Forecasting and predicting stochastic agent-based model data with biologically-informed neural networks

John T. Nardini[1]

[1] *Department of Mathematics and Statistics, The College of New Jersey, Ewing, NJ, 08628, USA.* ,

nardinij@tcnj.edu

August 14, 2024

Abstract

Collective migration is an important component of many biological processes, including wound healing, tumorigenesis, and embryo development. Spatial agent-based models (ABMs) are often used to model collective migration, but it is challenging to thoroughly predict these models' behavior throughout parameter space due to their random and computationally intensive nature. Modelers often coarse-grain ABM rules into mean-field differential equation (DE) models. While these DE models are fast to simulate, they suffer from poor (or even ill-posed) ABM predictions in some regions of parameter space. In this work, we describe how biologically-informed neural networks (BINNs) can be trained to learn interpretable BINN-guided DE models capable of accurately predicting ABM behavior. In particular, we show that BINN-guided partial DE (PDE) simulations can 1.) forecast future spatial ABM data not seen during model training, and 2.) predict ABM data at previously-unexplored parameter values. This latter task is achieved by combining BINN-guided PDE simulations with multivariate interpolation. We demonstrate our approach using three case study ABMs of collective migration that imitate cell biology experiments and find that BINN-guided PDEs accurately forecast and predict ABM data with a one-compartment PDE when the mean-field PDE is ill-posed or requires two compartments. This work suggests that BINN-guided PDEs allow modelers to efficiently explore parameter space, which may

enable data-driven tasks for ABMs, such as estimating parameters from experimental data. All code and data from our study is available at https://github.com/johnnardini/Forecasting_predicting_ABMs.

1 Introduction

Many population-level patterns in biology arise from the actions of individuals. For example, predator-prey interactions determine ecological population dynamics; individuals' adherence to public health policies limit disease spread; and cellular interactions drive wound healing and tumor invasion. Mathematical modeling is a useful tool to understand and predict how such individual actions scale into collective behavior [1, 2, 3, 4, 5, 6, 7]. In particular, stochastic agent-based models (ABMs) are a widely-used modeling framework where autonomous agents mimic the individuals of a population [8, 9, 10, 11]. ABMs are advantageous because they capture the discrete and stochastic nature of many biological processes [12]. However, ABMs are computationally intensive, and their simulations can become time-consuming to perform when the population is comprised of many individuals [13, 14]. This computational restraint prevents modelers from efficiently exploring how model parameters alter model outputs. As such, there is a need for the development of methods to efficiently and accurately predict ABM behavior [14, 15, 16].

Modelers often perform ABM prediction by coarse-graining ABM rules into continuous differential equation (DE) models [8, 13]. Ordinary DEs (ODEs) describe how a quantity (e.g., agent density) changes over time, and Partial DEs (PDEs) describe how spatially-varying ABMs change with time [13]. Such DE models are useful surrogates for ABMs because they are cheap and efficient to simulate. Mean-field DE models, which assume agents respond to the average behavior of their neighbors, have been shown to accurately predict ABM behavior at some parameter values. Unfortunately, these models can poorly predict ABM outputs when the mean-field assumption is violated [8, 17]. For example, Baker and Simpson 2010 [8] demonstrated that the mean-field DE model for a population growth ABM only accurately predict ABM data when agents proliferate slowly. A further complication of mean-field DEs is that they may be ill-posed at certain parameter values. Anguige and Schmeiser 2009 [1] used a stochastic space-jump model to study how cell adhesion impacts collective migration and found that the resulting mean-field PDE model is ill-posed (and thus cannot predict ABM behavior) for large adhesion values.

Despite the inability of mean-field DE models to predict ABM behavior at all parameter values, ABM simulations do obey conservation laws (e.g., conservation of mass for spatial ABMs) [18]. Alternative DE models may thus be capable of accurately describing ABM behavior. Equation learning (EQL) is a new area of research on the development and application of algorithms to discover the dynamical systems model that best describes a dataset [19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29]. Brunton et al. 2016 [19] introduced a sparse regression-based EQL approach to learn DE models from data with a user-specified library of candidate terms. This method has proven very successful in recovering informative models from simulated and experimental data [30]. There is a growing understanding that EQL methods can aid the prediction of ABM data [14, 31, 32, 33]. For example, we recently demonstrated that the least squares EQL approach learns ODE equations that accurately describe simulated ABM data, even when the collected data is incomplete or sparsely sampled [14]. Supekar et al. 2023 [33] coupled this method with spectral basis representation data to discover PDE models that capture the emergent behavior found in active matter ABMs. Another popular EQL approach includes physics-informed neural networks (PINNs), where modelers embed physical knowledge (in the form of a known PDE framework) into the training procedure for artificial neural networks (ANNs) [34, 35, 36, 37, 38]. Trained PINN models can predict complex, sparse, and noisy data while also obeying known physical principles. Lagergren et al. 2020 [28] extended the PINNs framework by replacing physics-based mechanistic terms with function-approximating multi-layer perceptions (MLPs) to develop the biologically-informed neural network (BINN) methodology. As a result, BINN models can learn PDE models from data with terms that depend on space, time, or agent density. Training the BINN to simulated ABM data ensures that a realization of this PDE that best matches the data is learned. Standard methods of DE analysis, including bifurcation analysis and pattern formation, can be used to understand the ABM's behavior. BINNs thus present a promising and interpretable tool for ABM forecasting and prediction. However, determining how BINNs can be used to learn predictive DE models for ABMs remains an open area of research.

In this work, we demonstrate how to combine BINNs and PDE model simulations to forecast and predict ABM behavior. Our approach leverages BINNs' vast data and modeling approximation capability with the computational efficiency of PDE models to develop a potent ABM surrogate modeling tool. In particular, we

demonstrate how to use trained BINN models to 1.) forecast future ABM data at a fixed parameter value, and 2.) predict ABM data at previously-unexplored parameter values. This latter task is achieved using multivariate interpolation, which provides a straightforward approach for inferring PDE modeling terms. We demonstrate that visually inspecting the BINN modeling terms over a range of ABM parameter values allows us to interpret how ABM parameters impact model behavior.

We apply the BINNs methodology to three case study ABMs in this work. Each case study models collective migration in cell biological experiments, such as barrier and scratch assays [5, 13, 28, 39, 40]. In a barrier assay, a two-dimensional layer of cells is cultured inside a physical boundary. Microscopy is used to image how the cell population migrates outwards once the barrier has been removed [40, 41]. Cells are closely packed in these experiments and thus interact with their neighbors. Our case study ABMs simulate how two stimuli, namely, cell pulling and adhesion, impact collectively migrating cell populations. These processes are ubiquitous in cell biology. For example, leader cells pull their followers into the wound area to heal wounded epithelial tissue, and cell adhesions in embryonic cells ensures the self organization of the different germ layers [42, 43, 44]. ABMs provide a promising avenue to model the impacts of these stimuli on collectively migrating cell populations.

We begin this work in Section 2 by presenting the case study ABMs and notation. In Section 3, we discuss our methodologies to forecast and predict ABM behavior. In Section 4, we detail our results on using these methods to forecast and predict data from the three case study ABMs; this section concludes with a brief discussion on the computational expenses of each method. We conclude these results and suggest areas for future work in Section 5.

2 The case study ABMs

We consider three case study ABMs that imitate collective migration during cell biological experiments, including scratch and barrier assays [5, 13, 28, 39, 40]. Each case study ABM models how cell pulling and adhesion impact collective cell migration during these experiments [45, 46]. The ABMs are two-dimensional cellular automata with pulling agents that perform cell pulling rules and/or adhesive agents that perform rules on cell adhesion. Each model is an exclusion process, meaning that each agent can only occupy one

Variable	Description	Range
r_m^{pull}	Pulling agent migration rate	$[0, \infty)$
r_m^{adh}	Adhesive agent migration rate	$[0, \infty)$
p_{pull}	Probability of successful pulling event	$[0, 1]$
p_{adh}	Probability of successful adhesion event	$[0, 1]$
α	Proportion of adhesive agents	$[0, 1]$

Table 1: ABM model parameters. We describe each model parameter and present their range of possible values.

lattice site at a time, and each lattice site is occupied by at most one agent. The first model is borrowed from [12] and consists only of pulling agents; the second model is inspired by the stochastic space jump model from [1] and consists only of adhesive agents; to the best of our knowledge, we are the first to study the third model, which consists of both pulling and adhesive agents.

In this section, we briefly introduce our case study ABMs and their rules on agent pulling and adhesion in Section 2.1; we then detail our ABM notation and simulation in Section 2.2. Additional details on the ABM rules and implementation can be found in electronic supplementary materials A and B, respectively.

2.1 Brief introduction to the case study ABMs and their model rules

Rules A-F governing agent pulling and adhesion are visually depicted in Figure 1, and the parameters for each rule are described in Table 1. In all rules, a *migrating agent* chooses one of its four neighboring lattice site to move into with equal probability (Figure 1(a)). A migration event is aborted if the lattice site in the chosen direction is already occupied (Figure 1(b)). We refer to a *neighboring agent* as an agent located next to the migrating agent in the direction opposite of the chosen migration direction.

Rules A, B, and E are initiated when a pulling agent attempts to migrate, which occurs with rate r_m^{pull}. Migratory pulling agents pull their neighboring agents along with them with probability p_{pull}. Rules C, D, and F are initiated when an adhesive agent attempts to migrate, which occurs with rate r_m^{adh}. Neighboring adhesive agents adhere to migrating agents and abort the migration event with probability p_{adh}. The parameter α corresponds to the proportion of adhesive agents in the simulation. Even though we eventually

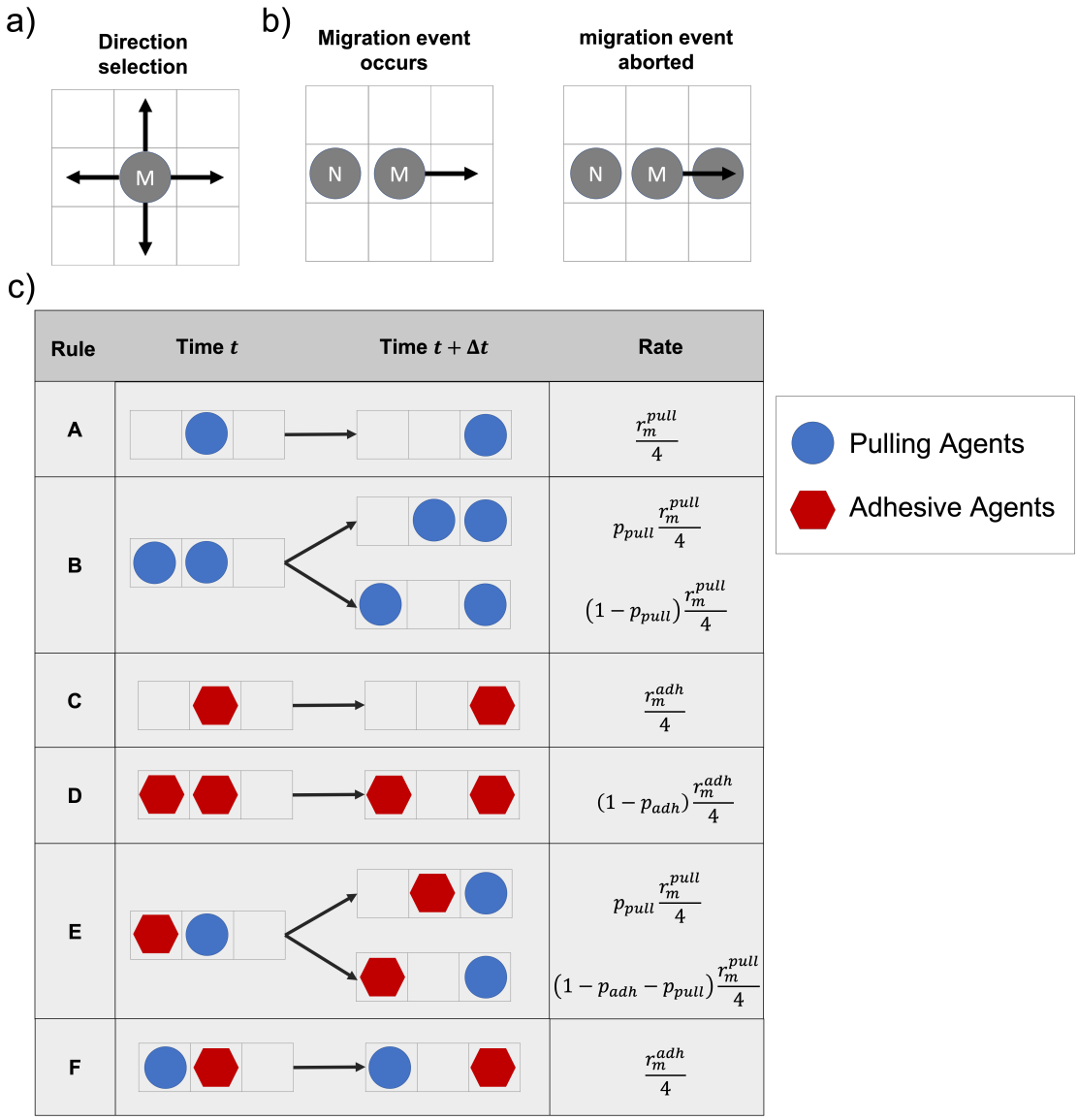

Figure 1: ABM rules on migration, pulling, and adhesion. a) When an agent performs a migration event, it chooses one of the four cardinal directions to move towards with equal probability; migration can also lead to a pulling or adhesion event in the chosen direction. The migrating agent is referred to as a migrating agent (M) b) A migration event requires the lattice site in the chosen migration direction to be empty; otherwise, the migration event is aborted. A neighboring agent (N) is an agent located in the direction opposite the chosen migration direction. c) Rules A-F dictate the rules on agent migration, pulling, and adhesion. Here, we show each rule when an agent chooses to move rightwards. Rule A prescribes how a pulling agent (blue circle) migrates when there is no neighboring agent. Rule B prescribes how a pulling agent migrates and attempts to pull a neighboring pulling agent with it. Rule C prescribes how an adhesive agent (red hexagon) migrates when there is no neighboring agent. Rule D prescribes how a neighboring adhesive agent attempts to adhere to a migrating adhesive agent and abort its migration event. Rule E prescribes how a migrating pulling agent attempts to pull its neighboring adhesive agent, while the adhesive agent attempts to adhere to the pulling agent. Rule F prescribes how a migrating adhesive agent and neighboring pulling agent do not interact with each other. The last column documents the rate at which each lattice site configuration at time t changes to the updated lattice site configuration at time $t + \Delta t$.

summarize each ABM simulation along the x-direction, all rules on migration, pulling, and adhesion occur in all four cardinal directions.

Our three case study ABMs are:

1. **The Pulling ABM**, which consists of rules A and B. This model has parameters $\boldsymbol{p} = (r_m^{pull}, p_{pull})^T$.
2. **The Adhesion ABM**, which consists of rules C and D. This model has parameters $\boldsymbol{p} = (r_m^{adh}, p_{adh})^T$.
3. **The Pulling & Adhesion ABM**, which consists of rules A-F. This model has parameters $\boldsymbol{p} = (r_m^{pull}, r_m^{adh}, p_{pull}, p_{adh}, \alpha)^T$.

2.2 ABM notation

All parameters used to configure ABM simulations are summarized in Table 2. Each model is simulated in the spatial domain $(x, y) \in [0, X] \times [0, Y]$. We represent this space with a two-dimensional lattice with square lattice sites of length $\Delta = 1$ to imitate a typical cell length. Let $N_P^{(r)}(x_i, t_j)$ and $N_H^{(r)}(x_i, t_j)$ denote the number of pulling and adhesive agents, respectively, in the i^{th} column at the j^{th} timepoint for $i = 1, \ldots, X$ and $j = 1, \ldots, T_f$ from the r^{th} of R identically prepared ABM simulations (the input model parameters are fixed but the R model initializations and subsequent agent behaviors are stochastic). Here, X and T_f denote the number of spatial columns and temporal grid points, respectively. To estimate the spatiotemporal pulling and adhesive agent densities from the r^{th} simulation, we compute

$$P^{(r)}(x_i, t_j) = \frac{N_P^{(r)}(x_i, t_j)}{Y} \text{ and } H^{(r)}(x_i, t_j) = \frac{N_H^{(r)}(x_i, t_j)}{Y}, \text{ for } i = 1, \ldots, X, \text{ and } j = 1, \ldots, T_f,$$

respectively. The *total* agent density in the r^{th} simulation is then estimated by

$$T^{(r)}(x_i, t_j) = P^{(r)}(x_i, t_j) + H^{(r)}(x_i, t_j).$$

Variable	Description	Value
R	Number of averaged ABM simulations per dataset	25
t_f	Ending simulation time	1000
Δt	Spacing between temporal gridpoints	10
T_f	Number of total timepoints	100
T_f^{train}	Number of training timepoints	75
T_f^{test}	Number of testing timepoints	25
X	Number of horizontal lattice sites	200
Y	Number of vertical lattice sites	40
Δx	Spacing between spatial points	1

Table 2: ABM configuration parameters. We describe each parameter used for ABM configuration and present the values used throughout this study.

To estimate the averaged pulling, adhesive, and total agent density in the i^{th} column from R identically prepared ABM simulations over time, we compute:

$$\langle P^{ABM}(x_i, t_j)\rangle = \frac{1}{R}\sum_{r=1}^{R} P^{(r)}(x_i, t_j);$$
$$\langle H^{ABM}(x_i, t_j)\rangle = \frac{1}{R}\sum_{r=1}^{R} H^{(r)}(x_i, t_j); \text{ and}$$
$$\langle T^{ABM}(x_i, t_j)\rangle = \frac{1}{R}\sum_{r=1}^{R} T^{(r)}(x_i, t_j), \quad \text{for } i = 1, \ldots, X \text{ and } j = 1, \ldots, T_f.$$

3 Methods to forecast and predict ABM data

In this section, we outline our methodologies for forecasting future ABM data and predicting ABM data at new parameter values. This begins with a description of how we generate ABM data in Section 3.1 followed by an overview of the four methods we use for ABM forecasting in Section 3.2. We then describe our approaches for ABM forecasting and prediction in Sections 3.3 and 3.4, respectively. We visualize how BINNs can be used for these processes in Figure 2. All methods are implemented using Python (version 3.9.12) with code available on GitHub at https://github.com/johnnardini/Forecasting_predicting_ABMs.

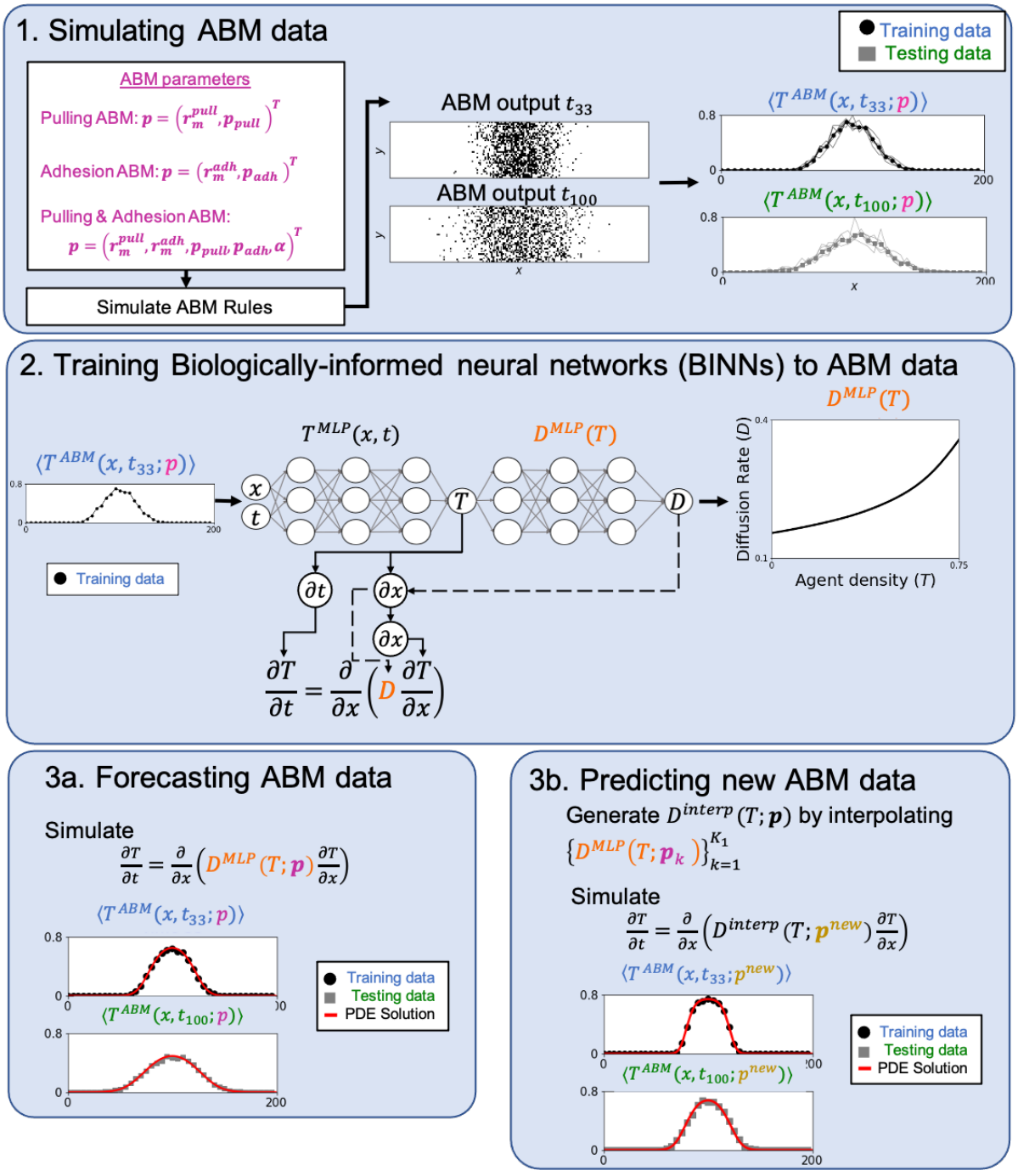

Figure 2: Forecasting and predicting ABM data with BINNs. 1. Simulating ABM data. For a given parameter, $\boldsymbol{p}$, we simulate the Pulling, Adhesion, or Pulling & Adhesion ABM. Each model outputs snapshots of agent locations over time; we summarize this data by estimating the average total agent density along the x-direction for each snapshot. We perform R total ABM simulations (shown as thin lines) for each $\boldsymbol{p}$ and average the total spatiotemporal agent density to obtain $\langle T^{ABM}(x,t;\boldsymbol{p})\rangle$; in this figure, $R = 5$. The first T_f^{train} timepoints are placed into a training ABM dataset, and the final T_f^{test} timepoints are placed into a testing ABM dataset. 2. Training biologically-informed neural networks (BINNs) to ABM data. Each BINN model consists of a data-approximating MLP, $T^{MLP}(x,t)$, and a diffusion-rate-approximating MLP, $\mathcal{D}^{MLP}(T)$. BINN models are trained so that $T^{MLP}(x,t) \approx \langle T^{ABM}(x,t;\boldsymbol{p})\rangle^{train}$ while T^{MLP} and $\mathcal{D}^{MLP}$ satisfy Equation (7). After model training, the inferred $\mathcal{D}^{MLP}(T)$ estimates the agent diffusion rate. 3a. Forecasting ABM data. Simulating the diffusion PDE framework with $\mathcal{D}^{MLP}(T)$ allows us to forecast the ABM training and testing data. 3b. Predicting new ABM data. We predict the rate of agent diffusion at a new parameter, $\boldsymbol{p}^{new}$, by interpolating $\mathcal{D}^{MLP}(T;\boldsymbol{p})$ over several $\boldsymbol{p}$ values to create $\mathcal{D}^{interp}(T;\boldsymbol{p})$. Simulating the diffusion PDE framework with $\mathcal{D}^{interp}(T;\boldsymbol{p}^{new})$ allows us to predict the new ABM training and testing data.

