# Acoustic Particle Control with Particle-Particle Interactions

**Author**
Mohamad Zowezer

**Supervisor**
Thierry Baasch

**Examiner**
Christian Antfolk

**Assistant Supervisors**
Alexander Edthofer

Shuo Huang

Guilherme Junqueira Perticarari

Master's Thesis in Electrical Measurements


LUND
UNIVERSITY

Department of Biomedical Engineering
Faculty of Engineering, Lund University

2026

# Abstract

Acoustophoresis, the manipulation of microscopic particles by acoustic forces, has found growing use in biomedical applications ranging from blood-cell separation to single-cell positioning in microfluidic chips[1, 2, 3, 4]. Existing multimodal control algorithms for such devices were developed under the assumption that particles do not influence one another. This simplification holds when only one particle is present or when particles are widely separated. In scenarios where particle trajectories can cross each other, or particles are present in close proximity, this simplification breaks down.

This thesis extends and evaluates a suite of acoustophoretic control algorithms, originally designed for non-interacting particles by Perticarari [5] and Öhrnberg [6], in the presence of acoustic and hydrodynamic particle-particle interactions. The work has two parts. In the first, the algorithms are re-benchmarked in the non-interaction setting. In the second, the same algorithms are evaluated inside an interaction-enabled simulation framework, where the Acoustic Interaction Force (AIF), hydrodynamic coupling between particles, and contact forces are fully accounted for.

Five controllers are studied: the $\varepsilon$-greedy algorithm [7], the model-based quadratic programme, here named (QP) or as named by Perticarari [5] Perfect-Information Local Optimisation (PILOt), the online-learning Vector-based Local Optimisation (VeLO) controller and its no-sweep variant, and a tile-coded tabular Q-learning. To ensure consistent and reproducible evaluation, a deterministic main benchmark consisting of a varied number of samples under multiple environment setups is developed and applied across all controllers.

The results show that QP, VeLO, and VeLO_No-Sweep can be deployed under interactions without any code changes, while $\varepsilon$-greedy was insufficient for multiple particle control. QP proved to be the most efficient and reached 100 % success on two and three interacting particles. Its per-step displacement matrix implicitly absorbs the acoustic interaction force. Two algorithmic improvements were identified: First, the tile-coded Q-learning with $\varepsilon$-decay, $\alpha$-decay, and a shaped reward achieves 100 % success at three particles with and without interactions. Second, a modified Hessian regularisation for the VeLO is introduced in this work, together with the VeLO_No-Sweep variant, which has no sweep phase. These modified versions of the controllers are reliable across the tested range, with a success rate between (90–100 %) at particle count

of 1, 2, 3, with and without interactions. VeLO_No-Sweep matches and sometimes exceeds standard VeLO at all conditions. Q-learning matches QP's success, but at a far greater computational cost. The long training time (roughly three hours per table in the three particle case) and the inflexibility of the learned policy make QP the more practical choice for deployment in simulations. A scalability stress test at $n_P = 10$ and $n_P = 20$, with targets arranged on a circle, further demonstrates that QP converges successfully under conditions involving particle counts far greater than three, including fully interaction-enabled dynamics.

*Keywords:* acoustophoresis, microfluidics, multi-particle control, quadratic programming, reinforcement learning, particle-particle interactions, acoustic radiation force, acoustic interaction force.

# Popular Science Summary

**Steering cells with sound: smarter algorithms for contactless particle control**

When we think about moving individual cells, we tend to imagine tiny motorized nanorobots performing the task. In fact, one of the most advanced techniques for cell manipulation requires no moving parts. Instead, it relies on carefully controlled sound waves known as acoustophoresis. Making it work reliably for multiple cells at once is the problem this study tackles.

Every year, thousands of patients undergo procedures that require isolating specific cells from blood: circulating tumour cells for cancer diagnosis, stem cells for therapy, and immune cells for immunotherapy. Current methods are often slow, imprecise, or might damage the cells. Acoustophoresis offers a gentler alternative, where a sound wave is sent through a tiny water-filled chip, and cells drift toward the quiet regions of the wave. By rapidly switching between different wave patterns, a control algorithm can push particles toward any chosen destination without any physical contact.

Previous research showed that this works well for a single cell, but real clinical samples contain many cells at once. When several particles share the same chip, each cell radiates its own tiny acoustic force that pushes its neighbours, and every moving cell drags a wave of fluid that affects the others. These "particle–particle interactions" had been left out of earlier control algorithms.

To find out whether those interactions actually matter, five different algorithms were tested on the same set of initial conditions, run for two and three particles at a time, with and without interactions.

The algorithms span a wide range of approaches. The most mathematically precise, a quadratic optimiser called Perfect-Information Local Optimisation (PILOt), probes the chip's response at every time step and solves an equation to find the best blend of wave patterns. An online learner called Vector-based Local Optimisation (VeLO) instead watches how the particles actually move and updates its model on the fly.

The $\varepsilon$-greedy bandit algorithm balances exploitation and exploration. It usually chooses the acoustic wave pattern known to work best, but occasionally takes a random action on an untried frequency to discover an even faster path. Finally, a more complex

reinforcement learning approach, Q-learning, expands on this concept by training offline and learning from observations.

The results were surprising in several ways. The quadratic optimiser actually improved when particle-to-particle interactions were introduced to the system. Because it probes the chip's behaviour at every time step, it captures the interaction features between cells and uses them as an extra steering force. The no-sweep variant of the online learner also performed well. It skips the calibration survey, which the standard VeLO starts with. VeLO_No-Sweep learns entirely from the cells' actual movements, including how they affect each other, never falling behind the standard version. Q-learning kept pace with the quadratic optimiser in step count and matched its success rate. Q-learning's main limitation is that each table takes about three hours to train and cannot adapt once learned. The $\varepsilon$-greedy algorithm was unable to reliably handle two cells simultaneously with an acceptable success rate.

These results show which algorithms can cope with particle–particle interactions, and point to better lab-on-a-chip designs for precision medicine.

# Acknowledgements

I would like to express my sincere gratitude to my supervisor, Thierry Baasch, for his continuous guidance, support, and valuable insights throughout this thesis.

I would like to extend my appreciation to my assistant supervisors, Alexander Edthofer, Shuo Huang, and Guilherme Junqueira Perticarari, for their support, feedback, and technical discussions during different stages of the project.

Special thanks are due to Christian Antfolk for serving as the examiner and for his time and contribution to evaluating this thesis.

This work also builds upon previous research conducted by Guilherme Junqueira Perticarari and Robin Öhrnberg, whose earlier theses established the methodological and theoretical basis that made this study possible.

Finally, I would like to thank my family and friends for their support and encouragement throughout my studies and during the completion of this thesis.

# Contents

**Abstract** **3**

**Popular Science Summary** **5**

**Acknowledgements** **7**

**Nomenclature** **13**

**1 Introduction** **15**

1.1 Background . . . . . 15

1.2 Thesis Goals and Research Questions . . . . . 16

1.3 Contributions . . . . . 16

1.4 Thesis Outline . . . . . 17

**2 Theoretical Background** **19**

2.1 Acoustic Standing Waves in Rectangular Chambers . . . . . 19

2.2 Acoustic Radiation Force . . . . . 20

2.2.1 Particle Velocity in the Overdamped Regime . . . . . 20

2.3 Acoustic Interaction Force . . . . . 21

2.4 Hydrodynamic Interactions . . . . . 21

2.5 Contact Force . . . . . 22

2.6 The 2D-SW Simulation Model . . . . . 22

**3 Control Problem Formulation** **25**

3.1 Control Systems and the Feedback Principle . . . . . 25

3.2 State and Action Representation . . . . . 27

**4 Control Algorithms** **29**

4.1 quadratic programme Controllers . . . . . 29

4.1.1 Quadratic Programming: Background . . . . . 29

4.1.2 The PILOt Framework . . . . . 30

4.1.3 QP . . . . . 30

4.2 Vector-based Local Optimisation (VeLO) . . . . . 31

4.2.1 Warmup Sweep . . . . . 31

4.2.2 Online Control Phase . . . . . 31

4.2.3 VeLO Without Sweep (VeLO_No-Sweep) . . . . . 32

4.2.4 Effect of Hessian Regularisation on the VeLO Family . . . . . 32

4.3 $\varepsilon$-Greedy Bandit . . . . . 33

4.4 Reinforcement Learning . . . . . 33

4.4.1 Markov Chains and the Markov Property . . . . . 34

4.4.2 Markov Decision Processes . . . . . 34

4.4.3 Q-Learning . . . . . 34

4.5 Q-Learning: Baseline Implementation (QG) . . . . . 35

4.6 Tile-Coded Q-Learning . . . . . 35

4.6.1 State Representation . . . . . 36

4.6.2 Enhancements Over QG . . . . . 36

4.6.3 Deployment . . . . . 37

**5 Methods: Simulation Framework and Benchmark** **39**

5.1 Simulation Architecture . . . . . 39

5.2 Setup Function and Device Parameters . . . . . 39

5.3 Deterministic Initial Conditions and Reproducibility . . . . . 40

5.4 BenchmarkB50 Suites . . . . . 40

5.5 Circle Stress Test . . . . . 42

5.6 Performance Metrics and Result Visualisation . . . . . 42

5.7 Software Environment . . . . . 45

**6 Results** **47**

6.1 Main Benchmark: Polystyrene Environment . . . . . 47

6.1.1 Without Particle-Particle Interactions . . . . . 47

6.1.2 With Particle-Particle Interactions . . . . . 48
6.2 Q-Learning vs. $\varepsilon$-Greedy: Polystyrene . . . . . 51
6.3 QP vs Q-Learning . . . . . 51
6.4 Scalability Stress Test: QP on Circle Arrangements . . . . . 54

**7 Discussion** **57**
7.1 Algorithmic Modifications for Robustness . . . . . 57
7.2 Adaptability Without Structural Modification . . . . . 58
7.3 How Interactions Affect Control Precision and Stability . . . . . 59
7.4 Practical Particle-Count Limits . . . . . 60
7.5 Limitations . . . . . 61

**8 Conclusion** **63**

**A Algorithms Pseudocode** **69**
A.1 One-Step Lookahead . . . . . 70
A.2 QP . . . . . 71
A.3 VeLO . . . . . 72
A.4 VeLO Without Sweep . . . . . 72
A.5 $\varepsilon$-Greedy Bandit . . . . . 73
A.6 Tile-Coded Q-Learning . . . . . 74
1.7 BenchmarkB50 . . . . . 76

# Nomenclature

| Symbol | Description |
|---|---|
| $a$ | Particle radius (m) |
| $c$ | Speed of sound in the fluid ($\mathrm{m\,s^{-1}}$) |
| $E_{\mathrm{ac}}$ | Time-averaged acoustic energy density ($\mathrm{J\,m^{-3}}$) |
| $f_1, f_2$ | Monopole and dipole acoustic scattering coefficients (-) |
| $F_{\mathrm{rad}}$ | Acoustic radiation force vector (N) |
| $F_{\mathrm{AIF}}$ | Acoustic interaction force vector (N) |
| $F_{\mathrm{drag}}$ | Stokes drag force vector (N) |
| $F$ | Displacement matrix ($\mathbb{R}^{2n_P \times M}$) |
| $\hat{F}$ | Estimated displacement matrix in VeLO |
| $h$ | Chamber height (m) |
| $H$ | Hessian matrix in QP |
| $k$ | Acoustic wavenumber ($\mathrm{m^{-1}}$) |
| $M$ | Number of acoustic modes |
| $n_P$ | Number of particles |
| $N_x, N_y$ | Mode numbers in $x$- and $y$-directions |
| $p$ | Acoustic pressure field (Pa) |
| $p_A$ | Acoustic pressure amplitude (Pa) |
| $q$ | Normalised state vector ($\mathbb{R}^{2n_P}$) |
| $q_r$ | Normalised Target state vector ($\mathbb{R}^{2n_P}$) |
| $t_k$ | Target displacement $q_r - q$ |
| $U$ | Gor'kov acoustic radiation potential (J) |
| $V_p$ | Particle volume ($\mathrm{m^3}$) |
| $w$ | Mode weight vector ($\mathbb{R}^M$, $w \geq 0$) |
| $w$ | Chamber width (m) |
| $\varepsilon$ | Exploration probability ($\varepsilon$-greedy) |
| $\eta$ | Dynamic viscosity of the fluid (Pa s) |
| $\gamma$ | EMA smoothing factor (VeLO) |
| $\kappa$ | Fluid compressibility ($\mathrm{Pa^{-1}}$) |
| $\rho$ | Fluid density ($\mathrm{kg\,m^{-3}}$) |
| $\Phi$ | Acoustic contrast factor |

| | |
|---|---|
| $\omega$ | Angular frequency ($\mathrm{rad\,s^{-1}}$) |

| **Abbreviation** | |
|---|---|
| ARF | Acoustic Radiation Force |
| AIF | Acoustic Interaction Force |
| EMA | Exponential Moving Average |
| IC | Initial Condition |
| PILOt | Perfect-Information Local Optimisation |
| QP | quadratic programme |
| RL | Reinforcement Learning |
| VeLO | Vector-based Local Optimisation |
| 2D-SW | Two-Dimensional Standing Wave |

# Chapter 1

# Introduction

## 1.1 Background

The ability to steer individual microscopic cells inside a fluid-filled chamber without physical contact has attracted interest in the biomedical community over the past two decades.

Acoustophoresis achieves this by exploiting the radiation pressure that arises when a sound wave scatters off a particle. For particles smaller than the acoustic wavelength, this force can be strong enough to move cells across a microfluidic chip, while leaving the cell unharmed [8, 2]. Traditional acoustophoretic devices operate with a single standing wave, focusing all particles toward the pressure nodes or antinodes. This depends on the material contrast between the particle and the fluid.

Although this bulk-focusing approach has proven effective for separation tasks [1], it offers little control over where individual particles end up.

A strategy that offers more freedom in control, enabling unreachable movements with a single pure mode, called multimodal acoustophoresis. This strategy drives the chamber with a rapidly switched mixture of resonance modes rather than a single fixed frequency.

The weighting between modes determines the net force experienced by a particle and can be chosen to steer each particle along a trajectory [5, 6].

A systematic simulation study of this approach was conducted by Perticarari [5], who investigated the feasibility of multimodal control and compared several algorithms.

Öhrnberg [6] subsequently extended that work with a broader algorithmic comparison, including quadratic programming based approaches and reinforcement learning (RL) methods.

Complementary work has used machine learning to sort particles in microfluidic

chips via bulk acoustic waves [7].

A key limitation shared by all algorithms developed so far is the assumption that the particles are dynamically independent. This limitation means that each particle responds only to the external acoustic field and the Stokes drag from the background fluid, and does not interact with neighbouring particles. In reality, this is never exactly true. Therefore, this work investigates how particle interactions impact the control task.

A vibrating particle radiates a secondary acoustic wave that produces an additional force on its neighbours, the acoustic interaction force (AIF) [9]. At the same time, a moving particle disturbs the fluid around it, changing the drag felt by nearby particles, a phenomenon known as hydrodynamic coupling [10, 11, 12]. At particle separations on the order of a wavelength or less, these interaction forces can be comparable in magnitude to the primary ARF and cannot be neglected without introducing errors in predicted particle trajectories.

