current environmental observable state; this mapping is known as the agent's policy. The RL description is summarized in Figure 3.1(b).

3.3.1 Formulation of observable state, action and reward

In our formulation, the observation of the agent is an incomplete knowledge of itself and its frictional environment. Given the established importance of proprioception [16] in locomotion, it is likely to be important in the learning process as well. A minimal approach accounting for this is via the observation o associated with the index of the segment that is most strongly contracted, since that requires knowledge of a single variable that can be easily computed via a series of pair-wise comparisons. Then,

$$o = \text{argmin}_{i\in(1,\ldots,N)} \left(u_i - u_{i-1}\right). \tag{3.10}$$

The action a is the input to the $\theta-$model that drives neuronal activity, resulting in muscle actuation, i.e., $I_i(t)$ in Equation 3.4. We further restrict this by allowing the input $I_i(t)$ to have values of 0 (OFF) or 1 (ON), with only one neuron active at a given time, as described in Equation 3.6.

Since the goal is to move forward, we set the reward r accordingly,

$$r = (\bar{u}_{t+\Delta t} - \bar{u}_t) - \epsilon r_2, \tag{3.11}$$

where $\bar{u}$ is the position of the centroid of the crawler, t denotes time, Δt is the size of the discrete time step (see Table B.1 in Appendix B), and $r_2 = \max_i \left(|u_{i+1} - 2u_i + u_{i-1}|\right)$ is a penalty on large variations in strain along the length of the crawler, with ϵ determining the relative contributions from this strain gradient to the reward r.

We use a form of RL known as Q-learning [23], with a discrete representation for the observation and action spaces. The entries in the Q-matrix, $Q(o, a)$, represent how much cumulative reward the crawler expects to get after taking an action a after an observation o, i.e., $\sum_{k=0}^{\infty} \gamma^k r_{t+(k+1)\Delta t}$, where r_t is the reward at time t and $\gamma \in [0, 1)$ is the discount factor (see Table B.1 in Appendix B) that weighs the long-term rewards relative to the short-term rewards. To maximize

the expected discounted cumulative sum of rewards, the entries $Q(o, a)$ are updated each time the agent takes an action after an observation, according to the update rule

$$Q(o, a) = (1 - \alpha)Q(o, a) + \alpha \left(r_t + \gamma \max_a (Q(o', a)) \right), \tag{3.12}$$

where α is the learning rate and o' is the subsequent observation made by the agent. The policy is a greedy policy, meaning that, after each observation, the agent takes the action that corresponds to the highest value. The learning is done in episodes; each episode corresponds to the crawler moving a fixed distance forward, after which it is reset to its original undeformed configuration. The crawler goes through a number of episodes in this manner, gaining experience in the interactions between neurons, body mechanics and environment, updating its Q-matrix as it goes through the episodes. It is worth emphasizing that our learning algorithm has just two parameters, a learning rate α and a discount factor γ, in contrast to many recent variants of RL that have many hyper-parameters; thus most reasonable choices for these will converge and yield similar policies. We choose $\alpha = 0.05$ to allow for stochastic effects and $\gamma = 0.95$ to strive towards the case of high long-time rewards [23].

3.3.2 Experimental results: regularized and unregularized gaits

We initialize the crawler in an undeformed state, with a Q-matrix of values that are uniform and high. Then the crawler is equally likely to take any action independent of the observable state of the crawler, and since the values are high, i.e., the reward is lower than the expected reward, the crawler explores other actions. This leads to uncoordinated gaits; an example is shown in Appendix B, Figure B.1. As the Q-matrix converges towards its steady-state value, the rewards become closer to the expectation of the crawler and the policy converges to a coordinated gait.

Figure 3.2 shows two coordinated gaits corresponding to two values of the regularization parameter $\epsilon = 0.01$ (Figure 3.2(a)) and $\epsilon = 0$ (Figure 3.2(b)), as defined in Equation 3.11. In both of the gaits, the crawler moves by means of a travelling wave of contraction from tail to head. The regularized gait corresponds to observations of a larva consistent with experiments, wherein a localized wave causing sequential segmental contraction moves from tail to head as shown in Figure 3.2(a). By contrast, the unregularized gait, corresponding to $\epsilon = 0$, is characterized by a 10% higher speed, and larger variations in segment

strain, and is due to the fact that some muscles are never activated (Figure 3.2(b), right), leading to pairs of segments moving together (see Appendix B.3). The policies for both gaits are shown in Figure 3.2(c). These results justify our use of a regularization penalty in the reward to recover gaits that are biologically plausible and are also consistent with the diagonal neuronal weights that result.

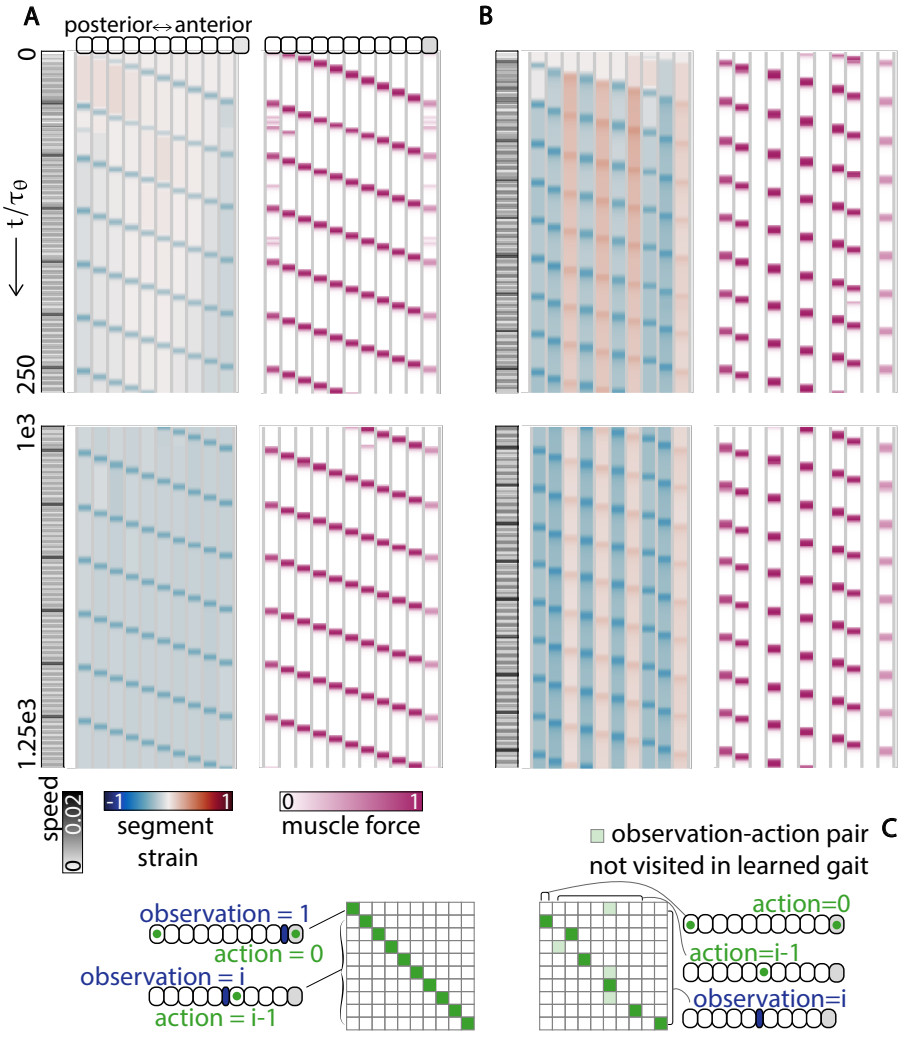


Figure 3.2: **Learning of coordinated gaits in a neurophysical model** determined using Equations 3.1–3.12. (a) A regularized gait ($\epsilon = 0.01$), with the speed of the centre of mass of the crawler shown in grey, the segment strains shown in blue-red and the muscle forces in each segment shown in white-pink, and (b) an unregularized gait ($\epsilon = 0$). The parameter values are summarized in Table B.1 in Appendix B. (c) Converged policy corresponding to the gaits in (a, b), with the dark and light green squares corresponding to $\pi(a|o) = 1$ in the final policy and light green squares corresponding to observations which are never reached in the converged gait.

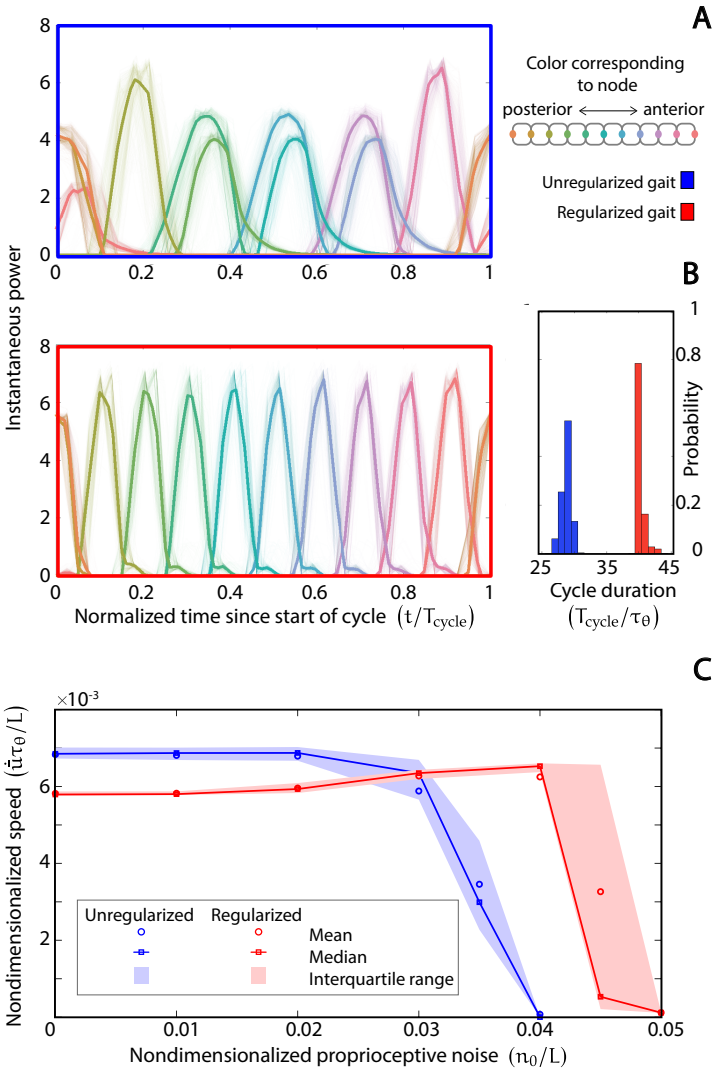


Figure 3.3: **Comparison of the unregularized (blue) and regularized (red) gaits.** (a) Power p_i computed using Equation 3.13 for each node $i \in [1, 10]$ as a function of phase in a cycle, for a number of cycles. The different colours correspond to node numbers as labelled. (b) Histogram for the duration of a cycle for the unregularized (blue) and regularized (red) gaits. (c) Speed versus proprioception noise for the unregularized (blue) and regularized (red) gaits.

To further compare the gaits, we show the power expenditure, cycle duration and robustness to noise in Figure 3.3. The power exerted at each node,

$$p_i = |f_i - f_{i-1}|\,|u_i|\,, \tag{3.13}$$

is a periodic function for both cases. For the regularized gait, the maximum power

and the duration for which power is non-zero are both more uniform across the interior nodes, while for the unregularized gait there is a larger variation in power across nodes (Figure 3.3(a)). Figure 3.3(b) shows the distribution of cycle duration for the two gaits, and shows that the higher speed of the unregularized gait is achieved via a faster propagation of waves along the length of the crawler.

To test whether these policies are robust, we explored the response of the two gaits to uncertainty in the crawler's ability to sense proprioceptive strain. We implement this by replacing the deterministic observation of the most compressed segment, given by Equation 3.10, by a noisy version with $o = \text{argmin}_{i\in(1,\ldots,N)}\,(u_i - u_{i-1} + U)$, where $U \in [-n_0, n_0]$ is a uniformly distributed random variable and n_0 is the maximum amplitude of the noise. We find that while the regularized gait has a lower speed than the unregularized gait at low levels of noise n_0, the regularized gait maintains its speed as the noise level increases, while the unregularized gait does not. This trade-off between speed and robustness to noise is demonstrated by the crossover in Figure 3.3(c). Comparing the segment strain over the course of a cycle, we observe that the unregularized gait varies over a smaller range than the regularized gait (denoted by a smaller contrast in colours for a particular segment in Figure 3.2(b) versus Figure 3.2(a)). This suggests that the unregularized gait should be more susceptible to proprioceptive noise, consistent with what is observed in Figure 3.3(c).

3.4 Discussion

There is now much interest in using a range of machine learning techniques for modelling movement control in the context of bipedal walking, running and jumping [10, 18, 12] and more complex modes of movement such as swimming, gliding, etc. [5, 19]. Our minimal approach to learning a coordinated gait for rectilinear crawling embeds the question of determining the neural weights via RL in a framework linking the brain, the body and the environment. Our observations and actions are all couched in terms of biophysically motivated quantities such as relative displacements, forces and neural spike patterns, along with rewards that are characterized by speed and internal strains and strain rates. The solutions that we converge to recover the wiring patterns and propagating contractile waves similar to experimental observations [16, 17]. Regularizing the reward to penalize strain gradients provides smooth gaits that expend power more uniformly in space and time, as well as gaits that are robust to uncertainty in the crawler's ability for proprioception, but at the cost of speed. Indeed there

is a trade-off between speed and robustness when these gaits are challenged by proprioceptive noise. This qualitative result of our study is suggested in the developmental neurobiology literature [9, 22] and has potential applications in the design of robotic analogues of crawlers [3].

