DATA ASSIMILATION WITH AGENT-BASED MODELS: A SAMPLING ALGORITHM

Daniel Tang
Leeds Institute for Data Analytics, University of Leeds, UK*
D.Tang@leeds.ac.uk

Nick Malleson
School of Geography, University of Leeds, UK

January 11, 2022

ABSTRACT

Every day, weather forecasting centres around the world receive noisy, incomplete observations of the atmosphere and use this data, along with their atmospheric models, to update their forecasts. This process, known as data assimilation or data fusion, is best expressed as Bayesian inference: given a set of observations, some prior beliefs and a model of the target system, what is the probability distribution of some set of observables? The observables are generally not variables that have been directly observed and may be in the future.

While data assimilation has developed rapidly in weather forecasting and other areas, relatively little progress has been made in performing data assimilation with agent based models. This has hampered the use of agent based models to make forecasts of real-world observables.

Here we present an algorithm that uses Markov-Chain-Monte-Carlo to generate samples of the states of an agent based model over a window of time given a set of noisy, incomplete observations of the system. We demonstrate the algorithm by performing data assimilation with an agent-based, spatial predator-prey model.

***K*eywords** Data assimilation, Bayesian inference, Agent based model, Integer linear programming, predator prey model

1 Introduction

Agent-based models (ABMs) have been widely used to model the interactions of many discrete, heterogeneous entities, such as humans or animals. These models are often used to demonstrate the existence of certain dynamical mechanisms or emergent properties. However, if we wish to use a model to make quantitative forecasts, we're often in a situation where we have noisy, aggregated and/or incomplete observations of the target system over some period of time and would like to use our ABM to infer the value of some unobserved variable given our observations. This is not straightforward as we don't generally know the complete start-state of the model, or the exact behavioural decisions the agents would need to make in order to reproduce our observations, so simply running the model forward is not an option. The problem is better framed in terms of probabilistic inference: given the model, our prior beliefs about the start state (or other boundary conditions) and our observations, what is the probability distribution of some unobserved measure? This is the problem of *data assimilation* (DA) and this paper presents a new method that allows such questions to be answered. Note that this problem is distinct from the problem of model estimation or parameter optimisation (Thiele, Kurth, & Grimm, 2014) which we do not directly address here.

DA has developed rapidly in applications such as weather forecasting (Kalnay, 2003) and the earth sciences more broadly (Reichle, 2008), but established DA techniques do not work well with ABMs and little progress has been made in developing ABM-specific techniques. Since ABMs consist of 'agents' that often make discrete choices from a number of possible actions, the space of model trajectories is not usually continuous and so techniques that require the gradient of the posterior (Talagrand, 1997) (Lewis, Lakshmivarahan, & Dhall, 2006) are not directly applicable. Some

*This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 757455)

studies have tried to apply particle filters and variants of the Kalman filter to ABMs in applications such as crime (Lloyd, Santitissadeekorn, & Short, 2016), bus routes (Kieu, Malleson, & Heppenstall, 2020), pedestrian dynamics (Wang & Hu, 2015; Ward, Evans, & Malleson, 2016; Clay, Kieu, Ward, Heppenstall, & Malleson, 2020; Malleson et al., 2020) and population movement (Lueck, Rife, Swarup, & Uddin, 2019), but these have not scaled well in practice.

Sampling algorithms offer an alternative approach, where we take samples of the posterior model trajectory over some time period then convert these to samples of the unobserved measure of interest. However, sampling is difficult in practice because the posterior has zero probability for the vast majority of model trajectories, so even finding a trajectory that fits the observations is difficult. Even though we can generate trajectories that fit the prior by simply performing a forward execution of the model from an initial state drawn from the prior, if the observations refute the trajectory the probability falls to zero. It doesn't take many observations until the probability of randomly choosing a trajectory that fits the observations becomes very small indeed. So simple techniques such as rejection sampling, for example, are not practical. It is for similar reasons that the techniques based on particle filtering fail. The Metropolis-Hastings algorithm is another widely used sampling algorithm. The challenge to using this algorithm for our purposes is to create a proposal function which randomly generates a proposed next sample given the current one. A common strategy is to generate a new sample by perturbing one or more elements of the previous sample at random. However, if we do this in a naive way it's very unlikely that the perturbed sample will be a trajectory that contains only possible actions and satisfies the observations. So, the proposed next sample would almost certainly be rejected and we'd probably end up stuck on the first sample until we grew old.

Here we show that the set of possible trajectories can be approximated as a high-dimensional convex polyhedron, and show how to use this approximation to construct a proposal function for the Metropolis-Hastings algorithm. We demonstrate the algorithm by performing data assimilation in an agent-based, spatial predator-prey model.

Alternative techniques for sampling from discrete points in a convex polyhedron do exist. For example, discrete hit-and-run (Baumert et al., 2009) has had some success, but because of the extreme sparsity of feasible points this is unlikely to be efficient when applied to our problem. Universal hashing (Meel et al., 2016) provides a promising alternative but we have found that this technique doesn't scale well to the number of dimensions needed for our application. This lack of applicable DA techniques has severely limited the use of ABMs for predictive modelling.

2 Formulation of the problem

2.1 Agents, States and Actions

Suppose we have a timestepping ABM that consists of agents with a finite number of possible internal states and a finite number of ways of acting on their world. Given this, we can define an ABM as:

- An ordered list of agent states $\mathcal{S} = \langle \sigma_0 ... \sigma_{S-1} \rangle$
- An ordered list of agent actions $\mathcal{A} = \langle \alpha_0 ... \alpha_{A-1} \rangle$
- An *agent timestep*, $\pi : \mathbb{Z} \times \mathbb{Z}^S \times \mathbb{Z} \to \mathbb{R}$, which defines the probability that an agent will act in a particular way such that $\pi(\psi, \Psi, a)$ gives the probability that an agent in state σ_ψ, surrounded by agents, Ψ, will perform action α_a (where Ψ is a vector whose i^{th} element is the number of agents in state σ_i at the start of the timestep).
- An *action function*, $F : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}^S$, which defines the effect of an action on the world such that $F(\psi, a)$ returns a vector whose i^{th} element gives the number of agents in state σ_i that result from an agent in state σ_ψ performing act α_a (including the final state of the acting agent).

As a simple illustration, we define the "cat and mouse" ABM which, using the definitions above, we define as follows:

Agent states, $\mathcal{S}$. An agent can be either a cat or a mouse and can be on one of two gridsquares, left or right, so

$$\mathcal{S} = \langle \text{left cat}, \text{right cat}, \text{left mouse}, \text{right mouse} \rangle$$

.

Agent actions, $\mathcal{A}$. In any given timestep, an agent can either move or stay still, so

$$\mathcal{A} = \langle \text{move}, \text{stay still} \rangle$$

.

Agent timestep, π. The agent timestep can be expressed as

$$\pi(\psi, \Psi, a) = \begin{cases} 0.5 & \text{if } \psi \in \{0, 1\} \\ 1 & \text{if } (\psi = 2, \Psi_1 = 0, a = 1) \text{ or } (\psi = 3, \Psi_2 = 0, a = 1) \\ & \text{or } (\psi = 2, \Psi_1 > 0, a = 0) \text{ or } (\psi = 3, \Psi_2 > 0, a = 0) \\ 0 & \text{otherwise} \end{cases}$$

The top line says that a cat will move or stay still with probability 0.5, irrespective of other agents. The next line says that a mouse will stay still if there are no cats on the same gridsquare while the third line says that a mouse will move if there are any cats on the same gridsquare. Finally, the last line says that any other behaviours have zero probability.