## 1.2 Thesis Goals and Research Questions

The main goal of this work is to enhance the existing suite of pure and multimodal acoustophoretic control algorithms to remain effective when particle-particle interactions are active, and to quantify how much those interactions alter performance. This involves two phases. In the first, the algorithms are benchmarked carefully in the non-interaction simulation framework. In the second, the same algorithms are run inside an interaction-enabled simulation framework and the performance is then analysed and compared against the non-interaction baseline to assess changes in control precision and stability.

**The questions guiding the study are:**

- What algorithmic modifications or enhancements improve robustness and controllability?
- Can existing control algorithms be adapted, without structural modification, to handle multiple interacting particles?
- How do acoustic and hydrodynamic particle-particle interactions affect the precision and stability of acoustic particle control?

## 1.3 Contributions

The principal contributions of this thesis relative to the work of Perticarari and Öhrnberg are as follows. First, Öhrnberg's QP implementation is extended to support the objectives and experimental requirements of the present work.

Second, a reproducible 50-sample benchmark, referred to throughout as *BenchmarkB50*, is introduced, built around a deterministic initial-condition generator that ensures every algorithm faces precisely the same set of start–goal pairs, regardless of implementation details.

Third, a tile-coded tabular Q-learning implementation is developed to improve the uniform-grid baseline of Perticarari, including $\varepsilon$-decay, $\alpha$-decay, a shaped reward with step penalties and success bonuses, and efficient parallel training across all initial conditions.

Fourth, a VeLO-based controller introduced by the co-supervisor Alexander Edthofer, is included in the study. In particular, the no-sweep variant (VeLO_No-Sweep) is evaluated in the comparative analysis.

Finally, an assessment of the algorithms' performance in both interaction and non-interaction environments is carried out.

## 1.4 Thesis Outline

Chapter 2 establishes the acoustic and mechanical physics underlying the simulation. Chapter 3 illustrates the manipulation task as a formal control problem. Chapter 4 describes each control algorithm in detail. Chapter 5 presents the experimental methods: the simulation framework, device presets, benchmark design, and the software environment. Chapter 6 reports results, with tables and figures. Chapter 7 interprets the findings, and Chapter 8 summarises conclusions and points toward future work. Appendix A collects the pseudocode for all algorithms.

# Chapter 2

# Theoretical Background

## 2.1 Acoustic Standing Waves in Rectangular Chambers

The simulation device is a rectangular microfluidic chamber filled with a liquid, typically water. Its walls are assumed rigid, and its dimensions are small enough that the acoustic field can be treated as two-dimensional, varying only in $x$ and $y$. The pressure field inside the chamber satisfies the wave equation [13]

$$\nabla^2 p(r,t) - \frac{1}{c^2}\frac{\partial^2}{\partial t^2} p(r,t) = 0, \tag{2.1}$$

where $c$ is the speed of sound and $r = (x,y)$. The acoustic velocity field must be tangent to each rigid wall, so that the wall-normal velocity must be zero: $v \cdot \hat{n} = 0$. To solve the wave equation, we write $p(x,y,t) = X(x)Y(y)T(t)$ a, which splits the problem into three separate ordinary differential equations. Applying the wall conditions to $X$ and $Y$ restricts the allowed wavenumbers to a discrete set, giving the resonance modes indexed by two non-negative integers $N_x$ and $N_y$:

$$p_{N_x,N_y}(x,y,t) = p_A \cos\left(\frac{N_x \pi}{w} x\right) \cos\left(\frac{N_y \pi}{h} y\right) \cos(\omega_{N_x,N_y} t), \tag{2.2}$$

where $w$ and $h$ are the chamber width and height, $p_A$ is the pressure amplitude, and the angular frequency satisfies the relation $\omega_{N_x,N_y} = c\sqrt{(N_x\pi/w)^2 + (N_y\pi/h)^2}$ [13]. Each pair $(N_x,N_y)$ defines one acoustic mode. The integer node counts $N_x$ and $N_y$ determine how many pressure half-wavelengths fit along each axis. A mode $(1,0)$ produces one pressure node line running parallel to the $y$-axis at $x = w/2$, while a mode $(2,2)$ creates a two-by-two chessboard pattern of pressure nodes and anti-nodes. This spatial variety is what makes multimodal control possible. Blending modes with different force landscapes, enable the controller to steer particles in directions no single mode could achieve.

## 2.2 Acoustic Radiation Force

A small sphere of radius $a \ll \lambda$ inside an acoustic field scatters the pressure wave around it. The time-averaged momentum carried by the scattered wave is transferred to the particle, producing a net acoustic radiation force. As Gor'kov [14] showed, this force can be derived from the Gor'kov potential $U$, which captures both the material properties of the particle and the local acoustic field:

$$U = V_p \left( \frac{f_1}{2\rho_0 c^2} \langle p^2 \rangle - \frac{3}{4} \rho_0 f_2 \langle v \cdot v \rangle \right), \tag{2.3}$$

so that $F_{\text{rad}} = -\nabla U$. Here, $V_p = \frac{4}{3}\pi a^3$ is the particle volume, $\rho_0$ is the fluid density, $\langle \cdot \rangle$ denotes a time average over one oscillation period, and $f_1$, $f_2$ are dimensionless acoustic scattering coefficients that describe how different the particle is from the surrounding fluid:

$$f_1 = 1 - \frac{\kappa_p}{\kappa_0}, \qquad f_2 = \frac{2(\tilde{\rho} - 1)}{2\tilde{\rho} + 1}, \qquad \tilde{\rho} = \frac{\rho_p}{\rho_0}. \tag{2.4}$$

$f_1$ measures the compressibility contrast between the particle and the fluid, while $f_2$ measures the density contrast. For polystyrene beads or biological cells in water, both coefficients are positive, so the combined acoustic contrast factor $\Phi = f_1/3 + f_2/2 > 0$ and particles are pushed toward the pressure nodes.

For a two-dimensional standing-wave mode $(N_x, N_y)$ the pressure and velocity fields are known analytically from Eq. (2.2). Plugging these into the Gor'kov potential and taking the gradient gives the ARF at any point $(x, y)$ in the chamber. The expression for one-dimension case:

$$F_x = 3V_p \Phi k E_{\text{ac}} \sin(2kx), \tag{2.5}$$

where $E_{\text{ac}} = p_A^2/(4\rho_0 c^2)$ is the acoustic energy density [15]. In two dimensions, the force landscape becomes more complex, with saddle points and local minima that depend on the mode numbers. This position dependence is exploited by the control algorithms.

### 2.2.1 Particle Velocity in the Overdamped Regime

At the micrometer scale, particles move at very low Reynolds numbers, meaning viscous drag dominates, and inertia can be ignored [16, 17]. The particle velocity at any instant is therefore set by the balance between the ARF and the Stokes drag force $F_{\text{drag}} = -6\pi\eta a \dot{r}$, where $\eta$ is the dynamic viscosity. Setting their sum to zero gives the first-order equation of motion:

$$\dot{r}(t) = \frac{F_{\text{rad}}(r,t)}{6\pi\eta a}, \tag{2.6}$$

which the simulation steps forward in time using an Euler scheme with time step $\Delta t$. The multimodal case is handled by linearity, each mode's velocity contribution is calculated independently and then combined as a weighted sum:

$$\dot{r} = \sum_{m=1}^{M} w_m \dot{r}_m(r), \tag{2.7}$$

where $w_m \geq 0$ is the weight assigned to mode $m$ and $\dot{r}_m$ is the velocity the mode would produce on its own.

## 2.3 Acoustic Interaction Force

When two or more particles are present in the same acoustic field, each particle scatters the incoming wave. The resulting scattered wave acts on the other particles. The physics behind the acoustic radiation forces between particles was first worked out by Doinikov [18]. This simulation uses the approach of Silva and Bruus [9], who derived a complete analytical expression for the acoustic interaction force (AIF) between small particles, valid when the particle radius is much smaller than the wavelength $ka$.

When two particles are close, they can strongly attract or repel each other, an effect explored in the acoustophoresis setting by Pavlič and Baasch [12]. The resulting interaction patterns depend on a combination of the physical properties of the particles and the acoustic field [19]. AIF is one of the main reasons multi-particle control is harder than handling each particle separately, because the AIF creates coupling between particles that the controller must either compensate for or exploit.

## 2.4 Hydrodynamic Interactions

When a particle moves through the fluid, it disturbs the surrounding flow, and this disturbance spreads slowly enough to reach neighbouring particles and alter their motion [10]. As a result, the velocity of each particle depends on the forces acting on all others, a coupling described by a mobility matrix $\mathbf{M}$ such that $\mathbf{U} = \mathbf{M} \cdot \mathbf{F}$, where $\mathbf{U}$ collects all particle velocities, and $\mathbf{F}$ all applied forces [10]. The hydrodynamic interaction between particles $i$ and $j$ separated by distance $r_{ij}$ can be characterised by the Oseen tensor [10, 11], which couples the velocity of each particle to the applied forces on all others. In this work, the mobility matrix $\mathbf{M}$ is approximated using the theory of Stokesian dynamics [10, 11], which constructs $\mathbf{M}$ from the far-field multipole contributions exchanged between particle pairs.

## 2.5 Contact Force

When the controller drives two particles into contact, a third interaction mechanism becomes active: the non-penetration constraint. It is expressed by Signorini's complementarity condition relating the gap $g_n$ between two particle surfaces and the normal contact force $F_n$ [16]: at any instant at most one of the two is positive. Either the particles touch, $g_n = 0$ and $F_n \geq 0$, or they are separated, $g_n \geq 0$ and $F_n = 0$.

In the interaction-enabled simulation, this constraint is enforced by an elastic repulsion penalty that activates only on overlap. That prevents the particles from passing through one another (Figure 2.1).

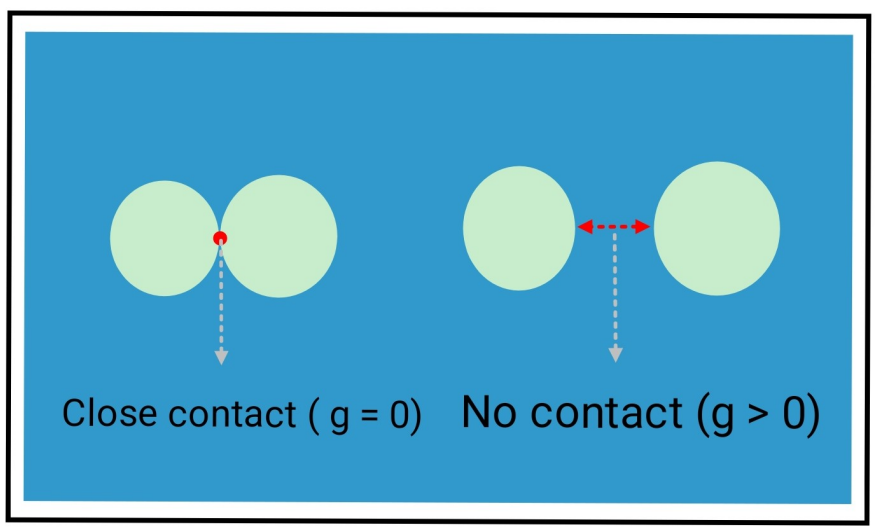



Figure 2.1: Two tangent particles close to contact ( touching at one point, pointed with a small red marker), where the contact force $F_n \geq 0$, and two separated particles (no contact with a red gap arrow) where the contact force $F_n = 0$.

## 2.6 The 2D-SW Simulation Model

The control algorithms in this thesis are evaluated in a two-dimensional standing-wave (2D-SW) rectangular chamber. The device is modelled as an acoustically lossless resonator with perfectly rigid walls, fluid and particle parameters are listed in Table 2.1. Two sets of initial conditions are used throughout: a "no-interaction" set where only the primary ARF is active, and an "interaction" set where both the AIF and hydrodynamic coupling are enabled. The dynamics in both cases are integrated with a forward Euler integrator. Contact handling (elastic repulsion) when two particles overlap is active only in the interaction-enabled setting.

Table 2.1: Simulation parameters for the polystyrene environment (`setupB40`). The pressure amplitude is 44.4 kPa for the main benchmark and 83 kPa for the Q-learning comparisons; mode counts per particle count are given in Table 5.1.

| Parameter | Symbol | Value |
|---|---|---|
| Chamber width | $w$ | 800 $\mu$m |
| Chamber height | $h$ | 600 $\mu$m |
| Pressure amplitude | $p_A$ | 44.4/83 kPa |
| Speed of sound | $c$ | 1500 m s$^{-1}$ |
| Fluid density | $\rho_0$ | 1000 kg m$^{-3}$ |
| Fluid compressibility | $\kappa$ | $458 \times 10^{-12}$ Pa$^{-1}$ |
| Dynamic viscosity | $\eta$ | $10^{-3}$ Pa s |
| Monopole / dipole coefficients | $f_1, f_2$ | 0.476, 0.0323 |
| Particle radius | $a$ | 3.5–7 $\mu$m |
| Time step | $\Delta t$ | 0.5 s |
| Convergence tolerance | $d_{\mathrm{tol}}$ | 15 $\mu$m |

# Chapter 3

# Control Problem Formulation

## 3.1 Control Systems and the Feedback Principle

Before diving into acoustophoretic manipulation mathematically, it helps to introduce some basic ideas from control theory, since these ideas shape how the algorithms in Chapter 4 are described and compared.

A control system is simply a way to steer a physical process called the plant so that its behaviour matches what we want [20, 21]. In a closed-loop system, the controller continuously checks how far the process has drifted from the desired reference to decide what to do next. If y is the measured output and r is the reference, the tracking error is : $e = r - y$.

The controller feeds this error into some function $u = C(e)$ to produce a control input that acts on the plant through an actuator. The plant then responds, and a new measurement of y is taken, and the cycle repeats [20] (Figure 3.1).

In the acoustophoretic manipulation setting, the four elements of the feedback loop are assigned as follows:

- **Plant**: the microfluidic chamber together with the particle dynamics.
- **Actuator**: the acoustic transducer array, which excites the chamber at the resonance frequencies of the selected modes, the weight vector $w \in \mathbb{R}^M$ is the control input $u$ that steers the particles in the chamber.
- **Sensor**: In the simulation, particle positions are read directly from the state struct at each discrete time step.
- **Reference**: the target position vector $q_r$ of particles.

At each time step, the controller observes the current positions $q$, computes the error $e = q_r - q$, and selects $w$ intended to reduce $e$ in the next step closing the feedback loop.

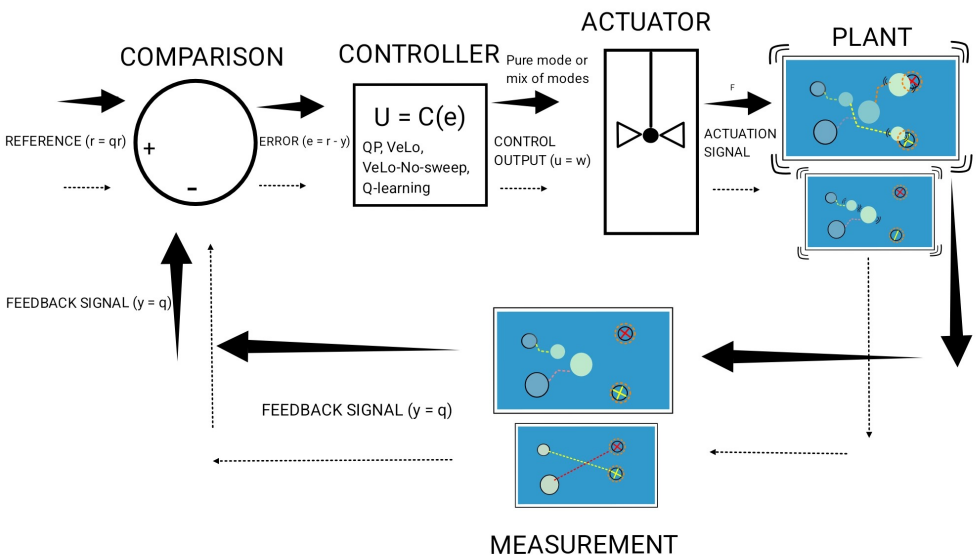



Figure 3.1: A control system loop. The dashed thin arrows represent the first iteration of the control loop, while the thick arrows represent the second iteration.

