Check for updates

METHOD ARTICLE

Data assimilation with agent-based models using Markov chain sampling

[version 1; peer review: 3 approved with reservations]

Daniel Tang [1], Nick Malleson [1-3]

[1]Leeds Institute for Data Analytics, University of Leeds, Leeds, UK
[2]School of Geography, University of Leeds, Leeds, UK
[3]Alan Turing Institute, London, UK

v1 **First published:** 31 May 2022, **2**:70
https://doi.org/10.12688/openreseurope.14800.1
Latest published: 31 May 2022, **2**:70
https://doi.org/10.12688/openreseurope.14800.1

Abstract

Every day, weather forecasting centres around the world make use of noisy, incomplete observations of the atmosphere to update their weather forecasts. This process is known as data assimilation, data fusion or state estimation and 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 unobserved quantities or latent variables at some time, possibly in the future?

While data assimilation has developed rapidly in some 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 quantitative claims about real-world systems.

Here we present an algorithm that uses Markov-Chain-Monte-Carlo (MCMC) methods to generate samples of the parameters and trajectories of an agent-based model over a window of time given a set of possibly noisy, aggregated and incomplete observations of the system. This can be used as-is, or as part of a data assimilation cycle or sequential-MCMC algorithm.

Our algorithm is applicable to time-stepping, agent-based models whose agents have a finite set of states and a finite number of ways of acting on the world. As presented, the algorithm is only practical for agents with a few bytes of internal state although we discuss ways of removing this restriction. We demonstrate the algorithm by performing data assimilation with an agent-based, spatial predator-prey model.

Keywords

Agent-based model, data assimilation, Bayesian inference, MCMC, Integer linear programming, predator prey model

Open Peer Review

Approval Status ? ? ?

	1	2	3
version 1 31 May 2022	? view	? view	? view

1. **Blas Kolic**, University of Oxford, Oxford, UK
2. **Yilin Huang**, Delft University of Technology, Netherlands, The Netherlands
3. **Corrado Monti**, CENTAI, Turin, Italy

Any reports and responses or comments on the article can be found at the end of the article.



This article is included in the Data Science gateway.



This article is included in the Horizon 2020 gateway.



This article is included in the European Research Council (ERC) gateway.

Corresponding author: Daniel Tang (D.Tang@leeds.ac.uk)

Author roles: Tang D: Conceptualization, Formal Analysis, Investigation, Methodology, Resources, Software, Validation, Visualization, Writing – Original Draft Preparation, Writing – Review & Editing; **Malleson N**: Funding Acquisition, Project Administration, Supervision, Writing – Review & Editing

Competing interests: No competing interests were disclosed.

Grant information: 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).
The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Copyright: © 2022 Tang D and Malleson N. This is an open access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

How to cite this article: Tang D and Malleson N. **Data assimilation with agent-based models using Markov chain sampling [version 1; peer review: 3 approved with reservations]** Open Research Europe 2022, **2**:70 https://doi.org/10.12688/openreseurope.14800.1

First published: 31 May 2022, **2**:70 https://doi.org/10.12688/openreseurope.14800.1

1 Introduction

Agent-based models (ABMs) have been widely adopted as an intuitive way to model systems that consist of a heterogeneous collection of autonomous, interacting agents. ABMs are sometimes used to show that an unexpected collective behaviour can emerge from a known agent behaviour. For example Schelling (1971) showed that agents with only a very slight racial bias can quickly form a highly racially-segregated population. This use of ABMs requires no assimilation of real-world data but generates results that cannot be used to assert anything about the real world without also providing a convincing reason to believe that the model is an abstraction of the real-world. In this paper, we'll deal with an alternative use case in which we start with the following information:

- A set of real-world observations, Ω, that may contain aggregated or macro-scale quantities and may be subject to noise during the measurement process.