Quantifying the functional form of this speed–robustness trade-off is a natural next question which will require specific choices for the form of the uncertainty in the environment (e.g., friction), in the body (e.g., mechanical properties that change due to developmental defects or injury) and in the brain (e.g., wiring that changes due to molecular mechanisms of external-induced injury). Looking ahead, our one-dimensional model needs to be augmented to account for bending deformations to reproduce larval motions that include both axial and transverse motions. Recent theoretical work [14] has suggested that a minimal reflexive strategy embedded in a neuromechanical model can give rise to a basis of exploratory locomotory gaits that include rectilinear crawling and turning. By modifying our model and changing the reward structure in our RL framework dynamically as a function of additional internal observations of the crawler, it might be possible to bias the gaits towards coordinated crawling that is exploitative, or towards random turning that is exploratory. More generally, our study is but the first step in determining neural actuation patterns for a range of complex tasks by combining models that couple the mechanics of the brain, body and environment with machine learning.

References

[1] Jimena Berni, Stefan R Pulver, Leslie C Griffith, and Michael Bate. Autonomous circuitry for substrate exploration in freely moving drosophila larvae. *Current Biology*, 22(20):1861–1870, 2012.

[2] Adam J Calhoun and Mala Murthy. Quantifying behavior to solve sensorimotor transformations: advances from worms and flies. *Current Opinion in Neurobiology*, 46:90–98, 2017.

[3] Shoue Chen, Yunteng Cao, Morteza Sarparast, Hongyan Yuan, Lixin Dong, Xiaobo Tan, and Changyong Cao. Soft crawling robots: design, actuation, and locomotion. *Advanced Materials Technologies*, 5(2):1900837, 2020.

[4] Hillel J Chiel and Randall D Beer. The brain has a body: adaptive behavior emerges from interactions of nervous system, body and environment. *Trends in Neurosciences*, 20(12):553–557, 1997.

[5] Simona Colabrese, Kristian Gustavsson, Antonio Celani, and Luca Biferale. Flow navigation by smart microswimmers via reinforcement learning. *Physical Review Letters*, 118(15):158004, 2017.

[6] Stewart Keith Eltringham. *Life in Mud and Sand*. Number 574.5263 E4. Crane Russak, 1971.

[7] G Bard Ermentrout and Nancy Kopell. Parabolic bursting in an excitable system coupled with a slow oscillation. *SIAM Journal on Applied Mathematics*, 46(2):233–253, 1986.

[8] Walter E Garrey and AR Moore. Peristalsis and coordination in the earthworm. *American Journal of Physiology-Legacy Content*, 39(2):139–148, 1915.

[9] Ellie S Heckscher, Shawn R Lockery, and Chris Q Doe. Characterization of drosophila larval crawling at the level of organism, segment, and somatic body wall musculature. *Journal of Neuroscience*, 32(36):12460–12471, 2012.

[10] Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, SM Eslami, Martin Riedmiller, and David Silver. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017.

[11] Cynthia L Hughes and John B Thomas. A sensory feedback circuit coordinates muscle activity in drosophila. *Molecular and Cellular Neuroscience*, 35(2):383–396, 2007.

[12] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26), 2019.

[13] Hiroshi Kohsaka, Pierre A Guertin, and Akinao Nose. Neural circuits underlying fly larval locomotion. *Current pharmaceutical design*, 23(12):1722–1733, 2017.

[14] Jane Loveless, Konstantinos Lagogiannis, and Barbara Webb. Modelling the mechanics of exploration in larval *Drosophila*. *PLoS computational biology*, 15(7):e1006635, 2019.

[15] Darshana Z Narayanan and Asif A Ghazanfar. Developmental neuroscience: How twitches make sense. *Current Biology*, 24(19):R971–R972, 2014.

[16] P Paoletti and L Mahadevan. A proprioceptive neuromechanical theory of crawling. *Proceedings of the Royal Society of London B: Biological Sciences*, 281(1790):20141092, 2014.

[17] Cengiz Pehlevan, Paolo Paoletti, and L Mahadevan. Integrative neuromechanics of crawling in *D. melanogaster* larvae. *Elife*, 5:e11031, 2016.

[18] Xue Bin Peng, Glen Berseth, KangKang Yin, and Michiel Van De Panne. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 36(4):1–13, 2017.

[19] Gautam Reddy, Antonio Celani, Terrence J Sejnowski, and Massimo Vergassola. Learning to soar in turbulent environments. *Proceedings of the National Academy of Sciences*, 113(33):E4877–E4884, 2016.

[20] Serge Rossignol, Réjean Dubuc, and Jean-Pierre Gossard. Dynamic sensorimotor interactions in locomotion. *Physiological Reviews*, 86(1):89–154, 2006.

[21] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354, 2017.

[22] Maximiliano L Suster and Michael Bate. Embryonic assembly of a central pattern generator without sensory input. *Nature*, 416(6877):174, 2002.

[23] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT press, 1998.

[24] Edwin Royden Trueman. *Locomotion of soft-bodied animals*. Edward Arnold, 1975.

[25] Rebecca D Vaadia, Wenze Li, Venkatakaushik Voleti, Aditi Singhania, Elizabeth MC Hillman, and Wesley B Grueber. Characterization of proprioceptive system dynamics in behaving drosophila larvae using high-speed volumetric microscopy. *Current Biology*, 29(6):935–944, 2019.

Chapter 4

Augmenting learning using symmetry in a biologically-inspired domain

This chapter has been published as **Mishra, S., Abdolmaleki, A., Guez, A., Trochim, P. and Precup, D., 2019. Augmenting learning using symmetry in a biologically-inspired domain. 33rd Conference on Neural Information Processing Systems 2019 (NewInML workshop)** [arXiv: 1910.00528]

Abstract

Invariances to translation, rotation and other spatial transformations are a hallmark of the laws of motion, and have widespread use in the natural sciences to reduce the dimensionality of systems of equations, e.g., [14, 3]. In supervised learning, such as in image classification tasks, rotation, translation and scale invariances are used to augment training datasets, such as in [8]. In this work, we use data augmentation in a similar way, exploiting symmetry in the quadruped domain of the DeepMind control suite [13] to add to the trajectories experienced by the actor in the actor-critic algorithm of Abdolmaleki et al. [1]. In a data-limited regime, the agent using a set of experiences augmented through symmetry is able to learn faster. Our approach can be used to inject knowledge of invariances in the domain and task to augment learning in robots, and more generally, to speed up learning in realistic robotics applications.

4.1 Introduction

Reinforcement learning (RL) agents are able to perform locomotion tasks using continuous actions and observations in animal-like domains such as the planar walker, half-cheetah, and humanoid [9, 7]. While the agents learn policies that successfully maximise a cumulative return, the behaviors of agents in such domains often appear idiosyncratic to humans. Notably, locomotion in animals is associated with a natural rhythm and stereotypy, such as in the case of a walking dog, galloping horse, or the tripod gait of an ant. In the absence of this spatial order and temporal rhythm in simulated agents, the behaviors of the learned policies can look unnatural.

Idiosyncratic behavior can arise in simulated agents because the physical simulation is incorrect, the task does not capture the tasks executed by embodied agents, or the constraints on a simulated agent are different from those on an embodied agent. The physical simulation itself is an inaccurate and incomplete model of the physical world – it allows perfect floors, instantaneous, noise-free actuation, and a windless environment. As such, simulated agents are allowed to overfit to peculiarities of the simulation environment. Embodied agents necessarily have behavior policies that are robust to deviations from any precise environment crafted for simulation. Additionally, the mechanical and biological constraints that restrict the behavior of an embodied agent, a robot or an animal, are different from those experienced by simulated agents. Embodied agents are also necessarily affected by notions of mechanical stress and fatigue. For example, an embodied walker would be less likely to drag a leg, due to the notion of wear, or flail around its arms, due to considerations of energy and efficiency.

There are several ways to mitigate the different sources of idiosyncrasy in the learned behavior of RL agents. For example, one can use more sophisticated and realistic physical simulations, reducing the possibility of over-fitting to incorrect physics. Naturally, this approach comes with added computational cost. We discuss some other approaches in Section 4.2.

In this work, we leverage order in the spatial structure of the domain. Specifically, the vast majority of animals have at least one plane of external symmetry, which is reflected in their stereotyped gaits when they are walking or running. Similarly, for simulated bodies with nearly identical limbs, if the agent knows how to move one limb relative to the body in a particular way, it can do the same for the other similar limbs. We use symmetry as a natural inductive bias in order to

reduce the space of behavior policies that can be achieved by agents, as well as to explore if this approach can boost the learning process.

4.2 Related work

In robotics and continuous control applications, some ways to improve the real-world suitability and sample efficiency of RL agents include curriculum learning [4], learning from motion-capture data [10], and using constraint-based objective functions [5]. Our work follows the same theme of making the policies learned by RL agents more realistic, and aiming to make the learning process efficient.

We use invariances in the physical domain and task to augment learning. Invariances to scale, orientation, translation and relative motion are common to the natural world and a hallmark of the dynamics of tangible objects and of continuous fields. The use of such invariances to reduce the dimensionality of the equations governing a physical process is thus ubiquitous in the natural sciences [3, 14]. In machine learning, supervised classification tasks use rotation, symmetry and scale invariances to augment training data to get improved performance [2, 12, 8]. In RL, symmetry and rotation invariances have been used to augment data in AlphaGo [11]. Work concurrent to ours proposes a very similar approach using data-augmentation as a way to inform the learned behavior policies of RL agents in locomotion tasks [6].

4.3 Domain and tasks

To illustrate the use of symmetry, we can use any animal-like domain with a saggital plane of symmetry, which is a plane of symmetry common to several animals. We use the quadruped domain from Tassa et al. (2018) [13]. The body of the quadruped comprises a torso with four legs. Each leg is identical, with three hinge joints, as shown in Figure 4.1 (left). The joints are controlled by torque actuators. We consider the *move* tasks in Tassa et al. (2018) [13], in which positive reward is given for an upright orientation relative to the horizontal plane and a forward velocity of the torso, relative to a specified desired velocity, in the frame of reference of the torso. Within the *move* tasks, the *walk* task and *run* task are specified by different values of desired velocity. Details of the reward function are given in Tassa et al. (2018) [13]. The observations comprise joint

angles and velocities, forces and torques on the joints, the state of the actuators, and the torso velocity.

This is illustrated in Figure 4.1 (right), by means of a pose with the legs shown in blue, and its corresponding *mirrored* pose, with the legs shown in pink. In the *move* tasks, for every observation-action pair that the agent encounters, there is a corresponding mirrored observation-action pair that can be obtained by reflection about the plane of symmetry. The mirroring operation for observations and actions are essentially matrix multiplications, and are obtained cheaply from the observations. The mirrored corresponding observation-action pair will result in the same forward displacement, and therefore the same reward. **Therefore, for every trajectory executed by the agent, there is a corresponding mirrored trajectory that results in the same sequence of rewards.**

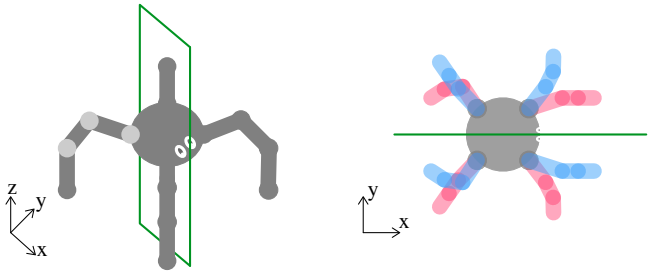


Figure 4.1: Left: **Quadruped domain**, showing the three joints on one of the legs (light grey), and the sagittal plane in green. Right: Top view of a quadruped, showing a fictitious adopted pose in terms of leg orientations (blue), and its corresponding pose (pink), mirrored about the axis of symmetry. The co-ordinate axes correspond to the local frame of reference of the torso.

4.4 Symmetric policy

A *symmetric gait* such as a human walk is characterized by a phase difference between the legs, and an *antisymmetric gait* such as that of a galloping horse is characterized by an in-phase synchrony between the left and right halves of the body. When we use the term *symmetric policy*, we mean that the probability $\pi\left(a^{\text{mirrored}}, s^{\text{mirrored}}\right) = \pi(a, s)$, where s^{mirrored} and a^{mirrored} are obtained by reflecting the state s and action a, respectively, about a plane of symmetry. By this definition, the biological *symmetric* and *antisymmetric* gaits are both executed by a symmetric policy. We note that in general, this does not mean that the probability of visiting a state, $p(s)$ is equal to the probability of visiting its mirrored counterpart, $p\left(s^{\text{mirrored}}\right)$; this depends on the transition dynamics.

4.5 Proposed algorithm

To demonstrate our approach, we use MPO, a state-of-the-art actor-critic algorithm described in Abdolmaleki et al. [1]. To encourage the agent to take symmetries into account, we augment the batch of experiences generated by the *actor* and stored in the *replay buffer*, by computing a set of corresponding mirrored trajectories for use by the *learner* of the MPO algorithm [1]. The MPO algorithm has two steps, **policy evaluation** and **policy improvement**. In this section, we summarize the algorithm and specify the modifications required to use data from mirrored trajectories, with a pseudocode of our approach given in Algorithm 1.

