Optimal control of active matter via automatic differentiation

Zar Rehman[1,*] and Luke K. Davis[2,3,†]

[1] *Department of Physics and Astronomy, University College London, 25 Gordon Street, London, England*
[2] *Isaac Newton Institute, University of Cambridge, Cambridge, CB3 0EH,England*
[3] *Department of Mathematics, University College London, 25 Gordon Street, London, England*

Active matter consists of constituents that produce entropy, usually resulting in sustained individual dynamics, which often leads to striking collective phenomenology. Indeed, such phenomenology is often richer than when the constituents do not produce entropy, such as the arising of motility induced phase separation (MIPS), and one can envisage controlling these active systems to perform functions that are not accessible to their passive counterparts. The far from equilibrium nature of active matter and their, often, non-trivial interactions among constituents and with the environment make the optimal and precise control of active matter difficult. Here, we develop an in-silico computational framework to explore the optimal control of modelled active matter systems, through the combination of molecular dynamics simulations and state-of-the-art automatic differentiation techniques (such as Enzyme).

I. PHYSICAL PROBLEM

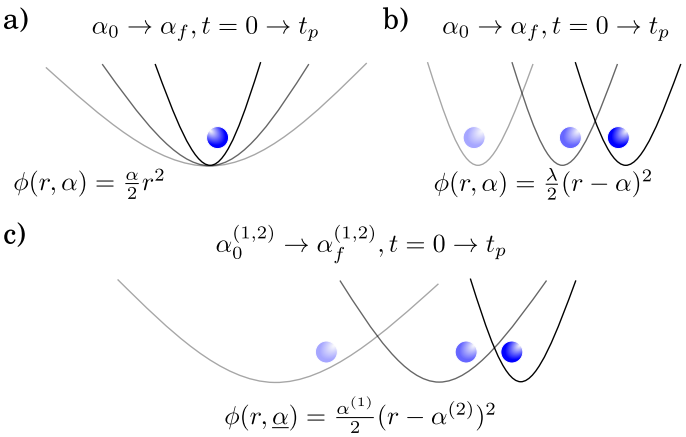


FIG. 1: Visual depiction of the physical control problem. **a)** Varying the trap stiffness. **b)** Varying the trap position. **c)** Varying both the trap stiffness and position.

We begin by considering a single particle within a potential trap defined by the function $\phi(r,\underline{\alpha})$, where r is the position and $\underline{\alpha}$ is a vector of parameters capturing the shape α_1 and position α_2 of the well as illustrated in figure 1. Such a potential is defined as

$$\phi(r,\underline{\alpha}) := \frac{\alpha_1}{2}(r-\alpha_2)^2, \tag{1}$$

and the resulting general dynamics of this system is governed by the following equation of motion:

$$\underbrace{m\ddot{r}}_{\text{Inertia}} = \underbrace{-\gamma r}_{\text{Drag}} - \underbrace{\partial_r \phi(r,\underline{\alpha})}_{\text{Interaction}} + \underbrace{\gamma\sqrt{2D}\eta}_{\text{Thermal Noise}} + \underbrace{\gamma v_a}_{\text{Propulsion}} \tag{2}$$

In this equation γ is determined by Stokes's Law as the friction exerted on the particle by the surrounding solvent and v_a is the self-propulsion. For thermal noise, D is taken to be the diffusion coefficient and η is a random variable from a Gaussian distribution with zero mean and unit variance (i.e $\langle\eta\rangle = 0$ and $\langle\eta(0)\eta(t-s)\rangle = \delta(t-s)$ for $t > s$). Imposing the over-damped regime leads to the following equation of motion

$$\dot{r} = -\mu\partial_r\phi(r,\underline{\alpha}) + \sqrt{2D}\eta + v_a, \tag{3}$$

where $\mu = \frac{1}{\gamma}$.