All five algorithms evaluated in this study implement this closed-loop structure. They differ in how they map $e$ and the current state to $w$: model-based methods like QP build a linearised model of the plant by probing the dynamics at each step, while model-free methods like Q-learning learn from observed transitions during the offline training.

## 3.2 State and Action Representation

The state of the system at any given time step is described by the positions of all particles. For the purpose of control, these are concatenated into a $2n_P$-dimensional vector and normalised by the chamber dimensions:

$$q = \begin{bmatrix} x_1/w \\ y_1/h \\ \vdots \\ x_{n_P}/w \\ y_{n_P}/h \end{bmatrix} \in \mathbb{R}^{2n_P}. \tag{3.1}$$

The target vector $q_r$ is constructed from the goal positions in the same normalised coordinates. The control task is to find, at each time step, a weight vector $w \in \mathbb{R}^M$ such that one step of forward Euler dynamics under the blended mode field brings $q$ as close as possible to $q_r$.

The action space is the set of $M$ acoustic modes available to the controller. The mode count $M = N_{\text{dim}}^2$ and the grid of available modes depend on the particle count, full details are given in Table 5.1. In the polystyrene environment used throughout, $N_{\text{dim}} = 6, 7, 8$ for $n_P = 1, 2, 3$, giving $M = 36, 49, 64$ modes respectively. The mode grid consists of pairs $(N_x, N_y)$ with $N_x, N_y \in \{1, \ldots, N_{\text{dim}}\}$. The weight vector satisfies $w_m \geq 0$ for all $m$, with a sum constraint $\sum_{m=1}^{M} w_m \leq 1$. A weight vector that uses a single mode, $w_m = 1$ and $w_{m'} = 0$ for $m' \neq m$, is referred to as a "pure mode" action.

# Chapter 4

# Control Algorithms

This chapter describes each of the five control algorithms evaluated in the benchmark, along with two earlier implementations included for context. All of them work within the same framework set out in Chapter 3: they take in the current state and output a weight vector $w$.

The simplest model-based strategy is to evaluate, for each mode separately, what the resulting particle positions would be after one time step, referred to as One-Step Lookahead, and then choose the mode that reduces the total distance to the goal the most. Formally, let $q_m$ denote the next state if mode $m$ is applied with full weight. A one-step lookahead controller selects

$$m^* = \arg\min_m \|q_m - q_r\|_2, \tag{4.1}$$

and sets $w_{m^*} = 1$, $w_{m'} = 0$ for $m' \neq m^*$. This requires $M$ calls to the dynamics function per step. Because it selects only pure modes, it cannot produce the blended force landscape needed to steer particles in directions that fall between two mode equilibria.

## 4.1 quadratic programme Controllers

### 4.1.1 Quadratic Programming: Background

The controllers in this section all frame the mode-selection problem as a quadratic programme (QP), that is, an optimisation problem with a quadratic objective and linear constraints[22, 23]:

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} x^T H x + f^T x \quad \text{s.t.} \quad A_{\text{eq}} x = b_{\text{eq}}, \quad A_{\text{ineq}} x \leq b_{\text{ineq}}, \tag{4.2}$$

where $H \in \mathbb{R}^{n \times n}$ is the Hessian matrix that encodes the curvature of the objective and $f \in \mathbb{R}^n$ is the linear term. When $H$ is positive semi-definite the objective is convex:

any local minimum is also a global minimum, and it can be solved in polynomial time using standard interior-point or active-set methods [23]. In MATLAB, this is handled by `quadprog` with the interior-point-convex method.

### 4.1.2 The PILOt Framework

The Perfect-Information Local Optimisation (PILOt) framework, introduced by Perticarari [5], replaces the greedy mode selection of the one-step lookahead with a quadratic programme (QP) that distributes weight across all modes simultaneously. The key insight is to treat one time step as a linear map: column $m$ of the displacement matrix $F \in \mathbb{R}^{2n_P \times M}$ records the normalised displacement that mode $m$ alone would produce from the current state,

$$F(:,m) = q_m - q, \qquad q_m = \text{next state under mode } m. \tag{4.3}$$

The objective is to choose weights $w$ such that the blended displacement $Fw$ is as close as possible to the desired target displacement $t_k = q_r - q$:

$$\min_{w} \quad \|Fw - t_k\|^2. \tag{4.4}$$

Expanding the squared norm and writing the problem in the standard quadratic programme form [22, 23], $\min \frac{1}{2} w^T H w + f^T w$, gives

$$H = 2(F^T F) + \lambda I, \qquad f = -2F^T t_k, \tag{4.5}$$

where $\lambda > 0$ is a small regularisation constant. The term $\lambda I$ is Tikhonov regularisation [22, 24] ensures that $H$ remains positive definite even when two modes produce nearly identical displacements, preventing the solver from returning degenerate weight vectors.

### 4.1.3 QP

The **QP** controller, based on the PILOt framework from Öhrnberg [6], builds the displacement matrix $F$ and solves a constrained least-squares QP.

$$H_{\text{QP}} = 2(F^T F) + 10^{-5} I, \qquad f_{\text{QP}} = -2F^T t_k. \tag{4.6}$$

The $10^{-5} I$ term is a small regulariser added to keep the problem well-conditioned when two or more modes produce nearly identical displacements. Without this term, the Hessian can become singular. The Weights are kept non-negative through a lower bound $w_m \geq 0$, and a single inequality caps the total weight:

$$\sum_{m=1}^{M} w_m \leq 1, \quad w_m \geq 0 \; \forall m. \tag{4.7}$$

Given $w_m \geq 0$ and $\sum w_m \leq 1$, each individual weight is implicitly bounded above by 1, so no explicit upper bound is needed.

## 4.2 Vector-based Local Optimisation (VeLO)

The **VeLO** controller, developed by the supervising group, learns the displacement model from observed particle motion rather than from re-simulation. It maintains a per-particle model $\hat{F} \in \mathbb{R}^{2\times M\times n_P}$: for each mode $m$ and particle $p$, $\hat{F}(:,m,p)$ holds an estimate of the 2D displacement that mode $m$ produces on particle $p$ at the current position. Similarly to QP, a $10^{-5}I_M$ regulariser is added to keep the problem well-conditioned, unlike the original implementation from the supervising group.

### 4.2.1 Warmup Sweep

When first called, VeLO sequentially fires each mode once, the sweep phase, and observes the resulting displacement. Positions are tracked in normalised coordinates throughout: before and after each probe step the position of particle $p$ is read as $(x_p/w,\ y_p/h)$, so the observed displacement

$$\Delta x_p = \begin{bmatrix} x_p^{\text{new}}/w - x_p^{\text{prev}}/w \\ y_p^{\text{new}}/h - y_p^{\text{prev}}/h \end{bmatrix} \tag{4.8}$$

is dimensionless and lives in the same space as the state vector $q$ (Eq. (3.1)). The model is updated using an exponential moving average (EMA) with smoothing factor $\gamma = 0.5$:

$$\hat{F}(:,m,p) \leftarrow \gamma \hat{F}(:,m,p) + (1-\gamma)\,\Delta x_p. \tag{4.9}$$

At the same time, a unit-direction version $F_{\text{unit}}(:,m,p)$ is maintained and used to compute a reliability score

$$\text{Var}(m,p) = 1 - \|F_{\text{unit}}(:,m,p)\|_2, \tag{4.10}$$

which approaches zero for modes that consistently push particle $p$ in the same direction, and approaches one for modes whose effect is position-dependent and variable. If all particles reach their targets before the sweep completes, the controller exits early.

### 4.2.2 Online Control Phase

Once the model is seeded, VeLO solves an unconstrained non-negative QP at each call:

$$z = \arg\min_{w\geq 0} \sum_{p=1}^{n_P} \|\hat{F}_p w - \alpha_s(q_{r,p} - q_p)\|^2 + \lambda\, \overline{\text{Var}}^T w, \tag{4.11}$$

where $\hat{F}_p = \hat{F}(:,:,p)$ is the $2\times M$ slice for particle $p$, $q_{r,p}$ and $q_p$ are the 2D normalised goal and current position, $\lambda = 0$ in the current configuration setting the variance inactive as the baseline VeLO, $\overline{\text{Var}}$ is the mean reliability penalty across particles, and $\alpha_s = 10^6$ is a target-scaling constant. Because $\hat{F}$ stores normalised displacements of low order

per step, the unscaled gradient $-2\hat{F}_p^T(q_{r,p} - q_p)$ is numerically too small. Therefore, multiplying the target by $\alpha_s$ restores a workable gradient magnitude. The accumulated Hessian $H = \sum_p \hat{F}_p^T \hat{F}_p$ is likewise of order $10^{-6}$ and can be near-singular. A small Tikhonov term [24] is added before solving, $H \leftarrow H + 10^{-5} I_M$, to ensure positive definiteness and a well-conditioned QP. The next step is selecting the five modes carrying the largest weights in the solution $z$.

$$[m_1, m_2, m_3, m_4, m_5] = \text{top-5}\big(z\big). \tag{4.12}$$

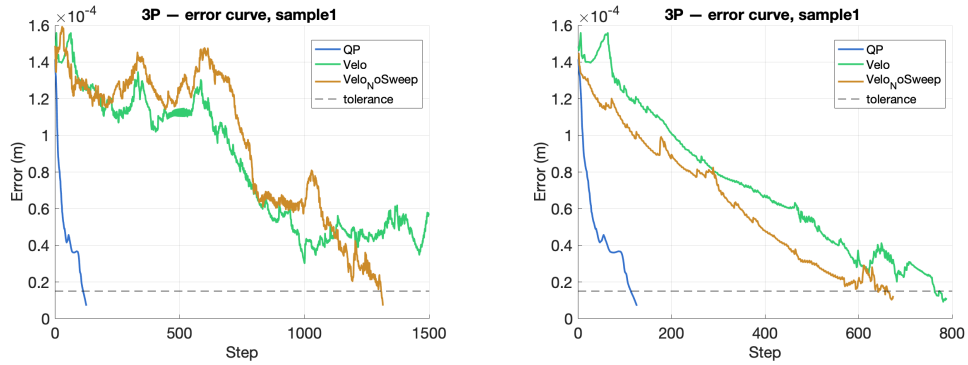
Each selected mode is applied as a pure single-mode step, the model is updated via Eq. (4.9) after each step, and the loop terminates early if all particles converge.

### 4.2.3 VeLO Without Sweep (VeLO_No-Sweep)

An alternative initialisation strategy, identified in this work as occasionally outperforming the full-sweep variant, skips the warm-up phase entirely. The model $\hat{F}$ is instead seeded with small random noise ($10^{-6} \cdot \mathcal{N}(0,1)$). After that, the controller starts acting immediately and learns each column of $\hat{F}$ the first time that mode is selected during the rollout. All other aspects are identical to the full-sweep variant. The motivation for skipping the sweep goes beyond just saving steps. Firing every mode in sequence during warm-up can push particles into poor positions that are difficult or even impossible to recover from within the remaining step budget. By starting exploitation immediately, VeLO_No-Sweep avoids this risk, but not entirely. The first several rollouts are guided by an essentially random model, so early moves may be inefficient.

### 4.2.4 Effect of Hessian Regularisation on the VeLO Family

The QP solved inside VeLO is numerically sensitive, and the reliability of the model-estimating controllers depends critically on the Tikhonov term $10^{-5} I_M$ added to the Hessian. Removing it degrades both VeLO variants substantially. Without regularisation, at $n_P = 2$ VeLO falls to 86 % (451 steps) and VeLO_No-Sweep to 80 % (460 steps). At $n_P = 3$ the gap widens, with VeLO at 78 % (859 steps) and VeLO_No-Sweep at 72 % (776 steps). Adding regularisation term lifts both variants into the 90–100 % range reported in the main benchmark (§6.1) and roughly reduces their step counts by a third. Figure 4.1 contrasts a representative $n_P = 3$ convergence history with and without the term. QP is only for reference use in that figure.




(a) Without Hessian regularisation

(b) With $10^{-5} I_M$ regularisation

Figure 4.1: Distance-to-goal convergence for QP, VeLO, and VeLO_No-Sweep at $n_P = 3$ in the polystyrene environment ($p_A = 44.4$ kPa, no interactions), without (a) and with (b) the $10^{-5} I_M$ Tikhonov term on the VeLO Hessian. Without regularisation the model-estimating controllers solve a near-singular QP and VeLO did not converge given 1500 steps. Adding the term restores convergence at a reasonable rate and step count. QP is included in the plot as a reference.

## 4.3 $\varepsilon$-Greedy Bandit

The **$\varepsilon$-greedy** controller treats mode selection as a multi-armed bandit problem. Each mode is assigned an estimated quality value $Q_m$, initialised to zero. At each step the controller either explores, with probability $\varepsilon$, picking a mode at random, or exploits, with probability $1-\varepsilon$, picking the mode with the highest $Q_m$. After each step, the quality estimate of the selected mode $a$ is updated by

$$Q_a \leftarrow Q_a + \frac{r - Q_a}{N_a}, \qquad N_a \leftarrow N_a + 1, \tag{4.13}$$

where $N_a$ is the visit count, and $r$ is a reward based on the displacement toward the goal:

$$r = \sum_{p=1}^{n_P} \Delta r_p \cdot \hat{g}_p, \qquad \hat{g}_p = \frac{g_p - x_p^{\text{prev}}}{\| g_p - x_p^{\text{prev}} \|}. \tag{4.14}$$

The exploration rate decays as $\varepsilon \leftarrow \max(0.01, \varepsilon \times 0.999)$, reducing exploration as the controller gains experience. Unlike QP, it never blends modes: the output is always a pure-mode weight vector.

## 4.4 Reinforcement Learning

The Q-learning controllers described in Sections 4.5 and 4.6 are instances of tabular Q-learning, a model-free reinforcement learning algorithm. To understand how it was developed, we need to take a closer look at the basic ideas behind it from Markov chains through Markov Decision Processes to the Q-learning update rule.

### 4.4.1 Markov Chains and the Markov Property

A Markov chain is a stochastic process $\{S_t\}$ over a discrete state space $\mathscr{S}$ satisfying the Markov property: the conditional distribution of the next state depends only on the current state, not on the full history of past states [25]:

$$\Pr(S_{t+1} = s' \mid S_t = s, S_{t-1}, \ldots, S_0) = \Pr(S_{t+1} = s' \mid S_t = s). \tag{4.15}$$

This memorylessness means that the current state alone is sufficient to predict all future behaviour. In the acoustophoretic context, the particle positions $q$ fully determine the acoustic forces through the Gor'kov potential Eq. (2.3), so the next state depends only on $q$ and the chosen mode weights, not on the history of how $q$ was reached.