Algorithm 1 MPO with Mirrored Data

given batch-size (K), number of actions (N), Q-function $\hat{Q}$, old-policy π_k and replay-buffer
initialize π_θ **from the parameters of** $\pi^{(k)}$
repeat
  Sample states with batch of size N from replay buffer
  Step 1: sample based policy (weights)
  $q(a_i|s_j) = q_{ij}$, **computed as:**
  for j = 1,...,K **do**
    for i = 1,...,N **do**
      $a_i \sim \pi_k(a|s_j)$
      $Q_{ij} = \hat{Q}(s_j, a_i)$
      $q_{ij} \propto \exp(Q_{ij}/\text{temperature parameter})$
    end for
  end for
  Calculate mirrored states and mirrored actions
  Step 2: update parametric policy
  Given the data-set $\{s_j, (a_i, q_{ij})_{i=1...N}\}_{j=1...K}$ and $\{s_j^{\text{mirrored}}, (a_i^{\text{mirrored}}, q_{ij})_{i=1...N}\}_{j=1...K}$,
  Update the policy by taking gradient of following weighted maximum likelihood objective
  $\max_\theta \left(\sum_j^K \sum_i^N q_{ij} \log \pi_\theta \left(a_i|s_j\right) + \sum_j^K \sum_i^N q_{ij} \log \pi_\theta \left(a_i^{\text{mirrored}}|s_j^{\text{mirrored}}\right)\right)$
  (subject to additional (KL) regularization)
until Fixed number of steps
return $\pi_{(k+1)}$

4.5.1 Policy evaluation

We employ a 1-step temporal difference (TD) learning, fitting a parametric Q-function $Q^{\pi}_{\phi}(s, a)$ with parameters ϕ by minimizing the squared TD error,

$$\min_{\phi} \left(r_t + \gamma Q^{\pi^{(k-1)}}_{\phi'} (s_{t+1}, a_{t+1}) - Q^{\pi^{(k)}}_{\phi} (s_t, a_t) \right)^2,$$

where $a_{t+1} \sim \pi^{(k-1)}(a|s_{t+1})$ and $r_t = r(s_t, a_t)$, which we optimize via gradient descent. We let ϕ' be the parameters of a target network that is held constant for 250 steps (and then copied from the optimized parameters ϕ). Using the fact that for a symmetric policy, $Q\left(s^{\text{mirrored}}, a^{\text{mirrored}}\right) = Q(s, a)$, we also fit the parametric Q-function for $Q\left(s^{\text{mirrored}}, a^{\text{mirrored}}\right)$,

$$\min_{\phi} \left(r_t + \gamma Q^{\pi^{(k-1)}}_{\phi'} (s_{t+1}, a_{t+1}) - Q^{\pi^{(k)}}_{\phi} \left(s^{\text{mirrored}}_t, a^{\text{mirrored}}_t \right) \right)^2.$$

4.5.2 Policy improvement

Step 1: We construct a non-parametric improved policy q. This is done by maximizing $\bar{J}(s, q) = \mathbb{E}_{q(a|s)}[\hat{Q}(s, a)]$ for states s drawn from a replay buffer $\mathcal{R}$ while ensuring that the solution stays close to the current policy π_k; i.e., $\mathbb{E}_{s\sim\mathcal{R}}[\mathrm{KL}\left(q(a|s), \pi_k(a|s)\right)] < \epsilon$, as detailed in Abdolmaleki et al. [1].
Step 2: We fit a new parametric policy to samples from $q(a|s)$ by solving the optimization problem $\pi_{k+1} = \mathrm{argmin}_{\pi} \mathbb{E}_{\mu(s)}[\mathrm{KL}\left(q(a|s) || \pi(a|s)\right)]$, where π_{k+1} is the new and improved policy, which employs additional regularization [1]. To learn from mirrored data, we repeat steps 1 and 2 for mirrored states and mirrored actions calculated from original data.

4.6 Experiments

We refer to training with the MPO algorithm [1] as "normal training" and training with the additional experience from the mirrored data, as described in Section 4.3, as "augmented training". For the normal and augmented training conditions, we use the same hyper-parameters as MPO [1], with one main modification: we slowed down the actor to get a rate of trajectory generation of 1000 s of environment steps every 30 s of real time, to bring it closer to the data-generation rate of real robots. The learner thus operates in a data-limited regime. The resulting cumulative reward as a function of gradient steps is shown in

Figure 4.2. For each of the training conditions, the augmented training condition achieves better performance, typically in fewer episodes. This preliminary result suggests that knowledge of symmetry, enforced through data augmentation in the policy optimization step, is a useful bias for the agent to shape its policy.

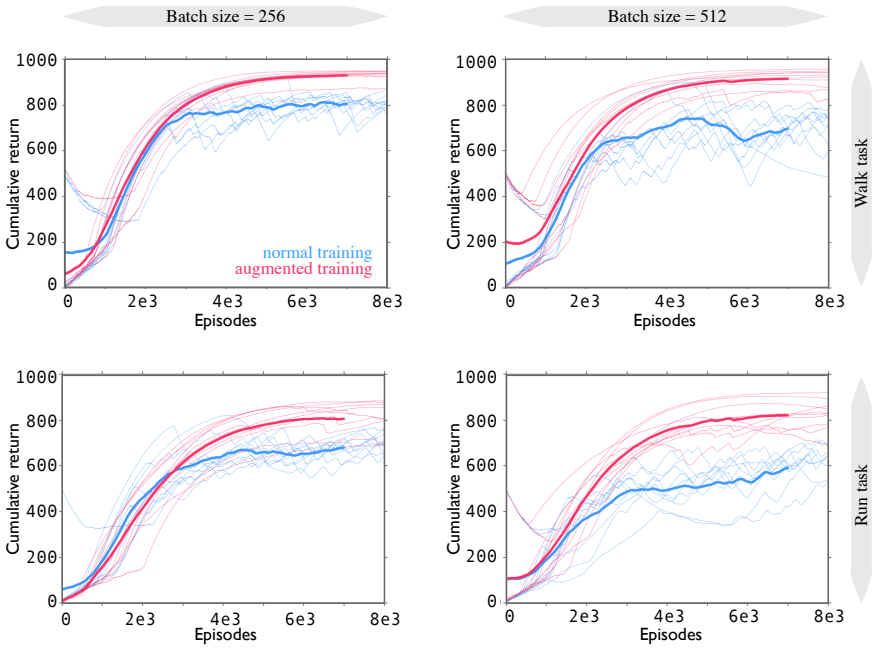


Figure 4.2: **Learning curves** for 10 seeds of the walk (top row) and run (bottom row) tasks, for batch sizes 256 (left column) and 512 (right column). The plot shows the cumulative reward as a function of gradient steps. The normal training and augmented training are in blue and pink, respectively. The thick line shows the mean.

4.7 Discussion

We presented an approach for incorporating symmetry in the environment in an actor-critic architecture by augmenting the experiences generated by the actor. Our work demonstrates a benefit of using symmetry in the walk and run tasks of the quadruped domain. We plan to extend our approach to investigate the effect of using symmetry to augment learning in other domains that exhibit symmetry, such as the humanoid and planar walker.

A different approach to inform an agent of symmetry in its domain could, for instance, compress the state representation, explicitly specifying the relationship between an observation-action pair and its mirrored version. This involves changing the underlying Markov Decision Process so that transitions are between

"canonical" observations. In general, this means that transitions do not arise from a physical process and may require modifications to the algorithm to ensure that no undesirable bias has been introduced.

A promising extension of this work would be to move towards a policy that explicitly encodes the symmetry using hierarchy, with similar legs sharing their lower-level policy with the corresponding actuators of other legs, and using experiences from the other legs to update their policies. Using symmetric policies leads to a change in the space of policies that can be achieved and therefore qualitative change in gait. We plan to investigate this in the future.

In general, the approach of using invariances in a Markov Decision Process to inform the learning of an agent can also be useful for transfer learning to tasks which do not share the same properties. For instance, in the context of the work presented here, an agent may first learn to walk using a knowledge of symmetry to be data-efficient, and then learn to navigate an external non-uniform landscape or modify its gait after sustaining an injury to one leg.

References

[1] Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018.

[2] Henry S Baird. Document image defect models. In *Structured Document Image Analysis*, pages 546–556. Springer, 1992.

[3] Grigory Isaakovich Barenblatt. *Scaling, self-similarity, and intermediate asymptotics: dimensional analysis and intermediate asymptotics*, volume 14. Cambridge University Press, 1996.

[4] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

[5] Steven Bohez, Abbas Abdolmaleki, Michael Neunert, Jonas Buchli, Nicolas Heess, and Raia Hadsell. Value constrained model-free continuous control. *arXiv preprint arXiv:1902.04623*, 2019.

[6] Zhaoming Xie Xue Bin Peng Farzad Abdolhosseini, Hung Yu Ling and Michiel van de Panne. On learning symmetric locomotion. *In Motion, Interaction and Games*, 2019.

[7] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[9] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *International Conference on Learning Representations*, 2016.

[10] Josh Merel, Leonard Hasenclever, Alexandre Galashov, Arun Ahuja, Vu Pham, Greg Wayne, Yee Whye Teh, and Nicolas Heess. Neural probabilistic motor primitives for humanoid control. *arXiv preprint arXiv:1811.11711*, 2018.

[11] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484, 2016.

[12] Patrice Y Simard, David Steinkraus, and John C Platt. Best practices for convolutional neural networks applied to visual document analysis. In *Icdar*, volume 3, 2003.

[13] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. DeepMind control suite. Technical report, January 2018.

[14] Kenneth G Wilson. Renormalization group and critical phenomena. I. Renormalization group and the Kadanoff scaling picture. *Physical Review B*, 4(9):3174, 1971.

Chapter 5

Skill composition using multi-objective policy optimization

The work in this chapter has been done in collaboration with Abbas Abdolmaleki, Nicolas Heess and Doina Precup, and involved discussions during meetings with Alexandre Galashov, Leonard Hasenclever, Josh Merel and others at DeepMind.

Abstract

We explore policy composition using the Multi-Objective Maximum a Posteriori Policy Optimization algorithm [1]. The idea is to use pre-existing teacher policies to enable reinforcement learning agents to learn successful policies for a new task that bears resemblance to the tasks that the teachers were trained on. The teacher policies are introduced as additional task objectives. We show that teacher policies can help speed up learning, particularly in the absence of shaping rewards. In two domains with continuous observation and action spaces, our agents successfully compose teacher policies spatially and temporally, and are also able to further extend the policies of the teachers. Depending on the particular combination of task and teacher(s), teacher(s) may naturally act to limit the final performance. The extent to which agents are required to adhere to teacher policies are determined by hyperparameters which determine both the effect of teachers on learning speed and the eventual performance of the agent on the task. Extensions of this work include equipping agents with various abilities to control the selection of teachers.

5.1 Introduction

In recent years, reinforcement learning (RL) agents have been trained to perform tasks to a level that meets or exceeds the abilities of human experts, particularly in the context of games [18, 14, 15, 13, 7]. In simulated robotics and animal-like domains, RL agents have been trained from scratch to move through their respective environments using various types of locomotion and navigation behaviors, involving crawling [6], gliding [11], running [3, 12], swimming [3, 12, 4], and traversing uneven terrain [5]. RL agents in these domains are characterized by significant simplifications in the mathematical models of the physics engines that create the environment, e.g., [17], resulting in unnatural behaviors [3, 12, 5], or by reduced observation and action spaces, specific to the task in consideration, rather than the agent and environment in general [6, 11, 4].

Limitations to the environmental complexity that RL agents can cope with are, in part, a result of the large amount of experience required by RL agents before they are able to execute tasks adequately. As such, efforts to improve sample efficiency in RL algorithms hold the promise to enable agents that can execute tasks in more realistic sensorimotor settings, both for understanding animals, and understanding and creating robots. Moreover, in biological and biologically-inspired domains, it is especially true that the learning of tasks can rely on policies that are successfully able to solve related tasks – a tiger cub learning to hunt does not have to relearn how to walk and run, but can call upon those behaviors it is already able to execute. Similarly, we might imagine that a robot that is learning to solve a navigational task does not need to relearn how to locomote in its specified domain. It makes sense for these agents to reuse knowledge to make learning more efficient.

5.2 Scope

Motivated by the considerations of computational efficiency and biological plausibility, we seek to develop a method for incorporating pre-existing *skills* in solving a task using RL. A task is described using the framework of a Markov Decision Process (MDP) [9], which is defined as a tuple,

$$\mathcal{M} \equiv (\mathcal{S}, \mathcal{A}, p, r, \gamma)\,, \tag{5.1}$$

where $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, respectively, of an agent in an environment, p and r specify the probability distribution of the next state s' and

the expected reward, respectively, after taking an action a in a state s. For a specified MDP, an agent's objective is to learn a policy $\pi(a|s)$ that maximizes a discounted sum of rewards,

$$\sum_{k=0}^{K} \gamma^k \mathbb{E}_{\pi(a|s)} [r_k], \tag{5.2}$$

with γ being the discount factor, and k denoting discrete steps taken by the agent in the space of states and actions.