Of interest is the total dissipation to the heat bath $\langle\Delta Q\rangle$ for a finite time period t_p defined as follows [1]:

$$\langle\Delta Q\rangle = \int_0^{t_p} dt \left\langle \frac{\dot{r}}{\mu} \circ (\dot{r} - \sqrt{2D}\eta) \right\rangle, \tag{4}$$

where $\langle\ldots\rangle$ denotes an average over all stochastic trajectories and $\circ$ is a Stratonovich product. Substituting the dynamics (3) into the above, and using the fact that $\dot{r}\partial_r\phi = \dot{\phi} + \dot{\underline{\alpha}}^T\partial_{\underline{\alpha}}\phi$, allows us to retrieve the first law of thermodynamics:

$$\begin{aligned}\langle\Delta Q\rangle &= \langle\Delta\phi(t_p, t=0)\rangle + \frac{\langle v_a(t_p)^2\rangle}{\mu}t_p + \\ &\int_0^{t_p} dt\langle\dot{\underline{\alpha}}^T\partial_{\underline{\alpha}}\phi(r,\underline{\alpha}) - \partial_r\phi(r,\underline{\alpha})v_a\rangle, \\ &\equiv \mathcal{J} = B\left[\{r(t_p,0), v_a(t_p)\}, \underline{\alpha}(t_p,0)\right] + \\ &\int_0^{t_p} dt\mathcal{L}\left[\{r(t), v_a(t)\}, \dot{\underline{\alpha}}(t)\right],\end{aligned} \tag{5}$$

where in the last line we recognise the heat has the general form of an objective function usually considered in optimal control problems: $\mathcal{J} = B[t_p, 0] + \int_0^{t_p} dt\mathcal{L}$, consisting of a boundary term B (or terminal cost) and an action-like term with a Lagrangian $\mathcal{L}$ (or running-cost). The optimal control here aims to derive protocols for the $\underline{\alpha}$ extremizing (5) denoted as $\underline{\alpha}^*$. Using the potential $\phi(r,\underline{\alpha})$ defined defined in (1) we compute the Lagrangian

* zcapehm@ucl.ac.uk
† ld731@cam.ac.uk

as follows:

$$\begin{aligned}\mathcal{L}(\underline{\alpha},\dot{\underline{\alpha}},r) &= \langle \dot{\underline{\alpha}}^T \partial_{\underline{\alpha}}\phi - \partial_r\phi\cdot v_a\rangle \\ &= \left\langle [\dot{\alpha_1},\dot{\alpha_2}] \begin{bmatrix}\partial_{\alpha_1}\phi \\ \partial_{\alpha_2}\phi\end{bmatrix} - \partial_r\phi v_a \right\rangle, \\ &= \left\langle \frac{\dot{\alpha_1}}{2}(r-\alpha_2)^2 - \dot{\alpha_2}\alpha_1(r-\alpha_2) \right. \\ &\qquad \left. - \alpha_1(r-\alpha_2)v_a \right\rangle\end{aligned} \tag{6}$$

We seek to arrive at the optimal protocol $\underline{\alpha}^*$ via functional gradient descent, by minimising the cost function $\mathcal{J}$ with the following updating mechanism:

$$\underline{\alpha}_{n+1} = \underline{\alpha}_n - l\nabla_{\underline{\alpha}}\mathcal{J}, \tag{7}$$

where l denotes the learning rate and the gradient, $\nabla_{\underline{\alpha}} \equiv \partial_{\underline{\alpha}}$ is computed at each iteration.

Recently, extremising the action $\mathcal{J}$ and therefore attaining $\underline{\alpha}^*$, has been accomplished in the context of active matter using techniques from response theory [2] under the restriction of smooth control. There is a natural desire to perform optimal control without such restrictions on the protocols, though it is not straightforward to derive a theoretical framework (equations) to accomplish this. However, recent advances in automatic differentiation and optimization present a promising avenue to accomplish this, indeed this has already begun in the context of non-equilibrium statistical mechanical systems [3], but not active matter.