Action function, F. F gives the result of an agent's actions. For example, $F(\psi = 1, a = 0)$ is the result of an agent in state $\psi = 1$ (*right cat*) performing the action $a = 0$ (*move*). If $F(1, 0) = \{1, 0, 0, 0\}$ then the result is one cat in state $\psi = 0$ (*left cat*). Similarly:

$$
\begin{aligned}
F(0,0) &= \{0,1,0,0\} \\
F(1,0) &= \{1,0,0,0\} \\
F(2,0) &= \{0,0,0,1\} \\
F(3,0) &= \{0,0,1,0\} \\
F(0,1) &= \{1,0,0,0\} \\
F(1,1) &= \{0,1,0,0\} \\
F(2,1) &= \{0,0,1,0\} \\
F(3,1) &= \{0,0,0,1\}
\end{aligned}
$$

2.2 Tensor notation

In the rest of this paper we'll make use of tensors. These are just arrays of numbers, like vectors or matrices, but arranged in any number of dimensions. Each dimension is either *covariant* or *contravariant* and each tensor has a *shape* which tells us how many co- and contravariant dimensions the tensor has and the number of elements there are in each dimension. It will be useful to refer to the set of all tensors of a given shape, for this we'll use $\mathbb{R}$ adorned with the tensor's shape. To write a tensor's shape we put covariant dimensions in the subscript position and contravariant dimensions in superscript position. So, for example, $\mathbb{R}^N_{SA}$ would describe the set of all 3-dimensional tensors that have one contravariant dimension of size N and two covariant dimensions of sizes S and A.

An element of a tensor is referred to by specifying the co- and contravariant coordinates of the element. For example if $T \in \mathbb{R}^N_{SA}$, then $T^t_{\psi a}$ refers to the element of T that has coordinate t in the contravariant dimension and coordinates (ψ, a) in the covariant dimensions. By convention, coordinates begin at 0.

We'll also use the Einstein summation convention, meaning that if the same index symbol is repeated in co- and contravariant positions in a term, then a summation over that symbol is implied. So, for example if $X \in \mathbb{R}_8$ and $Y \in \mathbb{R}^8$ then

$$X_\psi V^\psi \equiv \sum_{\psi=0}^{7} X_\psi V^\psi$$

When the same symbol is repeated in the same position, this implies universal quantification. For example if $Y \in R_8$ then

$$X_\psi = Y_\psi$$

is equivalent to

$$\forall (0 \leq \psi < 8) : X_\psi = Y_\psi$$

We refer to *slices* of a tensor by using the $*$ symbol in an index position. So, for example if $T \in \mathbb{R}^N_{SA}$ then $T^t_{\psi *}$ refers to the 1-dimensional tensor in $\mathbb{R}_A$ comprised of the elements of T with fixed t and ψ but ranging over all values of a.

The symbols $\mathbf{0}$ and $\mathbf{1}$ represent tensors whose elements are all 0 or 1 respectively. Their shape should be unambiguous from the context.

Finally, a tensor $Y \in \mathbb{R}^J$ is considered to be equivalent to a J-dimensional column vector, a tensor $X \in \mathbb{R}_I$ equivalent to an I-dimensional row vector and a tensor $M \in \mathbb{R}^J_I$ equivalent to a matrix with I rows and J columns, so we use normal matrix notation with these tensors without ambiguity.

2.3 Trajectories

Let a model timestep be a tensor $E \in \mathbb{R}_{SA}$ whose elements $e_{\psi a}$ are the number of agents in state ψ that perform act a in this timestep. For example, the timestep shown in Figure 1 for the cat and mouse example would be

$$
E = \begin{array}{c} \\ \sigma_0 \\ \sigma_1 \\ \sigma_2 \\ \sigma_3 \end{array}
\begin{array}{c} \begin{array}{cc} \alpha_0 & \alpha_1 \end{array} \\
\left[\begin{array}{cc} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{array} \right]
\end{array}
$$

where one agent in state σ_1 (*right cat*) performs action α_0 (*move*) and one agent in state σ_2 (*left mouse*) performs action α_1 (*stay still*).

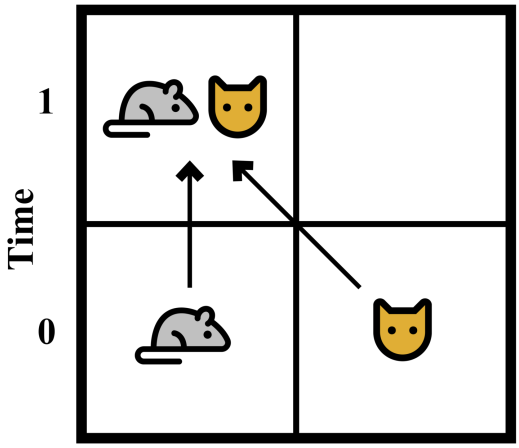

Figure 1: A simple cat and mouse model.

Let a model trajectory, T, be a tensor in $\mathbb{R}^N_{SA}$ that represents N timesteps of a model with S agent states and A actions, so that $T^t_{\psi a}$ denotes the $(\psi, a)^{th}$ element of the t^{th} timestep matrix.

A tensor must satisfy a number of constraints in order to be a valid trajectory of an ABM. Since the elements of a trajectory are counts of agents, they must be non-negative integers. We'll call this the *non-negative integer constraint* and define the set of all non-negative integer tensors

$$\mathbb{N}^N_{SA} = \left\{T \in \mathbb{R}^{\mathbb{N}}_{\mathbb{SA}} : T^t_{\psi a} \geq \mathbf{0}^t_{\psi a}, T^t_{\psi a} \in \mathbb{Z}\right\} \tag{1}$$

A trajectory must also be *continuous* by which we mean that the number of agents in each state at the end of timestep $t-1$ must be the number of agents in each state at the beginning of timestep t[2]. We call this the *continuity constraint* and define the set of continuous tensors, with respect to an action function F:

$$\mathcal{C}^N_{SA}(F) = \left\{T \in \mathbb{R}^N_{SA} : \forall\, (1 \leq \bar{t} < N) : F^{\psi a}_{\phi} T^{t-1}_{\psi a} = \mathbf{1}^b T^t_{\phi b}\right\} \tag{2}$$

where $F \in \mathbb{R}^{SA}_S$ is the tensor such that $F^{\psi a}_* = F(\psi, a)$. Note that this assumes that a timestep is performed by updating all agents in parallel, i.e. agents must act with no information about the actions of other agents in the same timestep. This is different from serial update, where agents are updated in a particular order within a timestep and have access to information about the actions of agents that come before them in the ordering. Note also that this assumes that agents in the same state are indistinguishable, i.e. if we swap two agents in the same state, the trajectory is unchanged.

So, we define the set of trajectories, $\mathcal{T}^N_{SA}(F)$, as set of tensors that satisfy (1) and (2).

$$\mathcal{T}^N_{SA}(F) = \mathbb{N}^N_{SA} \cap \mathcal{C}^N_{SA}(F) \tag{3}$$

2.4 The posterior

Our ultimate goal is to generate samples from the Bayesian posterior over the trajectories, given a set of observations, Ω

$$P\left(T|\Omega\right) \propto P\left(\Omega|T\right) P(T)$$

The prior over the trajectory, $P(T)$, can be composed from terms involving the agent timestep function, $\pi(\psi, \Psi, a)$, which gives the probability that a single agent will perform action a given that it starts in state ψ and is surrounded by agents Ψ. First consider all the agents that are in state ψ at the beginning of timestep t. Let $R \in \mathbb{R}_A$ be a vector whose elements R_a are the number of these agents that perform act a in timestep t. The probability distribution of R is just a multinomial distribution

$$P\left(R \mid \Psi, \psi\right) = \begin{cases} \Psi_\psi! \prod_a \frac{\pi(\psi,\Psi,a)^{R_a}}{R_a!} & \text{if } R_a \mathbf{1}^a = \Psi_\psi \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

If we now let $P(\Psi^0)$ be the prior probability over the number of agents in each state at the beginning of the trajectory then the prior over trajectories can be written

$$P(T) = \begin{cases} P(\Psi^0 = T^0_{*c}\mathbf{1}^c) \prod_{t,\psi} \left(T^t_{\psi b}\mathbf{1}^b\right)! \prod_a \frac{\pi(\psi, T^t_{*d}\mathbf{1}^d, a)^{T^t_{\psi a}}}{T^t_{\psi a}!} & \text{if } T \in \mathcal{T}^N_{SA}(F) \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

[2]This does not mean that agents cannot leave or enter the system, only that if they do then that change must be defined as part of an action.