We are concerned with scenarios where a RL agent has access to pre-existing skills, in the form of policies. We refer to a pre-existing policy as a *teacher policy*, $\pi_{\text{teacher}}(o|a)$, which is the probability π of taking an action $a \in \mathcal{A}$ for a given observation $o \in \mathcal{O}$, where $\mathcal{A}$ and $\mathcal{O}$ are the action and observation spaces, respectively. Here, $\mathcal{O} \subseteq \mathcal{S}$ is the part of the state-space that can be accessed by the agent. Transitions take place from one state to the next, and the policy is executed over the space of observations. An RL agent does not, a priori, have a policy that can solve the task. It may use the teacher policies, $\pi_{\text{teacher}}(o|a)$, to inform its learning. For simplicity, our work is limited to the case where, for a particular task, the observation and action spaces, $(\mathcal{O}, \mathcal{A})$, are the same across all teachers and the task, and therefore do not need to be distinguished.

In general, teacher policies are limited in scope; imitating the teacher policy may not be sufficient to solve the task. Additionally, teacher policies may compete with each other and be counterproductive to solving the task in different parts of $(\mathcal{O}, \mathcal{A})$. The method for combination of the teacher policies can be a choice made by the agent, or encoded as part of the learning algorithm.

Governed by our choice to study RL in biologically-inspired settings, we consider tasks in continuous-control settings, where the observation and action spaces are both vectors with components that are real-valued, i.e., $(\mathcal{O}, \mathcal{A}) \in \left(\mathbb{R}^{|\mathcal{O}|}, \mathbb{R}^{|\mathcal{A}|}\right)$, with $|\mathcal{O}|$ and $|\mathcal{A}|$ being the number of components in the observation and action vectors, respectively.

5.3 Literature on skill composition

As with most scientific endeavors, our work bears connections and parallels to work in many fields. While we touch on these throughout the text, this section focuses on methods in the literature on composing skills using RL, and introduces concepts that carry over into the remainder of this chapter. Broadly, there are two sets of approaches to composing skills: policy–based composition

(e.g., [10, 8]), and value–based composition (e.g., [2]). In both of these, the composition of skills is done by attributing weights to the underlying skills, and combining the weighted skills in an additive [10, 2] or multiplicative [8] manner. The goal of the RL agent is to learn a combination of weights through the task objective, specified in terms of a reward r.

The difference between policy–based methods and value–based methods arises in the notion of *skill*, and how these may be combined. In policy–based methods, a skill i is specified in terms of a policy $\pi_i(a|o)$. While, in general, the policy is stochastic, it does not change as a function of the task. In value–based methods, skills are combined using an internal notion of *value*. This difference can be illustrated through a 1-step MDP, i.e., $K = 0$ in Equation 5.2. The value function $Q(s_0, a) \equiv r(a)$ for a single initial state s_0. We define reward functions $r^{(T_1)}$ and $r^{(T_2)}$ for two tasks, T_1 and T_2, in a discrete action space $\mathcal{A} \equiv \{a^{(1)}, a^{(2)}, a^{(3)}\}$. Table 5.1 shows these example MDPs, as well as MDPs constructed by linearly combining the rewards associated with these primitive MDPs. In the combined MDPs, policy–based methods will choose from the actions corresponding to the primitive skills, in this case actions $a^{(1)}$ and $a^{(3)}$, whereas value–based methods will choose from actions corresponding to the highest value for the combined task reward, which also allows for the possibility of choosing action $a^{(2)}$. Moreover, value–based methods are sensitive to the scale of the rewards, which is shown by comparing the actions with the highest value in the last two rows of Table 5.1, which are different as the scale of the reward for T_2 changes.

Table 5.1: **Q-values for 1-step MDPs** under primitive, rescaled, and composed reward functions, r. For each reward, the action, a, with the highest value, $Q(a)$, is highlighted in **bold text**.

	$Q\left(a^{(1)}\right)$	$Q\left(a^{(2)}\right)$	$Q\left(a^{(3)}\right)$
$r^{(T_1)}$	**0.6**	0.4	0.0
$r^{(T_2)}$	0.0	0.4	**0.6**
$r^{(T_2),\text{ rescaled}} = 10r^{(T_2)}$	0.0	4.0	**6.0**
$r^{\text{new}} = 0.5\left(r^{(T_1)} + r^{(T_2)}\right)$	0.3	**0.4**	0.3
$r^{\text{new, rescaled}} = 0.5\left(r^{(T_1)} + r^{(T_2),\text{ rescaled}}\right)$	0.6	2.2	**3.0**

5.4 Method

In the context of a RL task, we define a *skill* to be a pre-existing policy, called a *teacher policy*, $\pi_{\text{teacher},i}(a|o)$, that can be accessed by an agent, where i denotes the index of a teacher. In this choice of what constitutes a skill, our work falls under the category of policy–based skill composition methods, as described in Section 5.3. In this section, we describe our method for skill composition, and reference the underlying algorithms that our method is based on.

We use the Multi-Objective Maximum a Posteriori Policy Optimization (MO-MPO) algorithm [1] to compose skills. Using this algorithm, we specify the objectives to correspond to the following:

- The standard RL objective for a task, as specified in Equation 5.2,
- The KL divergence from one or a number of teacher policies,

$$\text{KL}\left(\pi(a|o)\|\pi_{\text{teacher},i}(a|o)\right). \tag{5.3}$$

Each of the objectives is weighted by a scalar parameter ϵ, which specifies the relative strength to which the MO-MPO agent is required to satisfy the corresponding objective.

5.4.1 Teacher policies

In Section 5.2, we stated that the teacher policies are pre-existing policies, as far as the agent is concerned. Unless specified otherwise, the teacher policies are obtained by training an agent on an appropriately specified task, denoted to be the *teacher task*. After the agent has learned a successful policy for the teacher task, this is frozen and used in Equation 5.3.

5.4.2 Types of composition

We consider the scenarios where teacher policies are relevant to distinct parts of the observation space, and scenarios where teacher policies that are relevant to overlapping parts of its observation space. In the framework of an MDP for continuous-control settings, these two scenarios approximate the temporal and spatial composition of policies, respectively. As such, our approach offers the

flexibility to bridge the spatial and temporal methods of policy composition, and compose expert knowledge from any subset of the task or observation description. We note that in general, one may consider analogous compositions in terms of different parts of the action space, but this is beyond the scope of the experiments conducted as part of this work.

5.4.3 Domains and tasks

We use two continuous-control domains from the DeepMind Control Suite [16]. The domains and corresponding tasks are chosen to highlight a particular type of skill composition that can be achieved with our method. The first domain is the `humanoid`. There are three tasks for this domain, `stand`, `walk` and `run`. For each of these tasks, the `humanoid` is initialized in a pose of random joint configurations, some distance above the ground. It then falls to the ground under gravity and the agent must either act to obtain an upright orientation (`stand` task) or maintain an upright orientation at a fixed value of forward velocity in its local frame of reference (`walk` and `run` tasks). The second domain is the `point_mass`, where a mass is initialized at a random location in a two-dimensional square arena. The agent's task is to minimize the distance of the `point_mass` from a specified target location.

5.5 Experiments

In this section, we describe experiments conducted using the method specified in Section 5.4. Each of our experiments is designed to highlight a particular type of composition, as described in Section 5.4.2. In each set of experiments that is described here, we first specify the teachers used by the agent(s), then describe the experiments conducted using these teachers, followed by a description of the results.

5.5.1 Humanoid domain, using a stand teacher

In these experiments, we consider scenarios where teachers are relevant for part of the observation space, but may be uninformative, or counterproductive, to attaining the task objective in a different part of the observation space. This shows the capability of our agents to use the task reward to go beyond what is specified via the teacher policies.

We use a single `stand` teacher, which is obtained by training a MO-MPO agent on the `humanoid` `stand` task. For the `stand` task, a `stand` teacher provides a successful policy. For the `walk` and `run` tasks, a `stand` teacher provides a portion of a successful policy – an agent that can get up after it has fallen on the ground. It must then further learn how to move forward at the respective speeds that correspond to the `walk` and `run` tasks. We ran experiments to see the effect of using a single `stand` teacher policy on tasks in the `humanoid` domain.

To understand the effect of teachers on the learning, we modified the task reward to be sparse along the parameters that specify the upright orientation and head height of the `humanoid`. Specifically, setting `upright_margin = 0` and `stand_margin = 0` make the shaping reward sparse for those aspects of the task, which are: to maintain an upright orientation of the torso, and to maintain a head height above a threshold.

Figure 5.1 shows the effect of using the `stand` teacher policy on the `stand` task. In all settings of reward sparsity, we see that increasing the constraint imposed by the teacher causes a speed up of learning. In the sparse reward settings, the agents with teachers do not see much reduction in learning speed, since a shaping reward is effectively provided by the teacher. In contrast, the agents without teachers learn more slowly.

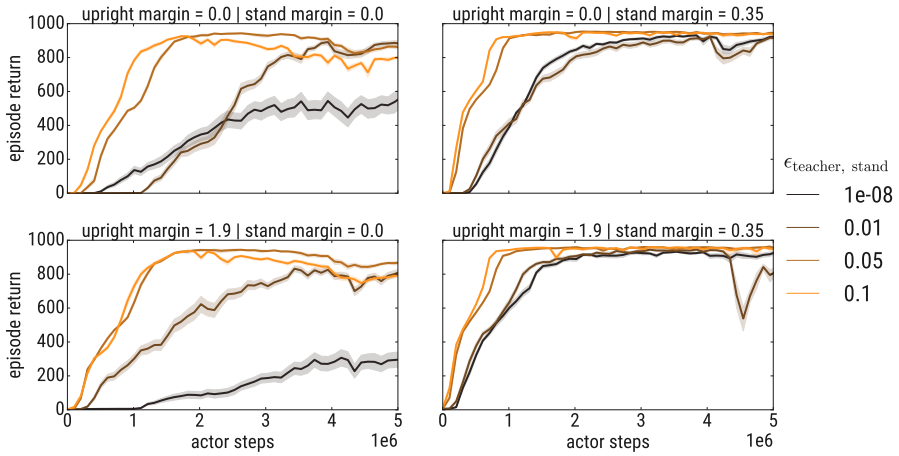


Figure 5.1: **Learning curves for 5 seeds of the `stand` task with a `stand` teacher**, for different values of $\epsilon_{\text{teacher}}$ (shown in different colours), with $\epsilon_{\text{task}} = 0.1$. The different panels correspond to different values of reward sparsity. The episode return is shown for actor steps at intervals of 1e5. The thick lines and shading correspond to the mean values and a 95% confidence interval, respectively.

Figures 5.2 and 5.3 show the effect of using the `stand` teacher policy on the

`walk` and `run` tasks, respectively. In all cases, the agents with teachers learn faster when they are early in learning. Later in learning, there may be a crossover, where some of the agents without teachers start to achieve higher task performance. This is because the agents with teachers have their performance limited by the requirement to adhere to the policy of `stand` teacher, which does not have any noticeable forward velocity. The existence and location of a crossover depends on the value of $\epsilon_{\text{teacher}}$.

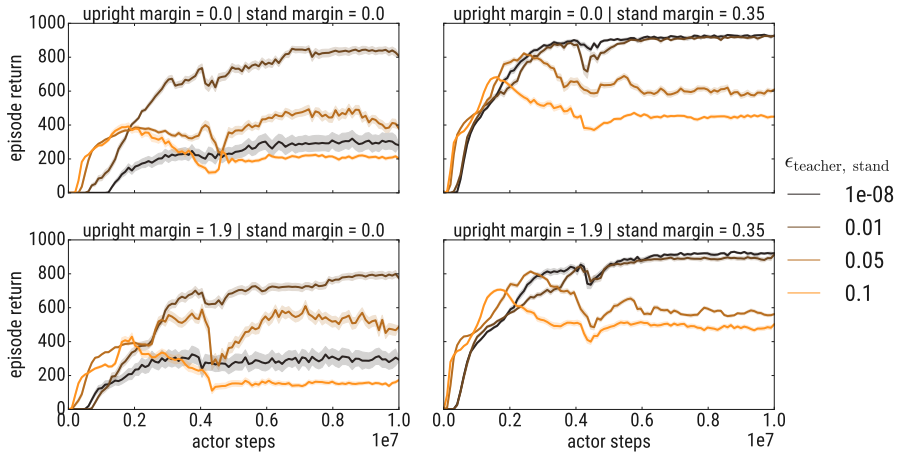


Figure 5.2: **Learning curves for 5 seeds of the `walk` task with a `stand` teacher**, for different values of $\epsilon_{\text{teacher}}$, with $\epsilon_{\text{task}} = 0.1$. The notation is the same as that of Figure 5.1.

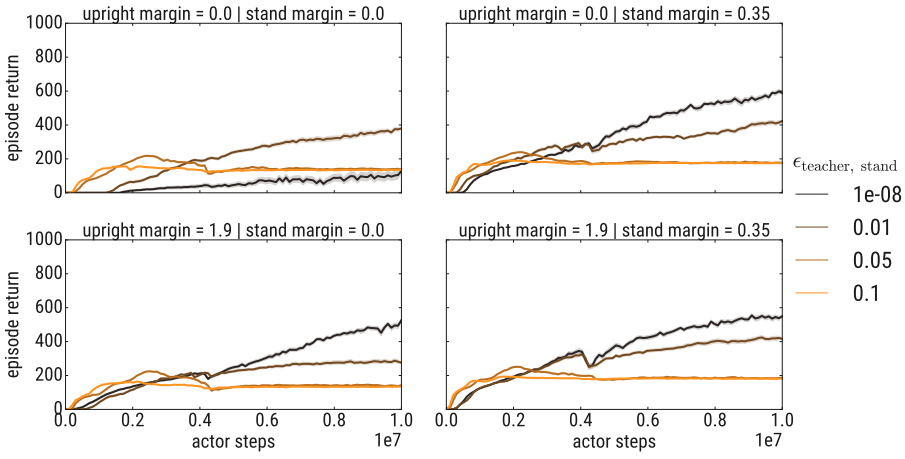


Figure 5.3: **Learning curves for 5 seeds of the `run` task with a `stand` teacher**, for different values of $\epsilon_{\text{teacher}}$, with $\epsilon_{\text{task}} = 0.1$. The notation is the same as that of Figure 5.1.