II. AUTOMATIC DIFFERENTIATION

For the purpose of optimisation we utilise automatic differentiation (AD), a robust machine learning technique, to compute derivatives for a variety of functions. AD is implemented via two different strategies: forward mode and reverse mode. Forward mode computes the numerical value of the partial derivative and simultaneously the value of the function, accumulating via the chain rule from the inner most variable to the outer function. Comparatively, Backward mode computes the derivative of the outer most function and accumulates the chain rules inwards.

$$\text{Forward mode}: \frac{\partial w_i}{\partial x} = \frac{\partial w_i}{\partial w_{i-1}}\frac{\partial w_{i-1}}{\partial x} \tag{8}$$

In this recursive definition, w_i is defined as the ith intermediate functions composing the objective function. This calculation is repeated until $w_i = \mathcal{J}(x)$, the objective function. Similarly, Backward mode utilises recursion but starting form the objective function and completing with $w_0 = x$, the dependant variable.

$$\text{Backward mode}: \frac{\partial \mathcal{J}}{\partial w_i} = \frac{\partial \mathcal{J}}{\partial w_{i+1}}\frac{\partial w_{i+1}}{\partial w_i} \tag{9}$$

AD utilises dual numbers defined by $d = a + \epsilon b$ where a and b are real numbers and ϵ is defined such that $\epsilon^2 = 0$. To use dual numbers, the standard operators defined for real number arithmetic need to be overloaded under new definitions inclusive of duals. The use of of daul numbers as inputs for a smooth function $f(a+\epsilon b)$ is convenient in computing the derivative f' as any term of order greater than or equal to $\mathcal{O}(f'')$ in the Taylor expansion will evaluate to 0 as follows:

$$\begin{aligned}f(a+\epsilon b) &= \sum_{k=0}^{\infty}\frac{f^{(k)}(a)}{k!}(a+\epsilon b - a)^k \\ &= \sum_{k=0}^{\infty}\frac{f^{(k)}(a)\epsilon^k b^k}{k!} \\ &= f(a) + bf'(a)\epsilon + \underbrace{\epsilon^2(\sum_{k=0}^{\infty}\frac{f^{(k)}(a)\epsilon^{k-2}b^k}{k!})}_{=0} \\ &= f(a) + bf'(a)\epsilon\end{aligned} \tag{10}$$

III. ENZYME

To implement AD we use Enzyme, an llvm (low level virtual machine) plugin, to generate an intermediate representation of the program. The advantage of this method is the versatility across multiple compilable programming languages (such as C++, Julia, Rust and Swift). For the purpose of optimised computing, llvm allows one to develop front ends and bookends of a compiler, to transform the source code into machine code specifically for a target machine architecture. Enzyme as a plugin for llvm is designed to optimise the computation of derivatives via AD. Enzyme works by parsing the llvm intermediate representation (IR) and applying novel transformation techniques to compute derivatives to any type of function (including recursively defined such as those used in propagation).

By leveraging llvm, Enzyme provides a fast and accurate approach to computing numerical derivatives via AD. For the purpose of active mater simulation, we utilise this within Julia to compute partial derivatives of the objective functional $\mathcal{J}$ (defined in equation 5).

A. Example test functions

Here we test few standard functions and compute the derivative via autodiff(mode, function, point) method in Enzyme.jl

1. $f(x) = C$, where $C \in \Re$

```
f(x)=5.0
println(autodiff(Reverse, f, Active,
        Active(1.0)))
```

```
Output: ((0.0,),)
```

2. $f(x) = Bx + C$, where $B, C \in \Re$

```
f(x)= 4.0*x + 5.0
println(autodiff(Reverse, f, Active,
        Active(1.0)))
Output: ((4.0,),)
```

3. $f(x) = \sum_{n=0}^{d} a_n x^n$ such that $\forall n, a_n \in \Re$

```
f(x)= 32.0*x^12.0 + 15.0*x^5 + x/13 + 6
println(autodiff(Reverse, f, Active,
        Active(5.0)))
Output: ((1.8750046875076923e10,),)
```

4. $f(x) = Bsin(x) + Ccos(x)$

```
f(x) = sin(x)+5.0*cos(x)
println(autodiff(Reverse, f, Active,
        Active(3.14)))
Output: ((-1.0079619963099737,),)
```

5. $f(x) = Be^x + Clog(x)$

```
f(x)= 10.0*exp(x) + 1/3*exp(-x^2) + log(x)
println(autodiff(Reverse, f, Active,
        Active(-0.5)))
Output: ((4.324906858150135,),)
```

6. $f(\underline{x}) = \underline{x}^T \underline{x}$ where $\underline{x} \in \Re^n$

```
f(x)= x[1]^2 + x[2]^2 + x[3]^2 + x[4]^2
println(gradient(Reverse, f,
        [1.0,2.0,3.0,4.0]))
Output: [2.0, 4.0, 6.0, 8.0]
```