The likelihood, $P(\Omega|T)$, can also be decomposed. Without loss of generality, we take Ω to consist of some number of observations that are independent of each other given the trajectory, so that Ω is a set of pairs (ω, v) that consist of a stochastic observation operator ω and an observed value v (which may be a vector). We write $P(\omega(T) = v)$ to denote the probability of observation operator ω making observation v on trajectory T. So

$$P(\Omega|T) = \prod_{(\omega,v)\in\Omega} P(\omega(T) = v)$$

and posterior can be written as

$$P(T|\Omega) \propto \begin{cases} \prod_{(\omega,v)\in\Omega,t,\psi,a} P(\Psi^0 = T^0_{*c}\mathbf{1}^c) P\left(\omega(T) = v\right) \left(T^t_{\psi b}\mathbf{1}^b\right)! \frac{\pi(\psi, T^t_{*d}\mathbf{1}^d, a)^{T^t_{\psi a}}}{T^t_{\psi a}!} & \text{if } T \in \mathcal{T}^N_{SA}(F) \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

3 Approximating the support of the posterior

Sampling from the posterior is difficult because it is hard to find proposal trajectories that have non-zero probability. Our strategy to solve this problem is to first derive an expression for a volume of trajectory space, $\mathcal{P}^N_{SA}$, that contains the support of the posterior, $\text{supp}(P(T|\Omega))$ (the support is just the set of trajectories that have non-zero probability). If we choose $\mathcal{P}^N_{SA}$ so that it is easy to sample from and a relatively tight approximation of $\text{supp}(P(T|\Omega)))$ then there's a good chance that a sample drawn from $\mathcal{P}^N_{SA}$ will have non-zero probaqbility. If a sample has zero probability we simply reject it and sample again.

From equation (6)

$$\begin{aligned} \text{supp}(P(T|\Omega)) = \bigcap_{(\omega,v)\in\Omega,t,\psi,a} & \mathcal{T}^N_{SA} \cap \\ & \text{supp}(P(\Psi^0 = T^0_{*c}\mathbf{1}^c)) \cap \\ & \text{supp}\left(P\left(\omega(T) = v\right)\right) \cap \\ & \left(\text{supp}\left(\pi(\psi, T^t_{*b}\mathbf{1}^b, a)\right) \cup \left\{T : T^t_{\psi a} = 0\right\}\right) \end{aligned} \tag{7}$$

i.e. in order for T to have non-zero posterior probability, it must be a trajectory of the ABM, it must have a start state that has non-zero prior probability, all the observation likelihoods must be non-zero and each element of T must denote an agent action with non-zero probability.

3.1 Convex $\mathbb{Z}$-polyhedra and $\mathbb{Z}$-distributions

Let a $\mathbb{Z}$-polyhedron be a set of tensors whose elements satisfy a set of linear constraints and belong to the set of integers:

$$\mathcal{P}^{\mathcal{N}}_{\mathcal{SA}} = \left\{T \in \mathbb{Z}^N_{SA} : L^i \leq C^{\psi ai}_t T^t_{\psi a} \leq U^i\right\}$$

where $C \in \mathbb{Z}^{SAI}_N$, $L \in \mathbb{Z}^I$ and $U \in \mathbb{Z}^I$ for some I. This is similar to the $\mathbb{Z}$-polyhedron described in (Quinton, Rajopadhye, & Risset, 1996)).

From equation 3 we can see immediately that the set of all trajectories, $\mathcal{T}^N_{SA}$, is a $\mathbb{Z}$-polyhedron. The supports of the prior, $P(\Psi^0)$, the observations, $P(\omega(T) = v)$, and the agent actions, $\pi(\psi, T^t_{*b}\mathbf{1}^b, a)$, can often be easily expressed as $\mathbb{Z}$-polyhedra. However, if this is not the case, each of the probability distributions can be expressed as computer programs. In practice, these computer programs will be simple and it will be possible to use a technique known as *abstract interpretation*(Cousot & Cousot, 1977) using the domain of convex polyhedra(Cousot & Halbwachs, 1978)(Becchi & Zaffanella, 2018)(Fukuda, 2020) to efficiently calculate a convex polyhedron that contains the support of the program. Software to perform abstract interpretation using convex polyhedra already exists(Henry, Monniaux, & Moy, 2012)(Gurfinkel & Navas, 2021)(Jeannet & Miné, 2009)(Bagnara, Hill, & Zaffanella, 2008) and the technique has been used in applications such as compiler optimization(Sjödin et al., 2009) and verification of safety-critical systems(Halbwachs, Proy, & Roumanoff, 1997). The programs may contain calls to a random number generator `Random()` that returns a random floating-point number $0 \leq r < 1$. This can be represented in the polyhedral domain by introducing a new variable, r, that satisfies $0 \leq r < 1$ for each call to `Random()`.

If the number of agents is very much smaller than the number of agent states (which is often the case with agent based models) then we may be willing to make the assumption that in any timestep there is at most one agent performing a given action from a given start state (i.e. $\forall \psi, a, t : T^{\psi a}_t \in \{0, 1\}$). Under this assumption, which we'll call the *Fermionic assumption*, the set of *Fermionic trajectories*, $\mathcal{F}^N_{SA} = \left\{T \in \mathcal{T}^N_{SA} : \forall \psi, a, t : T^{\psi a}_t \in \{0, 1\}\right\}$, all lie on the corners of the unit hypercube. So the intersection of $\mathcal{F}^N_{SA}$ with any set is a $\mathbb{Z}$-polyhedron and $\text{supp}(P(T|\Omega))$ can always be exactly represented as a $\mathbb{Z}$-polyhedron.

So, if we let $\mathcal{P}^N_{SA}(f)$ be a $\mathbb{Z}$-polyhedron that contains $\mathrm{supp}(f)$ on the integer points then from equation (7)

$$\begin{aligned}\mathrm{supp}(P(T|\Omega)) \subseteq \mathcal{P}^N_{SA}(P(T|\Omega)) = \bigcap_{(\omega,v)\in\Omega,t,\psi,a} & \mathcal{T}^N_{SA} \cap \\ & \mathcal{P}^N_{SA}(P(\Psi^0 = T^0_{*c}\mathbf{1}^c)) \cap \\ & \mathcal{P}^N_{SA}\left(P\left(\omega(T) = v\right)\right) \cap \\ & \left(\mathcal{P}^N_{SA}\left(\pi(\psi, T^t_{*b}\mathbf{1}^b, a)\right) \cup \left\{T : T^t_{\psi a} = 0\right\}\right)\end{aligned} \tag{8}$$