5.5.2 Humanoid domain, using stand and walk teachers

In this section, we consider the `humanoid walk` task with two teachers, to explore composition with multiple teacher policies. In addition to a `stand` teacher policy, we have a `walk` teacher policy. This policy has been trained on a `humanoid walk` task with a modification: the agent is always initialized in an upright orientation. If the height of its head above the ground is less than a threshold, the episode is terminated. This means that the policy learns how to execute a `walk` from an initial upright orientation. However, it never learns to regain balance once it has been lost. This is to separate the expertise of the `walk` teacher from the `stand` teacher; if the `walk` teacher were able to regain balance, there would be no need for a distinct `stand` teacher. This configuration of teachers means that the teacher policies are relevant to distinct parts of the observation space, and the agent is required to compose policies temporally.

As in Section 5.5.1, we consider tasks with and without sparse rewards. In Section 5.5.1, we already saw the effect of the `stand` teacher in speeding up learning for different values of reward sparsity for the components of the reward corresponding to an upright orientation and head height above a threshold. As such, in this section, we consider sparsity of the reward only for the speed of the `humanoid`.

In Figures 5.4 and 5.5, we see the effect of using `stand` and `walk` teachers on the `walk` task, for regular and sparse versions of the `move` component of the reward, respectively. For the `walk` and `run` tasks, the `move` reward is the component of the reward for attaining forward velocity in the local frame of reference of the `humanoid`. For both panels, the baseline performance is the no teacher case, which is the black curve in panel 1 (left most panel). In Figure 5.4, with the usual reward parameters, we do not see a speed up in learning. Additionally, when $\epsilon_{\text{teacher, walk}} \gg \epsilon_{\text{teacher, stand}}$, there is a collapse of the returns (panel 1 of Figure 5.4). This is likely because the `walk` teacher does not have an informative policy for being able to regain balance, and as such, adhering strongly to only the `walk` teacher will cause a collapse in performance. In Figure 5.5, with the sparse reward setting, we do see some effect of the teachers in speeding up the learning. However, this is sensitive to the specific values of $\epsilon_{\text{teacher, stand}}$ and $\epsilon_{\text{teacher, walk}}$.

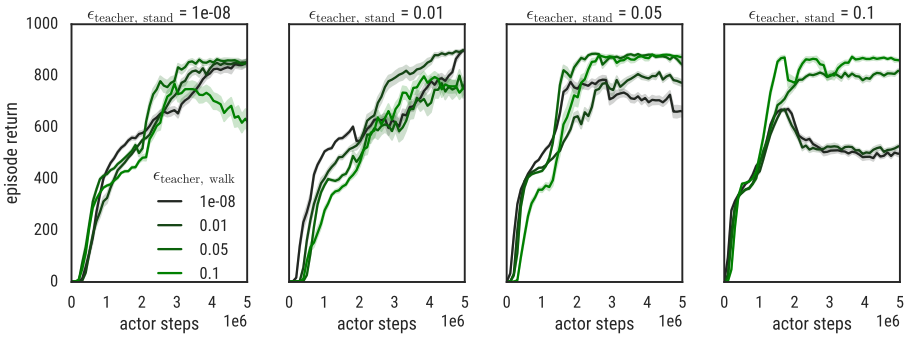


Figure 5.4: **Learning curves for 5 seeds of the `walk` task with `stand` and `walk` teachers.** The different panels show different values of $\epsilon_{\text{teacher, stand}}$, increasing from left to right. The different colours show different values of $\epsilon_{\text{teacher, walk}}$, increasing from black to green. The episode return is shown for actor steps at intervals of 1e5. The thick lines and shading correspond to the mean values and a 95% confidence interval, respectively.

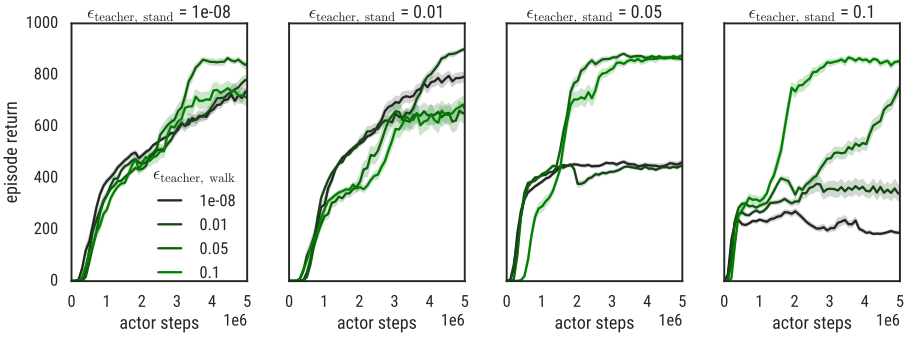


Figure 5.5: **Learning curves for 5 seeds of the `walk` task with `stand` and `walk` teachers. The task reward is sparse for the `move` component of the reward.** The notation is the same as that of Figure 5.4.

5.5.3 Humanoid domain, demarcating the observation space

Noting that the `stand` and `walk` teacher policies are relevant for different parts of the observation space, we encode this information explicitly in the teacher policy. Specifically, we now have only one teacher policy which is a function of the observation, $\pi_{\text{teacher}}(o)$. When the head height of the `humanoid` is below a threshold, $\pi_{\text{teacher}}(o) = \pi_{\text{teacher, stand}}$. When the head height of the `humanoid` is above a threshold, $\pi_{\text{teacher}}(o) = \pi_{\text{teacher, walk}}$. The `walk` component of the teacher is obtained from the `walk` teacher of Section 5.5.2. This setting is designed to further explore the case of an agent being able to usefully compose between

multiple teachers with different specializations relevant to different parts of the observation space, i.e., compose policies temporally.

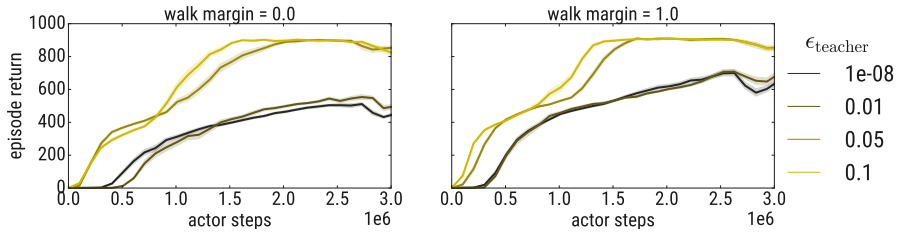


Figure 5.6: **Learning curves for 5 seeds of the `walk` task with the `stand` and `walk` teachers being active in distinct parts of the observation space.** The episode return is shown for actor steps at intervals of 1e5. The thick lines and shading correspond to the mean values and a 95% confidence interval, respectively.

Figure 5.6 shows learning curves for the `walk` task with the `stand` and `walk` teachers being active in distinct parts of the observation space. In this case, there is a single value of $\epsilon_{\text{teacher}}$ that applies to the active teacher. We see a speed up in learning for the sparse as well as regular settings of the reward. The learning is stable across the different values of $\epsilon_{\text{teacher}}$. Moreover, we note that agents are able to successfully stitch together between the discontinuous sub-policies of the teacher, and likely use the objective corresponding to the task reward to enable this. This offers a flexibility beyond the use of skills as components to a weighted policy.

5.5.4 Point mass domain

We constructed a task where the teacher policies are relevant in the same part of the observation space, i.e., composition of teacher policies must be done spatially. Here, each of the teachers solves a task to go to one of two lines, $x = 0$ or $y = 0$. Figure 5.7 shows the trajectories of the `point_mass` in its two-dimensional arena for the policies of the two teachers.

Figure 5.8 shows the trajectories of the composed policies in the `point_mass` domain. We can see that agent policies are successfully composed from the two teachers, without an explicit task reward. For each teacher policy, the `point_mass` goes to one of two intersecting lines. In the trained policies from a composition of these teachers, the `point_mass` goes to the intersection of these lines. Figures 5.9 and 5.10 show that the teachers act to speed up learning.

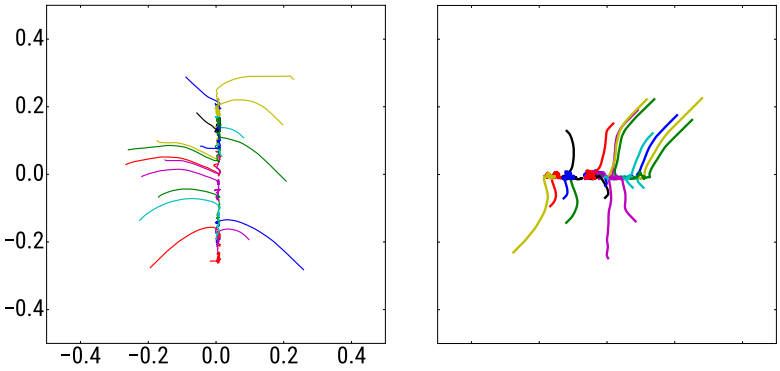


Figure 5.7: **Trajectories for the trained teacher policy** for the target $x = 0$ (left) and $y = 0$ (right).

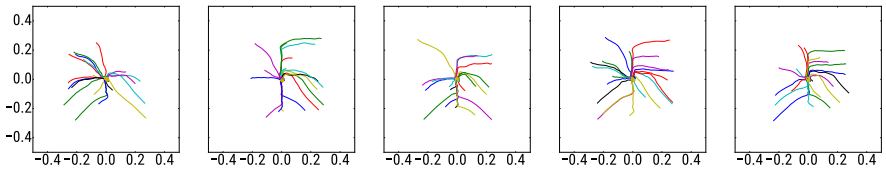


Figure 5.8: **Trajectories for the composed policy**, with both teachers having equal values of ϵ, and very small ϵ_{task}. Each panel is a different seed.

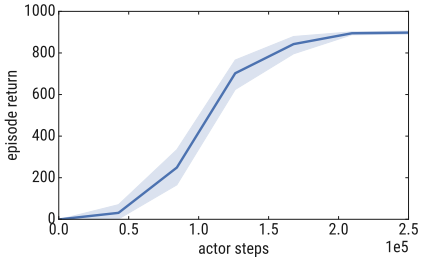


Figure 5.9: **Learning curves for 5 seeds of the `point_mass` task with the target at $(0, 0)$, with no teachers.** The thick line and shading correspond to the mean and a 95% confidence interval, respectively.

5.6 Discussion and future work

We used the MO-MPO algorithm [1] to incorporate, in addition to the task objective, a penalty on the divergence from teacher policies that are known to be experts for sub-tasks that are characterized either based on the observation (e.g., the `stand` and `walk` tasks in the `humanoid` domain) or based on the action

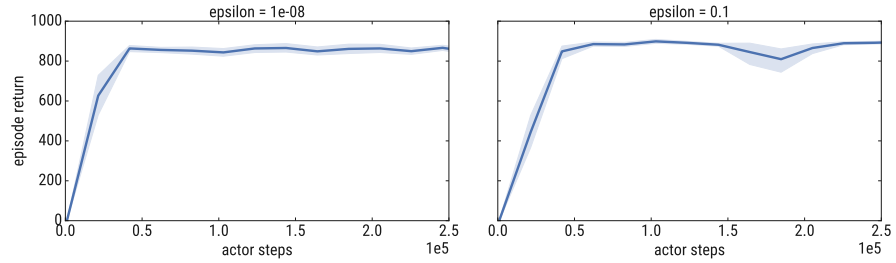


Figure 5.10: **Learning curves for 5 seeds of the** `point_mass` **task to go to the target at** $(0, 0)$**, with teachers** that go to one of two intersecting lines ($x = 0$ or $y = 0$). Both teachers are active at the same time, with $\epsilon = 0.1$. The two panels show different values of ϵ_{task}. The thick lines and shading correspond to the mean values and a 95% confidence interval, respectively.

(e.g., move vertically and horizontally in the `point_mass` domain). In doing so, we successfully show that our agents can compose policies in multiple ways, including both spatial and temporal composition, as well as stitch together between discontinuous teacher policies. We also observe that, as expected, the behavior of agents with composed policies is closer to the behaviour of the teachers than the case where teachers are not incorporated in the learning algorithm.

5.6.1 Principled selection of teachers

We have thus far explored selections of teachers where the teacher policies are relevant to overlapping or distinct parts of the observation space. In both of these cases, the relevance of the teacher has been specified through hard-coding the $\epsilon_{\text{teacher},i}$. In general, an agent may want to control this.

5.6.1.1 Teacher selection by the agent

The experiments described in Section 5.5 consider teachers that are specified to the agent. In some cases, such as in Section 5.5.3, the teacher policies are a function of the observation, but this function is specified to the agent.