- An agent-based model expressed as a set of agent behaviours. These will generally be stochastic, in order to account for our uncertainty in agent behaviour, and will be a function of some set of unknown parameters, θ, in order to account for our uncertainty over which behavioural model best represents the real-world target entities. Note that the parameters can, and generally should, contain values that control the size of the model stochasticity in order to account for "unknown unknowns". The agents may also interact with a world outside of the model. These "boundary conditions" necessarily consist of agents being injected into the model at certain times and/or external influences on modelled agents' behaviour[1]. The start state of the model can also be treated as a boundary condition consisting of the injection of agents into the model at time $t = 0$. Although the boundary conditions are semantically distinct from the parameters, their mathematical treatment will be identical so we assume our prior beliefs about boundary conditions is contained in θ. We'll show later how to express an ABM as a "forecast" distribution, $P(\tau|\theta)$, which is the probability that a set of agents would exhibit behaviours τ on a model execution given parameters/boundary conditions θ. We'll call τ a model trajectory.

- A set of probabilistic prior beliefs, $P(\theta)$, about the model parameters and boundary conditions (after accounting for any micro-calibration or other relevant direct observations we may have).

Our aim will be to calculate the expectation of some (possibly vector-valued) unobserved quantity, Λ, over the posterior distribution $P(\tau, \theta|\Omega)$

$$\mathbb{E}_{P(\tau,\theta|\Omega)}(\Lambda(\tau,\theta)) = \int \Lambda(\tau,\theta)P(\tau,\theta\,|\Omega)d\tau d\theta. \tag{1}$$

Inserting Bayes' rule gives us

$$\mathbb{E}_{P(\tau,\theta|\Omega)}(\Lambda(\tau,\theta)) = \int \Lambda(\tau,\theta)\frac{P(\Omega|\,\tau)P(\tau\,|\,\theta)P(\theta)}{P(\Omega)}d\tau d\theta \tag{2}$$

where $P(\Omega|\tau)$ is the observation likelihood, which we assume is easy to calculate given a trajectory that covers the observed period, and $P(\Omega)$ is just a normalising constant which is not usually explicitly calculated when sampling. Note that any information about $P(\tau, \theta|\Omega)$ can be expressed without loss of generality in this way.

There are various approaches in the literature to solving this equation for ABMs. One strategy is to first transform the problem into a simpler form, such as a set of partial differential equations (Lloyd *et al.*, 2016) or a graphical model (Liao & Barooah, 2010), for which there exists a well known method of solving the problem. Tang (2019) expressed the problem in terms of creation and annihilation operators then used symbolic computation to evaluate the integral in (2). However, the most popular strategy in the literature is to sample from the posterior, $P(\tau, \theta|\Omega)$, and use the samples to approximate the integral using Monte Carlo integration. This is the technique we employ here.

1.1 Importance sampling from the prior

The simplest Monte Carlo approach would be to approximate $\mathbb{E}_{P(\tau,\theta|\Omega)}(\Lambda(\tau,\ \theta))$ using importance sampling as follows:

1. Taking a sample from the prior parameters/boundary conditions $\theta_i \sim P(\theta)$.
2. Executing the model forward with the given θ_i to generate a trajectory $\tau_i \sim P(\tau|\theta_i)$.
3. Calculating a weight $w_i = P(\Omega|\tau_i)$ to create a weighted sample $\langle\tau_i, \theta_i, w_i\rangle$.

[1] we take the purist view that an ABM models everything as an agent

4. Repeating N times from step 1 to get a set of weighted samples $\{\langle\tau_1, \theta_1, w_1\rangle \ldots \langle\tau_N, \theta_N, w_N\rangle\}$.
5. Approximating $\mathbb{E}_{P(\tau,\theta|\Omega)}(\Lambda(\tau,\theta)) \approx \frac{1}{\sum_{j=1}^{N} w_j} \sum_{i=1}^{N} w_i \Lambda(\tau_i,\theta_i)$.

However, Chatterjee & Diaconis (2018) showed that the information contained in the weighted samples decreases exponentially with the KL-divergence from the prior to the posterior, $D_{KL}\,(P(\tau, \theta|\Omega) \parallel P(\tau|\theta)P(\theta))$. This divergence can be thought of as a measure of the information contained in the observations Ω. For reasons we'll see later, in the case of ABMs, the observations usually contain enough information to make the set of weighted samples less informative than a single unweighted sample, even when N is astronomically high. Therefore, this algorithm will not work unless the observations contain very little information.