The intersection of two $\mathcal{Z}$-polyhedra is easy to express as another $\mathcal{Z}$-polyhedron by just concatenating the constraints

$$\begin{aligned}\left\{T \in \mathbb{Z}^N_{SA} : L \leq C^{\psi a *}_t T^t_{\psi a} \leq U\right\} \cap \left\{T \in \mathbb{Z}^N_{SA} : L' \leq D^{\psi a *}_t T^t_{\psi a} \leq U'\right\} \\ = \left\{T \in \mathbb{Z}^N_{SA} : \begin{pmatrix} L \\ L' \end{pmatrix} \leq \begin{pmatrix} C^{\psi a *}_t \\ D^{\psi a *}_t \end{pmatrix} T^t_{\psi a} \leq \begin{pmatrix} U \\ U' \end{pmatrix}\right\}\end{aligned} \tag{9}$$

so the only difficulty in calculating $\mathcal{P}^N_{SA}(P(T|\Omega))$ from (8) is the union in the final term. To transform this into an intersection we introduce an auxiliary variable z and use the identity

$$\begin{aligned}\left\{T \in \mathbb{Z}^N_{SA} : L^i \leq C^{\phi bi}_s T^s_{\phi b} \leq U^i\right\} \cup \left\{T : T^t_{\psi a} = 0\right\} = \\ \left\{T \in \mathbb{Z}^N_{SA}, z \in \{0,1\} : \right. \\ C^{\phi bi}_s T^s_{\phi b} + (\overline{B}^i - U^i)z \leq \overline{B}^i, \\ \underline{B}^i \leq C^{\phi bi}_s T^s_{\phi b} + (\underline{B}^i - L^i)z, \\ 0 \leq uz - T^t_{\psi a}, \\ \left. z - T^t_{\psi a} \leq 0\right\}\end{aligned} \tag{10}$$

where u is the maximum value that any element of T can take, the elements of $\overline{B} \in \mathbb{R}^I$ are defined as

$$\overline{B}^i = \frac{u\sum_{s,\phi,b}\left(C^{\phi bi}_s + \left|C^{\phi bi}_s\right|\right)}{2}$$

and the elements of $\underline{B} \in \mathbb{R}^I$ are defined as

$$\underline{B}^i = \frac{u\sum_{s,\phi,b}\left(C^{\phi bi}_s - \left|C^{\phi bi}_s\right|\right)}{2}$$

To see why this identity holds, note first that the constraints on z makes it into an indicator variable that is 0 if $T^t_{\psi a} = 0$ or 1 otherwise. When $z = 1$ the first set of constraints is equal to $C^{\phi bi}_s T^s_{\phi b} \leq U^i$ and the second is equal to $L^i \leq C^{\phi bi}_s T^s_{\phi b}$ so their intersection is $L^i \leq C^{\phi bi}_s T^s_{\phi b} \leq U^i$ as required, whereas when $z = 0$ we have the constraints $\underline{B}^i \leq C^{\phi bi}_s T^s_{\phi b} \leq \overline{B}^i$. But $\underline{B}^i$ and $\overline{B}^i$ are lower and upper bounds on the value of $C^{\phi bi}_s T^s_{\phi b}$ so this is satisfied for all trajectories, as required.

There are two things worth noting here. Firstly if we make the Fermionic assumption then $z = T^t_{\psi a}$ and the auxiliary indicator variables become unnecessary. Secondly, we must impose a finite value for u, the maximum value that elements of the trajectory can take. In practice, this is not a problem as we can give u a value such that the probability of any trajectory of interest having any element larger than u is small.

Using (8), (9) and (10) the support of the posterior can be reduced to a $\mathbb{Z}$-polyhedron.

The idea of a $\mathbb{Z}$-polyhedron as the support for a probability distribution naturally leads to the idea of a $\mathbb{Z}$-distribution which is a discrete probability distribution defined over the members of a $\mathbb{Z}$-polyhedron. From the above, it can be seen that the posterior distribution of an ABM trajectory can be expressed as a $\mathbb{Z}$-distribution.

As an illustration, consider a two-timestep trajectory of the cat and mouse model described in section 2.1. Suppose we flip a fair coin to decide whether each agent state is occupied or empty at $t = 0$. Suppose also that we observe a cat in the left grid-square at time $t = 1$. Our aim is to construct a $\mathbb{Z}$-polyhedron, $\mathcal{P}^2_{4\,2}(P(T|\Omega))$, that describes the support of the posterior.

Working through (8) term by term, the $\mathcal{T}^2_{4\,2}$ term is just the continuity constraints in (2), which are already in linear form so we're done. The second term is the support of the prior. This constrains each agent state at $t = 0$ to be at most 1, which can be expressed as

$$\left\{T : T^0_{\psi 0} + T^0_{\psi 1} \leq \mathbf{1}_\psi\right\}$$

The third term is the support of the observation. Since we observe a cat in the left grid-square at time $t = 1$ we need to add the constraint

$$T^1_{00} + T^1_{01} = 1$$

The final term is the constraint due to agent interactions. The impossible interactions are a mouse staying put when there is a cat on the same gridsquare or moving when there are no cats, which translates to the four cases

$$\begin{aligned}
\mathrm{supp}(\pi(2, T^t_{*a}\mathbf{1}^a, 0)) &= \left\{T : -T^t_{00} - T^t_{01} \leq -1\right\} \\
\mathrm{supp}(\pi(3, T^t_{*a}\mathbf{1}^a, 0)) &= \left\{T : -T^t_{10} - T^t_{11} \leq -1\right\} \\
\mathrm{supp}(\pi(2, T^t_{*a}\mathbf{1}^a, 1)) &= \left\{T : T^t_{00} + T^t_{01} \leq 0\right\} \\
\mathrm{supp}(\pi(3, T^t_{*a}\mathbf{1}^a, 1)) &= \left\{T : T^t_{10} + T^t_{11} \leq 0\right\}
\end{aligned} \tag{11}$$

for all t. If, for simplicity, we make the Fermionic assumption by adding the constraints

$$\mathbf{0}^t_{\psi a} \leq T^t_{\psi a} \leq \mathbf{1}^t_{\psi a}$$

then using the identity in (10) to take the union of each constraint in (11) with $\left\{T : T^t_{\psi a} = 0\right\}$ finally gives the four constraints

$$\begin{aligned}
-T^t_{00} - T^t_{01} + T^t_{20} &\leq 0 \\
-T^t_{10} - T^t_{11} + T^t_{30} &\leq 0 \\
T^t_{00} + T^t_{01} + 2T^t_{21} &\leq 2 \\
T^t_{10} + T^t_{11} + 2T^t_{31} &\leq 2
\end{aligned}$$

for each timestep $t = 0$ and $t = 1$ to describe the agent interactions.

Taken together, these constraints define a $\mathbb{Z}$-polyhedron that is the set of (Fermionic) trajectories for the cat and mouse ABM, and when combined with equation (6) defines $P(T|\Omega)$ as a $\mathbb{Z}$-distribution.

4 Sampling from a $\mathbb{Z}$-distribution

Having shown how to express $P(T|\Omega)$ as a $\mathbb{Z}$-distribution, we now show how to construct a Markov process which will allow us to sample from a $\mathbb{Z}$-distribution.

To do this we need to define

- a set of Markov states, $\mathcal{M}$
- a probability measure $P : \mathcal{M} \to \mathbb{R}$ which gives the probability of each Markov state (this need not be normalised, though, as the Metropolis Hastings algorithm only ever needs probability ratios)
- a stochastic proposal function $f : \mathcal{M} \to \mathcal{M}$ from which we can generate transitions to a new Markov state given the current Markov state
- a mapping $E : \mathcal{M} \to \mathbb{R}^T_{SA}$ which maps Markov states to trajectories so we can recover the sample.

In order to be of use in practice, the proposal function, f, must have the following properties:

- For any two Markov states there should exist a sequence of transitions which forms a path between those states and has non-zero probability of being proposed.
- For any proposal from state $S_a \to S_b$ with non-zero probability, the probability of the reverse transition from $S_b \to S_a$ should also be non-zero. This allows us to attain detailed balance in the Metropolis Hastings algorithm. The average ratio of forward and backward probabilities times the ratio of start and end state probabilities should be close to 1 to ensure that a reasonable proportion of proposals are accepted.
- Given a current Markov state, there should be computationally efficient procedure to generate a proposal and calculate its acceptance probability.