3.1 Simulating ABM data

The process of simulating ABM data is illustrated in Part 1 of Figure 2. At the parameter value $\boldsymbol{p}$, we calculate $\langle T^{ABM}(x,t;\boldsymbol{p})\rangle = \{\langle T^{ABM}(x_i,t_j;\boldsymbol{p})\rangle\}_{i=1,\dots,X}^{j=1,\dots,T_f}$. For subsequent model training and validation purposes, we split $\langle T^{ABM}(x,t;\boldsymbol{p})\rangle$ into training and testing datasets by setting

$$\begin{aligned}\langle T^{ABM}(x,t;\boldsymbol{p})\rangle^{train} &= \left\{\langle T^{ABM}(x_i,t_j;\boldsymbol{p})\rangle\right\}_{i=1,\dots,X}^{j=1,\dots,T_f^{train}}, \text{ and}\\ \langle T^{ABM}(x,t;\boldsymbol{p})\rangle^{test} &= \left\{\langle T^{ABM}(x_i,t_j;\boldsymbol{p})\rangle\right\}_{i=1,\dots,X}^{j=T_f^{train}+1,\dots,T_f^{train}+T_f^{test}}.\end{aligned} \tag{1}$$

Here, T_f^{train} and T_f^{test} denote the number of training and testing timepoints, respectively, and $T_f = T_f^{train} + T_f^{test}$.

3.2 Models to forecast ABM data

We now describe the four models we use to forecast future ABM data. Namely, these models are the mean-field PDE, ANN, BINN, and BINN-guided PDE models.

The mean-field and BINN-guided PDE models consist of simulating a PDE of the form[1]:

$$\frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(\mathcal{D}(T)\frac{\partial T}{\partial x}\right), \tag{2}$$

where $T = T(x,t) = P(x,t) + H(x,t)$ denotes the total agent density over space and time. The form of $\mathcal{D}(T)$ in Equation (2) changes based on the ABM and the modeling approach being used. For the mean-field PDE, we determine the form of $\mathcal{D}(T)$ by converting discrete ABM rules into their continuous counterparts and invoking the mean-field assumption, which may be invalid at some parameter values. BINNs, on the other hand, are a data-driven approach to infer $\mathcal{D}(T)$ from the data without any such *a priori* assumptions.

The ANN and BINN models consist of training a prescribed neural network to ABM data and then using the trained neural network to forecast future data.

3.2.1 Mean-field PDE Models

Here, we present the mean-field PDE models for each case study ABM. More detailed information on how the ABM rules are coarse-grained into these models are provided in electronic supplementary material D.

[1] with the exception of the mean-field PDE for the Pulling & Adhesion ABM, which requires simulating the two-compartment PDE given by Equation (5) in Section 3.2.1

Our numerical method to numerically integrate these PDE models is provided in electronic supplementary material F.

The Pulling ABM: The Pulling ABM includes only pulling agents and consists of Rules A-B from Figure 1. In electronic supplementary material D.1, we show that these rules can be coarse grained into the Pulling ABM's mean-field PDE model:

$$\frac{\partial P}{\partial t} = \nabla \cdot \left(\mathcal{D}^{pull}(P) \nabla P \right), \quad \mathcal{D}^{pull}(P) = \frac{r_m^{pull}}{4} \left(1 + 3 p_{pull} P^2 \right) \tag{3}$$

where $P = P(x, y, t)$ denotes the spatiotemporal pulling agent density.

The Adhesion ABM: The Adhesion ABM includes only adhesive agents and consists of Rules C-D from Figure 1. In electronic supplementary material D.2, we show that these rules can be coarse grained into the Adhesion ABM's mean-field PDE model:

$$\frac{\partial H}{\partial t} = \nabla \cdot \left(\mathcal{D}^{adh}(H) \nabla H \right), \quad \mathcal{D}^{adh}(H) = \frac{3 r_m^{adh}}{4} \left(p_{adh} \left(H - \frac{2}{3} \right)^2 + 1 - \frac{4 p_{adh}}{3} \right) \tag{4}$$

where $H = H(x, y, t)$ denotes the spatiotemporal adhesive agent density.

Notice that $\mathcal{D}^{adh}(H)$ from Equation (4) becomes negative for some density values when $p_{adh} > 0.75$. This PDE thus fails to provide an ABM prediction at these parameter values because negative diffusion is ill-posed [1].

The Pulling & Adhesion ABM: The Pulling & Adhesion ABM includes both pulling and adhesive agents, and consists of Rules A-F from Figure 1. In electronic supplementary material D.3, we show that these rules can be coarse-grained into the Pulling & Adhesion ABM's mean-field PDE model:

$$\begin{aligned}
\frac{\partial P}{\partial t} =& \frac{r_m^{pull}}{4} \nabla \cdot \Big((1 - T) \nabla P + P \nabla T \Big) \\
&+ p_{adh} \frac{r_m^{pull}}{4} \nabla \cdot \Big(-3P(1 - T) \nabla H - H(1 - T) \nabla P - HP \nabla T \Big) \\
&+ p_{pull} \frac{r_m^{pull}}{4} \nabla \cdot \Big(3P^2 \nabla T \Big) \\
\frac{\partial H}{\partial t} =& \frac{r_m^{adh}}{4} \nabla \cdot \Big((1 - T) \nabla H + H \nabla T \Big) \\
&+ p_{adh} \frac{r_m^{adh}}{4} \nabla \cdot \Big(-4(1 - T) H \nabla H - H^2 \nabla T \Big) \\
&+ p_{pull} \frac{r_m^{pull}}{4} \nabla \cdot \Big(-(1 - T) H \nabla P + (1 - T) P \nabla H + 3HP \nabla T \Big).
\end{aligned} \tag{5}$$

This two-compartment PDE describes the spatiotemporal densities of pulling agents, $P(x, y, t)$, and adhesive agents, $H = H(x, y, t)$. The total agent density is given by $T = T(x, y, t) = H(x, y, t) + P(x, y, t)$. To the best of our knowledge, it is not possible to convert Rules A-F into a single-compartment PDE model describing $T(x, y, t)$

3.2.2 The ANN model

ANNs have recently gained traction as surrogate models for ABMs [16, 47]. Here, we consider a simple multilayer perceptron (MLP) model, $T^{MLP}(x, t)$, to predict the total agent density at the spatiotemporal point (x, t). We provide a brief description of the model architecture and training procedure in this section; more detailed information can be found in electronic supplementary material E.

The ANN architecture: $T^{MLP}(x, t)$ has a two-dimensional input, (x, t), and one-dimensional output, $T(x, t)$. This model has three hidden layers, each with 128 neurons. The hidden layers all have sigmoidal activation functions, and the output layer has a softplus activation function.

ANN model training: The ANN model is trained to minimize

$$\mathcal{L}_{ANN} = \mathcal{L}_{WLS}, \tag{6}$$

where $\mathcal{L}_{WLS}$ is given by Equation (29) in electronic supplementary material E and computes a weighted mean-squared error (MSE) between $T^{MLP}(x, t)$ and $\langle T^{ABM}(x, t)\rangle^{train}$. Here, extra weight is assigned to data from the first timepoint to ensure that T^{MLP} closely agrees with the ABM's initial data.

We use the ADAM optimizer with default hyperparameter values to minimize Equation (6). We perform 10^4 epochs with an early stopping criterion of 10^3 epochs.

3.2.3 The BINN model

We provide a brief overview of our BINN model architecture and training procedure, which closely follow the implementation from the original BINN model study in [28]. More detailed information can be found in electronic supplementary material E.

The BINN architecture: We construct BINN models that consist of two sequential MLP models: $T^{MLP}(x, t)$ predicts the total agent density at the point (x, t), and $\mathcal{D}^{MLP}(T)$ predicts the agent diffusion

rate at the density value T (Part 2 of Figure 2). The architecture for $T^{MLP}(x,t)$ here is identical to the ANN architecture. The architecture for $\mathcal{D}^{MLP}(T)$ also has three hidden layers (each with 128 neurons), and the same hidden and output activation functions. However, this model has a one-dimensional input, T, and one-dimensional output, $D(T)$.

BINN model training: The two MLPs comprising the BINN model are trained to concurrently fit the given dataset, $\langle T^{ABM}(x,t)\rangle^{train}$, and solve the PDE given by

$$\frac{\partial}{\partial t}T^{MLP} = \frac{\partial}{\partial x}\left(\mathcal{D}^{MLP}(T^{MLP})\frac{\partial}{\partial x}T^{MLP}\right). \tag{7}$$

This is achieved by minimizing the following multi-term loss function:

$$\mathcal{L}_{BINN} = \mathcal{L}_{WLS} + \epsilon\mathcal{L}_{PDE} + \mathcal{L}_{constr}. \tag{8}$$

The equation for $\mathcal{L}_{WLS}$ is identical to Equation (6), $\mathcal{L}_{PDE}$ computes the MSE between the left- and right-hand sides of Equation (7) to ensure both MLPs satisfy this diffusion framework, and $\mathcal{L}_{constr}$ penalizes the two MLPs for violating user-defined criteria (such as lower and upper bounds on $\mathcal{D}^{MLP}$). The equations for these three terms are provided in Equations (29), (30), and (31) from electronic supplementary material E. The ϵ parameter is chosen to ensure the $\mathcal{L}_{WLS}$ and $\mathcal{L}_{PDE}$ terms are weighted equally.

Following [36], we minimize Equation (28) in a two-step process. In the first process, we minimize Equation (6) over 10^4 epochs with an early stopping criterion of 10^3 epochs. In the second process, we minimize Equation (28) over 10^6 epochs with an early stopping criterion of 10^5 epochs. The ADAM optimizer is used during both steps with its default hyperparameter values.

3.2.4 The BINN-guided PDE model

BINN models are trained to satisfy Equation (7). The *BINN-guided PDE model* computes this learned equation by simulating Equation (2) with $\mathcal{D}(T) = \mathcal{D}^{MLP}(T)$. Our numerical method to numerically integrate this PDE is provided in electronic supplementary material F.

3.3 Forecasting future ABM data

We use the four models introduced in Section 3.2 to forecast future ABM data (Part 3a of Figure 2). In *forecasting*, we assess the ability of a model to compute future ABM data at a fixed parameter value from

previous ABM data. This could correspond to inferring the future behavior of a computationally-intensive ABM simulation or an expensive experimental procedure.

We perform ABM forecasting by training each model to the training ABM dataset and then computing the model prediction over all space- and timepoints. The mean-field PDE model does not require any model training because we can directly compute it from the ABM parameter values. We then partition each model's prediction into training and testing datasets to match the ABM training and testing datasets from Equation (1). We report the training MSE from each model prediction as:

$$\frac{1}{XT_f^{train}} \sum_{i=1}^{X} \sum_{j=1}^{T_f^{train}} \left(T^{model}(x_i, t_j) - \langle T^{ABM}(x_i, t_j)\rangle\right)^2,$$

and the testing MSE as:

$$\frac{1}{XT_f^{test}} \sum_{i=1}^{X} \sum_{j=T_f^{train}+1}^{T_f} \left(T^{model}(x_i, t_j) - \langle T^{ABM}(x_i, t_j)\rangle\right)^2.$$

3.4 Predicting new ABM data using BINN-guided PDE models

We combine BINN modeling, multivariate interpolation, and numerical integration of PDEs to predict new ABM data (Part 3b of Figure 2). In *predicting*, we assess the ability of our proposed approach to compute ABM data at a parameter value that has not been seen previously. This could correspond to exploring an ABMs' parameter space, or predicting the output of an experimental procedure for different experimental conditions, such as drug concentration or the initial number of agents.

We perform multivariate interpolation using BINNs' computed diffusion rates to predict density-dependent diffusion rates for new ABM data. We define a prior parameter collection and a new parameter collection as

$$\mathcal{P}^{prior} = \{\boldsymbol{p}_k\}_{k=1}^{K_1} \text{ and } \mathcal{P}^{new} = \{\boldsymbol{p}_k^{new}\}_{k=1}^{K_2}.$$

Our workflow for predicting ABM data from $\mathcal{P}^{new}$ proceeds as follows:

1. Generate the prior and new ABM data collections by simulating the ABM at all parameters from the prior and new parameter collections:

$$\mathcal{T}^{prior} = \left\{\langle T^{ABM}(x, t; \boldsymbol{p}_k)\rangle\right\}_{k=1}^{K_1} \text{ and } \mathcal{T}^{new} = \left\{\langle T^{ABM}(x, t; \boldsymbol{p}_k^{new})\rangle\right\}_{k=1}^{K_2}.$$

2. Train a BINN model to each k^{th} training ABM dataset from $\mathcal{T}^{prior}$ and extract $\mathcal{D}^{MLP}(T;\boldsymbol{p}_k)$ from the trained BINN model.

3. Perform multivariate interpolation on $\{\mathcal{D}^{MLP}(T;\boldsymbol{p}_k)\}_{k=1}^{K_1}$ to create an interpolant, $\mathcal{D}^{interp}(T;\boldsymbol{p})$, that matches the concatenated vector $[T,\boldsymbol{p}_k]$ to the diffusion rate $\mathcal{D}^{MLP}(T;\boldsymbol{p}_k)$ for $k=1,\ldots,K_1$.

4. Predict the new ABM dataset, $\langle T^{ABM}(x,t;\boldsymbol{p}_k^{new})\rangle$, by simulating Equation (7) with $\mathcal{D}=\mathcal{D}^{interp}(T;\boldsymbol{p}_k^{new})$ to create $T^{interp}(x,t;\boldsymbol{p}_k^{new})$. Partition $T^{interp}(x,t;\boldsymbol{p}_k^{new})$ into its training and testing datasets to match the ABM data's training and testing datasets.

5. Compute the training and testing MSEs between $T^{interp}(x,t;\boldsymbol{p}_k^{new})$ and $\langle T^{ABM}(x,t;\boldsymbol{p}_k^{new})\rangle$ to summarize the predictive performance of $T^{interp}(x,t;\boldsymbol{p}_k^{new})$ for $k=1,\ldots,K_2$.

We implement multi-dimensional radial basis function interpolation using Sci-kit Learn's (version 0.24.2) **RBFInterpolator** command to create $\mathcal{D}^{interp}(T;\boldsymbol{p})$.

4 Results

4.1 Mean-field and BINN-guided PDEs accurately forecast baseline ABM simulations

We simulated the three case study ABMs using the configuration values provided in Table 2. These values were chosen to match previous studies [12, 13]. For ABMs of collective migration, one often chooses a large spatiotemporal domain to ensure ample ABM behavior is observed (e.g., the population spreads) while ensuring the boundary does not affect this behavior. In Table 3, we provide baseline model parameter values for each case study ABM; these values were arbitrarily chosen to demonstrate typical ABM behavior characterized by moderate population spread. The ABM outputs are depicted against each ABM's mean-field PDE in Figure 3. The mean-field PDE models accurately describe the baseline simulations for all three ABMs.

We investigate the performance of the mean-field PDE, ANN, BINN, and BINN-guided PDE models in forecasting Pulling ABM data from the baseline parameter values provided in Table 3. Visual inspection

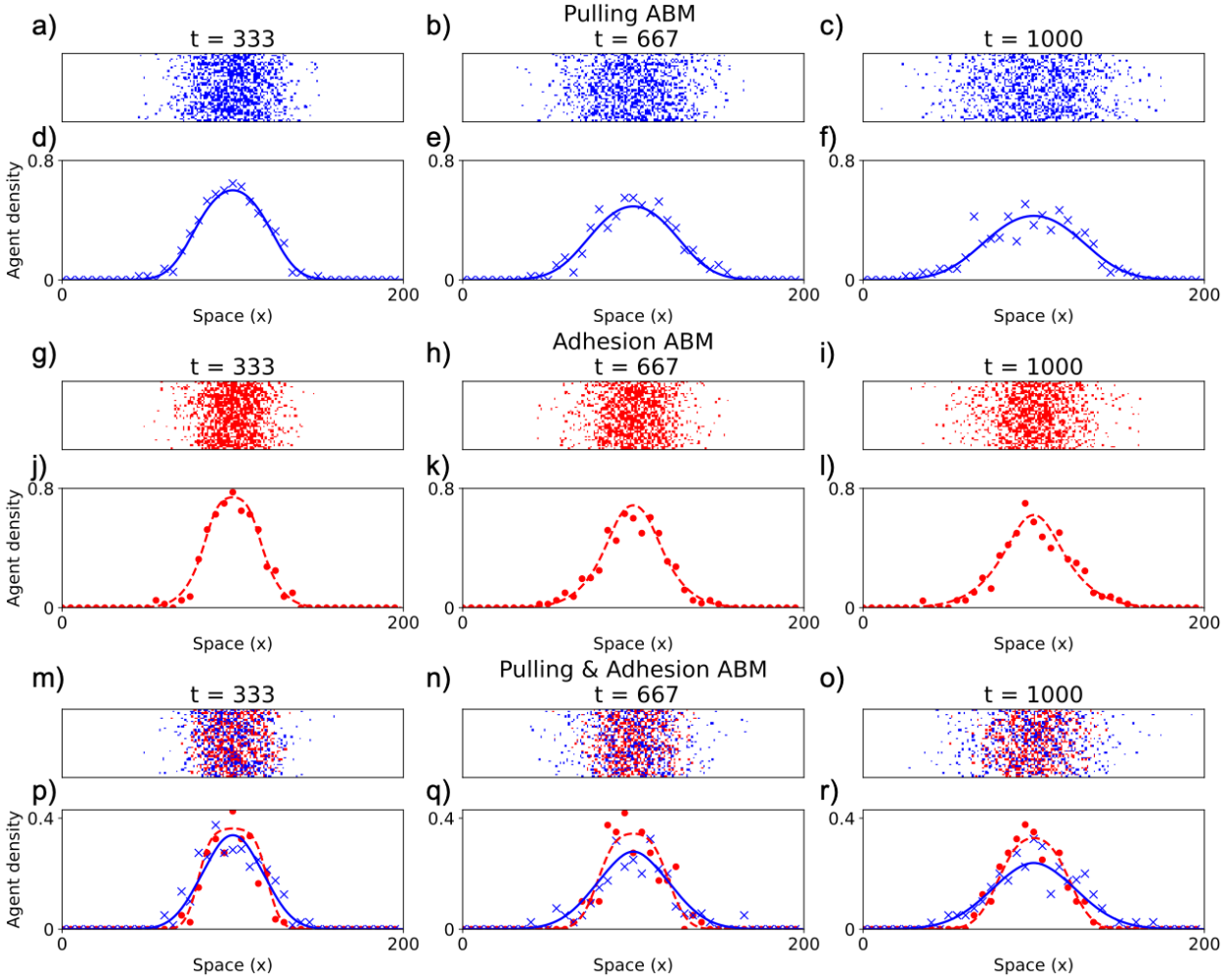

Figure 3: Baseline ABM simulation snapshots and the mean-field PDE models for the Pulling, Adhesion, and Pulling & Adhesion ABMs. Blue pixels denote pulling agents and red pixels denote adhesive agents. All ABMs were simulated on rectangular 200×40 lattices. (a-c) Snapshots of the Pulling ABM for $r_m^{pull} = 1.0, p_{pull} = 0.5$. (d-f) The output spatiotemporal pulling agent density (blue 'x' marks) is plotted against the solution of the mean-field PDE (solid blue line) given by Equation (3). (g-i) Snapshots of the Adhesion ABM for $r_m^{adh} = 1.0, p_{adh} = 0.5$. (j-l) The output spatiotemporal adhesive agent density (red dots) is plotted against the solution of the mean-field PDE (dashed red line) given by Equation (4). (m-o) Snapshots of the Pulling & Adhesion ABM for $r_m^{pull} = 1.0, r_m^{adh} = 0.25, p_{pull} = 0.33, p_{adh} = 0.33, \alpha = 0.5$. (p-r) The output spatiotemporal pulling and adhesive agent densities are plotted against the solution of the mean-field PDE given by Equation (5).