1.2 Particle filtering

In the literature, the most popular way of solving (2) for ABMs is to use some form of particle filtering, also known as "sequential Monte Carlo". This splits the time period of interest into smaller, contiguous time windows with (not necessarily equidistant) end times $\langle t_1 \ldots t_N\rangle$ where $t_m < t_{m+1}$. The trajectory and observations are similarly split based on which window they occur in, $\tau = \langle\tau_1 \ldots \tau_N\rangle$ and $\Omega = \langle\Omega_1 \ldots \Omega_N\rangle$. The posterior distribution of the whole trajectory can then be split into the product of contributions from each window given the distribution over the previous window, leading to a recursion relation

$$P(\tau_{1:t+1},\theta\,|\Omega_{1:t+1}) = \frac{P(\Omega_{t+1}|\,\tau_{t+1})P(\tau_{t+1}\,|\,\tau_{1:t},\theta)P(\tau_{1:t},\theta\,|\Omega_{1:t})}{P(\Omega_{t+1}|\Omega_{1:t})} \tag{3}$$

where $\tau_{1:t} = \langle\tau_1 \ldots \tau_t\rangle$ and $P(\tau_{t+1}|\tau_{1:t}, \theta)$ is the probability that an execution of the ABM that starts with trajectory $\tau_{1:t}$ would go on to produce τ_{t+1} given the parameters/boundary conditions θ. The final posterior $P(\tau_{1:N}, \theta|\Omega_{1:N})$ can now be built up in steps, starting with the first window and moving to the last, using the recursion as we go and approximating each $P(\tau_{1:t}, \theta|\Omega_{1:t})$ as a set of samples, or "particles". This makes the problem easier in two respects: firstly the dimension of the distributions we need to deal with at each recursion step are reduced, since if we consider each particle separately, we're only concerned with the trajectory in one window at a time; secondly, we split the information in the observations into smaller chunks, so at each recursion the information in Ω_t is more likely to be small enough to make it practical to perform importance sampling from the prior.

If the unobserved quantity of interest, Λ, depends only on the states of the agents, σ_N, at time t_N and/or the parameters/boundary conditions, θ, rather than on the full trajectory, $\tau_{1:N}$, then we can reduce the dimensionality even further by performing the recursion on model states rather than trajectories (where a model state tells us how many agents there are in each agent state). First, we arrange the windows so that each observation lies at the end of a window, then marginalise equation (3) over $\tau_{1:t+1}$ for a fixed σ_{t+1} to get

$$P(\sigma_{t+1},\theta\,|\Omega_{1:t+1}) = \frac{P(\Omega_{t+1}|\,\sigma_{t+1})}{P(\Omega_{t+1}|\Omega_{1:t})} \int P(\sigma_{t+1}\,|\,\sigma_t,\theta)P(\sigma_t,\theta\,|\Omega_{1:t})\,d\sigma_t \tag{4}$$

where we've assumed that the observation likelihoods depend only on the model state and made explicit in $P(\sigma_{t+1}|\sigma_t, \theta)$ that the probability of the state σ_{t+1} depends only on σ_t and θ[2]. Note that the definition of "model state" can always be re-defined in such a way as to make these assumptions true. In fact, we can see that equation (3) can be thought of as a special case of equation (4) where the model state, σ_t, is replaced by the trajectory, $\tau_{1:t}$, in which case the integral over σ_t is non-zero for only a single trajectory, since $P(\tau'_{1:t+1}|\tau_{1:t}, \theta) = \delta_{\tau 1:t}(\tau'_{1:t})P(\tau'_{t+1}|\tau_{1:t}, \theta)$, where δ here is the multivariate delta distribution.

So, we can consider a recursion on generalised trajectories, X_t,

$$P(X_{t+1}\,|\Omega_{1:t+1}) = \frac{P(\Omega_{t+1}|\,X_{t+1})}{P(\Omega_{t+1}|\Omega_{1:t})} \int P(X_{t+1}\,|\,X_t)P(X_t\,|\Omega_{1:t})\,dX_t \tag{5}$$

so that equation (3) and (4) would correspond to $X_t = \langle\tau_{1:t}, \theta\rangle$ and $X_t = \langle\sigma_t, \theta\rangle$ respectively.