### 4.4.2 Markov Decision Processes

A Markov Decision Process (MDP) extends a Markov chain by adding an agent that picks actions [26, 25]. Formally, an MDP is the tuple $(\mathscr{S}, \mathscr{A}, P, R, \gamma)$, where:

- $\mathscr{S}$ is the state space, discretised particle positions.
- $\mathscr{A}$ is the action space, set of $M$ acoustic modes.
- $P(s' \mid s,a)$ is the transition probability of reaching state $s'$ from $s$ under action $a$.
- $R(s,a)$ is the expected immediate reward for taking action $a$ in state $s$.
- $\gamma \in [0,1)$ is the discount factor, which downweights rewards that arrive in the distant future.

The agent's goal is a policy $\pi : \mathscr{S} \to \mathscr{A}$ that maximises the expected discounted return $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$. The Bellman optimality equations characterise the optimal policy through the optimal action-value function $Q^*$:

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s'} P(s' \mid s,a) \max_{a'} Q^*(s',a'). \tag{4.16}$$

If $Q^*$ is known, the optimal action at any state is simply $\pi^*(s) = \arg\max_a Q^*(s,a)$, without the need to calculate anything at runtime.

### 4.4.3 Q-Learning

The difficulty of solving Eq. (4.16) directly is that the transition probabilities $P(s' \mid s,a)$ are typically unknown, and the sum over all next states is computationally intractable for large $|\mathscr{S}|$. Q-learning, introduced by Watkins and Dayan [27], sidesteps both issues

by updating a tabular estimate of $Q^*$ directly from observed experience. After taking action $a_t$ in state $s_t$, receiving reward $r_t$, and transitioning to state $s_{t+1}$, the update is:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\Big[r_t + \gamma \max_{a'} Q(s_{t+1},a') - Q(s_t,a_t)\Big], \tag{4.17}$$

where $\alpha \in (0,1]$ is the learning rate and the temporal-difference (TD) error is the gap between the current estimate and the target $r_t + \gamma \max_{a'} Q(s_{t+1},a')$. Watkins and Dayan proved that if every state-action pair is visited infinitely often, and the learning rate satisfying the Robbins–Monro summability conditions ($\sum_t \alpha_t = \infty,\ \sum_t \alpha_t^2 < \infty$) the table $Q(s,a)$ converges to $Q^*(s,a)$ for all $s,a$ [27].

During training, the agent must balance exploration, trying actions whose value is uncertain, with exploitation, choosing the currently estimated best action. The $\varepsilon$-greedy policy used throughout this thesis resolves this by selecting a random action with probability $\varepsilon$ and the greedy action otherwise. $\varepsilon$ decays over training episodes so that exploration is concentrated in the early stages when estimates are most uncertain. The primary architectural choice separating the two Q-learning implementations below is the state discretisation: a uniform grid (Section 4.5) versus tile coding (Section 4.6), the latter reducing state aliasing without a proportional increase in memory.

## 4.5 Q-Learning: Baseline Implementation (QG)

Perticarari [5] implemented a tabular Q-learning controller, referred to here as **QG**, based on a uniform-grid state discretisation. The state space is partitioned with a fixed spatial spacing, yielding a grid of states.

Training applies the update of Eq. (4.17) with fixed learning rate $\alpha = 0.5$, discount factor $\gamma = 0.99$, and fixed exploration probability $\varepsilon = 0.1$. The reward is the raw change in distance to the goal: $r = \|q - q_r\| - \|q' - q_r\|$, with no step penalty or success bonus.

## 4.6 Tile-Coded Q-Learning

The principal limitation of QG is state aliasing: the uniform grid cannot distinguish states that fall within the same bin, so particles in similar but not identical positions receive identical Q-values.

Tile coding [26] addresses this by overlaying multiple offset grids (tilings) over the state space and representing each state as a combination of one index of each tiling. This generates a richer feature representation without a proportional increase in memory (Figure 4.2).

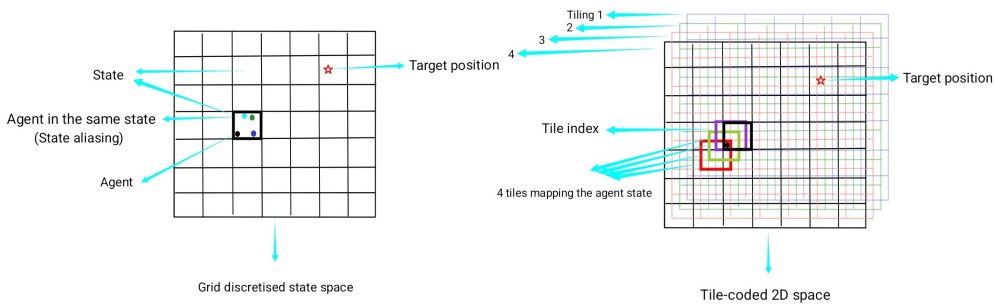



Figure 4.2: Tile-coding in two dimensions. Several offset tilings overlay the continuous state space. A state activates exactly one tile per tiling, here four, and the sum of those tiles represents it. This yields finer resolution than a single grid without a proportional increase in memory.

### 4.6.1 State Representation

With $N_T = 4$ tilings and resolution $R$ per dimension, each tiling uses a grid of $R^{2n_P}$ cells. The $t$-th tiling is offset by $(t-1)/N_T$ of the bin width relative to the origin, so adjacent tilings disagree by exactly one cell at each boundary. The Q-table has shape $(R^{2n_P}) \times M \times N_T$, and the value estimate for state $q$ under action $m$ is the sum of the table entries across all tilings:

$$\hat{Q}(q,m) = \sum_{t=1}^{N_T} Q(\mathrm{idx}_t(q), m, t)\,. \tag{4.18}$$

The resolution is set to $R = 20$ for $n_P = 1$, $R = 12$ for $n_P = 2$, and $R = 8$ for $n_P = 3$, keeping the total memory tractable.

### 4.6.2 Enhancements Over QG

Four modifications distinguish this implementation from QG:

1. **Epsilon decay.** The exploration rate follows $\varepsilon_k = \max(\varepsilon_{\text{floor}}, \varepsilon_0/(1+k\,\delta_\varepsilon))$ where $k$ is the episode index. For $n_P = 1$, the parameters are $\varepsilon_0 = 0.3$, $\varepsilon_{\text{floor}} = 0.001$, $\delta_\varepsilon = 0.03$, for $n_P \geq 2$ the decay is slower to allow more exploration of the larger joint state space.

2. **Alpha decay.** Rather than a fixed learning rate, a Robbins–Monro schedule is used: $\alpha_{s,m} = 1/(1+N_{s,m})^{0.7}$, where $N_{s,m}$ is the number of times the state-action pair $(s,m)$ has been visited. This satisfies the Robbins–Monro summability conditions introduced in Section 4.4, ensuring convergence guarantees while still allowing fast early learning.

3. **Shaped reward.** The reward includes a step penalty and a success bonus: $r = (\|q - q_r\| - \|q' - q_r\|) \times 2 \times 10^5 - \text{pen}$, where the step penalty encourages the agent to converge quickly. Upon reaching the goal, an additional success bonus equal to $\|q_0 - q_r\| \times 2 \times 10^5$ is added.

4. **Parallel training.** Tables for all initial conditions are trained simultaneously using MATLAB's `parfor`, reducing total training time substantially.

### 4.6.3 Deployment

At deployment time, the controller loads the pre-trained Q-table from a `.mat` file, computes the tile indices for the current state, sums the per-tiling Q-values, and selects the action with the highest aggregate value:

$$a^* = \arg\max_m \hat{Q}(q, m). \tag{4.19}$$

For the benchmark comparison, training time is folded into the reported computational time.

# Chapter 5

# Methods: Simulation Framework and Benchmark

## 5.1 Simulation Architecture

The simulation environment has a set of MATLAB structs ( `device`, `fluid`, `particles`, `modes`, `trajectories`, and `timeStepper`) that collectively specify the physical parameters and state of the system. The dynamics are encapsulated in the function `func_positionUpdate_2`, which accepts the current state and a weight vector and returns the updated state after one time step. This interface is shared across all algorithms, ensuring that differences in performance arise from the control policy. For the interaction-enabled setting, the dynamics function is replaced by `func_positionUpdate_TB`.

## 5.2 Setup Function and Device Parameters

A single setup function, `setupB40`, configures the simulation for every controller. It fixes one physical environment throughout the thesis: an $800 \times 600$ $\mu$m chamber filled with water ($c = 1500$ m/s, $\rho = 1000\,\mathrm{kg/m^3}$, $\kappa = 458 \times 10^{-12}\,\mathrm{Pa^{-1}}$). The chamber holds polystyrene beads with scattering coefficients $f_1 = 0.476$ (monopole) and $f_2 = 0.0323$ (dipole) and radii 3.5–7 $\mu$m, and $\Delta t = 0.5$ s. The mode count grows with particle count, $N_{\mathrm{dim}} = 6, 7, 8$ for $n_P = 1, 2, 3$ (giving $M = 36, 49, 64$ modes).

The only parameter that varies is the acoustic pressure amplitude $p_A$. The main benchmark (QP, VeLO, VeLO_No-Sweep) uses $p_A = 44.4$ kPa. The Q-learning comparisons and the related suites use $p_A = 83$ kPa, because a stronger acoustic field produces larger per-step displacements and therefore faster convergence. That has substantially shortened the offline Q-table training without changing the physical environment. All other parameters are identical across controllers, so performance differences reflect the

control policy rather than environment tuning. Table 5.1 summarises the configuration.

Table 5.1: Mode count ($N_{\text{dim}}^2$) and pressure amplitude $p_A$ per particle count for the polystyrene environment. The main benchmark uses $p_A = 44.4$ kPa. The Q-learning comparisons use $p_A = 83$ kPa to speed up training. All other parameters are shared.

| | $n_P = 1$ | $n_P = 2$ | $n_P = 3$ |
|---|---|---|---|
| Mode count $M = N_{\text{dim}}^2$ | $6^2 = 36$ | $7^2 = 49$ | $8^2 = 64$ |
| $p_A$, main benchmark (kPa) | 44.4 | 44.4 | 44.4 |
| $p_A$, Q-learning comparisons (kPa) | 83 | 83 | 83 |

## 5.3 Deterministic Initial Conditions and Reproducibility

Reproducible benchmarking requires that every algorithm faces exactly the same start-goal pairs. This is achieved by the `getSampleIC2` function, which generates positions deterministically from a seeded random number generator. For the sample index $r$, and particle count $n_P$, the seed is

$$s_{\text{seed}} = n_P \times 1000 + r, \tag{5.1}$$

and positions are drawn uniformly from the active chamber area $[x_{\text{buffer}}, w/2 - x_{\text{buffer}}] \times [y_{\text{buffer}}, h/2 - y_{\text{buffer}}]$ with a buffer of 60 $\mu$m to avoid placing particles near the walls. A minimum separation check of 30 $\mu$m is applied to the starting positions to prevent particle overlap at initialisation.

Stochastic controllers (VeLO, $\varepsilon$-greedy, and Q-learning during training) produce variable results across runs, which can mask genuine performance differences if the random state is not controlled. The seed is therefore fixed before each sample using `rng(nP * 100 + r)`. The resulting generator state is captured, and restored identically before every controller runs on that sample. This guarantees that the results are exactly reproducible from the saved seed, enabling verification of the reported numbers. See Section 5.7 for details on the toolboxes and machine used.

## 5.4 BenchmarkB50 Suites

Two benchmark suites are used, each with 50 samples per particle count and collectively referred to as **BenchmarkB50**. Both run in the single polystyrene environment defined in Section 5.2. They differ only in the controller group and the pressure amplitude:

- **Suite A** (main benchmark, $p_A = 44.4$ kPa): evaluates QP, VeLO, and VeLO_No-Sweep.

- **Suite B** (Q-learning comparisons, $p_A = 83\,\text{kPa}$): evaluates QP, **$\varepsilon$-greedy**, and tile-coded Q-learning. The higher pressure amplitude produces larger per-step displacements, shortening the offline Q-table training.

The step limits differ by particle count to account for longer convergence times at higher $n_P$: $T_{\max} = 500/1000/1500$ for $n_P = 1/2/3$ across the QP-based comparisons. The exact budget for each comparison is listed in Table 5.2.

One point about the design should be kept in mind when reading the results: **QP appears in both suites**, but at different pressure amplitudes (44.4 kPa in Suite A, 83 kPa in Suite B), therefore, its step counts differ between the two and must be compared within a suite, not across. Table 5.2 summarises the full set of comparisons.

Table 5.2: Overview of the BenchmarkB50 comparisons. All use the polystyrene environment with $\Delta t = 0.5\,\text{s}$ and are run both without and with interactions. Mode counts are $M = N_{\text{dim}}^2$ with $N_{\text{dim}} = 6,7,8$ for $n_P = 1,2,3$. This choice sits well above the $M \geq 4P$ controllability threshold reported by Perticarari [5], whose heuristic $R(M,P) \approx 1 - P/M$ relates the success rate to the mode-to-particle ratio. The circle stress test (Table 6.2) is the only experiment that departs from these mode counts. $p_A$ in kPa. $N$ is the number of samples per particle count.

| | Main bench. (§6.1) | Q vs $\varepsilon$-greedy (§6.2) | QP vs Q-learning (§6.3) |
|---|---|---|---|
| Controllers | QP, VeLO, VeLO_No-Sweep | Q-learning, $\varepsilon$-greedy | QP, Q-learning |
| $p_A$ (kPa) | 44.4 | 83 | 83 |
| $n_P$ | 1,2,3 | 2 | 1,2,3 |
| Modes $M$ | 36/49/64 | 49 | 36/49/64 |
| $T_{\max}$ | 500/1000/1500 | 1000 | 500/1000/1500 |
| $N$ | 50 | 50 | 50 / 10[†] |

[†] The interaction-trained three-particle comparison uses 10 samples rather than 50, because each interaction-environment Q-table requires $\approx 3\,\text{h}$ to train.

The shared structure for both suites is:

1. For each particle count $n_P \in \{1,2,3\}$ and each sample index $r \in \{1,\ldots,50\}$:
   (a) Generate initial and goal positions using `getSampleIC2(nP, r)`.
   (b) Set the RNG state to `rng(nP * 100 + r)` and capture the state.
   (c) For each controller $c$ in the controller list:
      i. Restore the captured RNG state.
      ii. Clear all persistent controller variables (`clear VeLO VeLO_No-Sweep`).
      iii. Initialise the environment via `setupB40`.
      iv. Run the episode until convergence or $T_{\max}$ steps.
      v. Record $T$, $e_{\text{final}}$, success/failure, and Computational time.

The step count for VeLO and VeLO_No-Sweep is measured as the growth of the trajectory array rather than the number of outer-loop iterations, because a single call to these controllers can advance the trajectory by up to five physical steps.

## 5.5 Circle Stress Test

To see how the system handles larger numbers of particles, samples with more than three particles are tested. A similar experiment was performed by Perticarari [5], without accounting for interactions. The test applies QP and covers $n_P \in \{10, 20\}$.