4.1 The set of Markov states

Given a $\mathbb{Z}$-polyhedron, we split the constraints into two sets: equalities (i.e. those whose lower and upper bound have the same value) and inequalities (i.e. those whose lower and upper bounds differ):

$$\mathcal{P} = \left\{T \in \mathbb{Z}^N_{SA} : L^i \leq C^{\psi a i}_t T^t_{\psi a} \leq U^i \cap D^{\psi a i}_t T^t_{\psi a} = E^i\right\} \tag{12}$$

Suppose there are N_e equality constraints (i.e. E is an N_e dimensional vector of integers) and we partition the elements of T into 'basic' and 'non-basic' elements, so that there are exactly N_e basic elements and $NSA - N_e$ non-basic elements (note that since the continuity constraints (2) are equality constraints then $N_e \geq (N-1)S$).

A partition of a tensor T can be defined as a pair of tensors $Q \in \{0,1\}^N_{SAN_e}$ and $R \in \{0,1\}^N_{SAJ}$, where $J = NSA - N_e$. Q and R take vectors of basic, $X \in \mathbb{R}^{N_e}$, and non-basic, $Y \in \mathbb{R}^J$, variables and pack them into a trajectory tensor, without changing their values. More formally, Q and R should satisfy

$$Q^t_{\psi ai}\mathbf{1}^i + R^t_{\psi aj}\mathbf{1}^j = \mathbf{1}^t_{\psi a}$$

$$Q^t_{\psi ai}\mathbf{1}^{\psi a}_t = \mathbf{1}_i$$

$$R^t_{\psi aj}\mathbf{1}^{\psi a}_t = \mathbf{1}_j$$

Given the partition, we can map from a vector of basic variables, X and a vector of non-basic variables, Y, to a trajectory

$$T^t_{\psi a} = Q^t_{\psi ai}X^i + R^t_{\psi aj}Y^j \tag{13}$$

If we now let

$$B^k_i = D^{\psi ak}_t Q^t_{\psi ai} \tag{14}$$

and

$$N^k_j = D^{\psi ak}_t R^t_{\psi aj}$$

then we can write the equality constraints as

$$B^k_i X^i + N^k_j Y^j = E^k \tag{15}$$

Now, since there are N_e basic variables and N_e equality constraints, B is square so if we choose the basic variables in such a way as to ensure that B has an inverse then

$$X^i = (B^{-1})^i_k(E^k + N^k_j Y^j) \tag{16}$$

inserting this into (13) gives

$$T^t_{\psi a} = Q^t_{\psi ai}(B^{-1})^i_k(E^k + N^k_j Y^j) + R^t_{\psi aj}Y^j \tag{17}$$

or equivalently

$$T^t_{\psi a} = K^t_{\psi a} + M^t_{\psi aj}Y^j \tag{18}$$

where

$$K^t_{\psi a} = Q^t_{\psi ai}(B^{-1})^i_k E^k$$

and

$$M^t_{\psi aj} = Q^t_{\psi ai}(B^{-1})^i_k N^k_j + R^t_{\psi aj}$$

If we also choose the basic variables such that all elements of B^{-1} are integer then we can define the set of Markov states to be

$$\mathcal{M} = \left\{Y \in \mathbb{Z}^J : \forall i : 0 \le Y_i \le u\right\}$$

where u is the upper limit on the occupancy of agent actions, and map each Markov state, Y, to a unique integer trajectory using (18).

4.2 The probability of a Markov state

Given a Markov state, Y, that satisfies $0 \le Y_j \le u$, the associated trajectory given by equation (18) is guaranteed to satisfy all the equality constraints. However, there is no guarantee that the basic variables given by eq (16) will be within their bounds or that the inequality constraints will be satisfied. To deal with this we introduce the concept of *infeasibility* which is a measure of the extent to which a Markov state fails to satisfy the inequality constraints.

Given the inequality constraints of the $\mathbb{Z}$-distribution (including the upper and lower limits of the trajectory elements themselves)

$$L^i \le C^{\psi ai}_t T^t_{\psi a} \le U^i$$

using (18) this can be written as

$$L^i \le J^i + Q^i_j Y^j \le U^i \tag{19}$$

where

$$J^i = C^{\psi ai}_t K^t_{\psi a}$$

and

$$Q^i_j = C^{\psi ai}_t M^t_{\psi aj}$$

let the infeasibility of the i^{th} constraint, $\iota(Y)_i$, be defined to be equal to 0 if the constraint is satisfied or equal to the distance to the nearest bound otherwise

$$\iota(Y)_i = \begin{cases} L^i - J^i - Q^i_j Y^j & \text{if } J^i + Q^i_j Y^j < L^i \\ J^i + Q^i_j Y^j - U^i & \text{if } J^i + Q^i_j Y^j > U^i \\ 0 & \text{otherwise} \end{cases}$$

and let the total infeasibility be the sum of infeasibilities of all constraints

$$\iota(Y) = \sum_i \iota(Y)_i$$

So, each Markov state is associated with a total infeasibility with respect to the inequalities of the $\mathbb{Z}$-distribution.

If the infeasibility is zero, then all the constraints are satisfied and the probability is defined to be proportional to the probability of the $\mathbb{Z}$-distribution

$$P(Y) \propto P(K^t_{\psi a} + M^t_{\psi aj} Y^j)$$

If the infeasibility is not zero, then the probability is defined in the following way. Given an upper bound on the trajectory's elements, u, we define the *clamped* trajectory

$$\underline{\overline{T^t_{\psi a}}} = \begin{cases} 0 & \text{if } T^t_{\psi a} < 0 \\ u & \text{if } T^t_{\psi a} > u \\ T^t_{\psi a} & \text{otherwise} \end{cases}$$

Furthermore, let $H^{\psi a}_t$ be a log-linear approximation to the target $\mathbb{Z}$-distribution so that for feasible trajectories

$$H^{\psi a}_t T^t_{\psi a} \approx \log(P(T))$$

For infeasible trajectories, we define the probability, P_ι, as

$$\log(P_\iota(Y)) = H^{\psi a}_t \underline{\overline{K^t_{\psi a} + M^t_{\psi aj} Y^j}} - \frac{\iota(Y)}{\tau} + A \tag{20}$$

where τ is a tunable parameter and A is a normalisation constant.

This means that infeasible states have non-zero probability and the Markov chain will pass through some infeasible states. However, if we just ignore these and only take samples from feasible states then we end up with samples from the target $\mathbb{Z}$-distribution. Allowing the Markov chain to pass through infeasible states has the advantage of ensuring that there exists a path between any two feasible states and improves mixing. The price we pay for this is the computational cost of moving through infeasible states that don't generate useful samples.

If we define the *energy* of a Markov state to be

$$E(Y) = \begin{cases} -\tau log(P(Y)) & \text{if } \iota(Y) = 0 \\ \iota(Y) - \tau H^{\psi a}_t \underline{\overline{K^t_{\psi a} + M^t_{\psi aj} Y^j}} & \text{otherwise} \end{cases}$$

then the probability of a Markov state can be written in the form

$$P(Y) \propto e^{\frac{-E(Y)}{\tau}} \tag{21}$$

This is the form of a Boltzmann distribution which describes a thermodynamic system at equilibrium with a heat bath of "temperature" τ. It has been shown that simulations of thermodynamic systems of this type are able to solve large integer optimisation problems (Kirkpatrick, Gelatt, & Vecchi, 1983). In our case, we don't need to change the temperature during the sampling process, but we do need to choose a value for τ. Higher temperatures will increase mixing in the Markov chain, but will also increase the proportion of time spent in infeasible states so we need to find a temperature that is high enough to ensure good mixing of the chain but low enough to ensure a reasonable proportion of samples are feasible. We have found that a good rule of thumb is to set the temperature so that 50% of the samples in a chain are infeasible.