A particle filter solves this recursion by taking a set of samples $X_t^{1:R}$ drawn from $P(X_t|\Omega_{1:t})$, and generating a set of samples $X_{t+1}^{1:R}$ drawn from $P(X_{t+1}|\Omega_{1:t+1})$. The simplest algorithm to do this is known as sequential importance resampling, which consists of the following steps:

1. Generate a set of samples $X_0^{1:R}$ from the prior $P(X_0)$. Set $t = 0$.

[2]technically, if θ contains time-dependent boundary conditions, the model states need to be timestamped so that the model knows which boundary conditions to apply

2. For each sample X_t^i, generate a sample from the forecast $\hat{X}_{t+1}^i \sim P(X_{t+1}|X_t^i)$ by executing the ABM from the end state of X_t^i to give a set of forecast samples $\hat{X}_{t+1}^{1:R}$.

3. Calculate a weight for each forecast sample $\hat{X}_{t+1}^i$

$$w_i = \frac{P(\Omega_{t+1}|\,\hat{X}_{t+1}^i)}{\sum_{j=1}^{R} P(\Omega_{t+1}|\,\hat{X}_{t+1}^j)}.$$

4. Resample from the weighted samples by taking R samples from the approximation

$$P(X_{t+1}\,|\Omega_{1:t+1}) \approx \sum_i w_i\, \delta_{\hat{X}_{t+1}^i}(X_{t+1}) \tag{6}$$

to give a new set of unweighted samples $X_{t+1}^{1:R}$. There are a few ways of performing this resampling (Douc & Cappé, 2005), the simplest being to draw an integer $i \in [1, R]$ with probability w_i, returning the sample $\hat{X}_{t+1}^i$ and repeating R times.

5. If $t < N$, increment t and repeat from step 2.

However, this algorithm suffers from the problem of "sample impoverishment" (Li *et al.*, 2014) or "sample deprivation" which describes the situation when the resampling step leaves many particles in the same state, so the number of distinct samples in our sample set is much smaller than the total number of samples. This is because, as we've seen, after each importance sampling step the information in the observations causes the effective sample size to decrease and therefore the resampling step is likely to generate repeated samples. This loss of effective sample size can accumulate and in practice as the number of windows increases, the number of particles required to maintain a desired accuracy often increases exponentially. In the worst case, we encounter an observation that is impossible for all particles and the algorithm fails completely. Malleson *et al.* (2020) showed that sample impoverishment is an issue when applied to data assimilation with an ABM of crowd movement. Khan *et al.* (2003) encountered the same problem in a model of ant movement.

To get a better understanding of impoverishment, suppose we use the above algorithm to generate a set of samples $\{\langle\tau_{1:N}^1,\theta^1\rangle \ldots \langle\tau_{1:N}^S,\theta^S\rangle\}$ of the full trajectory $P(\tau_{1:N}, \theta|\Omega_{1:N})$, from which we generate marginalisations over each window $W_t = \{\langle\tau_t^1, \theta^1\rangle \ldots \langle\tau_t^S, \theta^S\rangle\}$. On the one hand, the repeated impoverishment at each step causes the number of distinct values in W_{N-L} to decrease exponentially as the lag, L, increases (*i.e.* as we look further back in time), meaning we lose information about the true distribution of $P(\tau_{N-L}, \theta|\Omega_{1:N})$. However, on the other hand, the model dynamics and observations make $P(\tau_N, \theta|\Omega_{1:N})$ increasingly independent of $P(\tau_{N-L}, \theta|\Omega_{1:N})$ as L increases. So if all we care about is the final window, W_N, (which is often the case) impoverishment only becomes a problem when the first effect (the rate of loss of information) dominates the second (the rate of increase in independence from the past).

Unfortunately every window's trajectory depends on the parameters, θ, but these are sampled in the first window and never subsequently change, so after assimilating only a few windows, all samples are likely to have collapsed to the same value of θ, giving us little information about the true distribution of the marginalised posterior $P(\theta|\Omega_{1:N})$. This is fine if the parameters are known at $t = 0$, but if there is prior uncertainty in θ and the observations don't reduce that uncertainty faster than the collapse in information contained in the samples, then impoverishment will be a problem (Andrieu *et al.*, 2004; Liu & West, 2001).