Initial positions are drawn uniformly at random from the full chamber $[0, W] \times [0, H]$. Target positions are placed on a circle of radius

$$r = \tfrac{1}{4} \min(W, H) = 150\,\mu\text{m}, \tag{5.2}$$

centered at $(W/2, H/2) = (400, 300)\,\mu$m, at angles $\theta_p = 2\pi(p-1)/n_P$ for $p = 1, \ldots, n_P$. The target positions are therefore separated by $2r\sin(\pi/n_P)$: approximately $93\,\mu$m for $n_P = 10$ and $47\,\mu$m for $n_P = 20$.

The mode count is set to $N_{\text{dim}} = 10$ ($M = 100$ modes) for $n_P = 10$ and $N_{\text{dim}} = 15$ ($M = 225$ modes) for $n_P = 20$. Each particle count is evaluated twice: once without interactions (ARF only) and once with interaction-enabled dynamics (AIF, hydrodynamic coupling, and contact). Because only a single sample is collected per condition, the test is a qualitative scalability probe rather than a statistically representative benchmark.

## 5.6 Performance Metrics and Result Visualisation

An episode is said to converge when all $n_P$ particles are simultaneously within the tolerance $d_{\text{tol}} = 15\ \mu$m ($d_{\text{tol}} = 10\ \mu$m for the circle stress test) of their current goal positions. Once the final goal of every particle is reached, the episode terminates. Four metrics are used to compare the controllers:

- **Step count** $T$: the number of integration steps from episode-start to convergence, or the maximum limit $T_{\max}$ if the episode does not converge.
- **Success rate** $S$: the fraction of trials in which the algorithm converges within the step limit.
- **Computational time** $t_w$: the Computational time per episode. For Q-learning, training time is included since the offline phase is an unavoidable cost of that approach.
- **Final error** $e_{\text{final}}$: the mean Euclidean distance from each particle's last position to its goal at episode end, averaged over all particles.

All metrics except the success rate are computed over successful episodes only, so that comparisons reflect how well each algorithm performs when it does converge, rather than being skewed by failed runs.

The benchmark results are summarised graphically rather than as raw tables. Three types of plot are used throughout the results chapter.

**Distribution (box) plots** (Figure 5.1) are used for the step count for each controller. The **box** covers the interquartile range (IQR), with the median shown as the line inside the box. The **whiskers** reach the farthest points within $1.5\times$IQR, and anything beyond them is an outlier. A **black diamond** marks the sample mean. A **vertical black bar** shows the 95 % confidence interval of the mean ($\bar{x} \pm 1.96\,\sigma/\sqrt{n}$). The number of successful samples in each box is shown above it as `n=`. Only successful episodes are included, following the same rule as used for the other metrics. **Success-rate bar charts** report the single fraction of converged episodes per controller, annotated with the percentage and a reference line at 100 %.

**Per-episode plots** show, for a representative sample, the particle trajectories in the chamber (start, goal, and final markers per particle). The **distance-to-goal convergence curve** (per-particle error against step number, with the $15\,\mu$m tolerance drawn as a horizontal line).

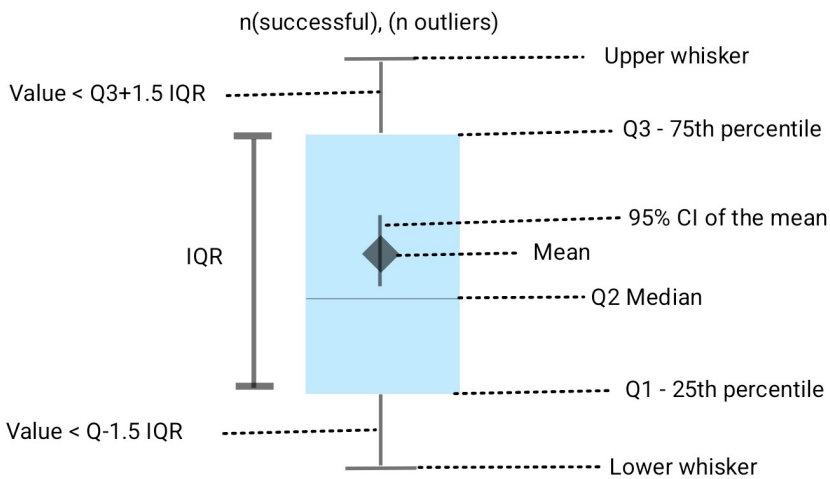



Figure 5.1: This figure shows the distribution box plot components

## 5.7 Software Environment

All simulations are implemented on a macOS operating system (Intel chip) in MATLAB (R2024b) and require the Optimisation Toolbox (for `quadprog`, version 24.2) and the Parallel Computing Toolbox (for `parfor` during Q-table training, version 24.2). All results, figures, and plots are generated using MATLAB.

Algorithm-specific persistent state is reset between episodes using `clear functions`, so no state from one sample or particle count carries over to the next. Q-table training is performed offline before the benchmark, the trained table file is loaded once at the start of each deployment episode via the global variable `QTABLE_FILE`.

# Chapter 6

# Results

This chapter presents the BenchmarkB50 results for the polystyrene environment across three particle counts ($n_P \in \{1,2,3\}$), run both without and with particle-particle interactions.

The chapter is organised as a sequence of specific questions. Section 6.1 reports the main benchmark (QP, VeLO, VeLO_No-Sweep), without and with interactions. Section 6.2 screens the two reinforcement-style controllers against each other. Section 6.3 then compares QP against tile-coded Q-learning, without and with interactions. Finally, Section 6.4 presents a scalability stress test at $n_P \in \{10,20\}$ with circle-arrangement targets, probing QP's performance well beyond the three-particle benchmark.

Because step count and Computational time are averaged over successful episodes only, they are not directly comparable between controllers with very different success rates: a controller that fails the harder initial conditions reports a mean step count over the easier ones it did solve.

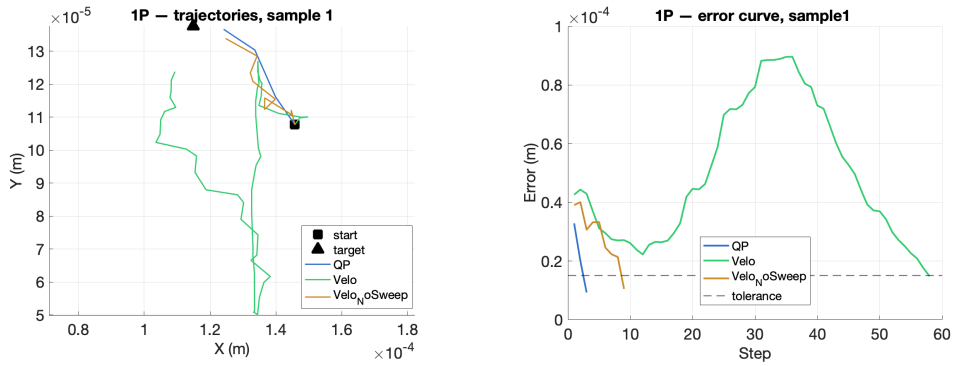
## 6.1 Main Benchmark: Polystyrene Environment

Chamber $800 \times 600$ $\mu$m, polystyrene beads ($f_1 = 0.476$, $f_2 = 0.0323$), $\Delta t = 0.5$ s, $p_A = 44.4$ kPa. Mode counts: $6^2 = 36$ ($n_P = 1$), $7^2 = 49$ ($n_P = 2$), $8^2 = 64$ ($n_P = 3$). Step limits are $T_{\max} = 500/1000/1500$ for $n_P = 1/2/3$. The non-interaction and interaction runs are reported together: the success rate and step distributions are shown side by side in Figures 6.2–6.3, and a representative two-particle episode in Figure 6.4—6.5.

### 6.1.1 Without Particle-Particle Interactions

All three controllers reach the targets reliably, and QP is the most step-efficient, requiring only $10-20\,\%$ of the total number of steps needed by VeLO variants. At $n_P = 1$ every

controller converges in all 50 samples. QP needs a mean of only 13.8 steps against 126.3 (VeLO) and 124.2 (VeLO_No-Sweep). VeLO, on average, spends most of its step budget on the warmup sweep before taking effective steps (Figure 6.1). At $n_P = 2$, QP holds 98 % success rate, with VeLO and VeLO_No-Sweep both at 92 %. At $n_P = 3$ QP returns to 100 % with 71.8 steps, VeLO_No-Sweep also reaches 100 %, in 518.8 steps, and VeLO is close behind at 96 %, in 555.9 steps (Figure 6.3, a).




(a) Trajectory

(b) Distance-to-goal convergence

Figure 6.1: Polystyrene environment, $n_P = 1$, no interactions. All three controllers converge. QP reaches the goal in 3 steps. VeLO_No-Sweep converges in 9 steps. VeLO must first complete the mode sweep and therefore converges after 58 steps.

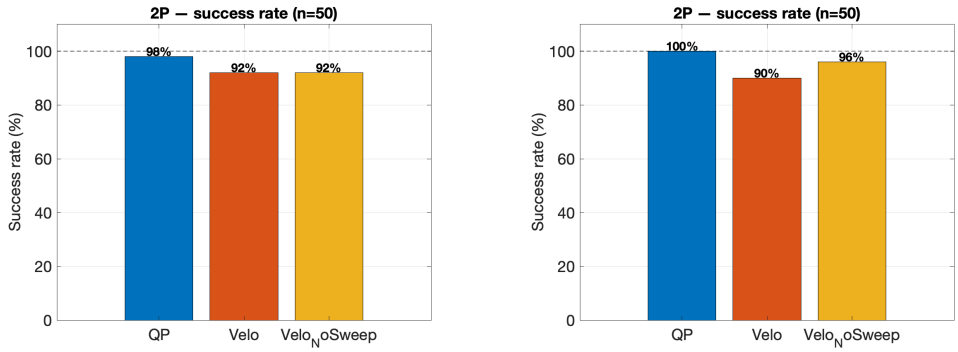
### 6.1.2 With Particle-Particle Interactions

At $n_P = 1$, the interaction run serves only as a dynamics check, with a single particle, there are no inter-particle forces, and the AIF evaluates to zero. Success rates and step counts are therefore identical to the non-interaction case and all three controllers at 100 %. Only Computational time increases, reflecting the overhead of evaluating the interactions at every step.

At $n_P \geq 2$, the AIF, hydrodynamic coupling, and contact handling are all active. The central finding is that, with Hessian regularisation (§4.2.3), all three controllers remain robust under interactions. **QP** rises to 100 % at both multi-particle counts (98 $\rightarrow$ 100 % at $n_P = 2$, 100 $\rightarrow$ 100 % at $n_P = 3$). The model-estimating controllers remain reliable: **VeLO_No-Sweep** is at 96 % at $n_P = 2$ and 98 % at $n_P = 3$, and **VeLO** at 90 % and 96 %. The success-rate comparison at $n_P = 2$ is shown in (Figure 6.2).

The step distributions (Figure 6.3) again separate the controllers on efficiency rather than reliability. QP remains low and tight under interactions 71.8 $\rightarrow$ 81.6 steps at $n_P = 3$. VeLO and VeLO_No-Sweep converge but take far longer and more variable trajectories $\approx 635$ and $\approx 708$ mean steps at $n_P = 3$. The computational cost of the interaction framework is also visible: at $n_P = 3$ the per-episode time rises from 1.2 s to 16.77 s for

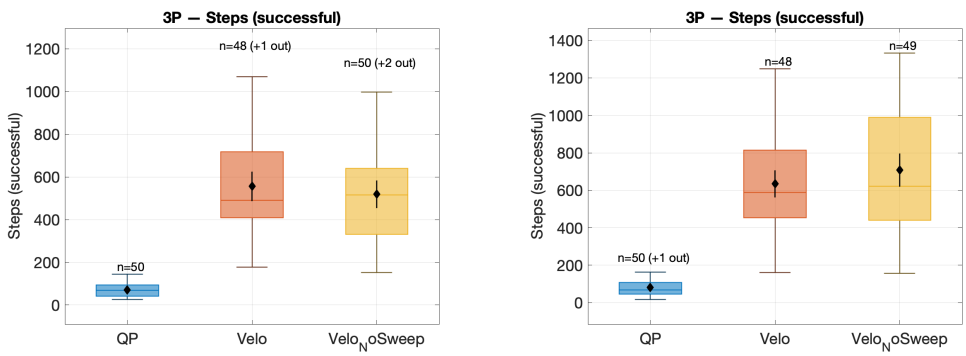
QP and from $\approx 2.4$ s to $\approx 28$ s for the model-estimating controllers.




(a) No interactions

(b) With interactions

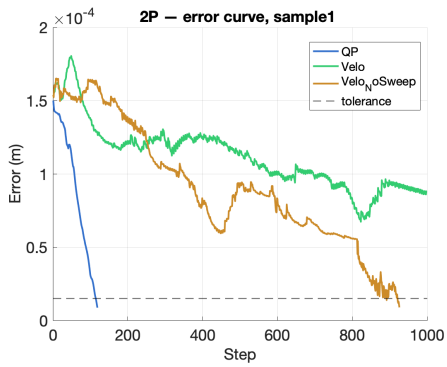
Figure 6.2: Success rate at $n_P = 2$ for the three main-benchmark controllers, polystyrene environment, without (a) and with (b) interactions. All three controllers remain at or above 90 % in both settings : QP 98–100 %, VeLO_No-Sweep 92–96 %, VeLO 90–92 %.



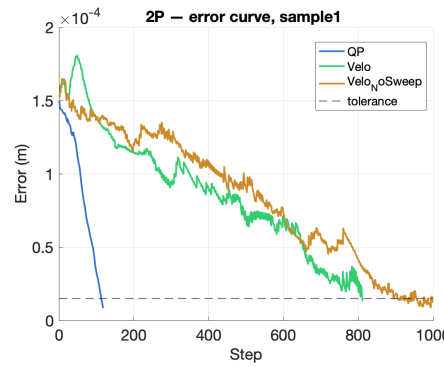

(a) No interactions

(b) With interactions

Figure 6.3: Distribution of step counts to convergence at $n_P = 3$, polystyrene environment, without (a) and with (b) interactions. QP's distribution is low and tight in both settings, while VeLO and VeLO_No-Sweep converge reliably but at an order of magnitude more steps.

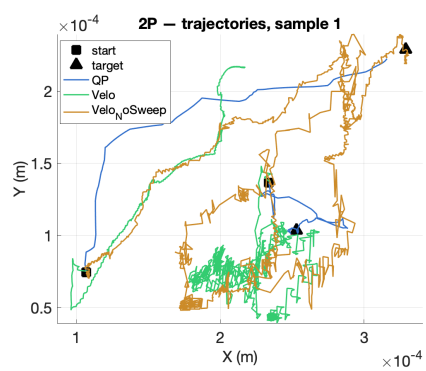



(a) Convergence, no interactions

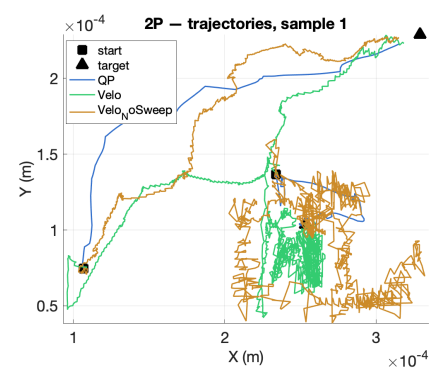



(b) Convergence, with interactions

Figure 6.4: Two-particle episode in the polystyrene environment (Suite A, $p_A = 44.4\,\text{kPa}$), shown without (a) and with (b) interactions. QP converges in 118 steps in both settings. The model-estimating controllers are far slower and benefit from the inter-particle interactions. Without them, VeLO fails to reach tolerance within the step budget, while VeLO_No-Sweep converges only after $\approx 900$ steps. With interactions, both converge $\approx 800$–$900$ steps. This illustrates that the interactions can assist the learned-model controllers when they point toward the goal, rather than acting purely as a disturbance.