IV. SIMULATION

We have built a self contained C++ script that simulated the motion of the particle and trap systems over time as outlined with (1)-(3). Since the dynamics in the overdamped regime is linear, we have the following update rule for the position:

$$r(t+\Delta t) = r(t) + \Delta t(-\mu \partial_r \phi(r, \underline{\alpha}) + v_a)(t) + \sqrt{2D\Delta t}\eta(t), \tag{11}$$

We define the dynamics of self-propulsion (the activity of the particle) as follows:

$$\begin{aligned} \tau \dot{v}_a(t) &= -v_a(t) + \sqrt{2D_v}\zeta(t), \\ \langle \zeta \rangle = 0, \quad \langle \zeta(t)\zeta(s) \rangle &= \delta(t-s) \end{aligned} \tag{12}$$

where τ is the persistence time of the direction of the propulsion and $\delta(\ldots)$ is the Dirac delta function. The propulsion dynamics is also linear, thus the following two difference equations are all the update rules needed to advance the position of the particle from some time t to a future time $t + \Delta t$:

$$r(t+\Delta t) = r(t) + \Delta t(-\mu \partial_r \phi(r, \underline{\alpha}) + v_a)(t) + \sqrt{2D\Delta t}\eta(t), \tag{13}$$

$$v_a(t+\Delta t) = v_a(t)\left(1 - \frac{\Delta t}{\tau}\right) + \sqrt{2D_v \Delta t}\zeta(t) \tag{14}$$

Where we stress that ζ, η are Gaussian white noises centered on zero with a standard deviation of 1. Thus, the simulation consists of specifying the following parameters/functions $\tau = 1, D = 1, \mu = 1, D_v = 1, \Delta t = 0.005, r(t=0) = 0, v_a(t=0) = 0, \alpha_1(t), \alpha_2(t)$. Here $\alpha_1(t), \alpha_2(t)$ are defined indolently and passed into the simulation as time arrays for each time step Δt.

The results of this simulation under differing values of trap intensity and trap position parameters, α_1 and α_2 respectively, are shown in FIG 2. and FIG 3. We observe a reduction in the volatility of particle position over time under the increased α_1 regimes in 2b and 2c, demonstrating the effect of increased trap intensity on the movement of the particle. Similarly, we observe the effect of changing the position of the trap relative to the particle under the varying α_2 regimes, where the time average of the particle's position increases over positive time versus the base case of the position being centred around 0 as shown in FIG 3a. We present a smooth linear increase in trap position over time (3b) versus changing trap position in discrete steps over time (3c). In both cases, the particle adapts to the change and on average follows the trap with some lag in time.

Next, we run 10000 instances of simulations for each of the following three cases and compute the relevant test statistics: Case 1: $\alpha_1 = 1$ and $\alpha_2 = 0$, Case 2: $\alpha_1 = 0$ and $\alpha_2 = 1$, Case 3: $\alpha_1 = 1$ and $\alpha_2 = 1$. For each case, we consider a simulation set with the activity turned off which is equivalent to a passive particle in a trap and a second simulation set with activity included. The test statistics computed are position squared (r^2) and velocity time position (rv). In each case and for both passive and active regimes, we compare these results with the theoretical expectations as derived by explicitly solving the equations of motion.

$$\text{Passive EOM:} \quad r'(t) = -\mu \partial_r \phi + \sqrt{2D}\eta(t) \tag{15}$$

$$\text{Active EOM:} \quad r'(t) = -\mu \partial_r \phi + \sqrt{2D}\eta(t) + v(t) \tag{16}$$

Passive Regime:
Case 1: $\alpha_1 > 0$, $\alpha_2 = 0$

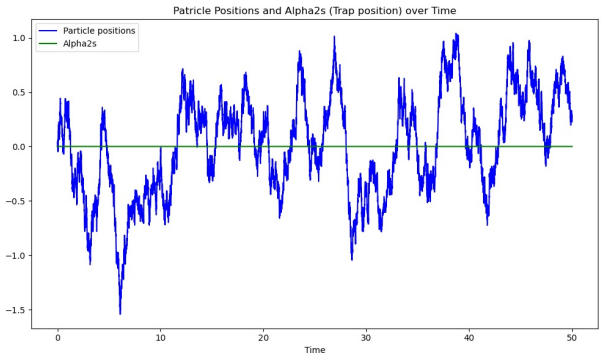
$$\langle r^2(t) \rangle = \frac{D}{\alpha_1 \mu}, \quad \langle (rv)(t) \rangle = 0 \tag{17}$$