In subsequent work, we would like to allow the agent to have control over the influence of the teacher policy. This can be done by modifying the action space, $\mathcal{A}$, of the original MDP to include additional action components, $a_i = \epsilon_{\text{teacher},i}(o)\ \forall\ i \in \{0, \ldots, N_{\text{teacher}} - 1\}$, where N_{teacher} is the number of teachers.

In doing so, the agent may encounter the following types of degeneracy:

- A distinct value of $\epsilon_{\text{teacher},i}(o)$ is chosen for each observation. Allowing the agent to have this level of control is likely to lead to overfitting, since we start with the assumption that relevant teacher policies help shape learning for finite portions of the observation space. Additionally, if this is not the case, then the teacher(s) will not provide a useful signal, as requiring a unique action component, $a_i = \epsilon_{\text{teacher},i}(o)$, for each observation, o, will essentially amount to learning a policy for the original MDP.
- A value of $\epsilon_{\text{teacher},i}(o) = 0$. We want to allow the agent to have the flexibility to choose this, just as it has the ability to choose $a = 0$ for any other component of the action.

To mitigate the first type of behaviour, we would like to induce a bottleneck from the observation space $\mathcal{O}$ to the space of parameters over which $\epsilon_{\text{teacher},i}(f(o))$ can be chosen.

5.6.1.2 *Population-based training*

For the tasks described thus far, population-based training can provide a way to select between the different options of teacher mixtures. Population-based training may have a greater effect on speeding up learning for tasks that can benefit from a higher dimensional space of teachers, where exploration of the appropriate combinations of $\epsilon_{\text{teacher},i}$ is difficult.

5.6.2 Policy composition in action-space

This type of composition can be useful for some types of tasks. For example, we may consider $\pi_{\text{teacher, walk}}$ to be a teacher policy that is able to walk in some arbitrary direction and $\pi_{\text{teacher, hold}}$ to be a teacher policy that is able to hold a box. A `carry` task can be constructed where the agent uses the $\pi_{\text{teacher, walk}}$ and $\pi_{\text{teacher, hold}}$ for different parts of the action space to carry a box from one location to another. In terms of the notation used in this chapter, we have allowed $\epsilon_{\text{teacher},i}$ where $\epsilon_{\text{teacher},i} = \epsilon_{\text{teacher},i}(o)$. In general, we may want $\epsilon_{\text{teacher},i}(o, a)$. The $\epsilon_{\text{teacher},i}(o, a)$ should pass through a bottleneck such that the space of $(\mathcal{O}, \mathcal{A})$ for the selection of $\epsilon_{\text{teacher},i}$ is restricted. Otherwise, this may degenerate into a restriction of the observation and action spaces, rather than informative sub-policies.

References

[1] Abbas Abdolmaleki, Sandy H Huang, Leonard Hasenclever, Michael Neunert, H Francis Song, Martina Zambelli, Murilo F Martins, Nicolas Heess, Raia Hadsell, and Martin Riedmiller. A distributional view on multi-objective policy optimization. *arXiv preprint arXiv:2005.07513*, 2020.

[2] André Barreto, Diana Borsa, Shaobo Hou, Gheorghe Comanici, Eser Aygün, Philippe Hamel, Daniel Toyama, Shibl Mourad, David Silver, and Doina Precup. The option keyboard: Combining skills in reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 13052–13062, 2019.

[3] Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva TB, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. Distributed distributional deterministic policy gradients. *arXiv preprint arXiv:1804.08617*, 2018.

[4] Simona Colabrese, Kristian Gustavsson, Antonio Celani, and Luca Biferale. Flow navigation by smart microswimmers via reinforcement learning. *Physical Review Letters*, 118(15):158004, 2017.

[5] Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, SM Eslami, Martin Riedmiller, and David Silver. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017.

[6] Shruti Mishra, Wim M van Rees, and L Mahadevan. Coordinated crawling via reinforcement learning. *Journal of the Royal Society Interface*, 17(169):20200198, 2020.

[7] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[8] Xue Bin Peng, Michael Chang, Grace Zhang, Pieter Abbeel, and Sergey Levine. MCP: Learning composable hierarchical control with multiplicative compositional policies. In *Advances in Neural Information Processing Systems*, pages 3686–3697, 2019.

[9] Martin L Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2 edition, 2014.

[10] Ahmed H Qureshi, Jacob J Johnson, Yuzhe Qin, Taylor Henderson, Byron Boots, and Michael C Yip. Composing task-agnostic policies with deep reinforcement learning. In *International Conference on Learning Representations*, 2020.

[11] Gautam Reddy, Antonio Celani, Terrence J Sejnowski, and Massimo Vergassola. Learning to soar in turbulent environments. *Proceedings of the National Academy of Sciences*, 113(33):E4877–E4884, 2016.

[12] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[13] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484, 2016.

[14] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018.

[15] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354, 2017.

[16] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. DeepMind control suite. Technical report, January 2018.

[17] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[18] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P Agapiou, Max Jaderberg, Alexander S Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L Paine, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Chapter 6

Overall summary and future directions

This thesis describes a series of investigations into the evolution and control of dynamical systems. In this chapter, we present an overview of the themes common to all of the studies. Specifically, we describe commonalities in the ways in which the evolution and control of the systems is considered, the role of numerical simulation in studying these systems, and some directions for future work in line with this theme. For detailed descriptions of each system and a corresponding set of discussions and conclusions, the reader is referred to the individual Chapters 2–5.

6.1 Leveraging structure in the evolution and control of dynamical systems

The work in all four chapters involves the modeling of processes in natural or artificial systems. In modeling and simulating these systems, we leveraged the existing structure in the systems considered. This section summarizes the ways in which structure was leveraged in these individual studies.

The physical process of drop impact on a solid surface in Chapter 2 was distilled to create a computational model, enabling numerical simulations of the fluid dynamics. In creating the computational model, asymptotic analysis of the relative sizes of different terms in the physical model led to a simplification of boundary conditions for the liquid drop. These simplifications also allowed for

a reduction in the domain size for the simulation of the liquid. Symmetry in the boundary and initial conditions about the vertical axis allowed for halving the computational domain for the fluid simulations.

A neuromechanical model of a soft-bodied crawler in Chapter 3 is based on many simplifications of the actual neuromechanics of a model organism, the *D. melanogaster* larva. Based on the considered problem of locomotion via forward crawling, the model was simplified to only consider one-dimensional locomotion. The neural control was achieved via reinforcement learning (RL), and the available observations and actions in the RL system were chosen to minimally and sufficiently represent the control task of the crawler.

Chapter 4 leveraged structure in the domain in the control algorithm – data augmentation was used to inject knowledge of symmetry in the domain, achieving a speed up in the learning of control policies in regimes where the time scales of interaction with the environment are similar to realistic robotic domains. Chapter 5 exploits pre-existing policies for related tasks to speed up the learning and shape the policies of agents in new tasks.

Existing RL algorithms have to deal with the issue of agents having to forget and re-learn how to perform tasks that, to humans, are only slightly different from those that the agents have already been trained on. Chapter 4 showed that when the experiences of the agents are limited, agents can exploit invariances in their environment to learn successful policies faster. Chapter 5 showed that a knowledge of related tasks can be used to speed up and shape the learning of new, more complicated tasks. Both of these chapters assume that a knowledge of the structure in the environment is available to the agent. Future work may consider how agents might be able to derive this knowledge through their experiences.

6.2 Role of numerical simulations

Our work in studying the evolution and control of dynamical systems has been made possible by the availability of numerical simulation tools. While the previous section focused on using system knowledge to enable numerical simulations, this section summarizes the role of numerical simulation in enabling each of the investigations presented in this thesis.

Chapter 2 considered the dynamics of a liquid drop falling towards a solid surface, and interacting with surrounding gas in this process. We modeled

the drop interacting with gas, and captured experimental observations on the effect of viscosity. In this chapter, simulation allowed us to study the evolution of a system that is theoretically intractable. In simulating the flow fields, we also probed flow-field variables that are not accessible to experiment. This is important to further characterize the cause of the viscous dynamics, towards understanding the mechanism. For this system, our work bridges the gap between theory and experiment. To enable the work presented in this chapter, we augmented a fluid solver to be able to describe the specified mathematical model, and released the code for the numerical simulations. By releasing code as part of our work, we further enable future work in using numerical simulation to probe the effect of viscosity, surface tension, and other parameters during the impact of a liquid drop on a solid surface.

Chapter 3 considered a neuromechanical system for a soft-bodied crawler, modeled by simple subsystems for the coupled interactions of the brain and body with the environment. By coupling this with a RL algorithm for the control policy of the neurons, we successfully recovered the biological gait of a peristaltic wave moving from back to front of the body to produce forward motion. In this chapter, numerical simulation allowed us to consider a minimal representation for the neuromechanical system of the crawler together with a RL algorithm, in a way that is not possible via experiments on animals, nor theoretically tractable on account of the complex dynamics even for the minimal model. With the development of connectome data, sophisticated learning algorithms and large computational capacity, future work can also consider the brain–body–environment triad with more realistic descriptions and fewer simplifications for each of these subsystems.

Chapters 4 and 5 considered simulated robot-like systems that are inspired by animals, and developed algorithms for learning control policies for agents to move through a simulated environment. Given the vast amounts of data required to develop successful control policies and the idiosyncratic behaviors that these policies may converge to, resulting, for example, in costly damage to a physical robot, numerical simulations have been crucial to the development of RL algorithms for agents in biologically–inspired environments. Moreover, numerical simulations enable a consideration of isolated and relevant parts of the physical dynamics of a robot, as well as allowing for asynchronously storing and accessing experiences.

Appendix A

Supplementary material: Computing the viscous effect in early-time drop impact dynamics

This has been published as part of **Mishra, S., Rubinstein, S. M. and Rycroft, C. H., 2021. Computing the viscous effect in early-time drop impact dynamics.** [arXiv: 2108.11497]

A.1 Calculations for the governing equation in the gas layer

A.1.1 Derivation of the gas layer pressure update equation

Substituting for ρ_g from the equation of state, Equation 2.5, into the lubrication equation for the pressure in the gas layer, Equation 2.4, we get

$$12\mu_g \left(\left(\frac{\rho_0}{p_0^{1/\gamma}} p_g^{1/\gamma} \right) h \right)_t = \left(\left(\frac{\rho_0}{p_0^{1/\gamma}} p_g^{1/\gamma} \right) h^3 p_{g,x} \right)_x . \tag{A.1}$$

Dividing both sides by $\rho_0/p_0^{1/\gamma}$

$$12\mu_g \left(p_g^{1/\gamma} h \right)_t = \left(p_g^{1/\gamma} h^3 p_{g,x} \right)_x . \tag{A.2}$$

Using the product rule to expand the brackets and dropping the subscript g for gas,

$$12\mu \left(\frac{1}{\gamma} p^{1/\gamma - 1} p_t h + p^{1/\gamma} h_t \right) = \frac{1}{\gamma} p^{1/\gamma - 1} p_x h^3 p_x + p^{1/\gamma} 3h^2 h_x p_x + p^{1/\gamma} h^3 p_{xx}. \tag{A.3}$$

Dividing both sides by $p_g^{1/\gamma-1}$,

$$12\mu\left(\frac{1}{\gamma}p_t h + p h_t\right) = \frac{1}{\gamma}p_x h^3 p_x + p3h^2 h_x p_x + p h^3 p_{xx}, \tag{A.4}$$

which can be rearranged to give Equation 2.17.

A.1.2 Numerical solution of the gas layer pressure

We now describe how to solve Equation 2.18 in the main text for updating the gas layer pressure using the Newton–Raphson method. Let P^k be a vector containing the pressure estimates at the kth Newton–Raphson iteration. As an initial estimate, we set P^0 to be the pressure values from the nth timestep. We then write Equation 2.18 as a nonlinear system $F(P) = 0$, where the ith component of F is given by the (L.H.S. – R.H.S.) of Equation 2.18 evaluated at the ith gridpoint. An improved estimate P^{k+1} for the pressure is given in terms of P^k by

$$J_F(P^k)\left(P^{k+1} - P^k\right) = -F(P^k), \tag{A.5}$$

where $J_F(P^k)$ is the Jacobian of F, which has components

$$J^{k+1}_{F,ij} = \frac{\partial F_i}{\partial P^{k+1}_j}. \tag{A.6}$$

Since the finite-difference stencils in Equation 2.18 only involve adjacent grid points, J_F is a tridiagonal system. It can be written as

$$J_F = \begin{pmatrix} b_0 & c_0 & & & \\ a_1 & b_1 & c_1 & & \\ & a_2 & b_2 & c_2 & \\ & & \ddots & \ddots & \ddots \end{pmatrix}, \tag{A.7}$$

where the terms are given by

$$a_i = \frac{\partial F}{\partial p^{n+1}_{i-1}} = \frac{1}{\gamma}\frac{\bar{h}^3}{\Delta x}\left(\frac{p^{n+1}_{i+1} - p^{n+1}_{i-1}}{2\Delta x}\right) - \frac{3\bar{h}^2\bar{h}_x}{2\Delta x}p_i^{n+1} + \frac{\bar{h}^3}{\Delta x^2}p_i^{n+1}, \tag{A.8}$$

$$b_i = \frac{\partial F}{\partial p^{n+1}_i} = -12\frac{\mu}{\gamma}\frac{\bar{h}}{\Delta t}(1) - 12\mu\bar{h}_t - \left[\frac{3\bar{h}^2\bar{h}_x}{2\Delta x}\left(p^{n+1}_{i+1} - p^{n+1}_{i-1}\right) + \frac{\bar{h}^3}{\Delta x^2}\left(p^{n+1}_{i+1} - 2p^{n+1}_i + p^{n+1}_{i-1}\right) + \frac{\bar{h}^3}{\Delta x^2}p_i^{n+1}(-2)\right], \tag{A.9}$$

$$c_i = \frac{\partial F}{\partial p^{n+1}_{i+1}} = -\frac{1}{\gamma}\frac{\bar{h}^3}{\Delta x}\left(\frac{p^{n+1}_{i+1} - p^{n+1}_{i-1}}{2\Delta x}\right) - \frac{3\bar{h}^2\bar{h}_x}{2\Delta x}p_i^{n+1} - \frac{\bar{h}^3}{\Delta x^2}p_i^{n+1}. \tag{A.10}$$

Solving Equation A.5 can be done efficiently using LAPACK's tridiagonal solver `dgtsv` [1]. In most cases, since the pressure does not change by a large amount per timestep, fewer than five iterations are required in order to achieve numerical convergence.