(a) Trajectories, no interactions




(b) Trajectories, with interactions

Figure 6.5: Two-particle episode in the polystyrene environment (Suite A, $p_A = 44.4\,\text{kPa}$), shown without (a) and with (b) interactions. Trajectories of all three controllers. Particle trajectory in green at the top center of (a) shows where VeLO fails to steer the first particle to its target position. QP is the smooth blue line that represents a clean low step-count path, while VeLO_No-Sweep struggles to steer the second particle shown at the bottom center of the plot(a). In (b), with interactions, VeLO successfully took the particle to its target, benefiting from the inter-particle interactions.

## 6.2 Q-Learning vs. $\varepsilon$-Greedy: Polystyrene

This section screens tile-coded Q-learning deployment against the $\varepsilon$-greedy bandit on the polystyrene environment at $n_P = 2$ (Suite B, 50 samples, $T_{\max} = 1000$), both without and with interactions. Q-learning training time is folded into the reported Computational time.

Q-learning achieves a perfect 100 % success rate 50/50 in both settings (Figure 6.6), at the cost of an expensive offline training phase. The $\varepsilon$-greedy bandit, by contrast, essentially fails. **Without interactions, it converges in 5 of the 50 episodes with a mean of 150 steps**, its online value estimates rarely stabilise within the 1000-step budget for the two-particle case, so its policy remains effectively random. **With interactions**, the bandit reaches a 22 % success rate, 11/50, with a mean of 215 steps, against Q-learning's 100 % success rate with a mean of 26 steps. The interaction-trained Q-table takes $\approx 1711$ s per episode with training time included versus the bandit's 7.54 s, but the $\varepsilon$-greedy online learning never approaches a usable success rate.

Given these results, $\varepsilon$-greedy is not investigated further. An algorithm that converges in 10 % of non-interacting two-particle episodes, and only 22 % under interactions, is not a viable multi-particle controller.

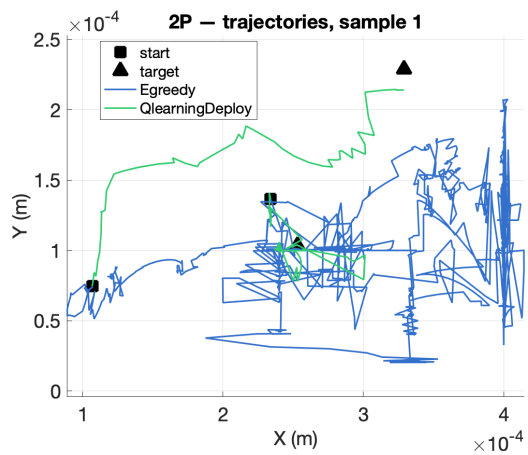



Figure 6.6: Q-learning vs. $\varepsilon$-greedy, polystyrene $n_P = 2$. Q-learning steers both particles directly to their targets, while $\varepsilon$-greedy wanders without converging.

## 6.3 QP vs Q-Learning

This section compares the model-based QP controller against tile-coded Q-learning in the polystyrene environment, across one to three particles with and without particle–particle interactions. For each setting, the Q-table is trained in the matching environment,

a non-interaction table for the non-interaction runs, and an interaction-trained table for the interaction runs. The setup follows Suite B ($M = 36/49/64$ modes for $n_P = 1/2/3$, $p_A = 0.83 \times 10^5$ Pa). Because in a three-particle interaction environment, the Q-table takes roughly three hours to train, only 10 samples were used for this case. All other conditions use 50. Throughout, the non-interaction run is shown beside the interaction run (Figures 6.7–6.9).

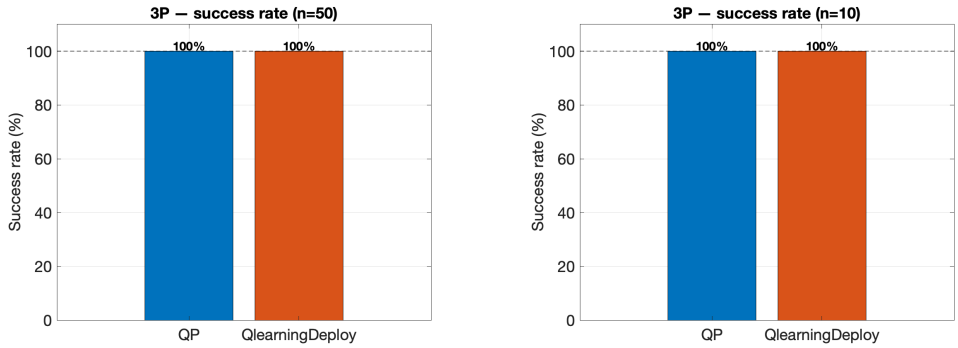
**Both controllers reach the targets reliably.** In every condition, both QP and Q-learning reach a 100 % success rate, the only exception being QP's single miss at $n_P = 2$ without interactions 98 %, 49/50 (Table 6.1, Figure 6.7). Enabling the interactions lowers neither controller's success, and Q-learning matches QP's reliability up to three interacting particles.

**QP is a relatively more efficient controller in terms of step count.** It converges in fewer steps at every particle count. Without interactions it needs $4.3/21.1/20.1$ steps at $n_P = 1/2/3$ against Q-learning's $6.4/23.3/31.0$. With interactions, the gap remains narrow at two particles 20.2 vs 26.1, but widens sharply at three 24.9 vs 139.6 (Figure 6.8). The three-particle interaction mean is heavy-tailed: a single hard configuration stretches Q-learning to 1033 steps against QP's 25. The same configuration is solved by Q-learning in only 35 steps once interactions are switched off (Figure 6.9).

**The decisive difference is the Computational time.** A single Q-learning episode with training included runs to $12.35/160.63/618.64$ s without interactions and to 1711 s ($n_P = 2$) and 10,642 s ($\approx 3$ h, $n_P = 3$) with interactions, against QP's 0.42–6.7 s throughout . Q-learning therefore matches QP in success and remains close in step count, but its training cost and the inflexibility of the frozen policy make QP the more practical choice for deployment.

Table 6.1: QP versus Q-learning in the polystyrene environment ($p_A = 0.83 \times 10^5$ Pa), without and with interactions. Success rate, mean step count, and mean per-episode Computational time, Q-learning's includes training time. The three-particle interaction run uses 10 samples. All other conditions use 50.

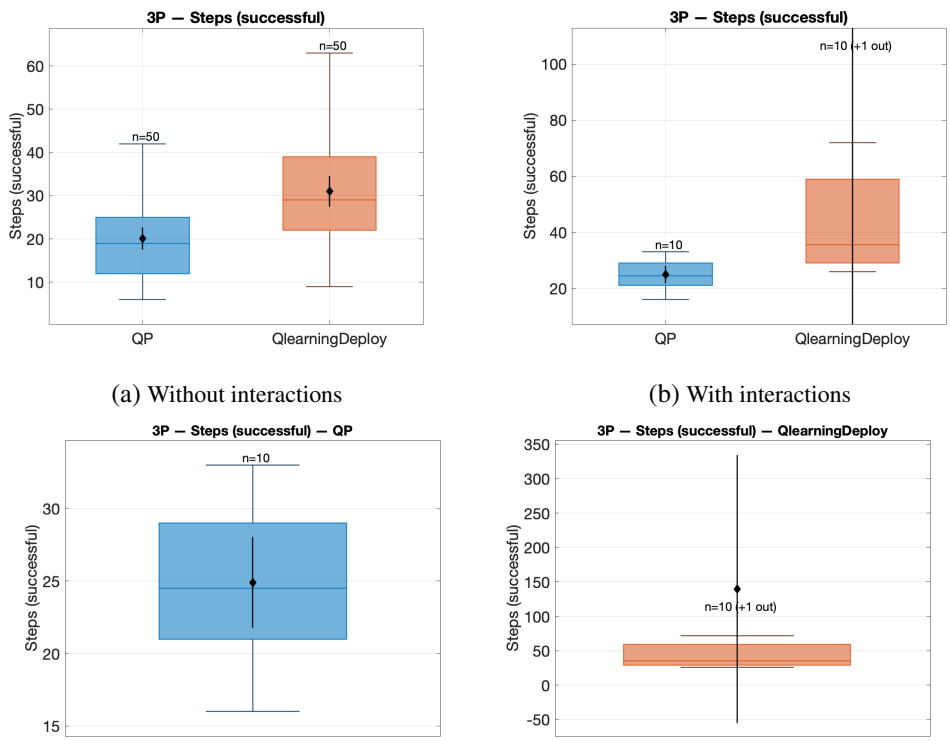
| | | QP | | | Q-learning | | |
|---|---|---|---|---|---|---|---|
| $n_P$ | Interactions | Succ. | Steps | Time (s) | Succ. | Steps | Time (s) |
| 1 | No | 100% | 4.3 | 0.42 | 100% | 6.4 | 12.35 |
| 2 | No | 98% | 21.1 | 0.66 | 100% | 23.3 | 160.63 |
| 3 | No | 100% | 20.1 | 0.7 | 100% | 31.0 | 618.64 |
| 2 | Yes | 100% | 20.2 | 3.33 | 100% | 26.1 | 1711 |
| 3 | Yes | 100% | 24.9 | 6.69 | 100% | 139.6 | 10642.55 |




(a) Without interactions

(b) With interactions

Figure 6.7: Success rate of QP and Q-learning at $n_P = 3$, without (a) and with (b) interactions. Both reach 100 % in either setting.



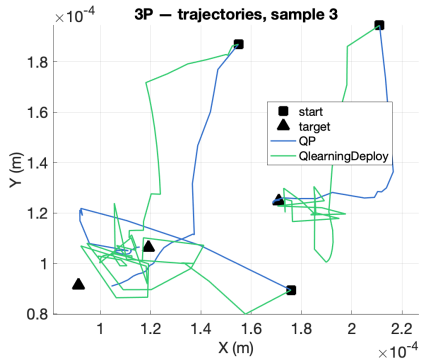

(a) Without interactions

(b) With interactions

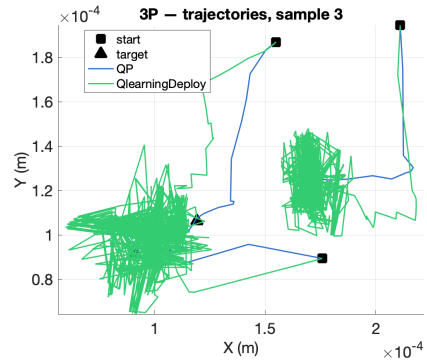
(c) QP With interactions

(d) Q-Learning With interactions

Figure 6.8: Step count distribution boxes of QP and Q-learning at $n_P = 3$, without (a) and with (b) interactions, and the with-interactions distributions shown separately for QP (c) and Q-learning (d). QP is more step-efficient in both settings. Enabling interactions raises Q-learning's tail mean 31.0 → 139.6, driven by an outlier at 1033 steps that dominates its mean and widens the 95% CI, while QP remains near 20 at 25 steps.




(a) Without interactions: QP 25, Q-learning 35 steps




(b) With interactions: QP 25, Q-learning 1033 steps

Figure 6.9: The same $n_P = 3$ configuration sample 3 for QP and Q-learning, without (a) and with (b) interactions. QP converges in $\approx 25$ steps in both. Q-learning still reaches all three targets but stretches from 35 to 1033 steps, showing that the heavy step shift is driven by the interaction physics, not by the controller. Interactions have driven the particles to rarely visited states under training, and that is causing instability of the Q-policy.

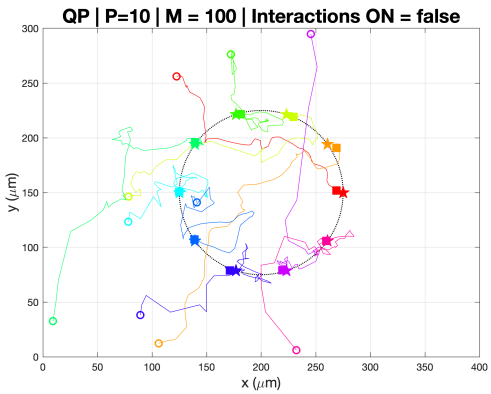
## 6.4 Scalability Stress Test: QP on Circle Arrangements

Rather than the three-particle benchmark used throughout the rest of this chapter, particle counts of $n_P = 10$ and $n_P = 20$ are evaluated, each with target positions equally spaced on a circle of radius $r = 150\,\mu$m centered at the chamber midpoint $(400, 300)\,\mu$m. Table 6.2 summarises the outcomes. QP converged successfully in all four conditions, with final positioning errors well below the $10\,\mu$m convergence tolerance in every case.

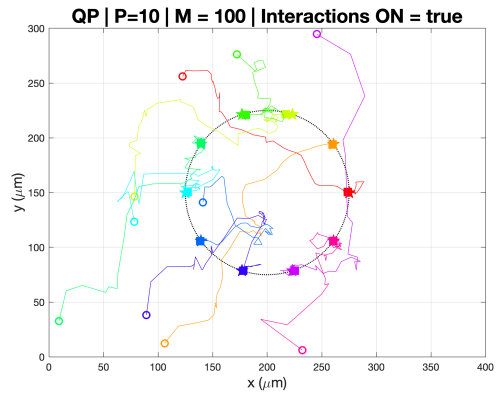
Table 6.2: QP stress-test results for circle arrangements at $n_P \in \{10, 20\}$. Steps to convergence, final positioning error, and Computational time are reported for one sample per condition. Interactions denote whether AIF, hydrodynamic coupling, and contact handling are enabled.

| $n_P$ | $M$ | Interactions | Steps | Final error (m) | Time (s) |
|---|---|---|---|---|---|
| 10 | 100 | No | 61 | $4.09 \times 10^{-6}$ | 1.75 |
| 10 | 100 | Yes | 87 | $1.93 \times 10^{-6}$ | 146.87 |
| 20 | 225 | No | 50 | $1.44 \times 10^{-6}$ | 2.64 |
| 20 | 225 | Yes | 61 | $1.45 \times 10^{-6}$ | 924.7 |

**Convergence is achieved at all tested scales.** QP converges for both $n_P = 10$ and $n_P = 20$ under both physical settings, demonstrating that the QP formulation remains well posed as the number of particles grows far beyond the three-particle benchmark.