4.3 Transitions between Markov states

Given a Markov state, we transition to another state by choosing an element of Y and perturbing it by ± 1 (if the element is on one of its bounds then perturb it away from the bound in the feasible direction. In the case of Fermionic trajectories, the elements of Y are always on their bounds so there is always exactly one perturbation per element: a swap to the other bound). This defines the set of transitions between Markov states. Let the set of all Markov states reachable from state Y be denoted by $\mathcal{D}(Y)$.

4.3.1 The probability of transitions

The probability of a proposed transition from Y to Y' is defined to be

$$P(Y \to Y') = \frac{\min\left(1, \frac{P_\iota(Y')}{P_\iota(Y)}\right)}{S(Y)}$$

where P_ι is the log-linear probability defined in (20) and

$$S(Y) = \sum_{Y' \in \mathcal{D}(Y)} \min\left(1, \frac{P_\iota(Y')}{P_\iota(Y)}\right)$$

Note that this is irrespective of whether Y or Y' are feasible.

Given this, the Metropolis-Hastings acceptance probability is

$$\alpha = \frac{P(T')P(Y' \to Y)}{P(T)P(Y \to Y')} = \frac{P_\iota(Y)}{P(Y)} \frac{P(Y')}{P_\iota(Y')} \frac{S(Y)}{S(Y')}$$

When Y is infeasible $\frac{P_\iota(Y)}{P(Y)} = 1$. When Y is feasible $P_\iota(Y)$ approximates $P(Y)$ so their ratio should be close to 1. Finally, $S(Y)$ will not change very much between transitions so the ratio $\frac{S(Y)}{S(Y')}$ should also be close to 1 meaning that the acceptance probability should also be close to 1.

4.4 Choosing a basis

The definition of the Markov chain depends on a partition of the trajectory into basic and non-basic variables such that B in equation (14) has an integer inverse $B^{-1} \in \mathbb{Z}^{N_e}_{N_e}$. In general there exist many possible partitions so it remains to define a method of choosing one. The choice we make affects the way Markov states are mapped to trajectories via equation (18) and how infeasibility is calculated via equation (19), where the columns of Q can be thought of as basis vectors in a *constraint space* and Y as a coordinate on the grid formed by that basis. The Markov chain can be seen as a random walk around this grid. If we choose a basis that consists of very dense basis vectors then the Markov states adjacent to a feasible trajectory are likely to have high infeasibility and low probability, leading to slow mixing of the chain, so we aim to choose a basis that has sparse basis vectors in the constraint space. This has the additional advantage that when we transition between adjacent Markov states, the probability distribution over transitions changes in only a few dimensions. This means we can use a binary tree to efficiently update and draw from the probability distribution over transitions(Tang, 2022).

Starting with a $\mathbb{Z}$-polyhedron expressed in the form

$$\mathcal{P} = \left\{T \in \mathbb{Z}^N_{SA} : L^i \leq C^{\psi ai}_t T^t_{\psi a} \leq U^i \cap D^{\psi ai}_t T^t_{\psi a} = E^i\right\}$$

suppose we begin with no basic variables so that R is an arbitrary ordering of the elements of a trajectory

$$T^t_{\psi a} = R^t_{\psi aj} Y^j$$

If we let

$$W^*_j = \begin{pmatrix} D^{\psi a*}_t R^t_{\psi aj} \\ C^{\psi a*}_t R^t_{\psi aj} \end{pmatrix}$$

and

$$F = \begin{pmatrix} E \\ \mathbf{0} \end{pmatrix}$$

then

$$\mathcal{P} = \left\{T \in \mathbb{Z}^N_{SA} : \begin{pmatrix} \mathbf{0} \\ L \end{pmatrix} \leq WY - F \leq \begin{pmatrix} \mathbf{0} \\ U \end{pmatrix}\right\} \tag{22}$$

We use a greedy algorithm to eliminate variables from Y and equality constraints from W one at a time until N_e variables and all equality constraints have been eliminated. The remaining variables in Y will make up the non-basic variables and the remaining constraints will define a basis in the constraint space.

Given a set of constraints in the form (22), if we choose an element W^i_j to be a *pivot point* such that $W^i_j = \pm 1$ and $i < N_e$, we can eliminate variable Y^j and the equality constraint W^*_i by performing Gaussian elimination on the j^{th} column of W and removing row i and column j from W, so that

$$\begin{pmatrix} \mathbf{0} \\ L \end{pmatrix} \leq GWH\hat{Y} - GF \leq \begin{pmatrix} \mathbf{0} \\ U \end{pmatrix}$$

Aget type	Adjacent agents	Behaviour	Probability
Prey	No predators	die	0.100
		give birth	0.156
		move	0.744
Prey	Predators > 0	die	0.100
		give birth	0.156
		be eaten	0.300
		move	0.444
Predator	No prey	die	0.100
		move	0.900
Predator	Prey > 0	die	0.100
		give birth	0.300
		move	0.600

Table 1: The probabilities of each behaviour in the predator-prey model

where $G \in \mathbb{Z}_{N_e}^{N_e-1}$

$$G = \begin{pmatrix} 1 & & & -\frac{W_j^0}{W_j^i} & & \\ & \ddots & & \vdots & & \\ & & 1 & -\frac{W_j^{i-1}}{W_j^i} & & \\ & & & -\frac{W_j^{i+1}}{W_j^i} & 1 & \\ & & & \vdots & & \ddots \\ & & & -\frac{W_j^n}{W_j^i} & & 1 \end{pmatrix}$$

$\hat{Y}$ is Y with the j^{th} element removed and H is the identity matrix with the j^{th} column removed. Letting $\hat{W} = GWH$ and $\hat{F} = GF$ recovers the canonical form

$$\begin{pmatrix} \mathbf{0} \\ L \end{pmatrix} \leq \hat{W}\hat{Y} - \hat{F} \leq \begin{pmatrix} \mathbf{0} \\ U \end{pmatrix}$$

Since $W_j^i = \pm 1$, B^{-1} exists and is integer, so $\hat{Y}$ maps to a unique integer trajectory, as required.

Note that WG differs from G in $(\left|G_j^*\right|_0 - 1)(\left|G_*^i\right|_0 - 1)$ elements, where $|\cdot|_0$ is the number of non-zero elements in a tensor. So, this gives an upper bound on the increase in L0-norm, $|WG|_0 - |G|_0$ and so gives a measure of how much sparsity may be lost by pivoting on W_j^i. It has been found that choosing W_j^i to be the element that minimises this measure (known as the Markowitz criterion) is a computationally efficient way of maintaining the sparsity of a matrix while performing Gaussian eliminatination(Markowitz, 1957)(Maros, 2002)(Suhl & Suhl, 1990). So, in order to find a sparse basis we choose the element that minimises $(\left|G_j^*\right|_0 - 1)(\left|G_*^i\right|_0 - 1)$, perform the elimination and repeat until no more equality constraints remain, $N_e = 0$.

5 Spatial predator-prey demonstration

We demonstrate the above techniques on a 32x32 grid of "predator" and "prey" agents. At each timestep a prey agent either moves to an adjacent grid-square (i.e. up, down, left or right with equal probability), dies or gives birth to another prey agent on the an adjacent grid-square. Predators also either move to an adjacent grid-square, die or give birth. However, predators only give birth when there is at least one prey on an adjacent grid-square. In addition, if a prey has any predators on adjacent grid-squares there is a chance it may be eaten. The probability of each behaviour is shown in table 1.

Simulated observation data was generated by profroming a forward simulation of the model. The number of predators and prey in each gridsquare at the start of the simulation was drawn from a Poisson distribution with rate parameter $\lambda = 0.05$. The model was then simulated forward for 16 timesteps. At each timestep, each gridsquare was observed with a probability of 0.02. If an observation was made, the number of predators in the gridsquare was observed. In order to simulate observational noise, each predator was observed with probability 0.9 (i.e. the number observed was drawn from a Binomial distribution). The same procedure was then repeated for prey.