A simple way of dealing with sample impoverishment is to add noise to the particles at each step. This is known as roughening (Gordon *et al.*, 1993; Li *et al.*, 2014) and is equivalent to replacing the delta function in equation (6) with some other function with wider support so we end up with a weighted kernel density estimation. Kieu *et al.* (2020) used this technique in the context of an ABM of public transport. However, Liu & West (2001) showed that this leads to over-dispersion unless we "shrink" the samples towards their mean value. More generally, the consequence of roughening is that a finite number of samples from a roughened particle filter is no longer a draw from the true posterior, when averaged over all possible draws of the same size, so we need to convince ourselves that the error introduced by the roughening process is small enough for our requirements; not always an easy task for an ABM where distributions can be highly discontinuous.

Wang & Hu (2015) dealt with sample impoverishment in the context of ABM by using the samples at time t to approximate the marginal probabilities of the state of each individual agent at the start of the next window, $P(a_j)$, then drawing new samples from the product of the marginals $P(a_1 \ldots a_n) = \prod_{j=1}^{N} P(a_j)$, *i.e.* sampling the state of the j^{th} agent from the marginal $P(a_j)$. This improves sample diversity at the price of losing all correlations between agents.

In the context of ABMs, sample impoverishment can manifest in its most extreme form by generating particles that have zero weight and so contribute no information to the resampling step. This happens because a draw from the forecast of an ABM $P(X_{t+1}|X_t)$ often has an extremely high probability of having a likelihood $P(\Omega_t|X_{t+1})$ of zero. For example, suppose we're tracking mobile phones by IMEI and receive intermittent signals from the phones identifying which cell they're in. Suppose for simplicity that if a phone is in a known cell at time t then it could be in any of eight cells at time $t + 1$, each with equal prior probability. Suppose at time $t + 1$ we get signals from 10 phones. Even if we know the correct model state at time t, the probability that a sample from the forecast at time $t + 1$ has a non-zero likelihood is 8^{-10}, about one in a billion. So sequential importance resampling will fail in a single step in this case.

1.3 Kalman filtering

One potential solution to the problem of impoverishment is to assume that some or all of the distributions are Gaussian. This is the basis of the data assimilation algorithms that have had a great deal of success in geophysical models (Carrassi *et al.*, 2018; Kalnay, 2003; Lewis *et al.*, 2006; Talagrand, 1997). If the forecast $P(X_{t+1}|\Omega_{1:t})$ and likelihood function $P(\Omega_{t+1}|X_{t+1})$ are multivariate Gaussians, the recursion in (5) can be solved analytically and evaluated even when the observations Ω_t are highly informative. This idea leads to the Ensemble Kalman filter (Evensen, 2003) and the Unscented Kalman filter (Wan *et al.*, 2001), both of which have been applied to data assimilation in ABM (Clay *et al.*, 2020; Ward *et al.*, 2016).

However, the Gaussian assumption is not usually a good one for ABMs. For example, suppose there is a sports hall containing N people and we take the model state to be the $2N$ dimensional vector consisting of the (x, y) coordinates of each person. Even if the initial state is distributed according to a $2N$ dimensional Gaussian distribution, the either/or decisions of the people can quickly transform this into a multimodal distribution. Suppose two exits are opened on opposite sides of the hall and each person moves towards their nearest exit. Each person makes an either/or decision which exit to move towards based on their position, so we can think of the prior Gaussian as being partitioned into 2^N regions corresponding to all possible sets of decisions made by the people. In the forecast distribution, the probability mass in each region will move towards a different model state (one of the 2^N states where all agents are at one or other of the exits), creating a highly non-Gaussian distribution with 2^N modes. In addition, the likelihood function is often highly non-Gaussian in this state space. For example, suppose there is a sensor at one of the exits that trips if any agents are close. Suppose for example that there are just two agents so the model state space is (x_1, y_1, x_2, y_2) and the origin of the agents' coordinate system is at the sensor. If the sensor trips, the likelihood function is zero everywhere except close to the perpendicular planes $(0, 0, x_2, y_2)$ and $(x_1, y_1, 0, 0)$. This clearly isn't well approximated by any Gaussian. This is a consequence of the fact that we don't know which agent caused the sensor to trip, a problem known as the "data association problem" (Lueck *et al.*, 2019). This problem becomes superexponentially worse if we have multiple sensors. If there are N agents and $M \leq N$ sensors the likelihood function is the union of $\frac{N!}{(N-M)!}$ hyperplanes.