Case 2: $\alpha_1 = 0$, $\alpha_2 \neq 0$

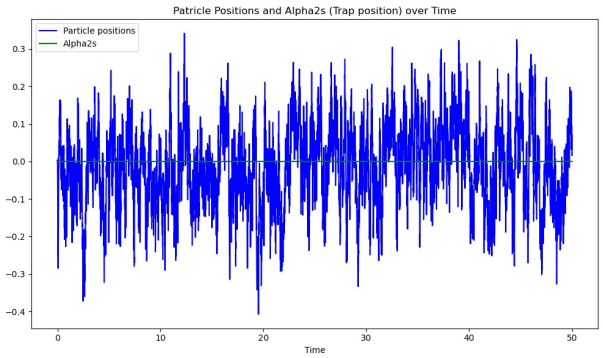
$$\langle r^2(t) \rangle = 2D(1+t), \quad \langle (rv)(t) \rangle = 0 \tag{18}$$

Case 3: $\alpha_1 > 0$,$\alpha_2 \neq 0$

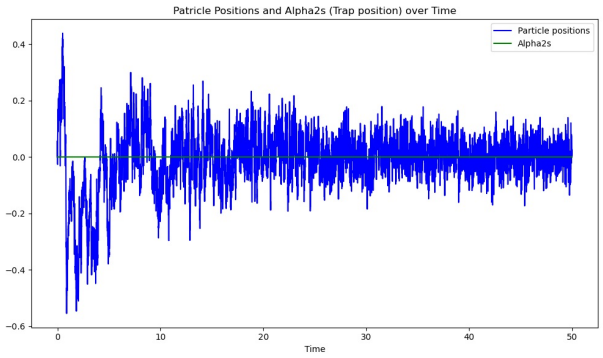
$$\langle r^2(t) \rangle = \alpha_2^2 + \frac{D}{\alpha_1 \mu}, \quad \langle (rv)(t) \rangle = 0 \tag{19}$$