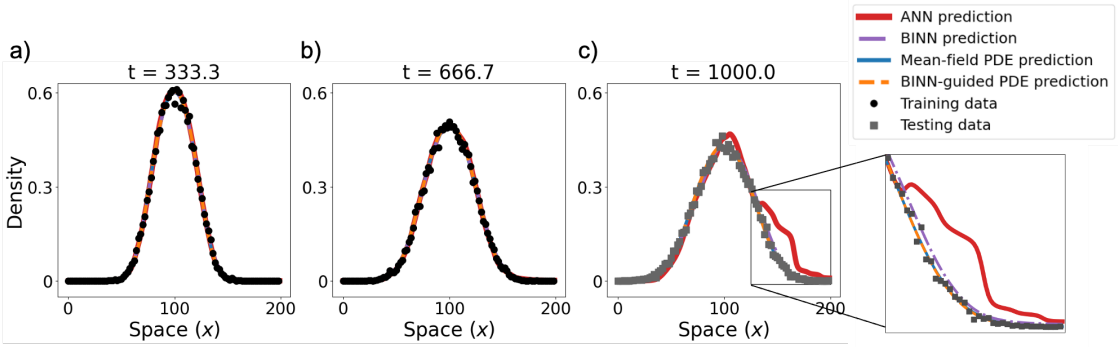

Figure 4: Forecasting Pulling ABM data with neural networks and PDEs. ANN and BINN models were trained to fit $\langle T^{ABM}(x,t)\rangle^{train}$ from the Pulling ABM with $\boldsymbol{p} = (r_m^{pull}, p_{pull})^T = (1.0, 0.5)^T$. These two neural networks and the mean-field and BINN-guided PDE simulations were then used to forecast (a-b) $\langle T^{ABM}(x,t)\rangle^{train}$ and (c) $\langle T^{ABM}(x,t)\rangle^{test}$.

suggests that all four models match the ABM training data well (Figure 4(a-b)). The computed training MSE values reveal that the mean-field and BINN-guided PDEs outperform the neural networks in describing this data (Table 3). The BINN, BINN-guided PDE, and mean-field PDE all accurately forecast the testing data (Figure 4(c)), but the two PDE models achieve smaller testing MSE values than the BINN model (Table 3). The ANN's prediction for the testing data has a protrusion that overpredicts all data for $x > 125$ (Figure 4(c) inset), which causes this model's computed testing MSE value to be almost an order of magnitude higher than all others. We obtain similar results when using the four models to predict data from the Adhesion ABM and Pulling & Adhesion ABM at their baseline parameter values (Table 3 and Supplementary Figure 12).

4.2 Forecasting ABM data for many parameter values with BINN-guided and mean-field PDE simulations

We now investigate the performance of BINN-guided and mean-field PDE simulations in forecasting ABM datasets over a wide range of parameter values for all three case study ABMs. We only consider the two PDE models (and exclude the neural network models) in this section due to their strong forecasting performance in Section 4.1.

Forecasting model	**Training MSE**	**Testing MSE**
The Pulling ABM with baseline parameters $\boldsymbol{p} = (r_m^{pull}, p_{pull})^T = (1.0, 0.5)^T$		
ANN	1.17×10^{-4}	9.36×10^{-4}
BINN	9.32×10^{-5}	1.47×10^{-4}
Mean-field PDE	7.45×10^{-5}	1.00×10^{-4}
BINN-guided PDE	7.64×10^{-5}	1.02×10^{-4}
The Adhesion ABM with baseline parameters $\boldsymbol{p} = (r_m^{adh}, p_{adh})^T = (1.0, 0.5)^T$		
ANN	1.55×10^{-4}	1.84×10^{-3}
BINN	8.54×10^{-5}	1.50×10^{-4}
Mean-field PDE	7.18×10^{-5}	9.21×10^{-5}
BINN-guided PDE	7.43×10^{-5}	1.02×10^{-4}
The Pulling & Adhesion ABM with baseline parameters $\boldsymbol{p} = (r_m^{pull}, r_m^{adh}, p_{pull}, p_{adh}, \alpha)^T = (1.0, 0.25, 0.33, 0.33, 0.5)^T$		
ANN	1.25×10^{-4}	2.67×10^{-3}
BINN	9.65×10^{-5}	9.96×10^{-5}
Mean-field PDE	7.50×10^{-5}	8.55×10^{-5}
BINN-guided PDE	6.55×10^{-5}	9.11×10^{-5}

Table 3: Computed training and testing MSE values. Computed MSE values when forecasting $\langle T^{ABM}(x,t)\rangle^{train}$ and $\langle T^{ABM}(x,t)\rangle^{test}$ from the three ABMs at their baseline parameter values. We used an ANN, BINN, mean-field PDE, and BINN-guided PDE to forecast each baseline ABM dataset.

4.2.1 The BINN-guided and mean-field PDEs both accurately forecast Pulling ABM data

The parameters for the Pulling ABM are $\boldsymbol{p} = (r_m^{pull}, p_{pull})^T$. To evaluate the BINN-guided and mean-field PDE models' performances in forecasting Pulling ABM data over a range of agent pulling parameter values, we computed eleven ABM datasets by varying $p_{pull} = 0.0, 0.1, 0.2, \ldots, 1.0$ while fixing r_m^{pull} at its baseline value of 1.0. The inferred rates of agent diffusion from both models propose that agents diffuse slower for low densities and faster for high densities (Figure 5(a)). While the mean-field diffusion rate at $p_{pull} = 0$ is constant, BINNs do not use this *a priori* information. Instead, their flexible nature leads to them learning a different diffusion rate from the data. The two PDE models achieve comparable training and testing MSE values for all values of p_{pull}, though the mean-field PDE usually attains slightly smaller values (Figure 5(b)). Snapshots of both simulated PDE models against data shows that their ABM predictions are visually indistinguishable (Supplementary Figure 13(a-c)).

To evaluate both PDE models' performances over a range of pulling agent migration values, we computed 10 Pulling ABM datasets with $r_m^{pull} = 0.1, 0.2, \ldots, 1.0$ while fixing p_{pull} at its baseline value of 0.5. We find close agreement between both models' inferred diffusion rates for all values (Figure 5(c)). Both models achieve similar computed training and testing MSE values (Figure 5(d)). Snapshots of both simulated PDE models against data reveals that their ABM predictions are visually indistinguishable (Supplementary Figure 13(d-f)).

4.2.2 BINN-guided PDEs accurately forecast Adhesion ABM data when the mean-field PDE is ill-posed

The parameters for the pulling ABM are $\boldsymbol{p} = (r_m^{adh}, p_{adh})^T$. To evaluate the BINN-guided and mean-field PDE models' performances over a range of agent adhesion parameter values, we computed eleven ABM datasets by varying $p_{adh} = 0.0, 0.1, 0.2, \ldots, 1.0$ while fixing r_m^{adh} at its baseline value of 1.0. The inferred rates of agent diffusion from both models decrease with agent density for most values of p_{adh} (Figure 6(a)). When $p_{adh} = 0$, the BINN-guided diffusion rate is slightly increasing and the mean-field model's diffusion rate is constant. The BINN-guided diffusion rates decline faster with agent density than the corresponding mean-field diffusion rates for low density values. We computed the training and testing MSEs for both

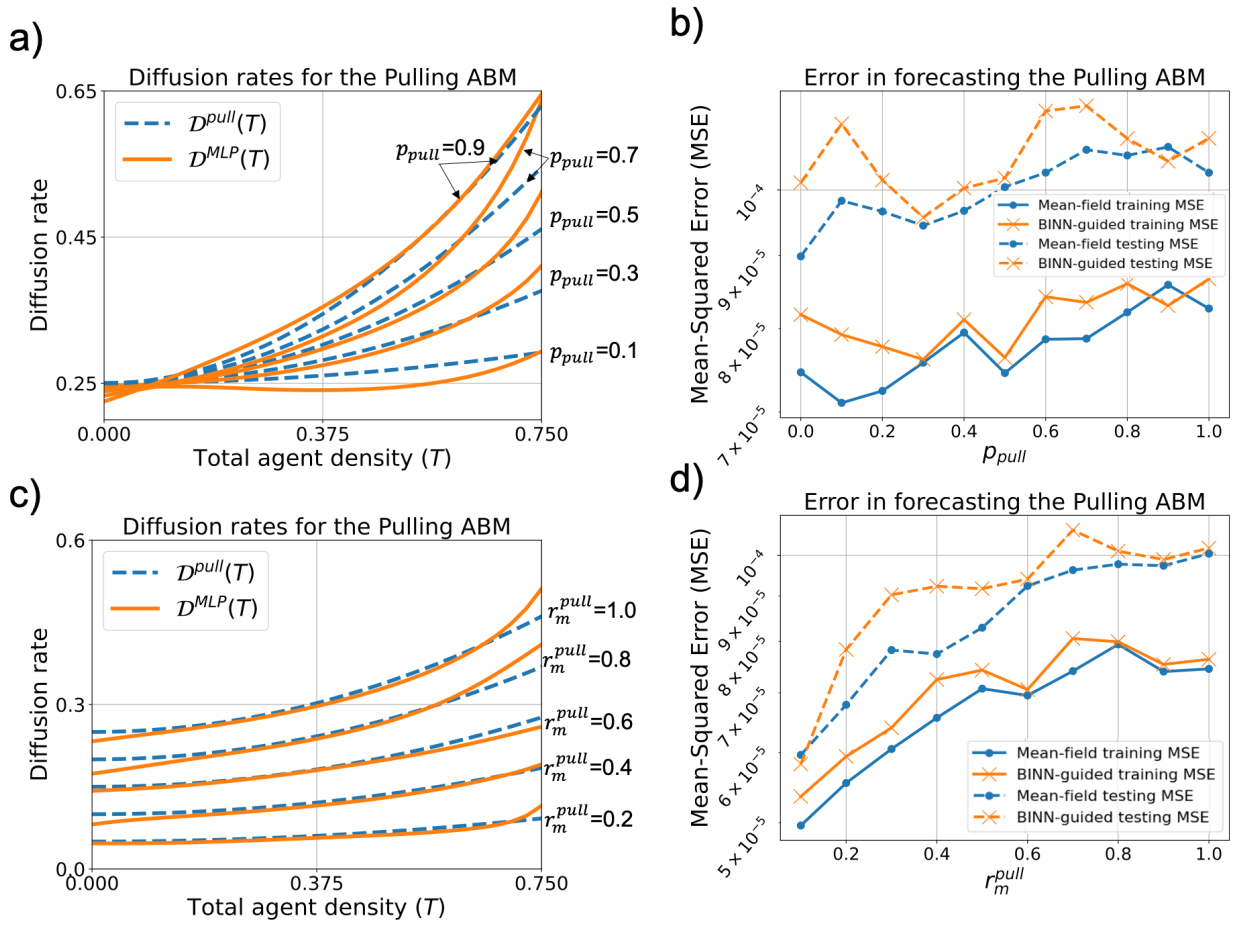

Figure 5: Forecasting Pulling ABM data with the mean-field (MF) and BINN-guided PDEs. (a) Plots of the mean-field diffusion rate, $\mathcal{D}^{pull}(T)$, from Equation (3) and the BINN-guided diffusion rate, $\mathcal{D}^{MLP}(T)$, for $p_{pull} = 0.1, 0.3, \ldots, 0.9$ (results not shown for $p_{pull} = 0.0, 0.2, \ldots, 1.0$ for visual ease) while fixing r_m^{pull} at its baseline value of 1.0. The horizontal axis ends at 0.75 instead of 1.0 because the ABM simulations begin with a density of 0.75 and will rarely exceed this initial value. The BINN cannot reliably predict the diffusion rate for densities outside the values observed in the data. (b) Plots of the mean-field and BINN-guided PDEs' computed training and testing MSE values while varying p_{pull} and fixing $r_m^{pull} = 1.0$. (c) Plots of $\mathcal{D}^{pull}(T)$ and $\mathcal{D}^{MLP}(T)$ for $r_m^{pull} = 0.2, 0.4, \ldots, 1.0$ while fixing p_{pull} at its baseline value of 0.5. (d) Plots of the mean-field and BINN-guided PDEs' computed training and testing MSE values while varying r_m^{pull} and fixing $p_{pull} = 0.5$.

models for all values of p_{adh} (Figure 6(b)) and partition the results as follows :

- **When $p_{adh} < 0.5$**: both models achieve similar training MSE values near 7×10^{-5} and testing MSE values around 10^{-4}.
- **When $0.5 \leq p_{adh} \leq 0.75$:** the mean-field PDE models' training and testing MSE values increase with p_{adh}, with a maximum computed value above 3×10^{-4}. The BINN-guided PDE model's training and testing MSE values remain near 7×10^{-5} and 10^{-4}, respectively.
- **When $p_{adh} > 0.75$:** the mean-field PDE model is ill-posed and cannot forecast this ABM data. The BINN-guided PDE model's computed training and testing MSE values increase with p_{adh} and have a maximum computed value of 2×10^{-4}.

Close inspection of snapshots from both PDE model simulations against ABM data from $p_{adh} = 0.7$ reveals that the mean-field PDE model slightly overpredicts the data at high densities above 0.5 and low densities below 0.1, whereas the BINN-guided PDE closely matches the data (Supplementary Figure 14(a-c)).

To evaluate both PDE models' performances over a range of adhesive agent migration values, we computed ten ABM datasets with $r_m^{adh} = 0.1, 0.2, \ldots, 1.0$ while fixing p_{adh} at its baseline value of 0.5. Both PDEs achieve similar computed training and testing MSE values for most values of r_m^{adh} (Figure 6(d)). When $r_m^{adh} = 0.1$, however, the BINN-guided PDE's testing MSE value is close to 10^{-4}, whereas the mean-field PDE attains a lower testing MSE value near 6×10^{-5}. Despite these differences, the two model simulations appear similar at these parameter values (Supplementary Figure 14(d-f)).

4.2.3 BINN-guided PDEs accurately forecast Pulling & Adhesion ABM data with a one-compartment model

The parameters for the Pulling & Adhesion ABM are $\boldsymbol{p} = (r_m^{pull}, r_m^{adh}, p_{pull}, p_{adh}, \alpha)^T$. We evaluate the performance of the BINN-guided and mean-field DE models in forecasting data from the Pulling & Adhesion ABM. We created 48 ABM datasets by fixing the baseline parameter values at $\boldsymbol{p}_{base} = (1.0, 0.25, 0.33, 0.33, 0.5)^T$ and then varying each parameter individually. We vary $r_m^{pull} = 0.5, 0.6, \ldots, 1.5$; $r_m^{adh} = 0.0, 0.1, \ldots, 1.0$; $p_{pull} = 0.1, 0.2, \ldots, 0.6, 0.67$; $p_{adh} = 0.1, 0.2, \ldots, 0.6, 0.67$; and $\alpha = 0.0, 0.1, \ldots, 1.0$. These parameter values were chosen to always satisfy $p_{pull} + p_{adh} \leq 1$.

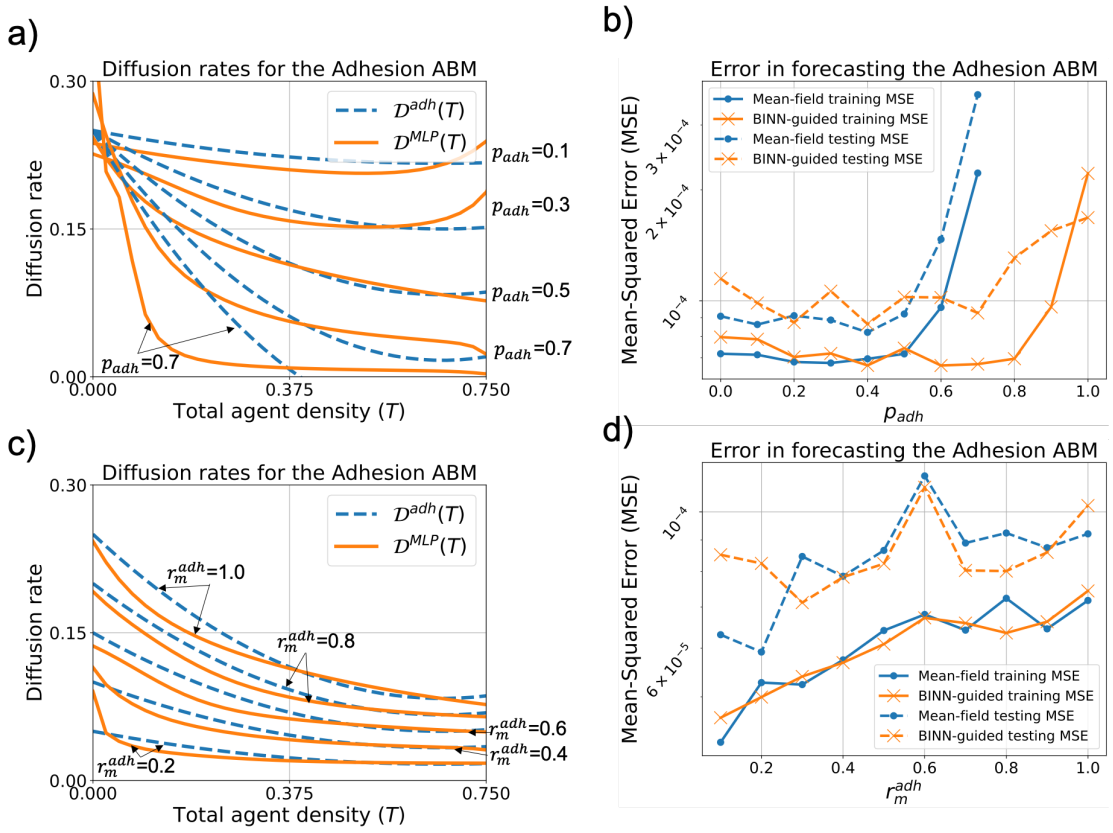

Figure 6: Forecasting Adhesion ABM data with the mean-field (MF) and BINN-guided PDEs. (a) Plots of the mean-field diffusion rate, $\mathcal{D}^{adh}(T)$, from Equation (4) and the BINN-guided diffusion rate, $\mathcal{D}^{MLP}(T)$, for $p_{adh} = 0.1, 0.3, \ldots, 0.9$ (results not shown for $p_{adh} = 0.0, 0.2, \ldots, 1.0$ for visual ease) while fixing r_m^{adh} at its baseline value of 1.0. (b) Plots of the mean-field and BINN-guided PDEs' computed training and testing MSE values while varying p_{adh} and fixing $r_m^{adh} = 1.0$. (c) Plots of $\mathcal{D}^{adh}(T)$ and $\mathcal{D}^{MLP}(T)$ for $r_m^{adh} = 0.2, 0.4, \ldots, 1.0$ while fixing p_{adh} at its baseline value of 0.5. (d) Plots of the mean-field and BINN-guided PDEs' computed training and testing MSE values while varying r_m^{adh} and fixing $p_{adh} = 0.5$.

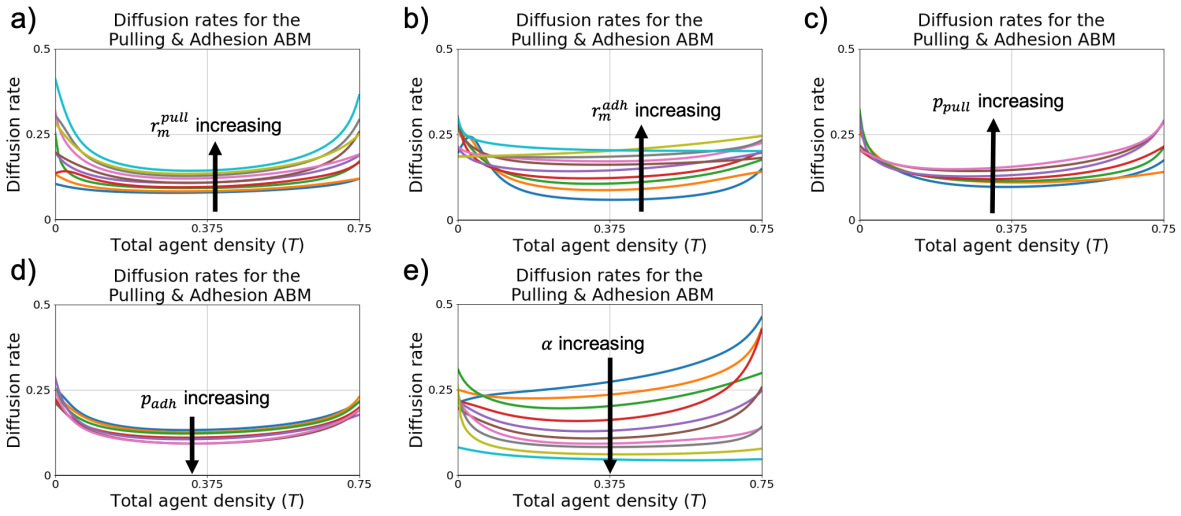

Figure 7: The BINN-guided diffusion rates for the Pulling & Adhesion ABM data. Plots of the BINN-guided diffusion rate, $\mathcal{D}^{MLP}(T)$, when varying (a) r_m^{pull}, (b) r_m^{adh}, (c) p_{pull}, (d) p_{adh}, and (e) α.