An alternative representation of the model state space is the occupation number representation where each agent state becomes a dimension in the state space and the coordinate values give the number of agents in each state. In this representation, Gaussians may be appropriate in some models if there are a large number of agents in every state. However, as the occupation numbers get smaller, an increasing proportion of a Gaussian approximation lies on states that contain negative occupation numbers, which is problematic as a negative occupation number has no natural interpretation. In very high dimensions, the vast majority of the probability mass will lie in a region with at least one negative occupation number. Likelihood functions in this space can also be highly non-Gaussian. For example, the sensor in (a discretised version of) the sports hall example above would have a likelihood function that is a step function on the dimension that corresponds to the state close to the door.

1.4 Metropolis-Hastings

In the following sections we present a solution to these problems by using the Metropolis-Hastings algorithm to construct a Markov process whose stationary state is $P(\tau_{1:N}, \theta|\Omega_{1:N})$. To our knowledge this is the first time this has been done in a way that is applicable to a wide range of ABMs and observations. For small N we can sample directly from this Markov process to get samples from the posterior. For larger N, or if we wish to perform online data assimilation where the observations form a continuous, effectively endless stream, we propose using the algorithm as part of a data assimilation cycle or MCMC Particle Filter algorithm (Finke *et al.*, 2020; Septier *et al.*, 2009). There are many specific ways of implementing this; unfortunately the most appropriate algorithm to use depends on the nature of the ABM dynamics, the likelihood function and the observation operator, Λ, so there's no silver bullet. We discuss this further in Section 5 and present some algorithms that are likely to be appropriate in the context of ABMs.

Since the nature of an ABM's parameters is highly model-specific, in the following we focus on the treatment of the trajectory, although in a way that is fully integrated with a model-specific treatment of the parameters for which we give more general design advice. In the case of state estimation (where the parameters are already known) the parameter-specific part can be left out completely.

2 Methods

2.1 Defining the problem

2.1.1 Definition of an ABM. Suppose we have a timestepping ABM that consists of agents with a finite number of possible internal states and a finite number of mutually exclusive ways of acting on their world. Given this, we formally 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 an S dimensional 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 $F(\psi, a)$ is an S dimensional 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 as follows:

Agent states An agent can be either a cat or a mouse and can be on one of two grid-squares, left or right, so

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

Agent actions In any given timestep, an agent can either move to the other grid-square or stay still, so

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

Agent timestep In a timestep, a cat will move or stay still with probability 0.5, a mouse will move if there are any cats on the same gridsquare or stay still otherwise. This behaviour can be expressed as the agent timestep:

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

Action function This gives the result of an agent's actions. For example, $F(1, 0)$ is the result of an agent in state $\psi = 1$ (*right cat*) performing the action $a = 0$ (*move*), which is one cat in state $\psi = 0$ (*left cat*). So $F(1, 0) = \{1, 0, 0, 0\}$. This model is simple enough that we can write down F explicitly for all ψ and a:

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

2.1.2 A note on tensor notation. In the rest of this paper we'll make use of multidimensional arrays. These are just arrays of numbers, like vectors or matrices, but arranged in any number of dimensions. Each array has a shape which tells us how many dimensions it has and the number of elements in each dimension. For notational

convenience, we'll distinguish between subscript dimensions and superscript dimensions[3] and will refer to the set of all arrays of a given shape with the symbol $\mathbb{R}$ adorned with the size of each subscript and superscript dimension. So, for example, $\mathbb{R}^N_{SA}$ describes the set of all three-dimensional arrays that have one superscript dimension of size N and two subscript dimensions of sizes S and A.

An element of an array is referred to by specifying the sub- and superscript 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 superscript dimension and coordinates (ψ, a) in the subscript dimensions. By convention, coordinates begin at 0.