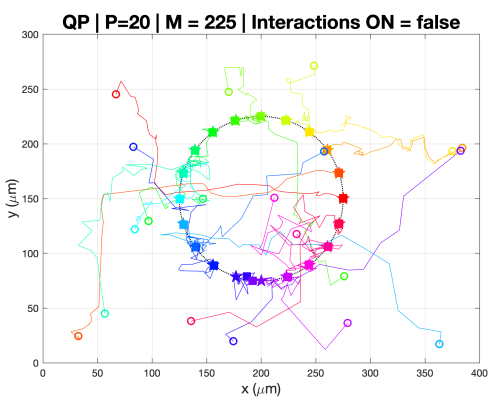



(a) No interactions $M = 100$, 61 steps, 1.75 s

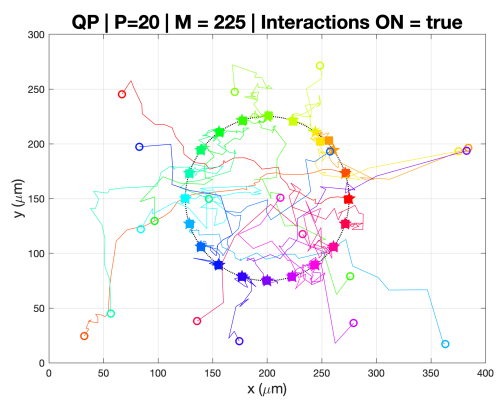



(b) With interactions $M = 100$, 87 steps, 146.87 s

Figure 6.10: QP circle stress test, $n_P = 10$, $M = 100$ modes. Each coloured line is a single particle trajectory. Circles denote random start positions, pentagrams denote target positions on the circle of radius 150 $\mu$m, and squares denote final positions. The dashed ring shows the target circle. a: ARF only. b: full interaction-enabled dynamics.

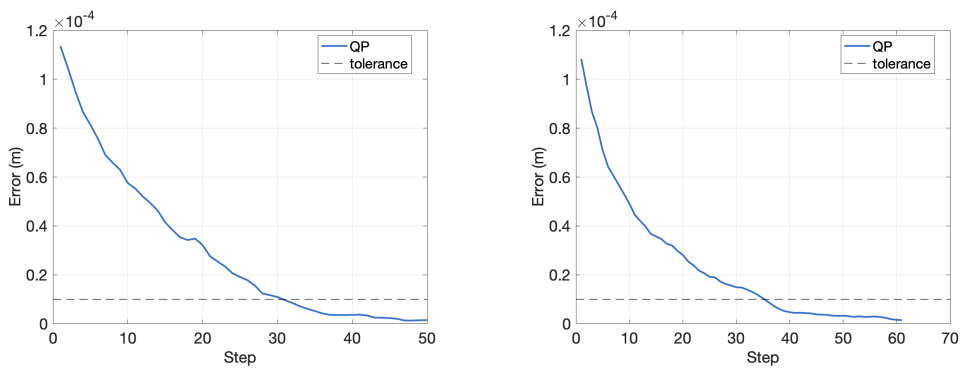



(a) No interactions $M = 225$, 50 steps, 2.64 s




(b) With interactions $M = 225$, 61 steps, 924.7 s

Figure 6.11: QP circle stress test, $n_P = 20$, $M = 225$ modes. Notation as in Figure 6.10. Targets are spaced $\approx 47\,\mu$m apart, approximately 7 particle radii at the nominal 3.5–7 $\mu$m size range. All 20 particles converge to their individually assigned positions under both interaction settings.




(a) No interactions

(b) With interactions

Figure 6.12: Per-particle positioning error versus step number for the $n_P = 20$ circle-arrangement run. a: no interactions 50 steps. b: interaction-enabled dynamics 61 steps. The convergence tolerance (10 $\mu$m) is shown as a horizontal dashed line.

# Chapter 7

# Discussion

## 7.1 Algorithmic Modifications for Robustness

Two implementations developed in this work demonstrate measurable improvements over their respective baselines.

**Tile-coded Q-learning improves state resolution without proportional memory cost.** The baseline Q-learning implementation by Perticarari (QG) discretises the state space with a single uniform grid, which creates state aliasing. Two particle positions that fall within the same cell receive identical Q-values even if they differ in distance to the goal or respond differently to an acoustic mode. Tile coding addresses this by overlaying $N_T = 4$ offset grids over the same state space. Each particle configuration is mapped to four separate bin indices simultaneously, one from each tiling, and the Q-value estimate for state $q$ under action $m$ is the sum of four table entries. Nearby states that share three of four tiling indices still disagree on the fourth, producing smoother interpolation across bin boundaries. With four times the resolution of a single-grid approach, tile-coded Q-learning substantially reduces aliasing. The combination of tiling with $\varepsilon$- and $\alpha$-decay schedules and a shaped reward with step penalty and success bonus, enables the tile-coded policy to achieve 100 % success at $n_P = 3$ in the polystyrene environment with and without interactions. (§6.3) (§6.1).

The fundamental limitation of tile coding in this context is that the state encodes only absolute particle positions that cannot represent the relative inter-particle geometry that the interactions depend on. Consequently, a Q-table trained without interactions cannot fully represent the interaction-dependent force landscape.

**Hessian regularisation and a no-sweep initialisation make the VeLO family robust.** Two changes to the model-estimating controller enhanced reliability. First, the QP solved inside VeLO is numerically sensitive, both the gradient and the Hessian are tiny, and the Hessian can be near-singular. Rescaling the target by $\alpha_s = 10^6$ and adding a Tikhonov term $10^{-5} I_M$ to the Hessian (Section 4.2.3) restores a well-conditioned

Hessian for the solver, enhancing both VeLO and VeLO_No-Sweep convergence rate.

Second, the VeLO_No-Sweep variant removes the warmup sweep, which for short episodes consumes a substantial fraction of the step budget and can push particles into unfavorable positions before control begins. Seeding $\hat{F}$ with small random noise and learning each column the first time its mode is selected, VeLO_No-Sweep is never worse than full-VeLO in any condition tested.

Under interactions, it holds 96 % at $n_P = 2$ and 98 % at $n_P = 3$, against VeLO's 90 % and 96 %. The reason is that it learns $\hat{F}$ from rollout observations taken at the positions the particles actually visit, capturing the beneficial component of the AIF at the true operating geometry. The full-sweep variant can probe the interactions at poor positions, especially at a close target IC as shown in Figure 6.4. The no-sweep initialisation is therefore the best of the VeLO family in terms of robustness.

## 7.2 Adaptability Without Structural Modification

**QP adapts automatically and successfully.** Because it treats the dynamics as a black box and queries it fresh at every step, turning on the interaction physics requires no changes to the controller code. One important implementation detail is that QP builds $F$ by testing each mode separately in the simulator with interactions turned on. These tests capture the combined effect of ARF and inter- particle interactions on each particle, making $F$ implicitly aware of the interaction field at the current positions. When the AIF or hydrodynamic interactions push particles in the desired direction, QP takes advantage of this extra force, essentially getting an additional actuation. That said, the AIF, hydrodynamic, and contact interactions depend on where all the particles are. These positions change as the particles move, so $F$ remains a one-step linearisation that may be less accurate when interactions are strong or varying quickly.

**VeLO's EMA model is interaction-aware and tracks the inter-particle interactions well enough to remain reliable.** The EMA update absorbs whatever displacement the interaction-enabled simulator produces, so the model reflects the combined ARF and inter-particle interactions at the positions visited during the observation. The inter-particle interactions do change continuously as particles move, and the EMA's $\gamma = 0.5$ smoothing needs several observations before a column reflects a new configuration, so VeLO is always a little behind the instantaneous field. Interactions neither help nor hurt VeLO dramatically.

**VeLO_No-Sweep adapts at least as well as full-VeLO across the board.** Because it begins updating $\hat{F}$ from rollout observations immediately, VeLO_No-Sweep is never worse than full-VeLO in any condition. With interactions, it reaches 96 % at $n_P = 2$ and 98 % at $n_P = 3$, while full-VeLO had 90 % and 96 %, respectively. VeLO_No-Sweep successfully captures the inter-particle interactions component at the operating geometry, rather than at swept positions where the particles never revisit. This makes it outperform

the full-sweep version.

**Q-learning adapts well after retraining and matches QP.** Unlike the three controllers above, Q-learning cannot sense or learn the interaction force online. Its table is frozen at deployment, and its state encodes only absolute positions. Deployed as trained, it cannot represent the interaction physics. The table is retrained inside the interaction-enabled simulator, after which it reaches 100 % at three interacting particles, matching QP (§6.3). Q-learning, therefore, does handle interactions, but through retraining rather than the live re-measurement that lets QP and the VeLO family adapt with no extra work.

## 7.3 How Interactions Affect Control Precision and Stability

Controllers that sense or learn the live dynamics absorb the inter-particle interactions. This is the main reason they remain robust, and it has been shown that QP even benefits, whereas a controller operating from an offline policy that cannot represent the interaction geometry is the one that suffers.

At $n_P = 1$, the interaction run is a pure dynamics check. With no second particle, the AIF is identically zero and contact never fires, so success rates and step counts are unchanged from the non-interaction case (§6.1). Only Computational time rises, because the interaction framework is evaluated at every step, even when it contributes nothing. The effects appear at $n_P = 2$ and $n_P = 3$, and they divide the controllers into two regimes.

**Interactions help or are learned by controllers that observe the live dynamics.** QP's success rate rises or remains at ceiling with interactions: from 98 % to 100 % at 2P, and at a steady 100 % at 3P. The model-estimating controllers, which learn $\hat{F}$ from the interaction-enabled rollouts, likewise remain robust : VeLO 90–96 %, VeLO_No-Sweep 96–98 %.

Notably, interactions degrade none of these controllers. QP rebuilds its displacement matrix $F$ by probing each mode in the live, interaction-enabled simulator, and every column of $F$ already encodes the combined ARF + inter-particle interactions response at the current geometry. When the inter-particle interaction points toward the goal, QP reads it as additional gain on the available modes and spends it as free actuation. When it points away, the same fresh probe lets the optimiser steer around it. This steering is only available for controllers that re-measure inter-particle interactions each step.

This benefit is not guaranteed in general. Whether the inter-particle interactions add or subtract steering authority depends on the inter-particle geometry and the sign of the combined interaction force.

**A lagging learned model is tolerable as long as it is learned at the operating geometry.** The model-estimating controllers do not re-measure the plant each step.

They carry an estimate $\hat{F}$ refreshed only occasionally, so they are always slightly behind the instantaneous inter-particle interactions. The earlier expectation was that this lag would make interactions harmful, but with the corrected scaling and regularisation it does not. The main reason is that $\hat{F}$ is built from rollouts taken at the positions actually visited. It captures enough of the inter-particle interactions to keep VeLO at 90–96 % and VeLO_No-Sweep at 96–98 %. VeLO_No-Sweep started the learning from the very first operating step, never falling below full-VeLO. The unifying principle is that the closer a controller's model is, in both time and space, to the geometry, the more the inter-particle interactions become an asset.

**A position-only state limits Q-learning but does not break it.** Tile-coded Q-learning uses only absolute particle positions to encode the state. A policy trained with interactions and operating under them will need a higher number of training episodes than one trained without them. When trained on the same number of episodes, the consequence is a limitation rather than a failure. In some deployed episodes, AIF, hydrodynamic interactions, and contact forces can steer the particles more frequently toward unvisited states or states that haven't been visited enough during training, causing an unstable policy that produces random or suboptimal actions. That explains why the trajectory of the Q-controller under interactions looks much more wobbly and messy for some episodes, like the one in 6.9 (b), compared to without interactions 6.9 (a).

Across both regimes, the final positioning error remains near the convergence tolerance ($\approx 1.5 \times 10^{-5}$ m). Interactions change whether a controller converges and how directly, not the accuracy once it does. The cost of QP's fidelity is computational. Evaluating the AIF and hydrodynamic forces for every particle pair at every step scales quadratically with $n_P$, and for QP at 3P in polystyrene, the per-episode time rises from 1.2 s to 16.77 s once interactions are enabled.

## 7.4 Practical Particle-Count Limits

Without interactions, all three controllers handle all three particle counts with high success. Q, QP, and VeLO_No-Sweep reach 100 % at 3P and VeLO 96 %. At 2P QP and the model-estimating variants both at $\geq 92$ %. With interactions, the picture is the same Q and QP at 100 %, while VeLO_No-Sweep drops a sample and VeLO at 96 % at 3P. The results show that reliability is no longer the question between controllers in this range.

What separates them is efficiency. QP converges in roughly an order of magnitude fewer steps than VeLO and VeLO_No-Sweep $\approx$ 72–82 versus $\approx$ 520–710 at the three-particles case. The reason that can cause this difference is that each model-estimating call advances by only a few single-mode steps, and the sweep adds a fixed overhead.

The circle-arrangement stress test (Section 6.4) pushes this picture to $n_P \in \{10, 20\}$. QP converges in all four tested conditions with and without interactions. Targets are

spaced only $\approx 47\,\mu$m apart, and all particles converge to their individually assigned positions with a sub-micrometre error. In this regime, the practical bottleneck shifts from algorithmic feasibility to computational cost. Interaction-aware QP runs in $\approx 146$ s for 10 particles and $\approx 924$ s for 20, a growth consistent with the $\mathcal{O}(n_p^2)$ scaling of the pairwise inter-particle interactions evaluation. Without interactions, the same conditions are completed in under 3 s. Closing this gap, for instance, through spatial cut-offs on the AIF summation or parallelisation of the per-step mode probing, is the key engineering challenge for deploying interaction-aware QP at much larger particle count.

## 7.5 Limitations

Some limitations should be kept in mind. All results come from simulation. The chamber is modelled with rigid walls and a two-dimensional field, so it approximates rather than reproduces a real device. The main benchmark covers only one to three particles, and the 10- and 20-particle runs are single-sample probes of QP rather than full statistical comparisons. Computational time is the largest practical constraint roughly three hours to train a single interaction-aware Q-table, and a per-step cost for QP that grows quickly with particle count.

# Chapter 8

# Conclusion

Five acoustophoretic control algorithms were benchmarked in a single polystyrene environment across three particle counts, both with and without particle-particle interactions. A reproducible 50-sample benchmark with deterministic seeding is used to guarantee reproducibility. Two algorithmic improvements were identified. First, tile-coded Q-learning with $\varepsilon$-decay, $\alpha$-decay, and a shaped reward achieves 100 % success at all IC without interactions. Second, Hessian regularisation ($10^{-5} I_M$) with the no-sweep variant. The modifications introduced make the model-estimating controllers reliable across the tested range. VeLO and VeLO_No-Sweep reach 90–100 % success at every particle count without interactions, and the no-sweep variant matches or exceeds VeLO at every condition.

All controllers were deployed under interactions without modifying any algorithm code. Only the simulator call was changed. QP proved the most efficient and reached 100 % on both two and three interacting particles. Rebuilding its per-step displacement matrix in the live simulator makes the acoustic interaction force observable and therefore exploitable. The model-estimating controllers, VeLO and VeLO_No-Sweep, proved equally reliable in success 90–98 % once their QP solve was regularised. They differ from QP mainly in step efficiency, requiring 9 times more steps to succeed. Interactions affect whether convergence is reached, not the accuracy once it is. $\varepsilon$-Greedy had the worst performance in both settings, reaching only 10 % success rate without interactions and 22 % with interactions for the two-particle case, reflecting its randomness in performance and unstable online learned policy.

Q-learning matches QP's success when its table is retrained directly in the interaction-enabled simulator, at two and three interacting particles, both 100 % , though at a far greater computational cost. Its remaining limitation arises from a position-only state that cannot represent the relative inter-particle geometry the interactions depend on. The long training time $\approx 3$ h per table for three-particles, and the inflexibility of the learned policy make QP the more practical choice for deployment in simulation.