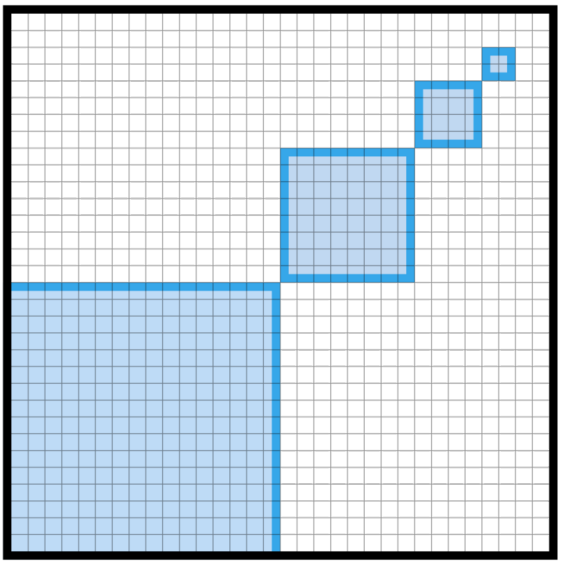

Figure 2: Summary statistics consist of the total number of agents in each of the four shaded regions.

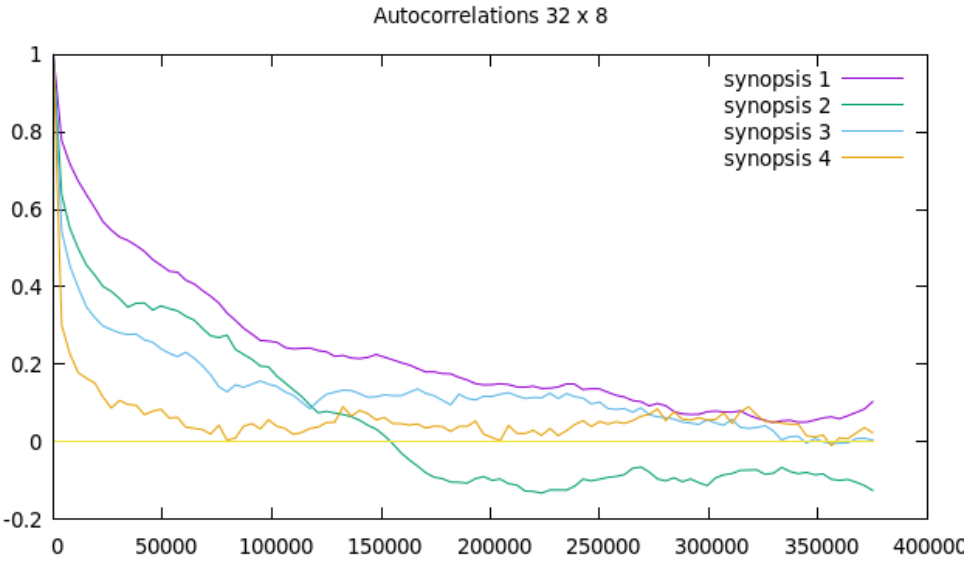

Figure 3: Mean autocorrelation for each scalar of the summary statistics, averaged over all chains.

Four separate Markov chains of 1,500,000 samples each were generated from the posterior using the method described above. The start state for each chain was drawn from the prior (i.e. model trajectory without any observations). The first 200,000 samples of each chain were discarded and the remaining samples were split into first and last halves to give 8 sample sequences in total.

In order to assess the convergence of the chains, a set of summary statistics were calculated for each sample of each chain. The summary statistics consisted of the total number of agents within the shaded regions shown in figure 2 measured after the last timestep of the trajectory. This arrangement of regions was chosen in order to capture the convergence at different spatial scales. From the summary statistics, we calculated the the Gelman-Rubin diagnostic (Gelman & Rubin, 1992), and approximated the autocorrelation at various time lags. Using the notation x_{ij} to refer to a scalar statistic of the i^{th} sample of the j^{th} sequence, given m sequences of n samples each, let $\bar{x}_j$ be the mean of the j^{th} sequence and $\bar{x}$ be the mean over all sequences

$$\bar{x}_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij}, \bar{x} = \frac{1}{m}\sum_{j=1}^{m} \bar{x}_j$$

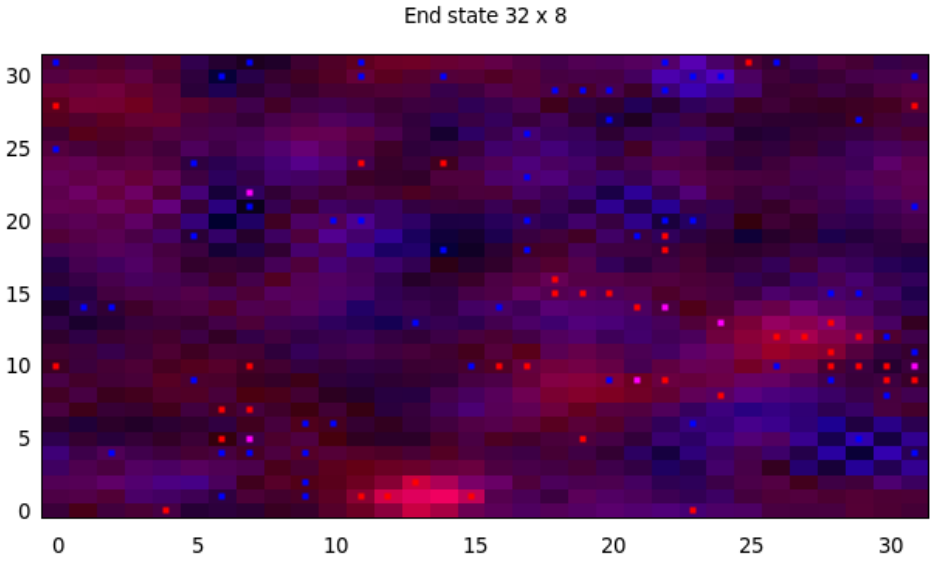

Figure 4: Mean number of agents in each gridsquare after the last timestep, averaged over all samples. Blue represents predator, red represents prey. The dots represent the final positions of agents in the simulation from which the observations were generated. The background colour of each square has red/blue intensity proportional to the log of the mean number of prey/predators in that square, respectively.

let W be the within-sequence variance

$$W = \frac{1}{m}\sum_{j=1}^{m}\frac{1}{n-1}\sum_{i=1}^{n}(x_{ij} - \bar{x}_j)^2$$

and B be the between-sequence variance

$$B = \frac{n}{m-1}\sum_{j=1}^{m}(\bar{x}_j - \bar{x})^2$$

Following(Gelman, Carlin, Stern, & Rubin, 2013), an overapproximation of the true variance of the statistic can be calculated as

$$\widehat{\text{var}}^+ = \frac{n-1}{n}W + \frac{1}{n}B$$

and the Gelman-Rubin statistic can be defined as

$$\hat{R} = \sqrt{\frac{\widehat{\text{var}}^+}{W}}$$

This gives a measure of the uncertainty in the standard deviation of the statistic due to the fact that we have only taken a finite number of samples. If this number is close to 1, then we have some justification in believing that we have taken enough samples.

Also following(Gelman et al., 2013), we approximated the autocorrelation as

$$\hat{\rho}_t = 1 - \frac{V_t}{2\widehat{\text{var}}^+}$$

where

$$V_t = \frac{1}{n-t}\sum_{i=1}^{n-t}(x_i - x_{i+t})^2$$

Figure 3 shows $\hat{\rho}_t$ as a function of t for each scalar in the summary statistics. A decline of $\hat{\rho}_t$ to zero long before t reaches the total number of samples, shows good convergence of the chain. The rate of this decline can be used to calculate the effective number of samples, defined as

$$\hat{n}_e = \frac{mn}{1 + 2\sum_t \hat{\rho}_t}$$

where the sum runs until $\hat{\rho}_t \leq 0$. This approximates of the number of samples from a perfect IID sampler that would give the same uncertainty in the mean of the statistic.