The BINN models' inferred diffusion rates, $\mathcal{D}^{MLP}(T;\boldsymbol{p})$, are often U-shaped with larger diffusion values at low and high agent densities and smaller values at intermediate densities (Figure 7). This U-shape tends to increase for larger values of r_m^{pull}, r_m^{adh}, and p_{pull} and decrease for larger values of p_{adh} and α. The inferred diffusion rates appear most sensitive to changes in the α parameter: at $\alpha = 0.0$, $\mathcal{D}^{MLP}(T;\boldsymbol{p})$ strictly increases with agent density and attains an average value of 0.289; at $\alpha = 1.0$, $\mathcal{D}^{MLP}(T;\boldsymbol{p})$ is strictly decreasing and has an average value of 0.051. The inferred diffusion rate is also sensitive to the r_m^{adh} and r_m^{pull} parameters: varying r_m^{adh} primarily alters the BINN diffusion rate at intermediate agent density values, whereas varying r_m^{pull} changes the BINN diffusion rate at low and high agent densitiy values.

The BINN-guided PDE computes a single compartment to forecast the total agent density, $T(x,t)$, whereas the mean-field PDE computes two compartments forecasting the Pulling and Adhesive agent densities, $P(x,t)$ and $H(x,t)$, respectively. We forecast the total agent density with the mean-field PDE by setting $T(x,t) = P(x,t) + H(x,t)$. The two PDE models achieve similar training MSE values for most parameter values that we considered (Figure 8). The mean-field model's testing MSE values are often smaller than the BINN-guided testing MSE values, though the BINN-guided PDE also achieves small testing MSE values. For example, both PDE simulations accurately predict ABM data when p_{adh} is set to 0.4, but visualizing both PDE simulations shows that the mean-field PDE better matches the elbow of the data than the BINN-guided

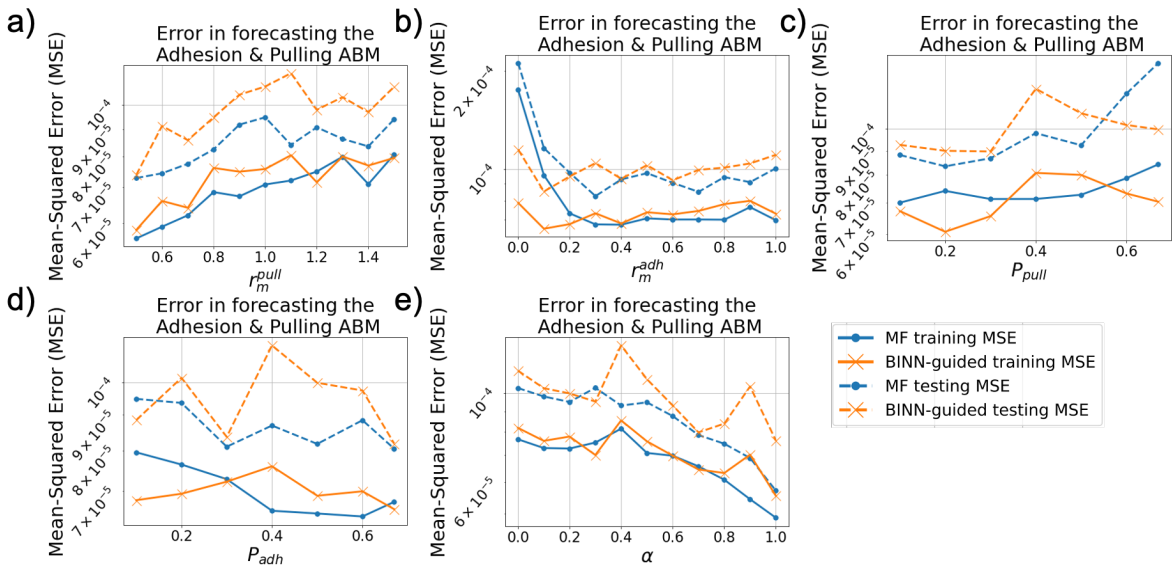

Figure 8: Forecasting Pulling & Adhesion ABM data with the mean-field and BINN-guided PDEs. Plots of the mean-field and BINN-guided PDEs' computed training and testing values while varying (a) r_m^{pull}, (b) r_m^{adh}, (c) p_{pull}, (d) p_{adh}, and (e) α.

PDE (Supplementary Figure 15(a-c)). The BINN-guided PDE outperforms the mean-field PDE in forecasting data for small values of r_m^{adh}: plotting both PDE simulations against data from $r_m^{adh} = 0.1$ shows that the mean-field PDE underpredicts the largest agent density values, while the BINN-guided PDE accurately matches this data (Supplementary Figure 15(d-f)).

4.3 Predicting ABM data at new parameter values

We now examine how performing multivariate interpolation on several BINN-guided diffusion rates, $\mathcal{D}^{MLP}(T; \boldsymbol{p})$, can aid the prediction of previously-unseen ABM data at new parameter values (see Section 3.4 for implementation details).

We predict new data from the Adhesion and Pulling & Adhesion ABMs in this section. We do not include the Pulling ABM in this work because the mean-field PDE model accurately forecasted ABM data for all parameter values that we considered in Section 4.2.1.

4.3.1 Predicting Adhesion ABM data

The parameters for the Adhesion ABM are $\boldsymbol{p} = (r_m^{adh}, p_{adh})^T$. We perform ABM data prediction for $p_{adh} \geq 0.5$ in this section because we found that the mean-field PDE model accurately forecasted ABM data for $p_{adh} \leq 0.5$ in Section 4.2.2.

We first predict ABM data when varying p_{adh} and fixing r_m^{adh}. The prior data collection consists of $K_1 = 6$ ABM datasets generated by varying $p_{adh} = 0.5, 0.6, 0.7, \ldots, 1.0$ while fixing r_m^{adh} at its baseline value of 1.0; the new data collection consists of $K_2 = 5$ ABM datasets generated by varying $p_{adh} = 0.55, 0.65, 0.75, 0.85,$ and 0.95 while fixing r_m^{adh} at its baseline value of 1.0. We performed multivariate interpolation over the six inferred $\mathcal{D}^{MLP}(T; \boldsymbol{p})$ terms from the prior data collection to generate $\mathcal{D}^{interp}(T; \boldsymbol{p})$. We use this interpolant to predict the diffusion rates for all parameters from the new data collection (Figure 9(a)). All interpolated diffusion rates decrease with agent density and tend to fall with larger p_{adh} values. Most of the computed training and testing MSE values on the new data collection are comparable to their counterparts from the prior data collection (Figure 9(b)). The lone exception occurs at $p_{adh} = 0.95$, where the testing MSE exceeds 5×10^{-4} while the testing MSEs at $p_{adh} = 0.9$ and 1.0 do not exceed 2.5×10^{-4}. Visual inspection of the simulated PDE prediction against ABM data at $p_{adh} = 0.95$ reveals that it matches the data well but slightly mispredicts the data's heel at later time points (Supplementary Figure 16(a-c)).

We next predict ABM data when varying both r_m^{adh} and p_{adh}. The prior data collection consists of $K_1 = 18$ ABM datasets generated by varying $r_m^{adh} = 0.1, 0.5, 1.0$ and $p_{adh} = 0.5, 0.6, \ldots, 1.0$; the new data collection consists of $K_2 = 10$ ABM datasets generated from a latin hypercube sampling of $(r_m^{adh}, p_{adh}) \in [0.1, 1.0] \times [0.5, 1.0]$ (Figure 10(a) and Supplementary Table 7). We performed multivariate interpolation over each $\mathcal{D}^{MLP}(T; \boldsymbol{p})$ from the prior data collection to generate $\mathcal{D}^{interp}(T; \boldsymbol{p})$. The predicted diffusion rates for the new data collection decrease with agent density, rise for larger r_m^{adh} values, and decrease faster for larger p_{adh} values (Figure 10(b)). We order the parameters from the new data collection by increasing training MSE values (Figure 10(c)). The four lowest training and testing MSE values are all below 1×10^{-4}, the eight lowest are all below 2×10^{-4}, and the highest testing MSE value reaches 1.6×10^{-3}. Visual inspection of the interpolated PDE prediction with the highest testing MSE value reveals that this simulation mispredicts the data's heel but otherwise matches the ABM data well (Supplementary Figure 17(a-c)). Visual inspection

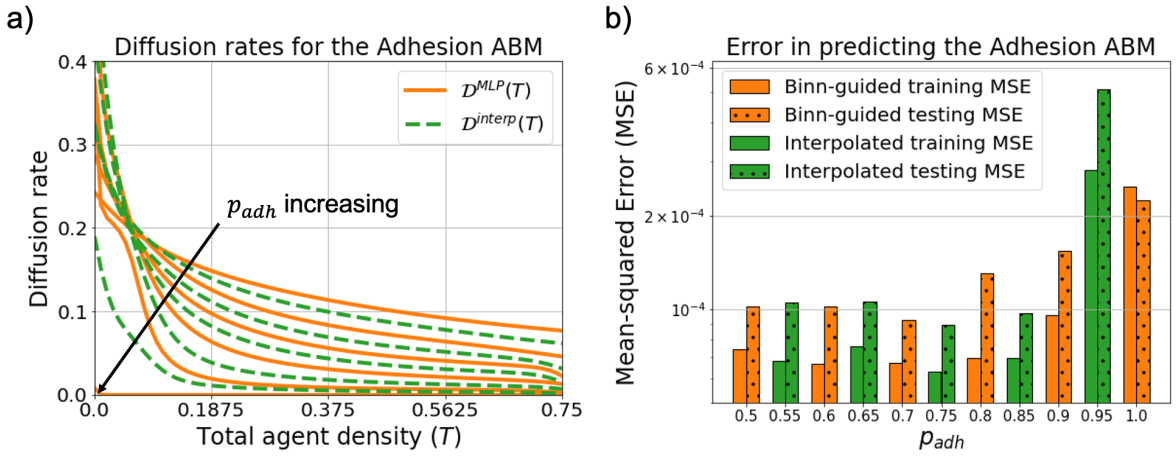

Figure 9: Predicting Adhesion ABM data with BINN-guided PDEs and multivariate interpolation for new p_{adh} values. The parameters for the Adhesion ABM are given by $\boldsymbol{p} = (r_m^{adh}, p_{adh})^T$. Here, we vary p_{adh} while fixing r_m^{adh} at its baseline value of 1.0. The prior data collection consists of $p_{adh} = 0.5, 0.6, \ldots, 1.0$ and the new data collection consists of $p_{adh} = 0.55, 0.65, \ldots, 0.95$ (a) Plots of the learned $\mathcal{D}^{MLP}(T; \boldsymbol{p})$ diffusion rates for the prior data collection. We performed multivariate interpolation on these rates to obtain $\mathcal{D}^{interp}(T; \boldsymbol{p})$, which we plot for the new data collection. (b) Plots of the BINN-guided PDEs' computed training and testing values on the prior data collection, and the interpolated PDE's training and testing values on the new data collection.

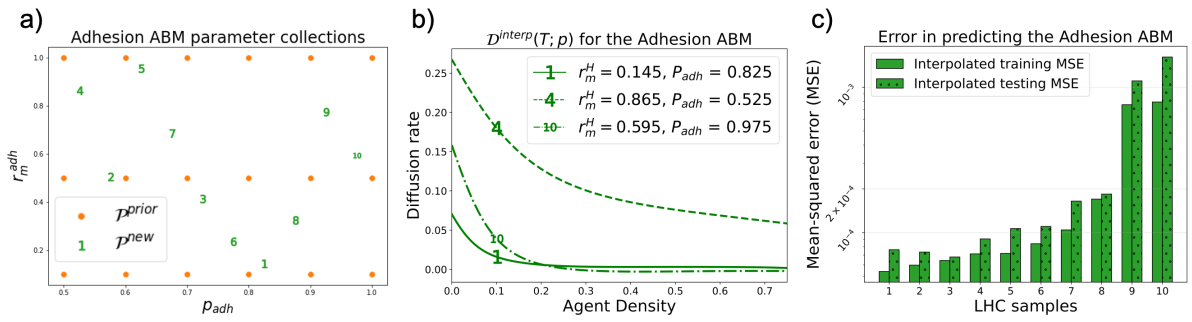

Figure 10: Predicting Adhesion ABM data with BINN-guided PDEs and multivariate interpolation for new r_m^{adh} and p_{adh} values. The parameters for the Adhesion ABM are given by $\boldsymbol{p} = (r_m^{adh}, p_{adh})^T$. Here, we vary both parameters. (a) The prior data collection consists of $r_m^{adh} = 0.1, 0.5, 1.0$ and $p_{adh} = 0.5, 0.6, \ldots, 1.0$ and the new data collection consists of a Latin hypercube (LHC) sampling of $\boldsymbol{p} \in [0.1, 1.0] \times [0.5, 1.0]$ with $K_2 = 10$ samples. (b) We performed multivariate interpolation on the $\mathcal{D}^{MLP}(T;\boldsymbol{p})$ rates on the prior data collection to obtain $\mathcal{D}^{interp}(T;\boldsymbol{p})$. We plot three illustrative $\mathcal{D}^{interp}(T;\boldsymbol{p})$ values from the new data collection. (c) Plots of the interpolated PDE's training and testing values on the new data collection.

of the interpolated PDE prediction with the third-highest MSE value shows that this simulation accurately matches the ABM data (Supplementary Figure 17(d-f)).

4.3.2 Predicting Adhesion & Pulling ABM data

The parameters for the Pulling & Adhesion ABM are $\boldsymbol{p} = (r_m^{pull}, r_m^{adh}, p_{pull}, p_{adh}, \alpha)^T$. We perform ABM data prediction over a large range of parameter values to determine if the one-compartment BINN-guided PDE simulations can predict this ABM's data, which results from two interacting subpopulations.

We perform multivariate interpolation over the p_{pull}, p_{adh}, and α parameters while fixing r_m^{pull} and r_m^{adh} at their baseline values of 1.0 and 0.25, respectively. The prior and new data collections consist of $K_1 = 40$ and $K_2 = 20$ ABM parameter combinations, respectively, that were generated from Latin hypercube samplings of $(p_{pull}, p_{adh}, \alpha) \in [0, 0.67] \times [0, 0.67] \times [0, 1]$ (Figure 11(a) and Supplementary Tables 8 and 9). We chose samplings where $p_{pull} + p_{adh} \leq 1.0$ for all samples. The computed training and testing MSE values for the new parameter collection suggest all simulated PDE predictions accurately match the ABM data at those parameters (Figure 11(b)). Of the $K_2 = 20$ computed testing MSE values in the new data collection, four

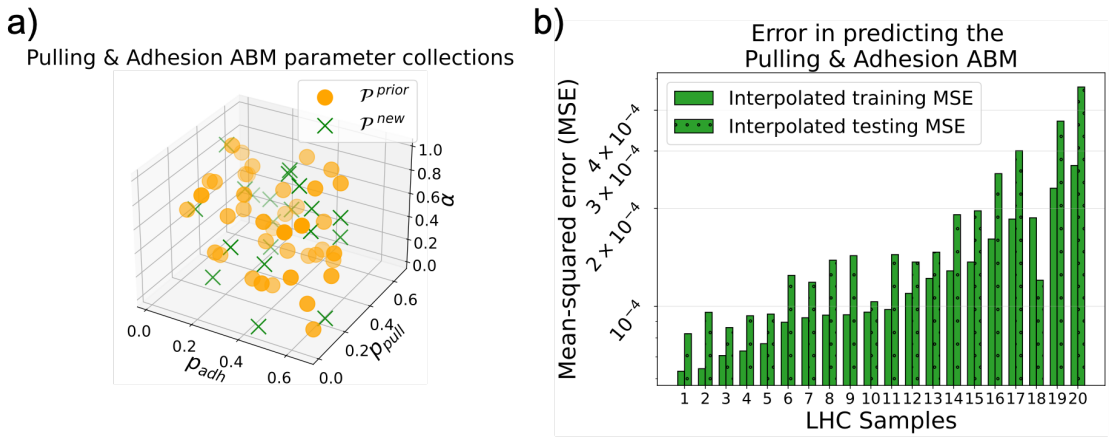

Figure 11: Predicting Pulling & Adhesion ABM data for new p_{pull}, p_{adh}, and α values. The parameters for the Adhesion ABM are given by $\boldsymbol{p} = (r_m^{adh}, r_m^{pull}, p_{adh}, p_{pull}, \alpha)^T$. Here, we vary p_{pull}, p_{adh}, and α while fixing r_m^{pull} and r_m^{adh} at their baseline values of 1.0 and 0.25, respectively. (a) The prior data consists of a Latin hypercube (LHC) sampling of $(p_{pull}, p_{adh}, \alpha) \in [0, 0.67] \times [0, 0.67] \times [0, 1]$ with $K_1 = 40$ samples and the new data consists of a LHC sampling of the same domain with $K_2 = 20$ samples. (b) Plots of the interpolated PDE's training and testing values on the new data, arranged by increasing training MSE values.

are below 1×10^{-4}, 16 are below 2×10^{-4}, and all are below 5×10^{-4}. The highest and third highest testing MSE value results from $(p_{pull}, p_{adh}, \alpha) = (0.218, 0.553, 0.675)$ and $(0.251, 0.486, 0.975)$, respectively. Visually inspecting the interpolated PDE predictions from these parameter values against ABM data reveals that both match the data well, though the worst prediction overpredicts the largest ABM density values (Supplementary Figure 18).

4.4 Comparing the computational expense of each modeling approach

We finish with a discussion on the computational expense of all approaches discussed in this work (Table 4 and Supplementary Figure 19). We recorded the computed wall times to simulate each ABM, train each BINN model, and simulate each PDE from Section 4.2. Averaging across all ABMs suggests that the average ABM dataset took 40.0 minutes to generate with a standard deviation of 15.6 minutes. The average mean-field PDE model simulations for the Pulling ABM and the Adhesion ABM took 0.6 and 0.5 seconds to complete, respectively, which are about 4,000 and 4,500 times faster than the average ABM simulation

ABM Name	ABM simulation	MF PDE simulation	BINN Training	BG PDE simulation
Adhesion	37.5 (15.4) minutes	0.5 (0.15) seconds	10.6 (4.44) hours	16.9 (23.65) seconds
Pulling	39.9 (15.8) minutes	0.6 (0.20) seconds	10.0 (3.99) hours	164.8 (156.9) seconds
Pulling & Adhesion	42.5 (15.52) minutes	4.7 (1.20) seconds	13.1 (4.54) hours	66.9 (50.81) seconds
Average	40.0 minutes	1.9 seconds	11.2 hours	82.9 seconds

Table 4: Computational expenses of each modeling approach. The mean wall time computations (standard deviation in parentheses) for ABM simulations, BINN training, mean-field (MF) PDE simulations, and BINN-guided (BG) PDE simulations for all three ABMs. The last row depicts the average mean computation time across all three ABMs.

time. The average mean-field PDE model simulation time for the Pulling & Adhesion ABM was 4.7 seconds, which is 542 times faster than the average ABM simulation time. Training a BINN model is the most time-consuming task with an average time of 11.2 hours across all ABMs with a standard deviation of 4.32 hours. The average BINN-guided PDE simulation takes 82.9 seconds with a standard deviation of 77.12 seconds, which is approximately 28 times faster than simulating the ABM.

5 Discussion and Future Work

In this work, we introduced how BINNs can be used to learn BINN-guided PDE models from simulated ABM data. BINN-guided PDE model simulations provide a new approach for forecasting and predicting ABM data. This methodology works by training a BINN model to match simulated ABM data while also obeying a pre-specified PDE model framework. After model training, future ABM data can be forecasted by simulating the BINN-guided PDE. Predicting ABM data at new parameters can be performed by simulating the pre-specified PDE framework with an interpolated modeling term. This model term is computed by interpolating over several learned BINN model terms and the parameter values that led to these terms.

It is challenging to predict how model parameters affect ABMs' output behavior due to their heavy computational nature. Mathematical modelers often address this limitation by coarse-graining ABM rules into computationally-efficient mean-field DE models. Unfortunately, these DE models may give misleading

ABM predictions; furthermore, they can be ill-posed for certain parameter values [1, 8]. Here, we demonstrated that BINN-guided PDE models accurately forecast future ABM data and predict ABM data from new parameter values. One benefit of this BINN-guided approach for ABM prediction is that BINNs can, in theory, be trained to simulated data from complex ABMs because BINN models are agnostic to the ABM rules. This is in contrast to the coarse-graining approach, which is limited to ABMs with simple rules to ensure a final PDE model can be recovered.