A circle-arrangement stress test at $n_P \in \{10, 20\}$, inspired by Perticarari's [5] large-scale arrangement experiments, using QP as a controller, confirms the scalability. All particles converge to their assigned positions on a 150 $\mu$m circle with a sub-micrometre error in all conditions. The primary bottleneck at large $n_P$ is computational. Interaction-aware QP takes 924 s for 20 particles against under 3 s without interactions, identifying the quadratic scaling of the AIF and hydrodynamic evaluation.

**Future work.** Two further directions are identified for future investigation. First, a systematic study of the state space structure. Mapping how the Gor'kov landscape changes as a function of mode weights would provide a principled basis for designing cost functions and transition models that remain valid under interactions, potentially benefiting all controllers. Second, a vision-based controller combining real-time particle tracking (e.g., via OpenCV image segmentation) with a large language model as a high-level action selector could capture complex multi-particle dependencies that structured state representations miss, eliminating the need for explicit state design and enabling rapid adaptation to new chip geometries.

# Bibliography


[1] T. Baasch, A. Edthofer, L. Péroux, O. Rengbrandt, L. Silversand, A. Lenshof, and T. Laurell. Optimizing the quality of acoustophoretic separation by the in-flow mobility-ratio method. *Physical Review Applied*, 23(1):014054, 2025.

[2] A. Urbansky, F. Olm, S. Scheding, T. Laurell, and A. Lenshof. Label-free separation of leukocyte subpopulations using high throughput multiplex acoustophoresis. *Lab on a Chip*, 19:1406–1416, 2019.

[3] Q. Zhou, V. Sariola, K. Latifi, and V. Liimatainen. Controlling the motion of multiple objects on a Chladni plate. *Nature Communications*, 7:12764, 2016.

[4] Selma Hevelius Bounja. Optimisation of acoustic manipulation for single cell manipulation. Master's thesis, Lund University, Faculty of Engineering LTH, May 2026. URL https://lup.lub.lu.se/student-papers/record/9225942.

[5] G. J. Perticarari. An investigation on the feasibility of multimodal acoustophoretic control. Master's thesis, Lund University, 2025.

[6] R. Öhrnberg. Algorithms for the multimodal acoustophoresis control of particle trajectories. Master's thesis, Lund University, 2025.

[7] K. Yiannacou and V. Sariola. Controlled manipulation and active sorting of particles inside microfluidic chips using bulk acoustic waves and machine learning. *Langmuir*, 37(14):4192–4199, 2021.

[8] A. Urbansky, P. Ohlsson, A. Lenshof, F. Garofalo, S. Scheding, and T. Laurell. Rapid and effective enrichment of mononuclear cells from blood using acoustophoresis. *Scientific Reports*, 7(1):17161, 2017.

[9] G. T. Silva and H. Bruus. Acoustic interaction forces between small particles in an ideal fluid. *Physical Review E*, 90:063007, 2014.

[10] L. Durlofsky, J. F. Brady, and G. Bossis. Dynamic simulation of hydrodynamically interacting particles. *Journal of Fluid Mechanics*, 180:21–49, 1987.

[11] L. J. Durlofsky and J. F. Brady. Dynamic simulation of bounded suspensions of hydrodynamically interacting particles. *Journal of Fluid Mechanics*, 200:39–67, 1989.

[12] A. Pavlič and T. Baasch. Acoustic nanoparticle trapping is driven by synergy between acoustic and hydrodynamic interactions. *Physical Review Letters*, 135: 187201, 2025.

[13] H. Bruus. Acoustofluidics 2: Perturbation theory and ultrasound resonance modes. *Lab on a Chip*, 12(1):20–28, 2012.

[14] L. P. Gor'kov. On the forces acting on a small particle in an acoustical field in an ideal fluid. *Soviet Physics Doklady*, 6:773, 1962.

[15] H. Bruus. Acoustofluidics 7: The acoustic radiation force on small particles. *Lab on a Chip*, 12(6):1014–1021, 2012.

[16] T. Baasch, I. Leibacher, and J. Dual. Multibody dynamics in acoustophoresis. *The Journal of the Acoustical Society of America*, 141(3):1664–1674, 2017.

[17] R. Barnkob. *Physics of microparticle acoustophoresis: Bridging theory and experiment*. PhD thesis, Danmarks Tekniske Universitet, 2012.

[18] A. A. Doinikov. Acoustic radiation interparticle forces in a compressible fluid. *Journal of Fluid Mechanics*, 444:1–21, 2001.

[19] Thierry Baasch, Wei Qiu, and Thomas Laurell. Gap distance between pearl chains in acoustic manipulation. *Phys. Rev. Appl.*, 18:014021, Jul 2022. doi: 10.1103/PhysRevApplied.18.014021. URL https://link.aps.org/doi/10.1103/PhysRevApplied.18.014021.

[20] K. J. Åström and R. M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2008.

[21] K. Ogata. *Modern Control Engineering*. Prentice Hall, 5th edition, 2010.

[22] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[23] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 2nd edition, 2006.

[24] A. N. Tikhonov and V. Y. Arsenin. *Solutions of Ill-Posed Problems*. V. H. Winston & Sons, 1977.

[25] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.

[26] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2nd edition, 2018.

[27] C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3–4):279–292, 1992.

# Appendix A

# Algorithms Pseudocode

This appendix collects the pseudocode for all control algorithms and the benchmark. For clarity, only the non-interaction versions are presented. Interaction-enabled variants differ only in the dynamics call (`positionUpdate_TB` instead of `positionUpdate_2`) and in setting `device.algorithm="FT"` and `device.aif="SilvaBruus2014"` in the setup file.

## A.1 One-Step Lookahead

**Algorithm 1** One-Step Lookahead

**Require:** modes, state $q$, target $q_r$, dynamics ADVANCE

**for** $m = 1$ to $M$ **do**
    $q_m \leftarrow$ ADVANCE(modes, $e_m$)  ▷ $e_m$: unit vector for mode $m$
**end for**
$m^* \leftarrow \arg\min_m \|q_m - q_r\|$
**return** $w = e_{m^*}$

## A.2 QP

**Algorithm 2** QP controller

**Require:** modes, trajectories, targetTraj, device, fluid, particles, timeStepper

$[q, q_r] \leftarrow$ GETACOUSTICSTATE(trajectories, targetTraj, device)
**for** $m = 1$ to $M$ **do**
    traj$' \leftarrow$ POSITIONUPDATE(modes$(m)$, $e_m$)
    $q_m \leftarrow [x_1'/w,\ y_1'/h,\ \ldots]^T$
    $F(:, m) \leftarrow q_m - q$
**end for**
$H \leftarrow 2(F^T F) + 10^{-5} I, \quad f \leftarrow -2F^T(q_r - q)$
$w \leftarrow$ `quadprog`$(H, f,\ \mathbf{1}^T,\ 1,\ [],\ [],\ \mathbf{0})$ ▷ $\sum w \leq 1,\ w \geq 0$
**return** $w$

## A.3 VeLO

The pseudocode below represents the normalised-coordinate variant used in this study. The original VeLO implementation from the supervising group tracked displacements in raw metre coordinates with a different error scaling. The normalised variant with $\alpha_s = 10^6$ is introduced for dimensional consistency with the state vector $q$.

**Algorithm 3** VeLO — normalised-coordinate variant ($\alpha_s = 10^6$)

**persistent** $\hat{F} \in \mathbb{R}^{2\times M\times n_P}$, $F_{\text{unit}}$, Var, warmup_done $\leftarrow$ false
**if** not warmup_done **then** ▷ sweep: fires each mode once in sequence
  **for** $m = 1$ to $M$ **do**
    $q_{\text{prev}} \leftarrow [(x_p/w,\ y_p/h)]_{p=1}^{n_P}$ ▷ record normalised positions
    Apply mode $m$; advance one step
    $\Delta x_p \leftarrow [(x_p^{\text{new}}/w - x_p^{\text{prev}}/w),\ (y_p^{\text{new}}/h - y_p^{\text{prev}}/h)]^T$
    $\hat{F}(:,m,p) \leftarrow \gamma \hat{F}(:,m,p) + (1-\gamma)\,\Delta x_p$ ▷ EMA, $\gamma = 0.5$
    Update $F_{\text{unit}}(:,m,p)$ and $\text{Var}(m,p)$
    **if** all particles within tolerance **then exit early**
    **end if**
  **end for**
  warmup_done $\leftarrow$ true; **return**
**end if**
$[q, q_r] \leftarrow$ GETACOUSTICSTATE() ▷ normalised coordinates
$H \leftarrow \sum_p 2\hat{F}_p^T \hat{F}_p$; $\quad H \leftarrow H + 10^{-5} I_M$ ▷ Tikhonov regularisation
$f \leftarrow -\sum_p 2\hat{F}_p^T \big[\alpha_s (q_{r,p} - q_p)\big]$, $\quad \alpha_s = 10^6$
$z \leftarrow \texttt{quadprog}(H, f, [], [], [], [], \mathbf{0})$
$[\sim, \text{idx}] \leftarrow \text{SORT}(z, \text{descend})$; $\quad [m_1, \ldots, m_5] \leftarrow \text{idx}(1{:}5)$ ▷ top-5 modes by weight
**for** $i = 1$ to 5 **do**
  Apply mode $m_i$; EMA-update $\hat{F}(:,m_i,:)$ using normalised $\Delta x_p$
  **if** all particles within tolerance **then break**
  **end if**
**end for**
**return** updated trajectory

## A.4 VeLO Without Sweep

Identical to VeLO above with `SWEEP = false`: the sweep phase is skipped and $\hat{F}$ is initialised with $10^{-6} \cdot \mathcal{N}(0,1)$ instead of zeros. The normalised-coordinate tracking, $\alpha_s = 10^6$ scaling, and $10^{-5} I_M$ Hessian regularisation are unchanged.

## A.5 $\varepsilon$-Greedy Bandit

**Algorithm 4** $\varepsilon$-Greedy Bandit

**persistent** $Q \in \mathbb{R}^M$, $N \in \mathbb{R}^M$, $\varepsilon = 0.3$, $a_{\text{prev}}$
$\varepsilon \leftarrow \max(0.01, \varepsilon \times 0.999)$
**if** previous step exists **then**
    Compute reward $r = \sum_p \Delta r_p \cdot \hat{g}_p$
    $N_{a_{\text{prev}}}$ += 1;  $Q_{a_{\text{prev}}}$ += $(r - Q_{a_{\text{prev}}})/N_{a_{\text{prev}}}$
**end if**
**if** rand() $< \varepsilon$ **then**
    $a \leftarrow$ randint($M$)  ▷ explore
**else**
    $a \leftarrow \arg\max Q$  ▷ exploit, break ties randomly
**end if**
$a_{\text{prev}} \leftarrow a$
**return** $w = e_a$

## A.6 Tile-Coded Q-Learning

**Algorithm 5** Tile-coded Q-learning, training One episode (IC)

**Require:** episodes $N_{\text{epi}}$, max steps $T_{\text{epi}}$, discount $\gamma$, tolerance $\theta$, resolution $R$, tilings $N_T$

Build $N_T$ offset grids: tiling $t$ shifted by $\frac{t-1}{N_T} \cdot \text{binWidth}$ per dimension
$Q(s,a,t) \leftarrow 0, \quad V(s,a,t) \leftarrow 0 \quad \forall s,a,t$ ▷ Q-values and visit counts
**for** $k = 1$ to $N_{\text{epi}}$ **do**
  $\varepsilon_k \leftarrow \max\big(\varepsilon_{\text{floor}},\ \varepsilon_0/(1+k\,\varepsilon_d)\big)$
  $q \leftarrow q_s$; waypoint indices $\leftarrow 2$
  **for** step $= 1$ to $T_{\text{epi}}$ **do**
    $\{s^{(1)},\ldots,s^{(N_T)}\} \leftarrow \textsc{TileIndices}(q)$ ▷ one index per tiling
    $\hat{Q}(a) \leftarrow \sum_{t=1}^{N_T} Q(s^{(t)},a,t)\ \ \forall a$ ▷ sum across tilings
    **if** rand() $< \varepsilon_k$ **then** $a^* \leftarrow \text{randint}(1,M)$
    **else** $a^* \leftarrow \arg\max_a \hat{Q}(a)$ with random tie-breaking
    **end if**
    $q' \leftarrow \textsc{System}(q, e_{a^*})$
    $d \leftarrow \|q - q_r\| - \|q' - q_r\|$ ▷ distance reduction
    $r \leftarrow d \cdot R_{\text{scale}} - c_{\text{step}}$; **done** $\leftarrow$ false ▷ shaped reward, step penalty
    **if** $\|q' - q_r\| \le \theta$ **then** $r \leftarrow r + R_{\text{bonus}}$; **done** $\leftarrow$ true
    **end if**
    $\{s'^{(1)},\ldots,s'^{(N_T)}\} \leftarrow \textsc{TileIndices}(q')$
    **if done then** $\hat{Q}'_{\max} \leftarrow 0$ ▷ no bootstrap at terminal
    **else** $\hat{Q}'_{\max} \leftarrow \max_a \sum_t Q(s'^{(t)},a,t)$
    **end if**
    $y \leftarrow r + \gamma \hat{Q}'_{\max}$ ▷ TD target
    **for** $t = 1$ to $N_T$ **do**
      $V(s^{(t)},a^*,t)\ +\!= 1$
      $\alpha_t \leftarrow \big(1 + V(s^{(t)},a^*,t)\big)^{-0.7}/N_T$
      $Q(s^{(t)},a^*,t)\ +\!= \alpha_t\,\big(y - \hat{Q}(a^*)\big)$
    **end for**
    $q \leftarrow q'$; **if done then break**
  **end for**
**end for**
Save $Q$, grid configuration, and metadata to file

**Algorithm 6** Q-learning greedy deployment

Load Q-table from `QTABLE_FILE`: $Q$, divisions, $N_T$, $n_{\text{dims}}$
$q \leftarrow$ GETSTATEVEC(trajectories, $n_P$)
Compute tile indices $s_1, \ldots, s_{N_T} \leftarrow$ GETTILEINDICES($q$, divisions, $N_T$)
$\hat{Q}(m) \leftarrow \sum_{t=1}^{N_T} Q(s_t, m, t)$ for each $m$
$a^* \leftarrow \arg\max_m \hat{Q}(m)$
**return** $w = e_{a^*}$

## 1.7 BenchmarkB50

**Algorithm 7** BenchmarkB50

**for** $n_P \in \{1,2,3\}$ **do**
  **for** $r = 1$ to 50 **do**
    $[\text{initX}, \text{initY}, \text{goalXY}] \leftarrow$ GETSAMPLEIC2($n_P$, $r$)
    `rng`$(n_P \times 100 + r)$; capture RNG state $s_r$
    **for** each controller $c$ **do**
      Restore RNG state $s_r$; clear persistent controller variables
      Initialise environment: SETUPB40($n_P$, $r$, `'default'`)
      Reset trajectories to initX, initY; reset currIdx $= 2$
      $T \leftarrow 1$; continueEpisode $\leftarrow$ true
      **while** continueEpisode **and** $T < 1000$ **do**
        $w \leftarrow c$(modes, targetTraj, particles, traj, device, fluid, ts)
        traj $\leftarrow$ POSITIONUPDATE(modes, $w$)
        Update currIdx, continueEpisode; $T \mathrel{+}= 1$
      **end while**
      Record $T$, $e_{\text{final}}$, success, time
    **end for**
  **end for**
  GENERATEBENCHMARKPLOTST; DISPLAYSUMMARYTABLE
**end for**