Finally, figure 4 shows the mean number of agents at the end of the simulation, averaged over all samples. This is a visual way of showing that the observations have given rise to a posterior that gives some information about the true positions of the agents.

6 Limitations and further work

[column generation for larger internal state]

[automatic linearisation]

[extension to PPL]

[model error]

[automated temperature setting]

7 Conclusion

We have described and demonstrated an algorithm to sample model trajectories from the posterior distribution of an agent based model given a set of observations. This allows us to use ABMs to perform inference about unobserved variables given a set of observations and to make forecasts given observations up to the present.

As presented, the algorithm is not applicable to agents that have a large amount of internal state (above a few bytes) but we suggest ways that this algorithm can be extended to apply in these cases.

Data assimilation with agent based models is currently in its infancy, and more powerful techniques for performing Bayesian inference with ABM could transform the usefulness and applicability of these models. The

References

Bagnara, R., Hill, P. M., & Zaffanella, E. (2008). The parma polyhedra library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems. *Science of Computer Programming*, *72*(1-2), 3–21.

Baumert, S., Ghate, A., Kiatsupaibul, S., Shen, Y., Smith, R. L., & Zabinsky, Z. B. (2009). Discrete hit-and-run for sampling points from arbitrary distributions over subsets of integer hyperrectangles. *Operations Research*, *57*(3), 727–739.

Becchi, A., & Zaffanella, E. (2018). An efficient abstract domain for not necessarily closed polyhedra. In *International static analysis symposium* (pp. 146–165).

Clay, R., Kieu, L.-M., Ward, J. A., Heppenstall, A., & Malleson, N. (2020). Towards Real-Time Crowd Simulation Under Uncertainty Using an Agent-Based Model and an Unscented Kalman Filter. In Y. Demazeau, T. Holvoet, J. M. Corchado, & S. Costantini (Eds.), *Advances in Practical Applications of Agents, Multi-Agent Systems, and Trustworthiness. The PAAMS Collection* (Vol. 12092, pp. 68–79). Cham: Springer International Publishing. doi: 10.1007/978-3-030-49778-1_6

Cousot, P., & Cousot, R. (1977). Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proceedings of the 4th acm sigact-sigplan symposium on principles of programming languages* (pp. 238–252).

Cousot, P., & Halbwachs, N. (1978). Automatic discovery of linear restraints among variables of a program. In *Proceedings of the 5th acm sigact-sigplan symposium on principles of programming languages* (pp. 84–96).

Fukuda, K. (2020). Polyhedral computation.
doi: https://doi.org/10.3929/ethz-b-000426218

Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (2013). *Bayesian data analysis* (3rd ed.). Retrieved 06/01/22, from `http://www.stat.columbia.edu/~gelman/book/`

Gelman, A., & Rubin, D. B. (1992). Inference from iterative simulation using multiple sequences. *Statistical science*, *7*(4), 457–472.

Gurfinkel, A., & Navas, J. A. (2021). Abstract interpretation of LLVM with a region-based memory model. In *Software verification - 13th international conference, VSTTE 2021, and 14th international workshop, NSV 2021, october 18-19, 2021, revised selected papers.*

Halbwachs, N., Proy, Y.-E., & Roumanoff, P. (1997). Verification of real-time systems using linear relation analysis. *Formal Methods in System Design*, *11*(2), 157–185.

Henry, J., Monniaux, D., & Moy, M. (2012). Pagai: A path sensitive static analyser. *Electronic Notes in Theoretical Computer Science*, *289*, 15–25.

Jeannet, B., & Miné, A. (2009). Apron: A library of numerical abstract domains for static analysis. In *International conference on computer aided verification* (pp. 661–667).

Kalnay, E. (2003). *Atmospheric Modeling, Data Assimilation and Predictability*. Cambridge University Press.

Kieu, L.-M., Malleson, N., & Heppenstall, A. (2020). Dealing with uncertainty in agent-based models for short-term predictions. *Royal Society Open Science*, *7*(1), 191074. doi: 10.1098/rsos.191074

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *science*, *220*(4598), 671–680.

Lewis, J. M., Lakshmivarahan, S., & Dhall, S. (2006). *Dynamic Data Assimilation: A Least Squares Approach*. Cambridge: Cambridge University Press.

Lloyd, D. J. B., Santitissadeekorn, N., & Short, M. B. (2016). Exploring data assimilation and forecasting issues for an urban crime model. *European Journal of Applied Mathematics*, *27*(Special Issue 03), 451–478. doi: 10.1017/S0956792515000625

Lueck, J., Rife, J. H., Swarup, S., & Uddin, N. (2019). Who Goes There? Using an Agent-based Simulation for Tracking Population Movement. In *Winter Simulation Conference, Dec 8 - 11, 2019.* National Harbor, MD, USA.

Malleson, N., Minors, K., Kieu, L.-M., Ward, J. A., West, A., & Heppenstall, A. (2020). Simulating crowds in real time with agent-based modelling and a particle filter. *Journal of Artificial Societies and Social Simulation*, *23*(3), 3. doi: 10.18564/jasss.4266

Markowitz, H. M. (1957). The elimination form of the inverse and its application to linear programming. *Management Science*, *3*(3), 255–269.

Maros, I. (2002). *Computational techniques of the simplex method* (Vol. 61). Springer Science & Business Media.

Meel, K. S., Vardi, M. Y., Chakraborty, S., Fremont, D. J., Seshia, S. A., Fried, D., ... Malik, S. (2016). Constrained sampling and counting: Universal hashing meets sat solving. In *Workshops at the thirtieth aaai conference on artificial intelligence.*

Motzkin, T. S., Raiffa, H., Thompson, G. L., & Thrall, R. M. (1953). The double description method. *Contributions to the Theory of Games*, *2*(28), 51–73.

Quinton, P., Rajopadhye, S., & Risset, T. (1996). On manipulating z-polyhedra. *Technical Report 1016, IRISA, Campus Universitaire de Bealieu, Rennes, France*.

Reichle, R. H. (2008, November). Data assimilation methods in the Earth sciences. *Advances in Water Resources*, *31*(11), 1411–1418. doi: 10.1016/j.advwatres.2008.01.001

Sjödin, J., Pop, S., Jagasia, H., Grosser, T., Pop, A., et al. (2009). Design of graphite and the polyhedral compilation package. In *Gcc developers' summit* (p. 113).

Suhl, U. H., & Suhl, L. M. (1990). Computing sparse lu factorizations for large-scale linear programming bases. *ORSA Journal on Computing*, *2*(4), 325–335.

Talagrand, O. (1997). Assimilation of Observations, an Introduction. *Journal of the Meteorological Society of Japan. Ser. II*, *75*(1B), 191–209.

Tang, D. (2022). *Mutable categorical distribution.* GitHub. Retrieved 06/01/22, from `https://github.com/danftang/MutableCategoricalDistribution`

Thiele, J. C., Kurth, W., & Grimm, V. (2014). Facilitating Parameter Estimation and Sensitivity Analysis of Agent-Based Models: A Cookbook Using NetLogo and 'R'. *Journal of Artificial Societies and Social Simulation*, *17*(3). doi: 10.18564/jasss.2503

Wang, M., & Hu, X. (2015). Data assimilation in agent based simulation of smart environments using particle filters. *Simulation Modelling Practice and Theory*, *56*, 36–54. doi: 10.1016/j.simpat.2015.05.001

Ward, J. A., Evans, A. J., & Malleson, N. S. (2016). Dynamic calibration of agent-based models using data assimilation. *Royal Society Open Science*, *3*(4). doi: 10.1098/rsos.150703