(a) $\alpha_1 = 1$

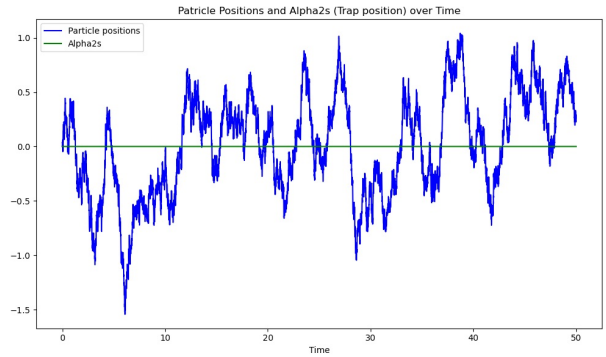


(b) $\alpha_1 = 10$

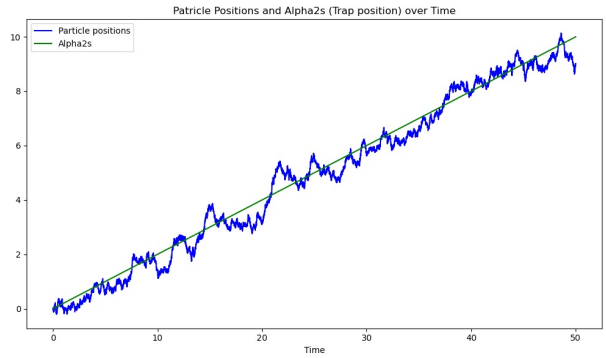


(c) $\alpha_1 = 1 + 0.00t$

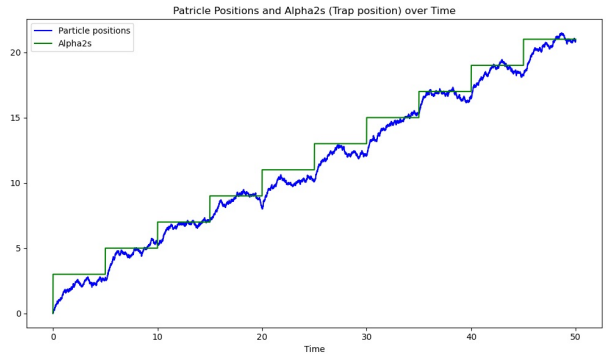
FIG. 2: Simulated particle positions over time under different α_1 regimes, holding $\alpha_2 = 0$ constant.



(a) Constant



(b) Linear



(c) Step function

FIG. 3: Simulated particle positions over time under different α_2 regimes, holding $\alpha_1 = 1$ constant.

Active Regime:

Case 1: $\alpha_1 > 0$, $\alpha_2 = 0$

$$\langle r^2(t)\rangle = \frac{D}{\alpha_1\mu} + \frac{D_r}{\alpha_1\mu(1+\alpha_1\mu\tau)}, \quad \langle (rv)(t)\rangle = \frac{D_r}{1+\alpha_1\mu\tau} \tag{20}$$

Case 2: $\alpha_1 = 0$, $\alpha_2 \neq 0$

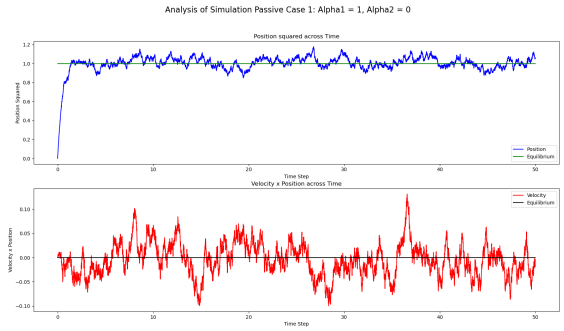
$$\langle r^2(t)\rangle = \int_\infty^t dt' \int_\infty^t dt'' [\frac{D_r}{\tau} e^{\frac{-|t'-t''|}{\tau}}] \quad + 2D(1+t) \tag{21}$$

$$\langle (rv)(t)\rangle = D_r \tag{22}$$

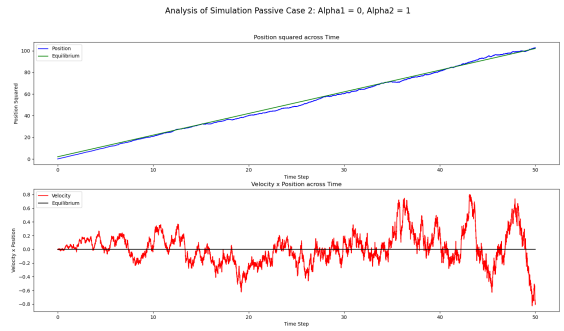
Case 3: $\alpha_1 > 0$,$\alpha_2 \neq 0$

$$\begin{aligned} \langle r^2(t)\rangle &= \alpha_2^2 + \frac{D_r}{\alpha_1\mu(1+\alpha_1\mu\tau)} + \frac{D}{\alpha_1\mu}, \\ \langle (rv)(t)\rangle &= \frac{D_1}{1+\alpha_1\mu\tau} \end{aligned} \tag{23}$$

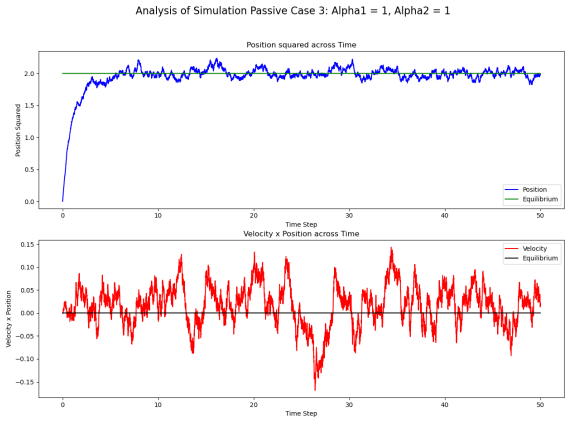
The results for repeated simulations of the Passive and Active models under different cases are summarised in FIG 4 and FIG 5 respectively. In all cases, the particle