A limitation of the BINN-guided approach for ABM forecasting and prediction is the computational expense of BINN model training. The average BINN training procedure in this study took 11.2 hours, which is 17 times longer than the average ABM data generation time of 40 minutes. Once a BINN model has been trained, however, the average BINN-guided PDE simulation took 83 seconds, which is 28 times faster than the average time to generate an ABM dataset. One possible source of these long BINN training times is our chosen BINN model architecture, which consists of over over 50,000 parameters to train. Kaplarevi-Malii et al. [35] proposed a genetic algorithm to identify the optimal model archictecture for PINN models. In future work, we plan to implement this algorithm to identify simpler BINN model architectures that can be efficiently trained to learn predictive PDE models for ABMs.

This work was purely computational, as we applied all prediction methodologies to simulated ABM data. It will be interesting in the future to validate the BINN-guided methodology on experimental data. Performing data-driven modeling techniques, such as parameter estimation, is challenging for ABMs due to their long simulation times. Our results suggest that BINN-guided PDE models may advance parameter estimation for ABMs by providing an accurate and efficient ABM surrogate model. For example, a typical approximate Bayesian computation (ABC) for parameter estimation requires performing 10,000 ABM simulations [48], which would require more than 6,600 computational hours. If we instead simulate the ABM at 10 parameter combinations, train BINN models to these data, and then use 10,000 interpolated BINN-guided PDE model simulations for ABC, then this total process would take 349 hours, a 19-fold reduction in time. This process will become even more efficient with new methodologies to expedite BINN model training.

Case study: collective migration. We studied three case study ABMs that are applicable to cell

	ABM prediction	Interpretability
Pulling ABM	MF PDE accurate for all parameters BG PDE accurate for all parameters	MF PDE is interpretable BG PDE is interpretable
Adhesion ABM	MF PDE accurate for $p_{adh} \leq 0.5$ BG PDE accurate for $p_{adh} \leq 0.9$	MF PDE is interpretable BG PDE is interpretable
Pulling & Adhesion ABM	MF PDE accurate for all parameters BG PDE accurate for all parameters	MF PDE not interpretable BG PDE is interpretable

Table 5: Highlighting the ability of mean-field (MF) and BINN-guided (BG) PDEs to accurately forecast simulated ABM data with interpretable PDE models.

biological experiments, such as barrier and scratch assays. Each ABM consists of rules governing how key cellular interactions (namely, pulling and adhesion) impact the collective migration of cell populations during these experiments [5, 17]. Table 5 summarizes the predictive and interpretative capabilities of the mean-field and BINN-guided PDE models for the three case study ABMs. For the Pulling ABM, both models use interpretable one-compartment PDEs that accurately predict ABM behavior for all parameter values. For the Adhesion ABM, the mean-field PDE predictions become less accurate for $p_{adh} \in [0.5, 0.75]$ and are ill-posed for $p_{adh} > 0.75$, whereas the BINN-guided PDEs make accurate predictions for $p_{adh} \leq 0.9$. For the Pulling & Adhesion ABM, both PDE models accurately forecast the total ABM data for most parameter values considered. The mean-field PDE model is not interpretable, as it contains two compartments that consist of many terms. The BINN-guided PDE, on the other hand, achieves similar accuracy to the mean-field PDE with an interpretable one-compartment PDE model.

We compared the performance of the mean-field and BINN-guided PDE models throughout this work. We emphasize, however, that these two approaches are complementary, and our thorough investigation highlights the strengths and limitations of each model. The mean-field PDE is fast to simulate but can provide inaccurate, ill-posed, and/or uninterpretable ABM predictions. The BINN-guided PDE accurately predicts ABM behavior with an interpretable PDE, but current BINN model training times are lengthy. We encourage modelers to refer to these guidelines when deciding which approach to use for their future

applications.

Acknowledgements: The author thanks R. Baker, J. Gevertz, and S. Nardini for helpful discussion and commentary.

Data Availability statement: All code and simulated data for this work is publicly available at https://github.com/johnnardini/Forecasting_predicting_ABMs.

Conflict of interest: The author declares no conflict of interest

Funding Statement: The author acknowledges use of the ELSA high performance computing cluster at The College of New Jersey for conducting the research reported in this paper. This cluster is funded in part by the National Science Foundation under grant numbers OAC-1826915 and OAC-2320244.

References

[1] Kathleen Anguige and Christian Schmeiser. A one-dimensional model of cell diffusion and aggregation, incorporating volume filling and cell-to-cell adhesion. *Journal of Mathematical Biology*, 58(3):395, March 2009.

[2] Fred Brauer, Carlos Castillo-Chavez, and Zhilan Feng. *Mathematical models in epidemiology*, volume 69 of *Texts in Applied Mathematics*. Springer, New York, NY, 2019.

[3] Theo Gibbs, Simon A. Levin, and Jonathan M. Levine. Coexistence in diverse communities with higher-order interactions. *Proceedings of the National Academy of Sciences*, 119(43):e2205063119, October 2022. Publisher: Proceedings of the National Academy of Sciences.

[4] A. Huppert and G. Katriel. Mathematical modelling and prediction in infectious disease epidemiology. *Clinical Microbiology and Infection*, 19(11):999–1005, November 2013.

[5] John T. Nardini, Douglas A. Chapnick, Xuedong Liu, and David M. Bortz. Modeling keratinocyte wound healing: cell-cell adhesions promote sustained migration. *Journal of Theoretical Biology*, 400:103–117, July 2016.

[6] Shahzeb Raja Noureen, Jennifer P. Owen, Richard L. Mort, and Christian A. Yates. Swapping in lattice-based cell migration models. *Physical Review E*, 107(4):044402, April 2023.

[7] Yanni Xiao and Lansun Chen. Modeling and analysis of a predator–prey model with disease in the prey. *Mathematical Biosciences*, 171(1):59–82, May 2001.

[8] Ruth E. Baker and Matthew J. Simpson. Correcting mean-field approximations for birth-death-movement processes. *Physical Review E*, 82(4):041905, October 2010.

[9] Volker Grimm, Eloy Revilla, Uta Berger, Florian Jeltsch, Wolf M. Mooij, Steven F. Railsback, Hans-Hermann Thulke, Jacob Weiner, Thorsten Wiegand, and Donald L. DeAngelis. Pattern-oriented modeling of agent-based complex systems: lessons from ecology. *Science*, 310(5750):987–991, November 2005. Publisher: American Association for the Advancement of Science.

[10] Brandon D. L. Marshall and Sandro Galea. Formalizing the role of agent-based modeling in causal inference and epidemiology. *American Journal of Epidemiology*, 181(2):92–99, January 2015.

[11] Melissa Tracy, Magdalena Cerdá, and Katherine M. Keyes. Agent-based modeling in public health: current applications and future directions. *Annual Review of Public Health*, 39(1):77–94, 2018. _eprint: https://doi.org/10.1146/annurev-publhealth-040617-014317.

[12] George Chappelle and Christian A. Yates. Pulling in models of cell migration. *Physical Review E*, 99(6):062413, June 2019.

[13] Matthew J. Simpson, Ruth E. Baker, Pascal R. Buenzli, Ruanui Nicholson, and Oliver J. Maclaren. Reliable and efficient parameter estimation using approximate continuum limit descriptions of stochastic models. *Journal of Theoretical Biology*, 549:111201, September 2022.

[14] John T. Nardini, Ruth E. Baker, Matthew J. Simpson, and Kevin B. Flores. Learning differential equation models from stochastic agent-based model simulations. *Journal of The Royal Society Interface*, 18(176):rsif.2020.0987, 20200987, March 2021.

[15] Le-Minh Kieu, Nicolas Malleson, and Alison Heppenstall. Dealing with uncertainty in agent-based models for short-term predictions. *Royal Society Open Science*, 7(1):191074, January 2020.

[16] Dale Larie, Gary An, and R. Chase Cockrell. The use of artificial neural networks to forecast the behavior of agent-based models of pathophysiology: an example utilizing an agent-based model of sepsis. *Frontiers in Physiology*, 12:716434, October 2021.

[17] Robin N. Thompson, Christian A. Yates, and Ruth E. Baker. Modelling cell migration and adhesion during development. *Bulletin of Mathematical Biology*, 74(12):2793–2809, December 2012.

[18] Daniel J. VandenHeuvel, Pascal R. Buenzli, and Matthew J. Simpson. Pushing coarse-grained models beyond the continuum limit using equation learning. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 480(2281):20230619, January 2024.

[19] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, April 2016.

[20] E. Kaiser, J. Nathan Kutz, and Steven L. Brunton. Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474(2219):20180335, November 2018.

[21] Samuel Rudy, Alessandro Alla, Steven L. Brunton, and J. Nathan Kutz. Data-driven identification of parametric partial differential equations. *SIAM Journal on Applied Dynamical Systems*, 18(2):643–660, January 2019. Publisher: Society for Industrial and Applied Mathematics.

[22] Kathleen Champion, Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116(45):22445–22451, November 2019. Publisher: National Academy of Sciences Section: Physical Sciences.

[23] Niall M. Mangan, Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Inferring biological networks by sparse identification of nonlinear dynamics. *IEEE Transactions on Molecular, Biological and Multi-Scale Communications*, 2(1):52–63, June 2016. Conference Name: IEEE Transactions on Molecular, Biological and Multi-Scale Communications.

[24] Niall M. Mangan, J. Nathan Kutz, Steven L. Brunton, and Joshua L. Proctor. Model selection for dynamical systems via sparse regression and information criteria. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2204):20170009, August 2017.

[25] Daniel A. Messenger and David M. Bortz. Weak SINDy: galerkin-based data-driven model selection. *Multiscale Modeling & Simulation*, 19(3):1474–1497, January 2021.

[26] Daniel A. Messenger and David M. Bortz. Weak SINDy for partial differential equations. *Journal of Computational Physics*, 443:110525, October 2021.

[27] John H. Lagergren, John T. Nardini, G. Michael Lavigne, Erica M. Rutter, and Kevin B. Flores. Learning partial differential equations for biological transport models from noisy spatio-temporal data. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 476(2234):20190800, February 2020.

[28] John H. Lagergren, John T. Nardini, Ruth E. Baker, Matthew J. Simpson, and Kevin B. Flores. Biologically-informed neural networks guide mechanistic modeling from sparse experimental data. *PLOS Computational Biology*, 16(12):e1008462, December 2020. Publisher: Public Library of Science.

[29] John T. Nardini, John H. Lagergren, Andrea Hawkins-Daarud, Lee Curtin, Bethan Morris, Erica M. Rutter, Kristin R. Swanson, and Kevin B. Flores. Learning equations from biological data with limited time samples. *Bulletin of Mathematical Biology*, 82(9):119, September 2020.

[30] Samuel H. Rudy, Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3(4):e1602614, April 2017. Publisher: American Association for the Advancement of Science Section: Research Article.

[31] Daniel A. Messenger, Graycen E. Wheeler, Xuedong Liu, and David M. Bortz. Learning anisotropic interaction rules from individual trajectories in a heterogeneous cellular population. *Journal of The Royal Society Interface*, 19(195):20220412, October 2022.

[32] Daniel A. Messenger and David M. Bortz. Learning mean-field equations from particle data using WSINDy. *Physica D: Nonlinear Phenomena*, 439:133406, November 2022.

[33] Rohit Supekar, Boya Song, Alasdair Hastewell, Gary P. T. Choi, Alexander Mietke, and Jörn Dunkel. Learning hydrodynamic equations for active matter from particle simulations and experiments. *Proceedings of the National Academy of Sciences*, 120(7):e2206994120, February 2023.

[34] Shengze Cai, Zhicheng Wang, Sifan Wang, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks for heat transfer Problems. *Journal of Heat Transfer*, 143(6):060801, June 2021.

[35] Ana Kaplarević-Malisić, Branka Andrijević, Filip Bojović, Srdan Nikolić, Lazar Krstić, Boban Stojanović, and Miloš Ivanović. Identifying optimal architectures of physics-informed neural networks by evolutionary strategy. *Applied Soft Computing*, page 110646, July 2023.

[36] Kevin Linka, Amelie Schäfer, Xuhui Meng, Zongren Zou, George Em Karniadakis, and Ellen Kuhl. Bayesian physics informed neural networks for real-world nonlinear dynamical systems. *Computer Methods in Applied Mechanics and Engineering*, 402:115346, December 2022.

[37] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, February 2019.

[38] Yeonjong Shin, Jerome Darbon, and George Em Karniadakis. On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type PDEs. *Communications in Computational Physics*, 28(5):2042–2074, June 2020. arXiv: 2004.01806.

[39] Stuart T. Johnston, Matthew J. Simpson, and Ruth E. Baker. Mean-field descriptions of collective migration with strong adhesion. *Physical Review E*, 85(5):051922, May 2012.

[40] Christine Decaestecker, Olivier Debeir, Philippe Van Ham, and Robert Kiss. Can anti-migratory drugs be screened in vitro? A review of 2D and 3D assays for the quantitative analysis of cell migration. *Medicinal Research Reviews*, 27(2):149–176, March 2007.

[41] Asha M. Das, Alexander M. M. Eggermont, and Timo L. M. ten Hagen. A ring barrier–based migration

assay to assess cell migration in vitro. *Nature Protocols*, 10(6):904–915, June 2015. Number: 6 Publisher: Nature Publishing Group.

[42] Jubin Kashef and Clemens M. Franz. Quantitative methods for analyzing cell–cell adhesion in development. *Developmental Biology*, 401(1):165–174, May 2015.

[43] Jan-Hendrik Venhuizen and Mirjam M. Zegers. Making heads or tails of it: cell–cell adhesion in cellular and supracellular polarity in collective migration. *Cold Spring Harbor Perspectives in Biology*, 9(11):a027854, November 2017.

[44] Medhavi Vishwakarma, Joachim P. Spatz, and Tamal Das. Mechanobiology of leader-follower dynamics in epithelial cell migration. *Current Opinion in Cell Biology*, 66:97–103, October 2020.

[45] Michalina Janiszewska, Marina C. Primi, and Tina Izard. Cell adhesion in cancer: beyond the migration of single cells. *Journal of Biological Chemistry*, 295(8):2495–2505, February 2020.

[46] Katheryn E. Rothenberg, Yujun Chen, Jocelyn A. McDonald, and Rodrigo Fernandez-Gonzalez. Rap1 coordinates cell-cell adhesion and cytoskeletal reorganization to drive collective cell migration in vivo. *Current Biology*, 33(13):2587–2601.e5, July 2023.

[47] Claudio Angione, Eric Silverman, and Elisabeth Yaneske. Using machine learning as a surrogate model for agent-based simulations. *PLOS ONE*, 17(2):e0263150, February 2022.

[48] Kyle C. Nguyen, Carter D. Jameson, Scott A. Baldwin, John T. Nardini, Ralph C. Smith, Jason M. Haugh, and Kevin B. Flores. Quantifying collective motion patterns in mesenchymal cell populations using topological data analysis and agent-based modeling. *Mathematical Biosciences*, 370:109158, April 2024.

[49] Alexander Kurganov and Eitan Tadmor. New high-resolution central schemes for nonlinear conservation laws and convection–diffusion equations. *Journal of Computational Physics*, 160(1):241–282, May 2000.

[50] Linda Petzold. Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations. *SIAM Journal on Scientific and Statistical Computing*, 4(1):136–148, March 1983.

Contents

A ABM Rules

A.1 The Pulling Model

The Pulling model consists of pulling agents that migrate with rate[2] r_m^{pull} and perform rules A and B from Figure 1. Suppose a pulling agent at lattice site (i, j) chooses to move rightwards into site $(i+1, j)$. If the lattice site $(i-1, j)$ is unoccupied, then the agent performs Rule A and moves into site $(i+1, j)$. If the lattice site $(i-1, j)$ is occupied, then the agent attempts Rule B on agent pulling. This event succeeds with probability p_{pull}, and the agent moves to site $(i+1, j)$ and pulls its neighbor into lattice site (i, j). This event fails with probability $1-p_{pull}$, in which the agent moves into site $(i+1, j)$ but the neighbor remains at lattice site $(i-1, j)$. These rules can be described by the following trimolecular reaction rates:

$$0_{i-1,j} + P_{i,j} + 0_{i+1,j} \xrightarrow{r_m^{pull}/4} 0_{i-1,j} + 0_{i,j} + P_{i+1,j}, \qquad \text{(Rule A)}$$

$$P_{i-1,j} + P_{i,j} + 0_{i+1,j} \xrightarrow{p_{pull} r_m^{pull}/4} 0_{i-1,j} + P_{i,j} + P_{i+1,j}, \qquad \text{(Rule B.1)}$$

$$P_{i-1,j} + P_{i,j} + 0_{i+1,j} \xrightarrow{(1-p_{pull}) r_m^{pull}/4} P_{i-1,j} + 0_{i,j} + P_{i+1,j}. \qquad \text{(Rule B.2)}$$

Equivalent reactions govern agent migration and pulling in the other three directions.

A.2 The Adhesion Model

The Adhesion model consists of adhesive agents that migrate with rate r_m^{adh} and perform rules C and D from Figure 1. Suppose an adhesive agent at lattice site (i, j) chooses to move rightwards into site $(i+1, j)$. If the lattice site $(i-1, j)$ is unoccupied, then the agent performs Rule C and moves into site $(i+1, j)$. If the lattice site $(i-1, j)$ is occupied, then the neighboring agent attempts Rule D to adhere to the migrating agent and abort their movement. This event succeeds with probability p_{adh}, and neither agent changes its location. This adhesion event fails with probability $1-p_{adh}$, and the migrating agent moves to site $(i+1, j)$ and the neighbor remains at lattice site $(i-1, j)$. These rules can be described by the following trimolecular

[2] Meaning that pulling agents attempt to migrate over an infinitesimal time interval of length dt with probability $r_m^{pull} dt$.

reaction rates:

$$0_{i-1,j} + H_{i,j} + 0_{i+1,j} \xrightarrow{r_m^{adh}/4} 0_{i-1,j} + 0_{i,j} + H_{i+1,j}, \quad \text{(Rule C)}$$

$$H_{i-1,j} + H_{i,j} + 0_{i+1,j} \xrightarrow{(1-p_{adh})r_m^{adh}/4} H_{i-1,j} + 0_{i,j} + H_{i+1,j}. \quad \text{(Rule D)}$$

A.3 The Pulling & Adhesion Model

The Pulling & Adhesion model consists of both pulling and adhesive agents. This model implements Rules A-F from Figure 1. Rules A-D are unchanged from their descriptions in Sections A.1 and A.2. If a pulling agent at lattice site (i, j) chooses to move rightwards into site $(i+1, j)$ while an adhesive agent occupies site $(i-1, j)$, then Rule E dictates the agents' attempts to pull and adhere to each other. The migrating pulling agent succeeds with probability p_{pull} and moves to site $(i+1, j)$ while pulling the neighboring adhesive agent into site (i, j); the neighboring adhesive agent successfully aborts the pulling agent's migration event with probability p_{adh}; both agents fail with probability $1-p_{adh}-p_{pull}$ and the pulling agent moves to site $(i+1, j)$ while the adhesive agent remains at site $(i-1, j)$. Based on our definition of this rule, it is not possible that both the pulling and adhesion events succeed, so the parameters must satisfy $0 \leq p_{pull} + p_{adh} \leq 1$. Rule E can be described by the following trimolecular reaction rate:

$$H_{i-1,j} + P_{i,j} + 0_{i+1,j} \xrightarrow{p_{pull}r_m^{pull}/4} 0_{i-1,j} + H_{i,j} + P_{i+1,j}, \quad \text{(Rule E.1)}$$

$$H_{i-1,j} + P_{i,j} + 0_{i+1,j} \xrightarrow{(1-p_{adh}-p_{pull})r_m^{pull}/4} H_{i-1,j} + 0_{i,j} + P_{i+1,j}. \quad \text{(Rule E.2)}$$

If an adhesive agent at lattice site (i, j) chooses to move rightwards into site $(i+1, j)$ while a pulling agent occupies site $(i-1, j)$, then Rule F dictates that the adhesive agent moves into site $(i+1, j)$ and the pulling agent remains at site $(i-1, j)$. Rule F can be described by the following trimolecular reaction rate:

$$P_{i-1,j} + H_{i,j} + 0_{i+1,j} \xrightarrow{r_m^{adh}/4} P_{i-1,j} + 0_{i,j} + H_{i+1,j}. \quad \text{(Rule F)}$$

B ABM implementation

Each model is simulated in the spatial domain $(x, y) \in [0, X] \times [0, Y]$. We choose $X = 200$ and $Y = 40$ to represent a thin rectangle where collective migration primarily occurs along the x-dimension and is not affected by the boundary in this dimension. We represent this space with a two-dimensional lattice with square lattice sites with length $\Delta = 1$ to imitate a typical cell length. The $(i, j)^{\text{th}}$ lattice site is centered at (x_i, y_j), where $x_i = (i - 0.5)\Delta,\ i = 1, \ldots, X$, and $y_j = (j - 0.5)\Delta,\ j = 1, \ldots, Y$. Each model is an exclusion process, meaning that each agent can only occupy one lattice site at a time, and each lattice site is occupied by at most one agent. The parameter $\alpha \in (0, 1)$ denotes the proportion of nonempty lattice sites that are occupied by adhesive agents in the simulation, and $(1 - \alpha)$ denotes the proportion of nonempty lattice sites that are occupied by pulling agents in the simulation.

All model simulations are initialized by populating 75% of the lattice sites in the middle 20% of columns, e.g., 75% of the lattice sites in $\{(x_i, y_j)\}_{j=1}^{Y}$ are initially occupied for $i = 80, \ldots, 120$. All other columns are initially empty. This initial condition is chosen to reflect a barrier assay [41]. Reflecting boundary conditions are used at the boundaries of lattice to enforce a no-flux condition in the spatial domain. We simulate each ABM using the Gillespie algorithm, which we provide for the Pulling & Adhesion ABM in Supplementary Algorithm 1 in electronic supplementary material C. All ABMs are simulated until $t = 1000$.