A.2 Tests of the simulation performance

Table A.1 contains statistics about the performance of the code for several simulations that were referenced in the main text. All tests were run using ten threads on an Ubuntu Linux computer with a ten-core 2.8 GHz Intel Core i9-10900 CPU, and the code was compiled with GCC version 10.3.

The initial dynamics simulations only need to capture the large-scale deformation of the drop, and can therefore be run on relatively coarse computational grids. Consequently, a typical simulation takes approximately 40 min of wall clock time to run. A large portion of the computation time is spent on solving the linear systems with the multigrid method. This should be expected since the multigrid V-cycles involve repeated scans over the entire grid. Collectively, the four multigrid solves take up 44% of the total computation time. The MAC and FEM linear systems take slightly more time and V-cycles, since apart from where Dirichlet boundary conditions are applied, these linear systems are only weakly diagonally dominant. By contrast, the linear systems for the implicit viscous term are strictly diagonally dominant. We note that the linear systems for the u and v velocity components are slightly different, since at $x = 0$ we apply different boundary conditions, $(u, v_x) = (0, 0)$. Hence it is not possible to vectorize this system, and there is no substantial computational advantage over solving for the updates to u and v separately.

Calculating the boundary conditions according to Equation 2.10 requires applying Simpson's rule along the M gridpoints on the bottom edge. This must be done for the top edge of length M, and the right edge of length N, requiring $O(M(M + N))$ work in total. This is sizable amount of work and takes 3.4% of the total computation time. Even though the gas layer involves several Newton steps and tridiagonal matrix solves, it only needs $O(M)$ work and therefore takes up a minimal amount of the total computation time.

Table A.1 also contains performance information for two lift-off simulations, which are run on larger computational grids to obtain high accuracy in the height of the gas layer. The wall clock time per timestep increases from 81 ms to

1.041 s. Based on a linear scaling with gridpoints, we would expect 81 ms to increase to (81 ms)$\frac{5120\times960}{2048\times256}$ = 760 ms. Thus the performance is comparable, but slightly worse than, linear scaling, which may be due to reduced cache efficiency for a larger grid. Overall, the percentages spent on the different parts of the simulation are comparable, although the fraction spent on multigrid V-cycles increases slightly, and the fraction spent on the gas layer (which scales like $O(M)$) decreases. In particular, more V-cycles are required to handle the implicit viscosity linear system when ν_l is larger.

Table A.1: Performance statistics for several different simulations, compiled using GCC 10.3 on an Ubuntu Linux computer with an 2.8 GHz Intel Core i9-10900 CPU. Ten threads were used, and all simulations use the baseline parameter choices in Tables 2.3 and 2.2, unless otherwise noted. The wall clock (WC) time is reported for each test, and the fraction of time on major components, such as the computation of boundary conditions (BCs), solving the marker-and-cell (MAC) linear system, and solving finite-element method (FEM) linear system, are reported.

	Initial dynamics	Lift-off dynamics	
Liquid viscosity, ν_l (mm^2/s)	10	10	100
Grid size	2048×256	5120×960	5120×960
Total WC time	0.641 h	20.6 h	29.8 h
Total timesteps	28500	71250	95000
WC time / timestep	81 ms	1041 ms	1130 ms
WC fraction on gas layer	0.25%	0.047%	0.045%
WC fraction on BCs	3.39%	1.61%	1.49%
WC fraction on MAC solve	11.21%	13.05%	12.09%
WC fraction on FEM solve	12.77%	14.31%	14.36%
WC fraction on viscous solve	20.11%	26.67%	31.12%
Mean MAC V-cycles	5	4.4	4.5
Mean FEM V-cycles	5.6	4.9	5.3
Mean viscosity V-cycles	3.7	4.3	5.5

A.3 Tests of the simulation accuracy

The simulation domain size, which is set via the non-dimensional parameter $\tilde{L}$ in Table 2.3 and the domain aspect ratio β, may have an effect on the results. Since the boundary conditions on the top and right boundaries are based on an inviscid assumption, the domain size affects the extent to which viscosity

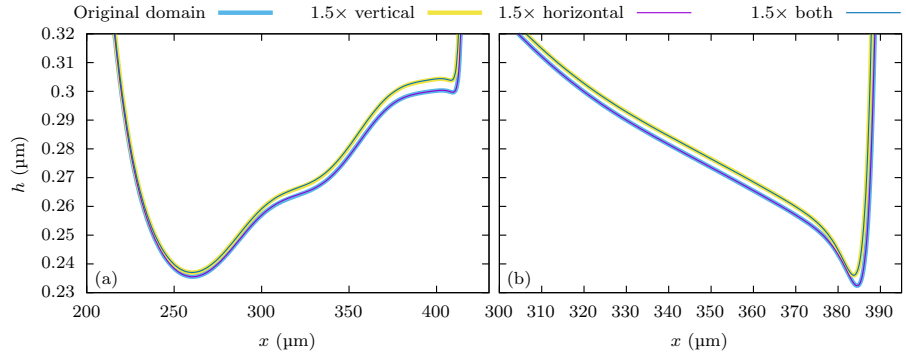


Figure A.1: Plots of the height profiles at t = 102.07 µs for simulations with (a) liquid viscosity $\nu_l = 10\,\mathrm{mm}^2/\mathrm{s}$ and (b) liquid viscosity $\nu_l = 100\,\mathrm{mm}^2/\mathrm{s}$. Baseline parameters are used although the original domain size for both values of ν_l uses $\tilde{L} = 30$ and $\beta = 16/3$. Results are also shown where the simulation domain is extended by a factor of 1.5 in either or both dimensions. In the extended simulations, the number of grid points is increased to keep the grid spacings $\Delta x = \Delta y$ the same.

is resolved. In addition, when solving for the pressure in the gas layer, the boundary condition of $p = P_0$ is imposed at $x = L$, and thus extending the domain affects the influence of this boundary condition on the simulation.

To test the sensitivity of the simulation results to the domain size, we performed simulations where either the horizontal dimension, vertical dimension, or both dimensions were extended by a factor of 1.5. In each of these simulations the grid spacings $\Delta x = \Delta y$ were kept the same, so that the horizontal extension increases the grid points from 5120 to 7680, and the vertical extension increases the grid points from 960 to 1440. Figure A.1 shows the height profiles in the thin gas layer for the four simulations, for (a) $\nu_l = 10\,\mathrm{mm}^2/\mathrm{s}$ and (b) $\nu_l = 100\,\mathrm{mm}^2/\mathrm{s}$. The vertical extensions give visually indistinguishable curves, indicating that vertical dimension is sufficiently large to resolve the viscous effects even for the larger value of ν_l. The horizontal extensions create minor vertical shifts in the curves, suggesting that the pressure boundary condition has an effect on the results. However, these shifts are still acceptably small, and result in no substantial change to the position and time at which lift-off occurs.

We also examined the sensitivity of the results to the numerical grid size. Figure A.2(a) shows height profiles for a simulation using the baseline parameters with liquid viscosity $\nu_l = 10\,\mathrm{mm}^2/\mathrm{s}$ and surface tension $\sigma = 0.072\,\mathrm{N/m}$. Simulations with higher resolutions of 8192×1536 and 12288×2304 are also plotted, resulting in visually indistinguishable results. Figure A.2(b) shows height profiles when the surface tension is set to zero and all other parameters

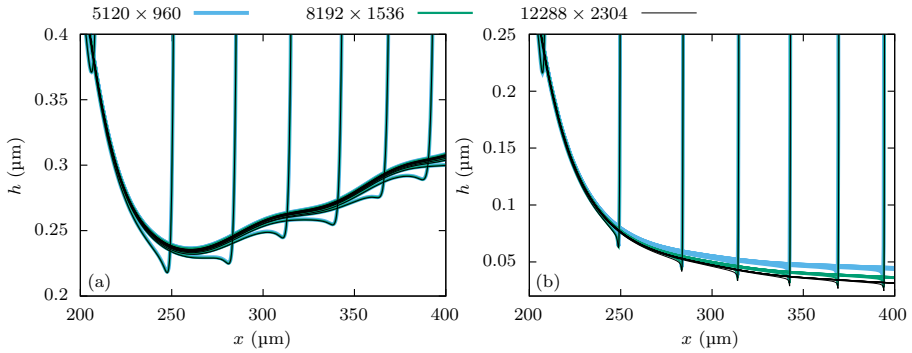


Figure A.2: Plots of the height profiles spaced 6.004 µs apart for three different resolutions using the baseline parameters and liquid viscosity $\nu_l = 10\,\mathrm{mm}^2/\mathrm{s}$, with (a) surface tension $\sigma = 0.072\,\mathrm{N/m}$, and (b) zero surface tension.

are kept the same. In this case, as discussed in Section 2.4.3, the leading tip becomes very sharp since the regularizing effect of surface tension is removed. As the tip moves across the grid, there will be a resolution-dependent numerical diffusion that will act as an effective small surface tension. Thus in this case there is a small shift in the height profiles. While this remains small, it makes the precise behavior of the leading tip difficult to resolve for small values of ν_l, requiring larger grid sizes as presented in Section 2.4.3.

A.4 Additional model fits for lift-off time

In Section 2.4.2, we examine how the lift-off time varies as a function of ν_l. For a sequence of measurements $(\nu_{l,k}, t_k)$, we fit the model $\tau_{dat} = t - t_0^{dat} = \gamma(\nu_l/\nu_{sc})^\alpha$ for the three parameters $(t_0^{dat}, \gamma, \alpha)$. Here $\nu_{sc} = 20\,\mathrm{mm}^2/\mathrm{s}$ is an arbitrary viscosity scale. We minimized the residual

$$S(t_0^{dat}, \gamma, \alpha) = \frac{1}{2}\sum_k \left(\log\gamma + \alpha \log\frac{\nu_{l,k}}{\nu_{sc}} - \log(t_k - t_0^{dat})\right)^2, \tag{A.11}$$

using the L-BFGS-B algorithm for bound-constrained nonlinear optimization [2], and we enforced $t_0^{dat} < 0.999 t_{min}$ and $\alpha > 0$, where $t_{min} = \min_k\{t_k\}$. Figure A.3 shows the results of this optimization for the six different initial drop velocities V, along with the values of the fitted parameters. The mean value of the residual is $\bar{S} = 0.254$.

In the experiments by Kolinski et al. [3], the parameter α is proposed to be $1/2$. We therefore ran an additional optimization where ν_l is constrained to be $1/2$.

The mean residual increases to $\bar{S} = 0.385$ and the model does not capture the curve in the data points (Figure A.4). We ran a further optimization where $\alpha = 1$ (Figure A.5), which results in $\bar{S} = 0.305$. We found that the value of α that minimizes the mean residual is $\alpha = 1.1181$. For this case $\bar{S} = 0.281$ and the optimization results shown in Figure A.6.

Figure A.7 shows an optimization when the time origin is fixed to the original proposed definition of $t_0 = H/V$, where H is the drop height above the surface. In Figure A.8 the exponent is additionally constrained to $\alpha = 1/2$. For each of the six optimizations, the values of S are reported in Table A.2.

Table A.2: Values of the residual from Equation A.11 for the six different optimizations considered. The residual S_V is shown for six different values of the initial drop V, shown in the first column. The different columns show residual in terms of additional constraints that are applied on the three parameters $(t_0^{dat}, \gamma, \alpha)$. The final row shows the mean residual taken over the six values of V.