(a) $PassiveCase1 : \alpha_1 = 1$

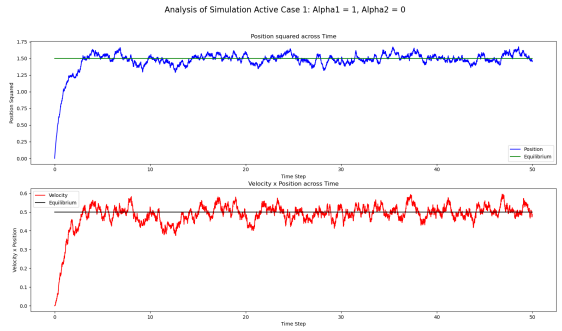


(b) $PassiveCase2 : \alpha_1 = 0$

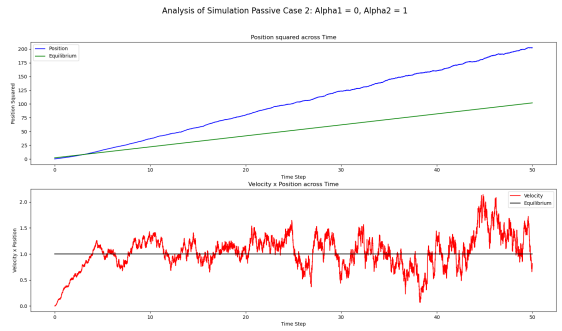


(c) $PassiveCase3 : \alpha_1 = 1$

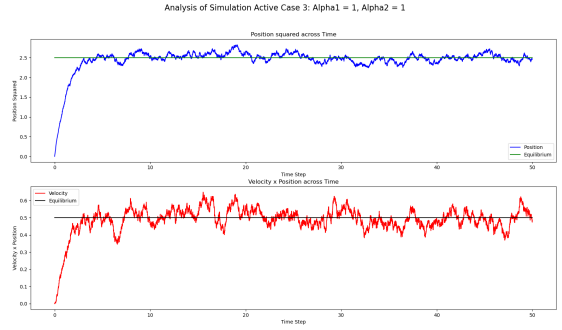
FIG. 4: Simulated passive particle statistics over time and the equilibrium value under each regime.



(a) $ActiveCase1 : \alpha_1 = 1$



(b) $ActiveCase2 : \alpha_1 = 0$



(c) $ActiveCase3 : \alpha_1 = 1$

FIG. 5: Simulated active particle statistics over time and the equilibrium value under each regime.

statistics reach the expected equilibrium value after some initial lag.

V. CONCLUSION

We have demonstrated control over the active particle using a trap under several regimes, varying both the strength and position to direct the motion of the particle over time. Additionally, we compared the results over many simulations with the theoretical expected equilibrium value of test statistics, showcasing the convergence to these states over time under the various cases for passive and active systems. Altogether, we have built a computational and theoretical framework to apply and verify control of the active particle under multiple regimes in one dimension. We also presented the use of Enzyme.jl to implement automatic differentiation of test functions. As next steps, this can be integrated into the simulation code to differentiate the $\langle \Delta Q \rangle$ in equation 6 for the purpose of optimal control and using stochastic gradient descent, can be used to design optimal protocols for $\underline{\alpha}$.

VI. ACKNOWLEDGEMENTS

I would like to express my sincerest gratitude to Dr Luke K. Davis for his continuous mentorship, collaboration and encouragement throughout this project. I am also thankful to the Laidlaw Foundation for their generous funding and training workshops that allowed me to undertake this project.

[1] Ken Sekimoto, "Langevin equation and thermodynamics," Progress of Theoretical Physics Supplement **130**, 17–27 (1998).

[2] Luke K. Davis, Karel Proesmans, and Étienne Fodor, "Active matter under control: Insights from response theory," Physical Review X **14** (2024), 10.1103/physrevx.14.011012.

[3] Megan C. Engel, Jamie A. Smith, and Michael P. Brenner, "Optimal control of nonequilibrium systems through automatic differentiation," Physical Review X **13** (2023), 10.1103/physrevx.13.041032.