C Gillespie algorithm

Our implementation of the Gillespie Algorithm for the Pulling & Adhesion ABM is provided in Supplementary Algorithm 1.

Algorithm 1: Gillespie algorithm for the Pulling & Adhesion ABM

Create an $X \times Y$ lattice with user-specified placement of agents

Set $t = 0$

Set maximum simulation time t_{end}

Set $P(t)$ and $H(t)$ equal to the number of Pulling and Adhesive agents on the lattice, respectively

while $t < t_{end}$ **do**

- Calculate the following random variables, uniformly distributed on $[0, 1] : \gamma_1, \gamma_2$
- Calculate the propensity function $a(t) = r_m^{pull} P(t) + r_m^{adh} H(t)$
- Calculate time step $\tau = -\ln(\gamma_1)/a(t)$
- $t = t + \tau$
- $R = a(t)\gamma_2$
- **if** $R < r_m^{pull} P(t)$ **then**
  - Perform Pulling agent migration (Supplementary Algorithm 2)
- **else if** $R < r_m^{pull} P(t) + r_m^{adh} H(t)$ **then**
  - Perform Adhesive agent migration (Supplementary Algorithm 3)

end

Algorithm 2: Pulling Agent migration

Randomly choose a pulling agent and determine its lattice site index, $\vec{x} = (i, j)^T$

Choose one of the four cardinal migration directions,

$\vec{dx} = (dx, dy)^T \in \{(1, 0)^T, (-1, 0)^T, (0, 1)^T, (0, -1)^T\}$, with equal probability, 1/4. The neighboring direction is given by $\hat{dx} = -\vec{dx}$

if $\vec{x} + \vec{dx}$ *is empty* **then**

- **if** $\vec{x} + \hat{dx}$ *is empty* **then**
  - `/* Rule A */`
  - Move the chosen pulling agent to lattice site $\vec{x} + \vec{dx}$
- **else if** $\vec{x} + \hat{dx}$ *is occupied by a Pulling agent* **then**
  - `/* Rule B */`
  - Calculate the random variable, γ_3, uniformly distributed on $[0, 1]$
  - **if** $\gamma_3 \leq p_{pull}$ **then**
    - Move the chosen pulling agent to lattice site $\vec{x} + \vec{dx}$
    - Move the neighboring agent to lattice site $\vec{x}$
  - **else if** $\gamma_3 > p_{pull}$ **then**
    - Move the chosen pulling agent to lattice site $\vec{x} + \vec{dx}$
- **else if** $\vec{x} + \hat{dx}$ *is occupied by an Adhesive agent* **then**
  - `/* Rule E */`
  - Calculate the random variable, γ_3, uniformly distributed on $[0, 1]$
  - **if** $\gamma_3 \leq p_{pull}$ **then**
    - Move the chosen pulling agent to lattice site $\vec{x} + \vec{dx}$
    - Move the neighboring agent to lattice site $\vec{x}$
  - **else if** $\gamma_3 \leq p_{pull} + 1 - p_{adh}$ **then**
    - Move the chosen pulling agent to lattice site $\vec{x} + \vec{dx}$

Algorithm 3: Adhesive agent migration

Randomly choose an adhesive agent and determine its lattice site index, $\vec{x} = (i, j)^T$

Choose one of the four cardinal migration directions,

$\vec{dx} = (dx, dy)^T \in \{(1, 0)^T, (-1, 0)^T, (0, 1)^T, (0, -1)^T\}$, with equal probability, 1/4. The neighboring direction is given by $\hat{dx} = -\vec{dx}$

if $\vec{x} + \vec{dx}$ *is empty* **then**

  if $\vec{x} + \hat{dx}$ *is empty* **then**

    `/* Rule C */`

    Move the chosen adhesive agent to lattice site $\vec{x} + \vec{dx}$

  else if $\vec{x} + \hat{dx}$ *is occupied by an adhesive agent* **then**

    `/* Rule D */`

    Calculate the random variable, γ_3, uniformly distributed on $[0, 1]$

    if $\gamma_3 \leq (1 - p_{adh})$ **then**

      Move the chosen adhesive agent to lattice site $\vec{x} + \vec{dx}$

  else if $\vec{x} + \hat{dx}$ *is occupied by a Pulling agent* **then**

    `/* Rule F */`

    Move the chosen adhesive agent to lattice site $\vec{x} + \vec{dx}$

D Coarse-graining ABM rules into PDE models

We will coarse-grain the Pulling, Adhesion, and Pulling & Adhesion ABMs into their mean-field PDE models. Each ABM consists of a combination of Rules A-F from Figure 1. Each rule updates the occupancies of three consecutive lattice sites, such as $\{(i, j-1), (i, j), (i, j+1)\}$. Let the variables $P_{i,j}(t)$, $H_{i,j}(t)$, and $0_{i,j}(t)$ denote the probabilities that lattice site (i, j) is occupied by a pulling agent, adhesive agent, or empty at time t, respectively. To convert each rule into a PDE model, we invoke the *mean-field assumption*, which supposes that all lattice site occupancies are independent of each other. This assumption simplifies model coarse-graining by allowing us to replace the joint probability of three lattice site occupancies with the product of the three individual lattice site occupancy probabilities. For example, under the mean-field assumption, we can write the probability that lattice sites $(i, j-1), (i, j),$ and $(i, j+1)$ are all occupied by pulling agents at time t as $P_{i,j-1}(t)P_{i,j}(t)P_{i,j+1}(t)$; otherwise, we must consider the joint occupancy probability for this triplet of lattice sites. Mean-field DE models can poorly predict ABM behavior when the mean-field assumption is violated during ABM simulations, see [8, 13, 14] for further details.

D.1 Coarse-graining the Pulling ABM

The Pulling ABM is composed of Rules A and B from Figure 1 and Section A.1. We begin coarse-graining this ABM into a PDE model by writing the master equation governing how $P_{i,j}(t)$ changes according to these rules:

$$\frac{\partial P_{i,j}(t)}{\partial t} = K^{\textit{Rule A}} + K^{\textit{Rule B.1}} + K^{\textit{Rule B.2}}. \tag{9}$$

Rule A specifies how pulling agents migrate into an empty lattice site with rate $r_m^{pull}/4$ when there is no neighboring agent in the lattice site opposite the direction of migration. This rate is divided by four because the agent randomly chooses to attempt to migrate into one of its four neighboring lattice sites. We write this rule in the master equation as:

$$\begin{aligned} K^{\textit{Rule A}} = &- \frac{2r_m^{pull}}{4}\left[0_{i,j-1}(t)P_{i,j}(t)0_{i,j+1}(t) + 0_{i-1,j}(t)P_{i,j}(t)0_{i+1,j}(t)\right] \\ &+ \frac{r_m^{pull}}{4}\left[0_{i,j-2}(t)P_{i,j-1}(t)0_{i,j}(t) + 0_{i,j}P_{i,j+1}0_{i,j+2} + 0_{i-2,j}P_{i-1,j}0_{i,j} + 0_{i,j}P_{i+1,j}0_{i+2,j}\right], \end{aligned} \tag{10}$$

where the first line describes how a pulling agent moves out of lattice site (i, j), and the second line describes

how a pulling agent moves into lattice site (i, j).

Rule B.1 specifies how a pulling agent migrates into an empty neighboring lattice site and pulls its neighbor along with it, which occurs with probability p_{pull}. We write this rule in the master equation as:

$$\begin{aligned} K^{\text{Rule B.1}} = -\frac{p_{pull} r_m^{pull}}{4} \Big[& P_{i,j}(t) P_{i,j+1}(t) 0_{i,j+2}(t) + 0_{i,j-2}(t) P_{i,j-1}(t) P_{i,j}(t) + \\ & P_{i,j}(t) P_{i+1,j}(t) 0_{i+2,j}(t) + 0_{i-2,j}(t) P_{i-1,j}(t) P_{i,j}(t) \Big] \\ \frac{p_{pull} r_m^{pull}}{4} \Big[& P_{i,j-2}(t) P_{i,j-1}(t) 0_{i,j}(t) + 0_{i,j}(t) P_{i,j+1}(t) P_{i,j+2}(t) + \\ & P_{i-2,j}(t) P_{i-1,j}(t) 0_{i,j}(t) + 0_{i,j}(t) P_{i+1,j}(t) P_{i+2,j}(t) \Big]. \end{aligned} \tag{11}$$

Rule B.2 specifies how a pulling agent migrates into an empty neighboring lattice site and fails to pull its neighbor along with it, which occurs with probability $1 - p_{pull}$. We write this rule in the master equation as:

$$\begin{aligned} K^{\text{Rule B.2}} = -\frac{(1 - p_{pull}) r_m^{pull}}{4} \Big[& P_{i,j-1}(t) P_{i,j}(t) 0_{i,j+1}(t) + 0_{i,j-1}(t) P_{i,j}(t) P_{i,j+1}(t) + \\ & P_{i-1,j}(t) P_{i,j}(t) 0_{i+1,j+1}(t) + 0_{i,j-1}(t) P_{i,j}(t) P_{i+1,j}(t) \Big] \\ + \frac{(1 - p_{pull}) r_m^{pull}}{4} \Big[& P_{i,j-2}(t) P_{i,j-1}(t) 0_{i,j}(t) + 0_{i,j}(t) P_{i,j+1}(t) P_{i,j+2}(t) + \\ & P_{i-2,j}(t) P_{i-1,j}(t) 0_{i,j}(t) + 0_{i,j}(t) P_{i+1,j}(t) P_{i+2,j}(t) \Big]. \end{aligned} \tag{12}$$

To obtain the resulting PDE model for the Pulling ABM, we substitute Equations (10), (11), and (12) into Equation (9) and set $0_{i,j} = 1 - P_{i,j}$. We replace each term with its Taylor expansion, up to second order:

$$\begin{aligned} P_{i \pm m, j}(t) &= P_{i,j}(t) \pm m\Delta (P_{i,j}(t))_x + \frac{m\Delta^2}{2} (P_{i,j}(t))_{xx} + \mathcal{O}(\Delta^3), && m = -2, -1, 0, 1, 2; \\ P_{i, j \pm n}(t) &= P_{i,j}(t) \pm n\Delta (P_{i,j}(t))_y + \frac{n\Delta^2}{2} (P_{i,j}(t))_{yy} + \mathcal{O}(\Delta^3), && n = -2, -1, 0, 1, 2; \end{aligned} \tag{13}$$

where subscripts denote differentiation with respect the the shown variable, and Δ is the length of each lattice site. As shown in the Mathematica notebook **Pulling_model_coarse_graining.nb**, taking the limit of the resulting expression as $\Delta \to 0$ leads to the mean-field PDE model for the Pulling ABM:

$$\frac{\partial P}{\partial t} = \nabla \cdot \left(\frac{r_m^{pull}}{4} \left(1 + 3 p_{pull} P^2 \right) \nabla P \right), \tag{14}$$

where $P = P_{i,j}(t)$.

D.2 Coarse-graining the Adhesion ABM

The Adhesion ABM is composed of Rules C and D from Figure 1 and Section A.2. We begin coarse-graining this ABM into a PDE model by writing the master equation governing how $H_{i,j}(t)$ changes according to these rules:

$$\frac{\partial H_{i,j}(t)}{\partial t} = K^{\textit{Rule C}} + K^{\textit{Rule D}}. \tag{15}$$

Rule C specifies how adhesive agents migrate into an empty lattice site with rate $r_m^{adh}/4$ when there is no neighboring agent in the lattice site opposite the direction of migration. We write this rule in the master equation as:

$$\begin{aligned} K^{\textit{Rule C}} = &-\frac{2r_m^{adh}}{4}\Big[0_{i,j-1}(t)H_{i,j}(t)0_{i,j+1}(t) + 0_{i-1,j}(t)H_{i,j}(t)0_{i+1,j}(t)\Big] \\ &+\frac{r_m^{adh}}{4}\Big[0_{i,j-2}(t)H_{i,j-1}(t)0_{i,j}(t) + 0_{i,j}(t)H_{i,j+1}(t)0_{i,j+2}(t) + \\ &\qquad 0_{i-2,j}(t)H_{i-1,j}(t)0_{i,j}(t) + 0_{i,j}(t)H_{i+1,j}(t)0_{i+2,j}(t)\Big], \end{aligned} \tag{16}$$

where the first line describes how an adhesive agent moves out of lattice site (i,j), and the second and third lines describe how an adhesive agent moves into lattice site (i,j).

Rule D specifies how adhesive agents migrate into an empty neighboring lattice site when a neighboring adhesive agent is in the lattice site opposite the direction of migration. The neighboring adhesive agent attempts to adhere to the migrating agent and abort the migration event. The adhesion event succeeds with probability p_{adh}, and neither agent changes its position. The adhesion event fails with probability $1-p_{adh}$, and the migrating agent shifts into the previously-empty lattice site while the neighboring agent remains in its previous lattice site. We write this rule in the master equation as:

$$\begin{aligned} K^{\textit{Rule D}} = &-\frac{(1-p_{adh})r_m^{adh}}{4}\Big[H_{i,j-1}(t)H_{i,j}(t)0_{i,j+1}(t) + 0_{i,j-1}(t)H_{i,j}(t)H_{i,j+1}(t) + \\ &\qquad H_{i-1,j}(t)H_{i,j}(t)0_{i+1,j}(t) + 0_{i-1,j}(t)H_{i,j}(t)H_{i+1,j}(t)\Big] \\ &+\frac{(1-p_{adh})r_m^{adh}}{4}\Big[H_{i,j-2}(t)H_{i,j-1}(t)0_{i,j}(t) + 0_{i,j}(t)H_{i,j+1}(t)H_{i,j+2}(t) + \\ &\qquad H_{i-2,j}(t)H_{i-1,j}(t)0_{i,j}(t) + 0_{i,j}(t)H_{i+1,j}(t)H_{i+2,j}(t)\Big]. \end{aligned} \tag{17}$$

To obtain the resulting PDE model for the Adhesion ABM, we substitute Equations (16) and (17) into

Equation (15) and set $0_{i,j} = 1 - H_{i,j}$. We replace each term with its Taylor expansion, up to second order:

$$H_{i\pm m,j}(t) = H_{i,j}(t) \pm m\Delta(H_{i,j}(t))_x + \frac{m\Delta^2}{2}(H_{i,j}(t))_{xx} + \mathcal{O}(\Delta^3), \qquad m = -2,-1,0,1,2;$$
$$H_{i,j\pm n}(t) = H_{i,j}(t) \pm n\Delta(H_{i,j}(t))_y + \frac{n\Delta^2}{2}(H_{i,j}(t))_{yy} + \mathcal{O}(\Delta^3), \qquad n = -2,-1,0,1,2. \tag{18}$$

As shown in the Mathematica notebook **Adhesion_model_coarse_graining.nb**, taking the limit of the resulting expression as $\Delta \to 0$ leads to the mean-field PDE model for the Adhesion ABM:

$$\frac{\partial H}{\partial t} = \nabla \cdot \left(\frac{r_m^{adh}}{4} \left(3p_{adh} \left(H - \frac{2}{3} \right)^2 + 1 - \frac{4p_{adh}}{3} \right) \nabla H \right) \tag{19}$$

where $H = H_{i,j}(t)$.

D.3 Coarse-graining the Pulling & Adhesion ABM

The Pulling & Adhesion ABM is composed of Rules A to F from Figure 1 and Sections A.1-A.3. We begin coarse-graining this ABM into a PDE model by writing the master system of equations governing how both $P_{i,j}(t)$ and $H_{i,j}(t)$ change according to these rules:

$$\frac{\partial P_{i,j}(t)}{\partial t} = K^{Rule\ A} + K^{Rule\ B.1} + K^{Rule\ B.2} + K_P^{Rule\ E.1} + K^{Rule\ E.2} \tag{20}$$
$$\frac{\partial H_{i,j}(t)}{\partial t} = K^{Rule\ C} + K^{Rule\ D} + K_H^{Rule\ E.1} + K^{Rule\ F}, \tag{21}$$

where $K_P^{Rule\ E.1}$ denotes how $P_{i,j}(t)$ is affected by Rule E.1 and $K_H^{Rule\ E.1}$ denotes how $H_{i,j}(t)$ is affected by Rule E.1. All other rules affect either $P_{i,j}(t)$ or $H_{i,j}(t)$, but not both. Rules A-D are described in Sections D.1 and D.2, and we do not restate them here.

Rule E specifies how a pulling agent migrates into an empty neighboring lattice site when a neighboring adhesive agent is present in the lattice site opposite the direction of migration. In Rule E.1, the pulling agent successfully pulls the adhesive agent as it migrates, which occurs with probability p_{pull}. In this scenario, the pulling agent shifts into the previously-empty lattice site and the adhesive agent moves into the site

previously occupied by the pulling agent. We write this rule in the master equation for $P_{i,j}(t)$ as:

$$K_P^{\text{Rule E.1}} = -\frac{p_{pull} r_m^{pull}}{4}\Big[H_{i,j-1}(t)P_{i,j}(t)0_{i,j+1}(t) + 0_{i,j-1}(t)P_{i,j}(t)H_{i,j+1}(t) + H_{i-1,j}(t)P_{i,j}(t)0_{i+1,j}(t) + 0_{i-1,j}(t)P_{i,j}(t)H_{i+1,j}(t)\Big] + \frac{p_{pull} r_m^{pull}}{4}\Big[H_{i,j-2}(t)P_{i,j-1}(t)0_{i,j}(t) + 0_{i,j}(t)P_{i,j+1}(t)H_{i,j+2}(t) + H_{i-2,j}(t)P_{i-1,j}(t)0_{i,j}(t) + 0_{i,j}(t)P_{i+1,j}(t)H_{i+2,j}(t)\Big], \tag{22}$$

and in the master equation for $H_{i,j}(t)$ as:

$$K_H^{\text{Rule E.1}} = -\frac{p_{pull} r_m^{pull}}{4}\Big[0_{i,j-2}(t)P_{i,j-1}(t)H_{i,j}(t) + H_{i,j}(t)P_{i,j+1}(t)0_{i,j+2}(t) + 0_{i-2,j}(t)P_{i-1,j}(t)H_{i,j}(t) + H_{i,j}(t)P_{i+1,j}(t)0_{i+2,j}(t)\Big] + \frac{p_{pull} r_m^{pull}}{4}\Big[H_{i,j-1}(t)P_{i,j}(t)0_{i,j+1}(t) + 0_{i,j-1}(t)P_{i,j}(t)H_{i,j+1}(t) + H_{i-1,j}(t)P_{i,j}(t)0_{i+1,j}(t) + 0_{i-1,j}(t)P_{i,j}(t)H_{i+1,j}(t)\Big]. \tag{23}$$

The neighboring adhesive agent successfully adheres to the migrating pulling agent and aborts its migration event with probability p_{adh}. Neither $P_{i,j}(t)$ or $H_{i,j}(t)$ changes in this scenario as no agents change their locations in response to the adhesion event. In Rule E.2, the adhesive agent fails to adhere to the pulling agent and the pulling agent fails to pull the adhesive agent, which occurs with probability $1 - p_{adh} - p_{pull}$. In this scenario, the pulling agent shifts into the previously-empty lattice site while the neighboring adhesive agent remains in its previous lattice site. We write this rule in the master equation as:

$$K^{\text{Rule E.2}} = -\frac{(1 - p_{adh} - p_{pull}) r_m^{pull}}{4}\Big[H_{i,j-1}(t)P_{i,j}(t)0_{i,j+1}(t) + 0_{i,j-1}(t)P_{i,j}(t)H_{i,j+1}(t) + H_{i-1,j}(t)P_{i,j}(t)0_{i+1,j}(t) + +0_{i-1,j}(t)P_{i,j}(t)H_{i+1,j}(t)\Big] + \frac{(1 - p_{adh} - p_{pull}) r_m^{pull}}{4}\Big[H_{i,j-2}(t)P_{i,j-1}(t)0_{i,j}(t) + 0_{i,j}(t)P_{i,j+1}(t)H_{i,j+2}(t) + H_{i-2,j}(t)P_{i-1,j}(t)0_{i,j}(t) + 0_{i,j}(t)P_{i+1,j}(t)H_{i+2,j}(t)\Big]. \tag{24}$$

Rule F specifies how adhesive agents migrate into an empty neighboring lattice site when a neighboring pulling agent is in the lattice site opposite the direction of migration. The two agents do not interact with each other in this scenario. As such, the adhesive agent migrates into the empty lattice site with rate $r_m^{adh}/4$.