	Constraints					
		$\alpha = 1/2$	$\alpha = 1$	$\alpha = 1.1181$	$t_0^{dat} = t_0$	$t_0^{dat} = t_0$, $\alpha = 1/2$
$S_{V=0.3\,m/s}$	0.019	0.226	0.093	0.039	0.218	0.227
$S_{V=0.45\,m/s}$	0.260	0.483	0.466	0.345	0.520	0.522
$S_{V=0.6\,m/s}$	0.415	0.523	0.417	0.446	0.572	0.602
$S_{V=0.75\,m/s}$	0.356	0.461	0.358	0.372	0.494	0.578
$S_{V=0.9\,m/s}$	0.264	0.332	0.266	0.273	0.351	0.488
$S_{V=1.05\,m/s}$	0.207	0.287	0.229	0.208	0.314	0.553
$\bar{S}$	0.2536	0.3855	0.3049	0.2805	0.4116	0.4952

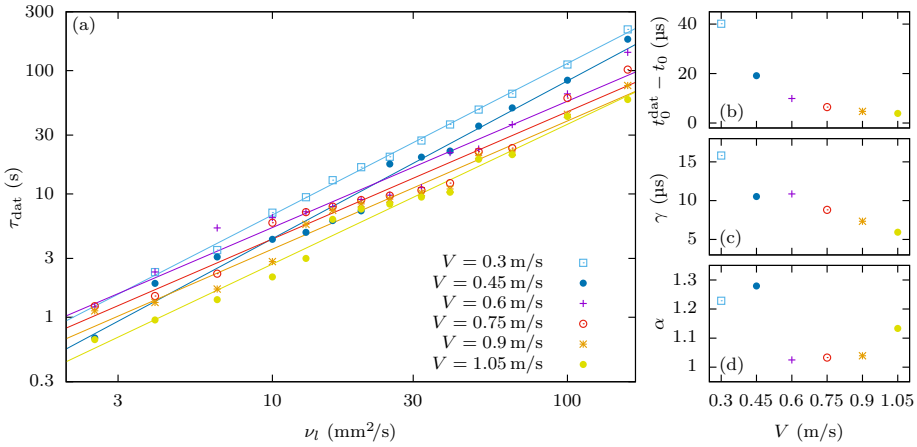


Figure A.3: (a) Model fits for minimizing the residual S for the three parameters $(t_0^{dat}, \gamma, \alpha)$. The straight lines show the fitted model $\tau_{dat} = \gamma(\nu_l/\nu_{sc})^{\alpha}$. (b–d) Values of the parameters t_0^{dat}, γ, and α.

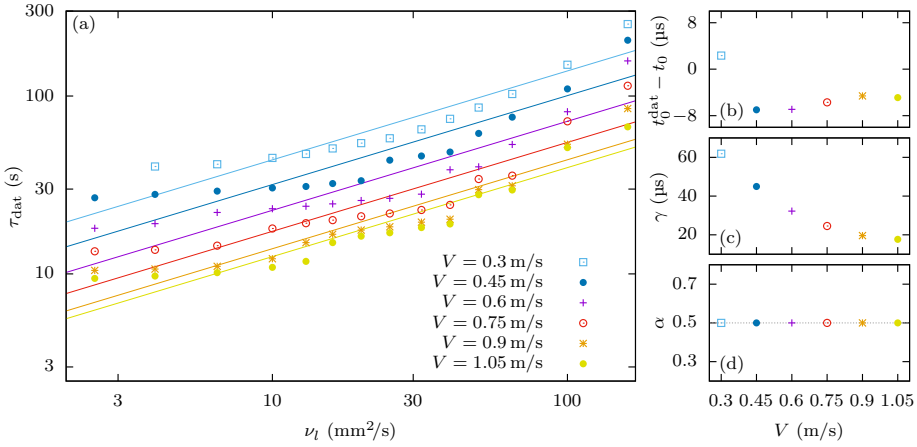


Figure A.4: (a) Model fits for minimizing the residual S for the two parameters (t_0^{dat}, γ) where $\alpha = 1/2$. The straight lines show the fitted model $\tau_{dat} = \gamma(\nu_l/\nu_{sc})^{\alpha}$. (b–d) Values of the parameters t_0^{dat}, γ, and α, with the dotted line indicating the constrained parameter.

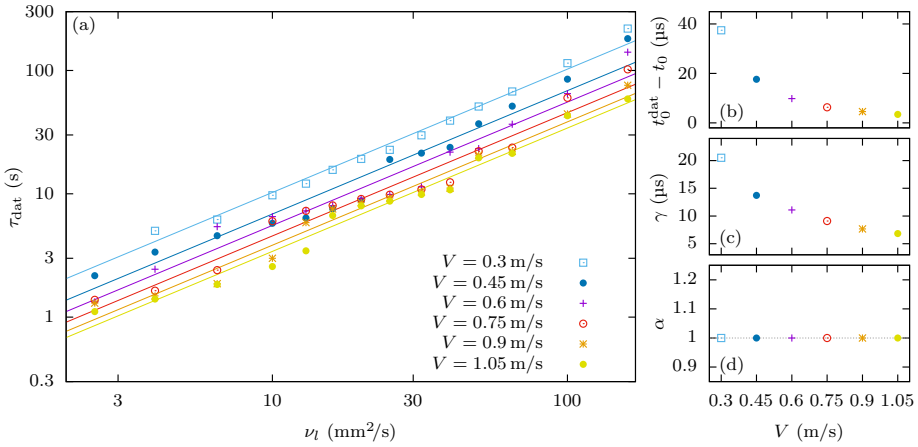


Figure A.5: (a) Model fits for minimizing the residual S for the two parameters (t_0^{dat}, γ) where $\alpha = 1$. The straight lines show the fitted model $\tau_{dat} = \gamma(\nu_l/\nu_{sc})^{\alpha}$. (b–d) Values of the parameters t_0^{dat}, γ, and α, with the dotted line indicating the constrained parameter.

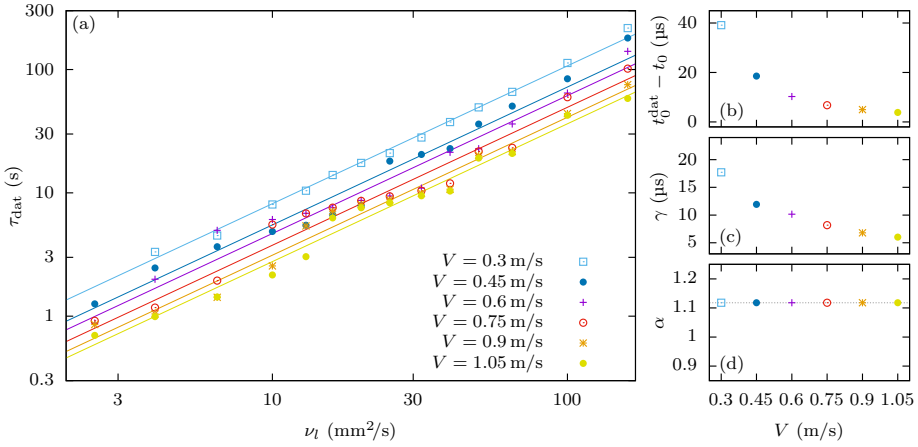


Figure A.6: (a) Model fits for minimizing the residual S for the two parameters (t_0^{dat}, γ) where $\alpha = 1.1181$. The straight lines show the fitted model $\tau_{dat} = \gamma(\nu_l/\nu_{sc})^{\alpha}$. (b–d) Values of the parameters t_0^{dat}, γ, and α, with the dotted line indicating the constrained parameter.

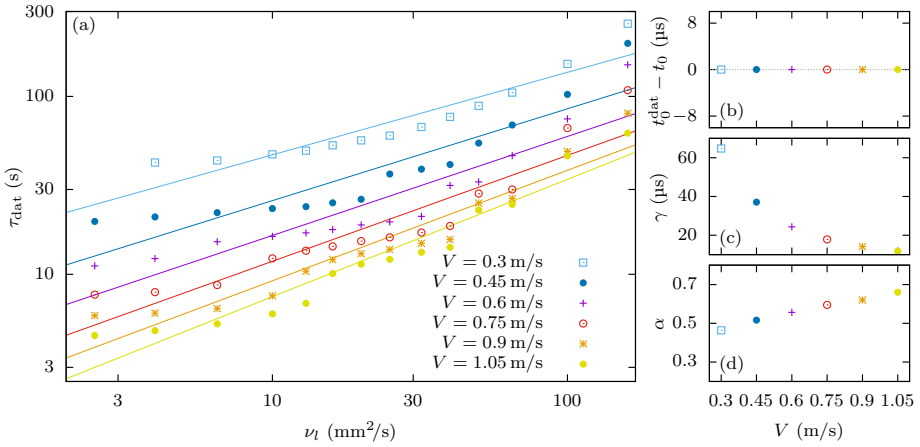


Figure A.7: (a) Model fits for minimizing the residual S for the two parameters (γ, α) where $t_0^{dat} = t_0$. The straight lines show the fitted model $\tau_{dat} = \gamma(\nu_l/\nu_{sc})^{\alpha}$. (b–d) Values of the parameters t_0^{dat}, γ, and α, with the dotted line indicating the constrained parameter.

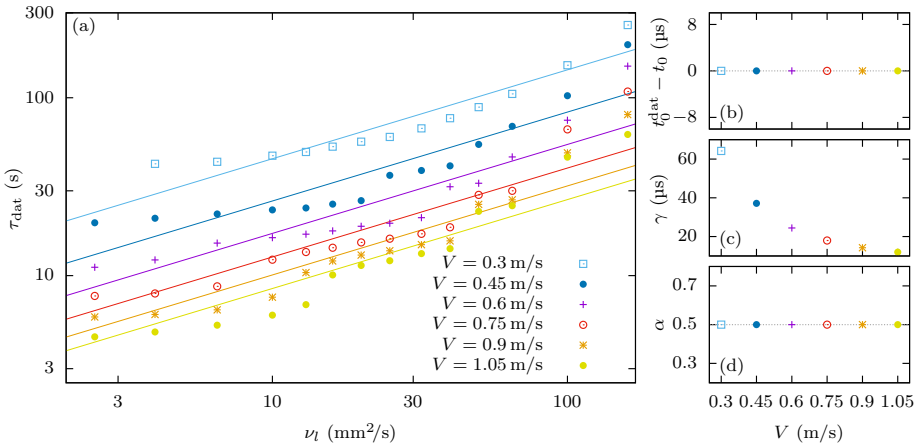


Figure A.8: (a) Model fits for minimizing the residual S for the parameter γ, where $t_0^{dat} = t_0$ and $\alpha = 1/2$. The straight lines show the fitted model $\tau_{dat} = \gamma(\nu_l/\nu_{sc})^{\alpha}$. (b–d) Values of the parameters t_0^{dat}, γ, and α, with the dotted lines indicating the constrained parameters.

References

[1] E Anderson, Z Bai, C Bischof, S Blackford, J Demmel, J. Dongarra, J Du Croz, A Greenbaum, S Hammarling, A McKenney, and D Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.

[2] Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

[3] John M Kolinski, L Mahadevan, and Shmuel M Rubinstein. Lift-off instability during the impact of a drop on a solid surface. *Physical Review Letters*, 112(13):134501, 2014.

Appendix B

Supplementary material: Coordinated crawling via reinforcement learning

This has been published as part of **Mishra, S., van Rees, W. M. and Mahadevan, L., 2020. Coordinated crawling via reinforcement learning. Journal of the Royal Society Interface 17: 20200198** [arXiv: 2003.12845]

B.1 Parameter values

Table B.1: Parameters and their values used in the simulation. All lengths are scaled by the segment length L and all times are scaled by the neuronal relaxation time τ_θ.

Symbol	Quantity	Value
L	segment length	1
τ_θ	neuronal timescale	1
$c\tau_\theta/k$	scaled damping	3.5
f^m_{max}/kL	scaled muscular force	1
τ_m/τ_θ	scaled muscular timescale	1
f^f_{max}/kL	scaled backward frictional force	9
ε^f	frictional smoothing	10^{-6}
η	friction anisotropy	30
Δt	scaled discrete timestep	0.01
α	learning rate	0.05
γ	discount rate	0.95

B.2 Unlearned gait

See Figure B.1.

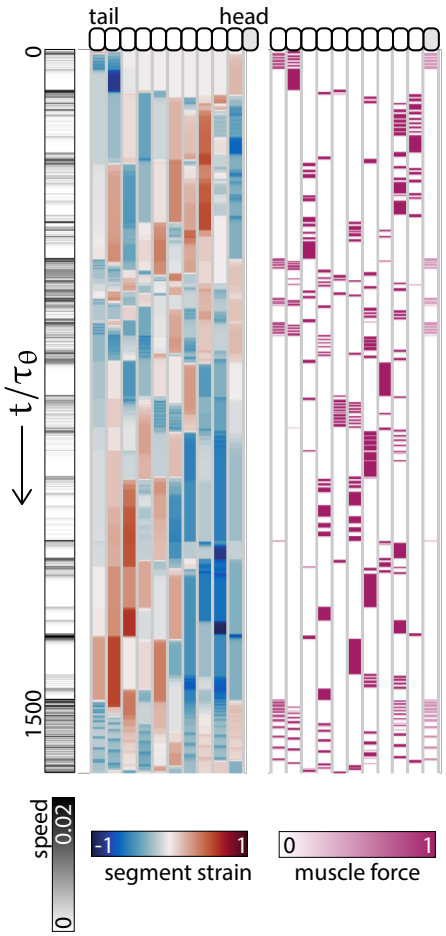


Figure B.1: Uncoordinated gait resulting from a fully connected neuronal network, as summarized by Equations 3.1–3.9 of the main text. The speed of the centre of mass is in grey (left), corresponding segment strains in blue-red (middle) and muscle force in pink (right). The parameter values are summarized in Table B.1.

B.3 Videos

We include two files that show the converged gait of the crawler with and without regularization, corresponding to Figure 3.2(a) and (b), respectively.

Video 1 – Regularized gait: Coordinated gait that arises from an initial uncoordinated gait with a regularization parameter $\epsilon = 0.01$.

Video 2 – Unregularized gait: Coordinated gait that arises from an initial uncoordinated gait with no regularization parameter, i.e., $\epsilon = 0$, which leads to motion where multiple segments that move concurrently. This gait is less robust to proprioceptive noise and is easily disrupted (see Figure 3.3(c) and corresponding text for details).