We write this rule in the master equation as:

$$
\begin{aligned}
K^{\text{Rule F}} = -\frac{r_m^{adh}}{4}\Big[&P_{i,j-1}(t)H_{i,j}(t)0_{i,j+1}(t) + 0_{i,j-1}(t)H_{i,j}(t)P_{i,j+1}(t) + \\
&P_{i-1,j}(t)H_{i,j}(t)0_{i+1,j}(t) + 0_{i-1,j}(t)H_{i,j}(t)P_{i+1,j}(t)\Big] \\
+\frac{r_m^{adh}}{4}\Big[&P_{i,j-2}(t)H_{i,j-1}(t)0_{i,j}(t) + 0_{i,j}(t)H_{i,j+1}(t)P_{i,j+2}(t) + \\
&P_{i-2,j}(t)H_{i-1,j}(t)0_{i,j}(t) + 0_{i,j}(t)H_{i+1,j}(t)P_{i+2,j}(t)\Big].
\end{aligned} \tag{25}
$$

To obtain the resulting system of differential equations for the Pulling & Adhesion ABM, we substitute Equations (10), (11), (12), (16), (17), (22), (23), (24), and (25) into Equation (21) and set $0_{i,j} = 1 - T_{i,j}$, where $T_{i,j} = P_{i,j} + H_{i,j}$. We replace each term with its Taylor expansion, up to second order, from Equations (13) and (18). As shown in the Mathematica notebook **Pulling-Adhesion_coarse_graining.nb**, taking the limit of the resulting expression as $\Delta \to 0$ leads to the mean-field system of PDEs for the Pulling & Adhesion ABM:

$$
\begin{aligned}
\frac{\partial P}{\partial t} =& \frac{r_m^{pull}}{4}\nabla\cdot\Big((1-T)\nabla P + P\nabla T\Big) \\
&+ p_{adh}\frac{r_m^{pull}}{4}\nabla\cdot\Big(-3P(1-T)\nabla H - H(1-T)\nabla P - HP\nabla T\Big) \\
&+ p_{pull}\frac{r_m^{pull}}{4}\nabla\cdot\Big(3P^2\nabla T\Big) \\
\frac{\partial H}{\partial t} =& \frac{r_m^{adh}}{4}\nabla\cdot\Big((1-T)\nabla H + H\nabla T\Big) \\
&+ p_{adh}\frac{r_m^{adh}}{4}\nabla\cdot\Big(-4(1-T)H\nabla H - H^2\nabla T\Big) \\
&+ p_{pull}\frac{r_m^{pull}}{4}\nabla\cdot\Big(-(1-T)H\nabla P + (1-T)P\nabla H + 3HP\nabla T\Big),
\end{aligned} \tag{26}
$$

where $P = P_{i,j}(t), H = H_{i,j}(t),$ and $T = T_{i,j}(t)$.

E BINN implementation and training

E.1 BINNs architecture

Following [28], we construct $T^{MLP}(x,t)$ using a fully-connected feed-forward MLP with three hidden layers, which can be written as:

$$\begin{aligned} z_0 &= [x,t] \\ z_1 &= \sigma\left(z_0 W_1 + b_1\right) \\ z_2 &= \sigma\left(z_1 W_2 + b_2\right) \\ z_3 &= \sigma\left(z_2 W_3 + b_3\right) \\ T^{MLP}(x,t) &= \psi\left(z_3 W_4 + b_4\right), \end{aligned} \tag{27}$$

where each z_k denotes the k^{th} hidden layer for $k = 1,2,3$; the W_k matrices and the b_k vectors provide the weights and biases of each hidden layer, respectively; σ denotes the sigmoid activation function $\sigma(x) = 1/(1+\exp(-x))$, and ψ denotes the softplus activation function $\psi(x) = \log(1+\exp(x))$. Each hidden layer in Equation (27) has 128 neurons, meaning that $W_1 \in \mathbb{R}^{2\times 128}; W_2, W_3 \in \mathbb{R}^{128\times 128}; W_4 \in \mathbb{R}^{128\times 1}; b_1, b_2, b_3 \in \mathbb{R}^{128}$; and $b_4 \in \mathbb{R}$.

The architecture of $\mathcal{D}^{MLP}(T)$ is identical to the architecture for T^{MLP} in Equation (27), except $\mathcal{D}^{MLP}$ has a one-dimensional input vector, T, instead of the two-dimensional input vector, $[x,t]$.

E.2 Loss Function

BINNs are trained to concurrently fit the given dataset, $\langle T^{ABM}(x,t)\rangle^{train}$, and solve Equation (7) by minimizing the following multi-term loss function:

$$\mathcal{L}_{total} = \mathcal{L}_{WLS} + \epsilon \mathcal{L}_{PDE} + \mathcal{L}_{constr}. \tag{28}$$

The ϵ parameter ensures the terms $\mathcal{L}_{WLS}$ and $\mathcal{L}_{PDE}$ are equally weighted because these terms can be of different orders of magnitude; we find good results for $\epsilon = 10^4$.

The $\mathcal{L}_{WLS}$ term of Equation (28) computes a weighted mean-squared error between $T^{MLP}(x,t)$ and

$\langle T^{ABM}(x,t)\rangle^{train}$:

$$\mathcal{L}_{WLS} = \frac{1}{T_f^{train} X} \sum_{i=1,j=1}^{X,T_f^{train}} w_{i,j} \left(T^{MLP}(x_i,t_j) - \langle T^{ABM}(x_i,t_j)\rangle \right)^2. \tag{29}$$

We set $w_{i,1} = 10.0$ for all values of i and all other $w_{i,j}$ values to 1.0 to ensure that T^{MLP} closely agrees with the ABM's initial data. By minimizing Equation (29), we ensure $T^{MLP}(x,t)$ closely approximates $\langle T^{ABM}(x,t)\rangle^{train}$.

The $\mathcal{L}_{PDE}$ term of Equation (28) quantifies how closely T^{MLP} and $\mathcal{D}^{MLP}$ follow Equation (7). To ensure the MLPs satisfy this PDE framework throughout the ABM's entire spatiotemporal domain, we uniformly sample 10,000 points, $\{(x_k,t_k)\}_{k=1}^{10,000}$, from $[0,X]\times[0,750]$. For notational convenience, let $\hat{T}_k = T^{MLP}(x_k,t_k)$ and $\hat{D}_k = \mathcal{D}^{MLP}\big(T^{MLP}(x_k,t_k)\big)$. We then compute the mean-squared error between the left- and right-hand sides of Equation (7) at all sampled points:

$$\mathcal{L}_{PDE} = \frac{1}{10,000} \sum_{i=1}^{10,000} \left[\frac{\partial}{\partial t}\hat{T}_k - \frac{\partial}{\partial x}\left(\hat{D}_k \frac{\partial}{\partial x}\hat{T}_k \right) \right]^2, \tag{30}$$

where differentiation of T^{MLP} and $\mathcal{D}^{MLP}$ is performed using automatic differentiation. Minimizing Equation (30) verifies that T^{MLP} and $\mathcal{D}^{MLP}$ together satisfy Equation (7).

The $\mathcal{L}_{constr}$ term of Equation (28) incorporates user knowledge into BINNs training. We penalize $\mathcal{D}^{MLP}$ for outputting values outside of the interval $[D_{\min}, D_{\max}]$. We set $D_{\min} = 0$ because Equation (7) is ill-posed if $\mathcal{D}(u) < 0$, and we set $D_{\max} = 1.0$ because the mean-field rates of diffusion are below one for all ABM simulations in this study. We compute this term by squaring any values of $\hat{D}_i$ that are not within $[D_{\min}, D_{\max}]$ and weighting these values by 10^{10}:

$$\mathcal{L}_{constr} = \frac{1}{10,000} \sum_{\substack{k=1 \\ \hat{D}_k \notin [D_{\min}, D_{\max}]}}^{10,000} 10^{10} (\hat{D}_k)^2. \tag{31}$$

This term regularizes the BINN training procedure to prevent $\mathcal{D}^{MLP}$ from outputting unrealistic values.

E.3 BINN Training Procedure

For BINN model training, we randomly partition the training ABM dataset into 80%/20% BINN training and BINN validation datasets. We train the BINN parameter values (i.e., the weights and biases for T^{MLP} and $\mathcal{D}^{MLP}$) to minimize a loss function, $\mathcal{L}$, using the gradient-based ADAM optimizer with its default

hyperparameter values on the BINN training dataset. For each new set of BINN parameters, we compute $\mathcal{L}$ on the BINN validation dataset and save the BINN parameters if the newly computed $\mathcal{L}$ value achieves a 1% or greater relative improvement over the previous smallest recorded value. Following [36], we perform training in a two-step process: in the first step, we train the BINN to match the ABM data by optimizing $\mathcal{L} = \mathcal{L}_{WLS}$ from Equation (29); in the second step, we train the BINN on $\mathcal{L} = \mathcal{L}_{total}$ from Equation (28). The first training step is performed for 10^4 epochs with an early stopping criterion of 10^3, meaning that training ends early if the smallest-computed $\mathcal{L}$ value on the validation data is unchanged for 10^3 epochs. The second step is performed for 10^6 epochs with an early stopping criterion of 10^5. Each epoch is computed in minibatches of size 10^3. BINN model training is performed using the PyTorch deep learning library (version 1.7.1).

Following [28], we train five separate BINNs for each ABM dataset using different BINN training and validation datasets because the final trained model can be sensitive to which data is included in these two datasets. We compute the five PDE forward simulations from these trained models and select whichever BINN achieves the smallest mean-squared error against the ABM training data as the final selected BINN model.

E.4 Comments on BINN training convergence

We depict the chosen hyperparameter values for BINN model training in Table 6. Many of these values were chosen to follow previous modeling studies [28, 36]. A current challenge in neural network training is determining the optimal choice of such hyperparameter values [35]. In our work, we found that BINN model training is most sensitive to the ϵ parameter as well as the number of epochs and early stopping number used during BINN model training (results not shown). If ϵ is too small, then the BINN will prioritize fitting the ABM data but not satisfying the PDE framework. Conversely, if ϵ is too large, then the BINN will ensure it satisfies a PDE framework while neglecting the data. We found a good balance between the two loss functions for $\epsilon = 1 \times 10^{-4}$. Training the BINN with a smaller number of epochs, such as 10^5 with an early stopping criterion of 10^4 led to a model that had not fully converged to the data and we found better convergence using 10^6 epochs with an early stopping criterion of 10^5.

Hyperparameter description	Value
Number of hidden layers	3
Number of neurons per hidden layer	128
Weighting between $\mathcal{L}_{WLS}$ and $\mathcal{L}_{PDE}$ (ϵ)	10^{-4}
Additional initial condition weighting in $\mathcal{L}_{WLS}$	10.0
Number of collocation points for $\mathcal{L}_{PDE}$	10,000
Penalty for $\mathcal{D}^{MLP}$ values outside of $[D_{min}, D_{max}]$	10^{10}
D_{min}	0.0
D_{max}	1.0
ANN epochs	10^4
ANN early stopping	10^3
BINN epochs	10^6
BINN early stopping	10^5

Table 6: Hyperparameter values used to perform BINN model training.

F Numerical integration of PDEs

When simulating Equation (2), we populate the middle 20% of the spatial dimension with 75% confluence and zero confluence everywhere else to match the initial ABM configurations and implement no-flux boundary conditions:

$$T(x,0) = \begin{cases} 0.75, & 80 \leq x \leq 120 \\ 0, & \text{otherwise,} \end{cases},$$
$$\frac{\partial T}{\partial x}(0,t) = \frac{\partial u}{\partial x}(X,t) = 0. \tag{32}$$

Before integration, we discretize the spatial domain as $x_i = i\Delta x$ with $i = 0, ..., 199$ and $\Delta x = 1.0$. For ease of notation, let $T_i(t) = T(x_i, t)$ and $\mathcal{D}_i(t) = \mathcal{D}(T_i(t))$. We then use the method of lines approach to integrate Equation (2). To discretize the right hand side of Equation (7), we let

$$\frac{\partial T_i(t)}{\partial x}\left(\mathcal{D}_i(t)\frac{\partial T_i(t)}{\partial x}\right) \approx \frac{P_{i+1/2}(t) - P_{i-1/2}(t)}{\Delta x},$$

where $P_{i\pm 1/2}(t)$ denotes the right or left flux through location x_i, respectively. Following [49], we approximate these fluxes by

$$P_{i+1/2}(t) = \frac{1}{2}\left(\mathcal{D}_i(t)\frac{T_{i+1}(t) - T_i(t)}{\Delta x} + \mathcal{D}_{i+1}(t)\frac{T_{i+1}(t) - T_i(t)}{\Delta x}\right)$$
$$P_{i-1/2}(t) = \frac{1}{2}\left(\mathcal{D}_{i-1}(t)\frac{T_i(t) - T_{i-1}(t)}{\Delta x} + \mathcal{D}_i(t)\frac{T_i(t) - T_{i-1}(t)}{\Delta x}\right). \tag{33}$$

To implement the no-flux boundary conditions, we incorporate the ghost points x_{-1} and x_{200} that enforce $u_{-1}(t) = u_1(t)$ and $u_{198}(t) = u_{200}(t)$ into Equation (33). We integrate Equation (2) using the `odeint` command from Scipy's integration package (version 1.8.0), which implements the Livermore Solver for Differential Equations (LSODA) method [50].

G Supplementary figures

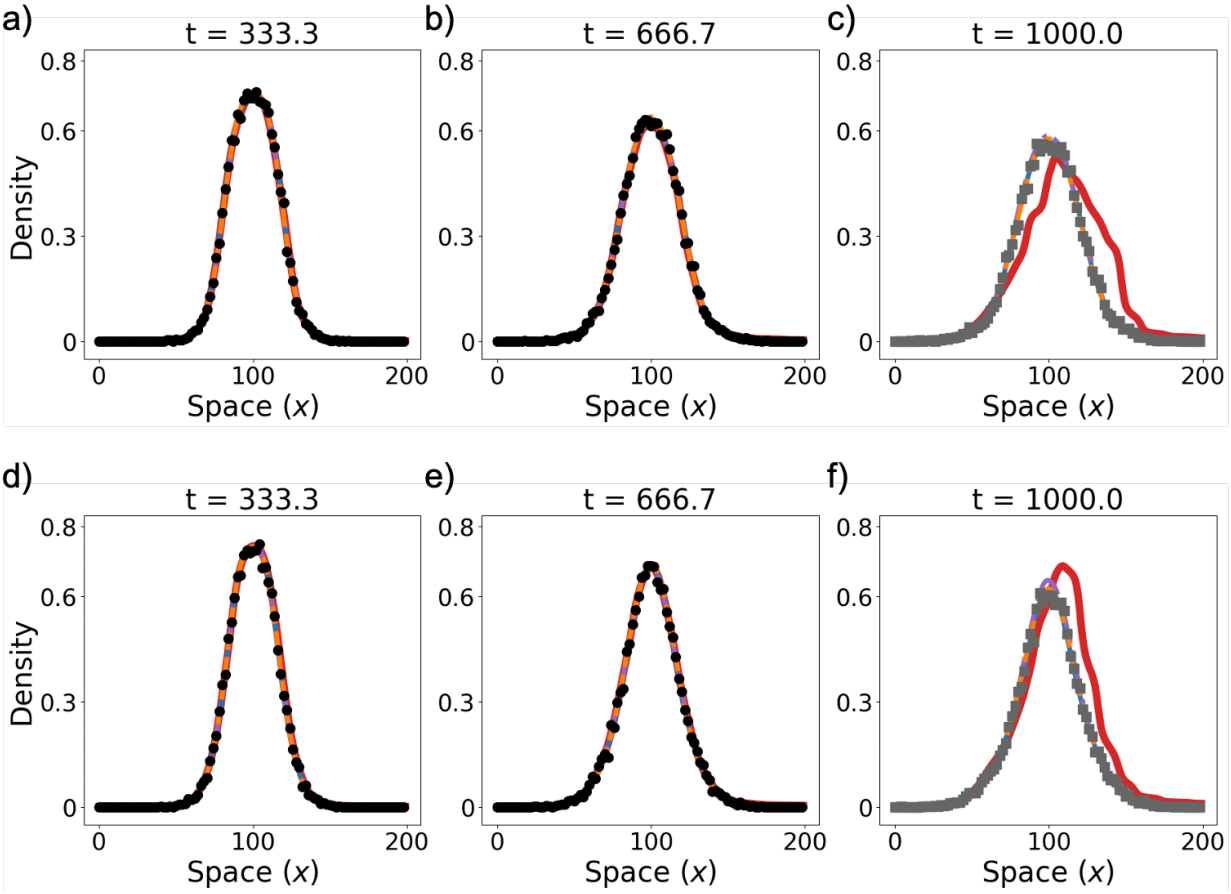

Figure 12: Forecasting ABM data with neural networks and PDEs. ANN and BINN models were trained to fit $\langle T^{ABM}(x,t)\rangle^{train}$. These two neural networks and the mean-field and BINN-guided PDE simulations were then used to forecast $\langle T^{ABM}(x,t)\rangle^{train}$ and $\langle T^{ABM}(x,t)\rangle^{test}$. This was performed for (a-c) the Adhesion ABM with $\boldsymbol{p} = (r_m^{adh}, p_{adh})^T = (1.0, 0.5)^T$ and (d-f) the Pulling & Adhesion ABM with $\boldsymbol{p} = (r_m^{pull}, r_m^{adh}, p_{pull}, p_{adh}, \alpha)^T = (1.0, 0.25, 0.33, 0.33, 0.5)^T$.

Sample	$\boldsymbol{p} = (r_m^{adh},\ p_{adh})^T$
1	$(0.145,\ 0.825)^T$
2	$(0.505,\ 0.575)^T$
3	$(0.415,\ 0.725)^T$
4	$(0.865,\ 0.525)^T$
5	$(0.955,\ 0.625)^T$
6	$(0.235,\ 0.775)^T$
7	$(0.685,\ 0.675)^T$
8	$(0.325,\ 0.875)^T$
9	$(0.775,\ 0.925)^T$
10	$(0.595,\ 0.975)^T$

Table 7: Latin hypercube sampling for the Adhesion ABM. The samples from the new parameter dataset for the Adhesion ABM when varying r_m^{adh} and p_{adh}. The samples are ordered by increasing testing MSE values (see Figure 10(c)).

Sample	$\boldsymbol{p} = (r_m^{pull},\ r_m^{adh},\ p_{pull},\ p_{adh},\ \alpha)^T$
1	$(1.0, 0.25, 0.394, 0.578, 0.912)^T$
2	$(1.0, 0.25, 0.293, 0.528, 0.938)^T$
3	$(1.0, 0.25, 0.008, 0.226, 0.988)^T$
4	$(1.0, 0.25, 0.511, 0.477, 0.862)^T$
5	$(1.0, 0.25, 0.41, 0.109, 0.962)^T$
6	$(1.0, 0.25, 0.075, 0.595, 0.888)^T$
7	$(1.0, 0.25, 0.042, 0.544, 0.838)^T$
8	$(1.0, 0.25, 0.327, 0.059, 0.712)^T$
9	$(1.0, 0.25, 0.444, 0.31, 0.662)^T$
10	$(1.0, 0.25, 0.209, 0.209, 0.612)^T$
11	$(1.0, 0.25, 0.126, 0.41, 0.762)^T$
12	$(1.0, 0.25, 0.193, 0.042, 0.588)^T$
13	$(1.0, 0.25, 0.059, 0.561, 0.462)^T$
14	$(1.0, 0.25, 0.243, 0.26, 0.788)^T$
15	$(1.0, 0.25, 0.427, 0.494, 0.512)^T$
16	$(1.0, 0.25, 0.595, 0.327, 0.812)^T$
17	$(1.0, 0.25, 0.025, 0.461, 0.388)^T$
18	$(1.0, 0.25, 0.377, 0.176, 0.488)^T$
19	$(1.0, 0.25, 0.226, 0.645, 0.538)^T$
20	$(1.0, 0.25, 0.528, 0.126, 0.688)^T$
21	$(1.0, 0.25, 0.561, 0.075, 0.562)^T$
22	$(1.0, 0.25, 0.142, 0.193, 0.362)^T$
23	$(1.0, 0.25, 0.31, 0.092, 0.738)^T$
24	$(1.0, 0.25, 0.176, 0.662, 0.412)^T$
25	$(1.0, 0.25, 0.645, 0.008, 0.638)^T$
26	$(1.0, 0.25, 0.343, 0.293, 0.312)^T$
27	$(1.0, 0.25, 0.092, 0.611, 0.238)^T$
28	$(1.0, 0.25, 0.109, 0.628, 0.012)^T$
29	$(1.0, 0.25, 0.159, 0.343, 0.212)^T$
30	$(1.0, 0.25, 0.26, 0.142, 0.188)^T$
31	$(1.0, 0.25, 0.36, 0.377, 0.262)^T$
32	$(1.0, 0.25, 0.276, 0.36, 0.038)^T$
33	$(1.0, 0.25, 0.578, 0.243, 0.288)^T$
34	$(1.0, 0.25, 0.628, 0.159, 0.062)^T$
35	$(1.0, 0.25, 0.477, 0.511, 0.138)^T$
36	$(1.0, 0.25, 0.611, 0.276, 0.338)^T$

Sample	$\boldsymbol{p} = (r_m^{pull},\ r_m^{adh},\ p_{pull},\ p_{adh},\ \alpha)^T$
1	$(1.0,\ 0.25,\ 0.285,\ 0.519,\ 0.775)^T$
2	$(1.0,\ 0.25,\ 0.419,\ 0.352,\ 0.875)^T$
3	$(1.0,\ 0.25,\ 0.486,\ 0.117,\ 0.525)^T$
4	$(1.0,\ 0.25,\ 0.553,\ 0.285,\ 0.375)^T$
5	$(1.0,\ 0.25,\ 0.385,\ 0.586,\ 0.475)^T$
6	$(1.0,\ 0.25,\ 0.586,\ 0.184,\ 0.175)^T$
7	$(1.0,\ 0.25,\ 0.62,\ 0.151,\ 0.325)^T$
8	$(1.0,\ 0.25,\ 0.184,\ 0.084,\ 0.625)^T$
9	$(1.0,\ 0.25,\ 0.352,\ 0.385,\ 0.925)^T$
10	$(1.0,\ 0.25,\ 0.653,\ 0.05,\ 0.275)^T$
11	$(1.0,\ 0.25,\ 0.151,\ 0.653,\ 0.075)^T$
12	$(1.0,\ 0.25,\ 0.452,\ 0.251,\ 0.125)^T$
13	$(1.0,\ 0.25,\ 0.084,\ 0.218,\ 0.225)^T$
14	$(1.0,\ 0.25,\ 0.318,\ 0.62,\ 0.725)^T$
15	$(1.0,\ 0.25,\ 0.519,\ 0.017,\ 0.825)^T$
16	$(1.0,\ 0.25,\ 0.117,\ 0.419,\ 0.425)^T$
17	$(1.0,\ 0.25,\ 0.251,\ 0.486,\ 0.975)^T$
18	$(1.0,\ 0.25,\ 0.017,\ 0.452,\ 0.025)^T$
19	$(1.0,\ 0.25,\ 0.05,\ 0.318,\ 0.575)^T$
20	$(1.0,\ 0.25,\ 0.218,\ 0.553,\ 0.675)^T$

Table 9: Latin hypercube sampling for the Pulling & Adhesion ABM. The samples from the new parameter dataset for the Pulling & Adhesion ABM when varying p_{pull}, p_{adh}, and α. The samples are ordered by increasing training MSE values.

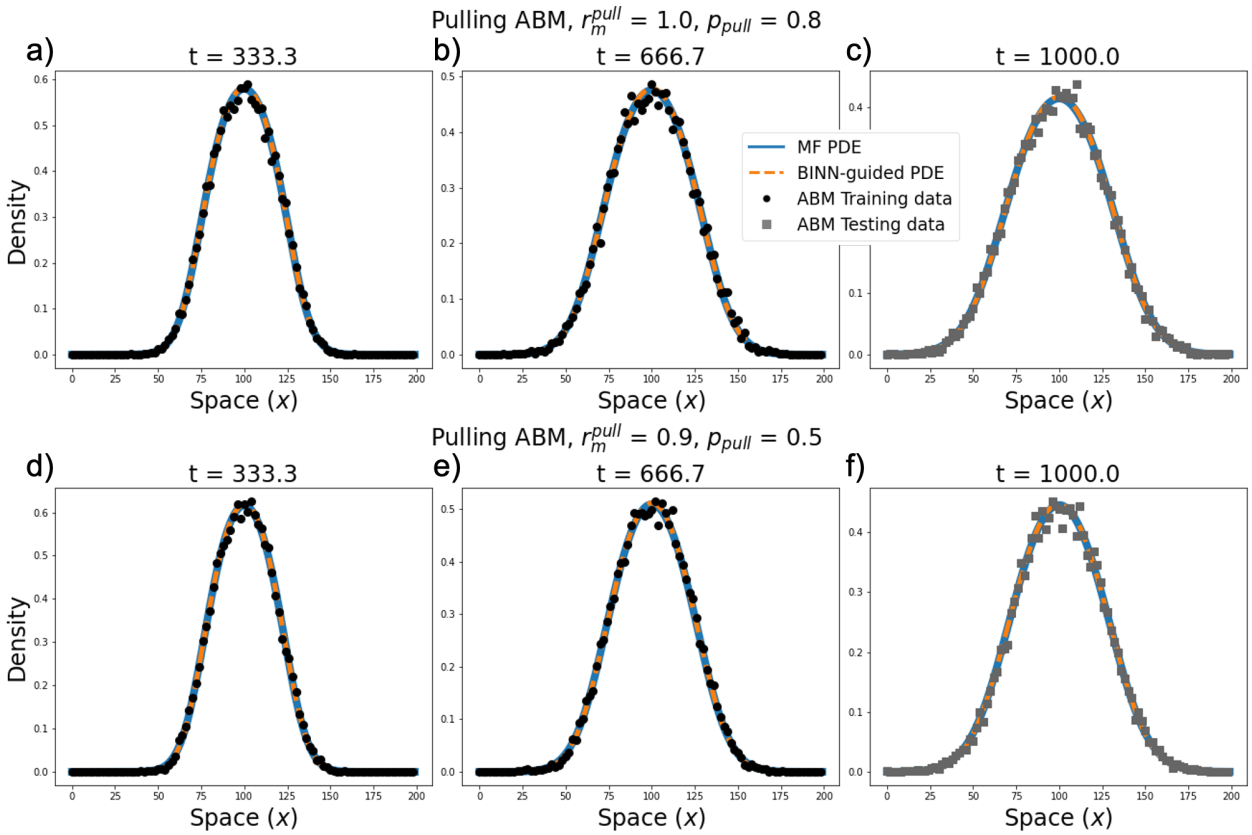

Figure 13: Forecasting Pulling ABM data with mean-field (MF) and BINN-guided PDE models. The mean-field and BINN-guided PDE simulations are used to forecast Pulling ABM data for (a-c) $r_m^{pull} = 1.0, p_{pull} = 0.8$ (d-f) $r_m^{pull} = 0.9, p_{pull} = 0.5$.

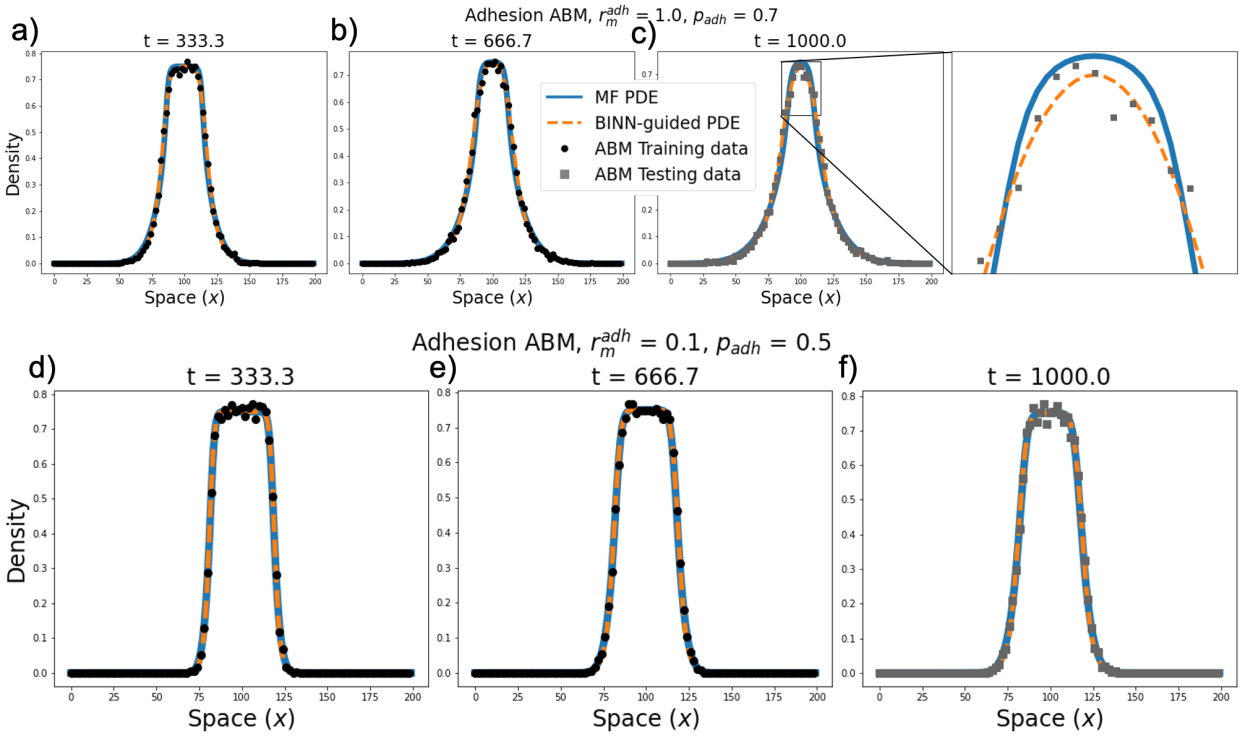

Figure 14: Forecasting Adhesion ABM data with mean-field and BINN-guided PDE models. The mean-field and BINN-guided PDE simulations are used to forecast Adhesion ABM data for (a-c) $r_m^{adh} = 1.0, p_{adh} = 0.7$ (d-f) $r_m^{adh} = 0.1, p_{adh} = 0.5$.

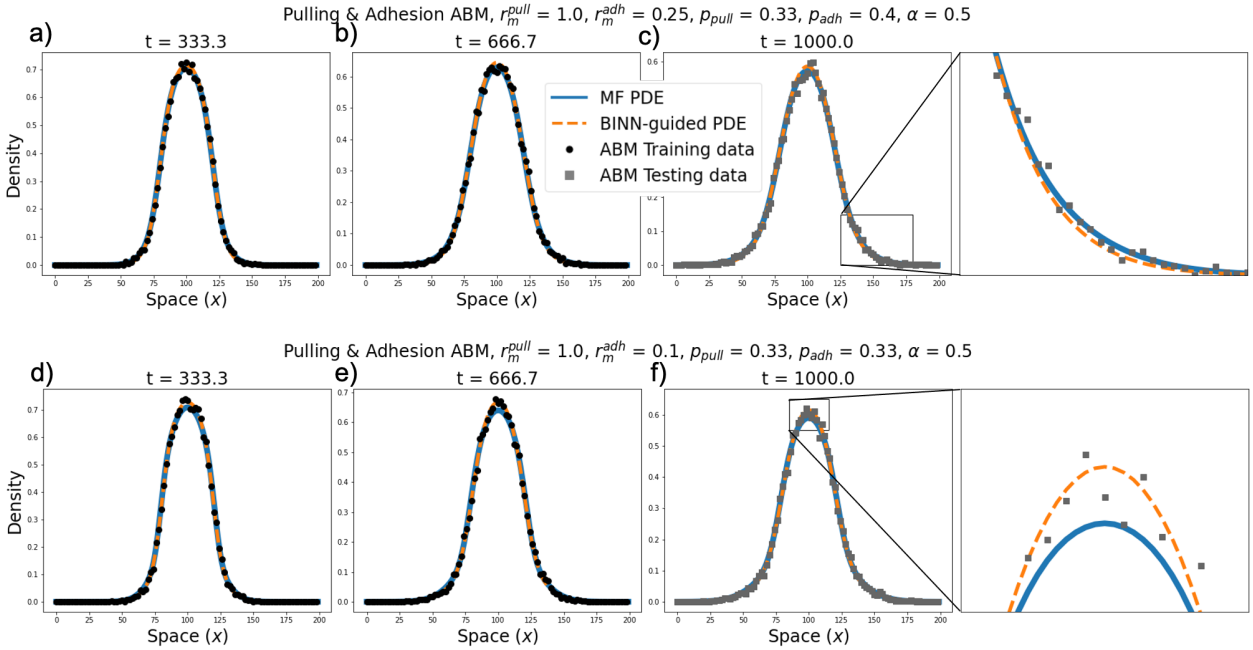

Figure 15: Forecasting Pulling & Adhesion ABM data with mean-field (MF) and BINN-guided PDE models. The mean-field and BINN-guided PDE simulations are used to forecast Pulling & Adhesion ABM data for the base parameter values ($r_m^{pull} = 1.0, r_m^{adh} = 0.25, p_{pull} = 0.33, p_{adh} = 0.33$, and $\alpha = 0.5$), except (a-c) $p_{adh} = 0.4$ (d-f) $r_m^{adh} = 0.1$.

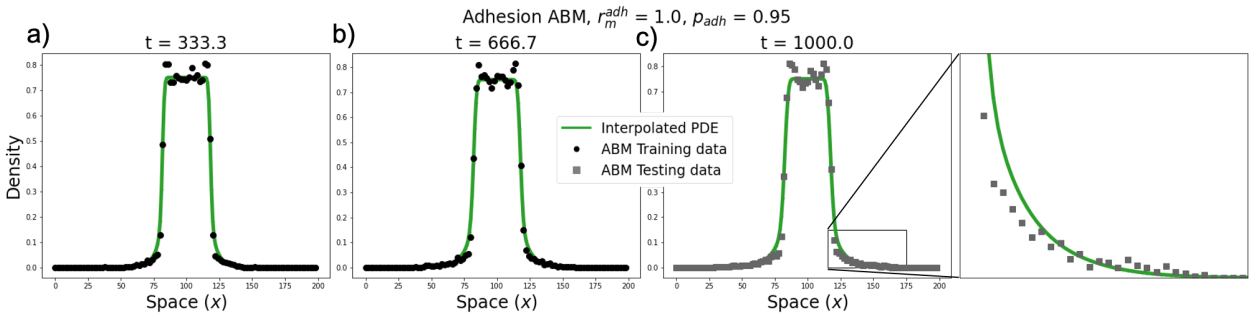

Figure 16: Predicting Adhesion ABM data with the interpolated PDE model. The interpolated PDE model predicts Adhesion ABM data for (a-c) $r_m^{adh} = 1.0$ and $p_{adh} = 0.95$.

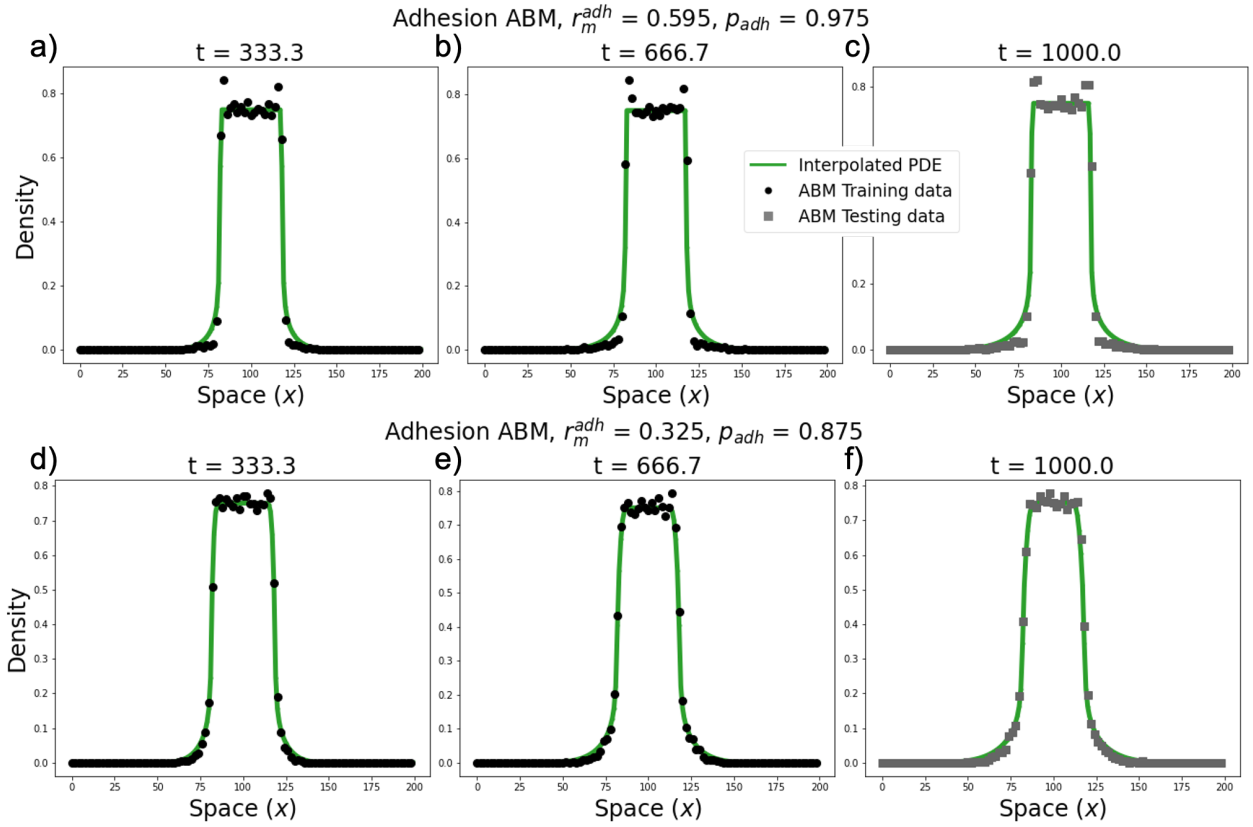

Figure 17: Predicting Adhesion ABM data with the interpolated PDE model. The interpolated PDE model predicts Adhesion ABM data for (a-c) $r_m^{adh} = 0.595$ and $p_{adh} = 0.975$ and (d-f) $r_m^{adh} = 0.325$ and $p_{adh} = 0.875$.

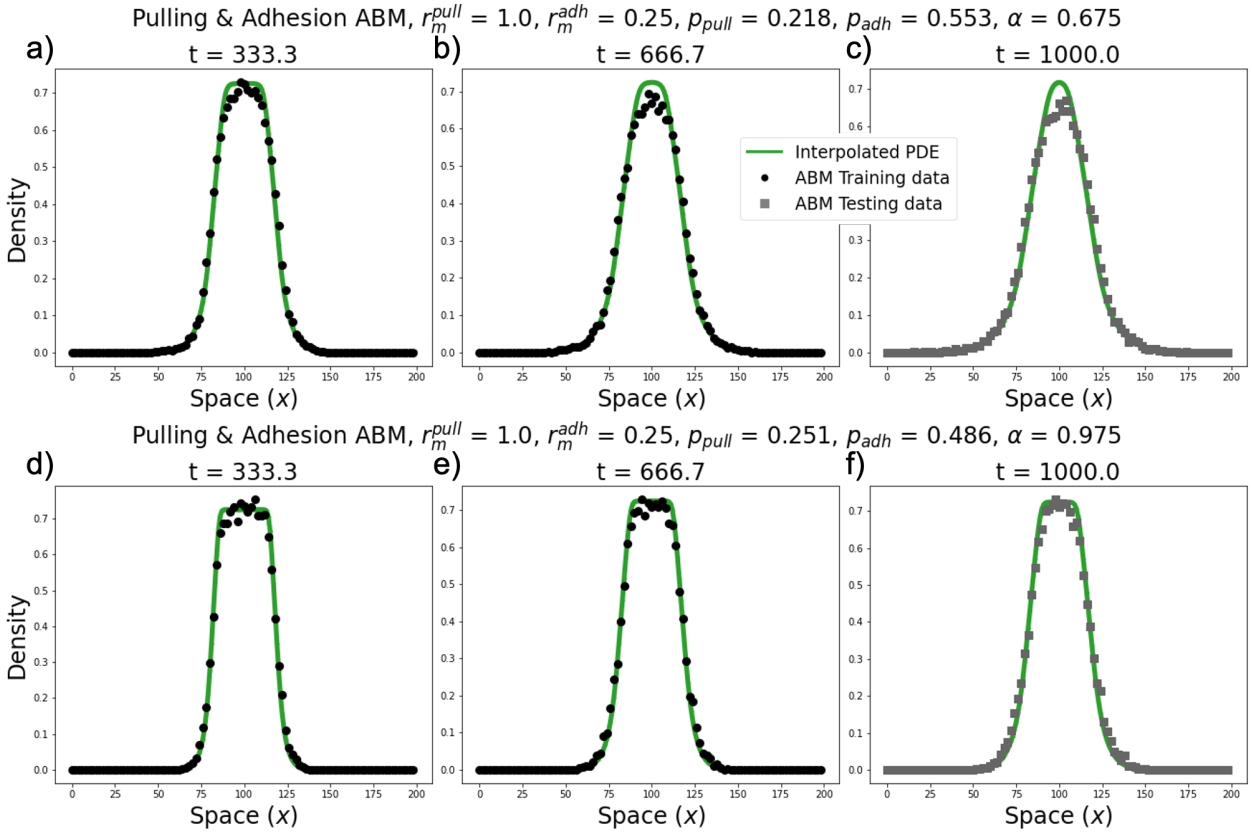

Figure 18: Predicting Pulling & Adhesion ABM data with the interpolated PDE model. The interpolated PDE model predicts Adhesion ABM data for $r_m^{pull} = 1.0$, $r_m^{adh} = 0.25$, and (a-c) $p_{pull} = 0.218$, $p_{adh} = 0.553$, and $\alpha = 0.675$ (d-f) $p_{pull} = 0.251$, $p_{adh} = 0.486$, and $\alpha = 0.975$.

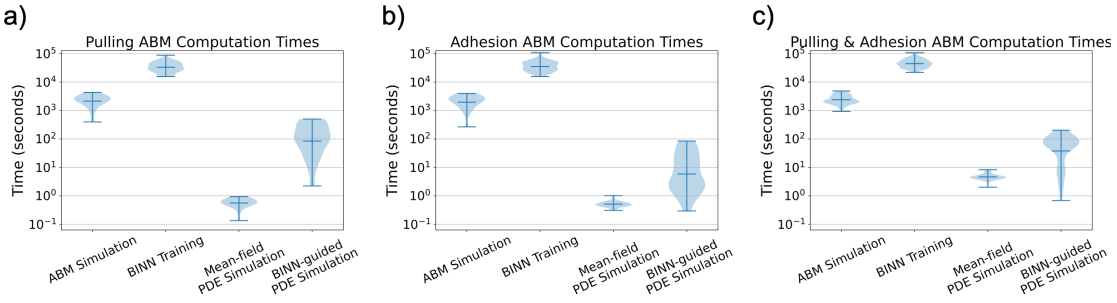

Figure 19: Computational expenses of each modeling approach. Violin plots represent the distribution of wall time computations for ABM simulations, BINN training, mean-field PDE simulations, and BINN-guided PDE simulations for the (a) Pulling ABM, (b) Adhesion ABM, and (c) Pulling & Adhesion ABM.