We'll also borrow from tensor notation by using the Einstein summation convention, meaning that if the same index symbol is repeated in sub- and superscript positions in a term, then a summation over that dimension is implied. So, for example if $X \in \mathbb{R}_8$ and $Y \in \mathbb{R}^8$ then

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

When the same symbol is repeated in the same position with no implied summation, this implies universal quantification. For example if $Y \in R_8$ then

$$X_\psi = Y_\psi$$

is equivalent to

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

We refer to "slices" of an array by using the * symbol in an index position. For example if $T \in \mathbb{R}^N_{SA}$ then $T^t_{\psi *}$ refers to the one-dimensional array in $\mathbb{R}_A$ comprised of the elements $\langle T^t_{\psi 0} \cdots T^t_{\psi A-1} \rangle$.

The symbols **0** and **1** represent arrays whose elements are all 0 or 1 respectively. Their shape should be unambiguous from the context.

2.1.3 Trajectories. Let a model timestep be an array $E \in \mathbb{R}_{SA}$ whose elements $E_{\psi a}$ give the number of agents in state σ_ψ that perform act α_a in this timestep. For example, the timestep shown in Figure 1 for the cat and mouse example would be

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

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).

Let a model trajectory, T, be an array 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.

[3]covariant and contravariant dimensions if you prefer

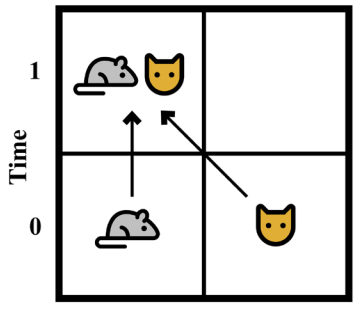


Figure 1. A simple cat and mouse model.

An array 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 denote the set of all non-negative integer arrays

$$\mathbb{N}_{SA}^{N} = \{T \in \mathbb{R}_{\mathbb{SA}}^{\mathbb{N}} : T_{\psi a}^{t} \geq \mathbf{0}_{\psi a}^{t}, T_{\psi a}^{t} \in \mathbb{Z}\}. \tag{7}$$

A trajectory must also be continuous, by which we mean that any agent that is the result of an action at timestep $t-1$ must be the cause of an action at timestep t, and any agents injected by the boundary conditions at timestep t must also act in timestep t[4]. We call this the "continuity constraint" and define the set of continuous arrays, with respect to an action function F:

$$\mathcal{C}_{SA}^{N}(F) = \left\{T \in \mathbb{R}_{SA}^{N} : I_{\phi}^{0} = T_{\phi b}^{0}\mathbf{1}^{b}, \forall(1 \leq t < N) : F_{\phi}^{\psi a} T_{\psi a}^{t-1} + I_{\phi}^{t} = T_{\phi b}^{t}\mathbf{1}^{b}\right\} \tag{8}$$

where $F \in \mathbb{R}_{S}^{SA}$ is the action array $F_{*}^{\psi a} = F(\psi, a)$ and I_{ϕ}^{t} is the number of boundary condition injections into state ϕ at time t. 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, $\mathcal{T}_{SA}^{N}(F)$, as a set of arrays that satisfy (7) and (8).

$$\mathcal{T}_{SA}^{N}(F) = \mathbb{N}_{SA}^{N} \cap \mathcal{C}_{SA}^{N}(F). \tag{9}$$

2.1.4 The posterior of an ABM. Our goal is to generate samples from the Bayesian posterior over the trajectories, boundary conditions and model parameters, given a set of observations, Ω

$$P(T, \theta \,|\, \Omega) \propto P(\Omega \,|\, T) P(T \,|\, \theta) P(\theta).$$

The model forecast, $P(T\,|\theta)$, can be decomposed into the product of conditionals for each timestep and agent state

$$P(T \,|\, \theta) \propto P(\Psi_{*}^{0} \,|\, \theta) \prod_{t,\psi} P(T_{\psi *}^{t} \,|\, \Psi_{*}^{t}, \psi, \theta)$$

where $\Psi_{*}^{t} \in \mathbb{N}_{S}$ is the model state at the start of timestep t and $P(\Psi_{*}^{0}|\theta)$ is defined by the boundary conditions[5]. The term $P(T_{\psi *}^{t}|\Psi_{*}^{t},\psi,\theta)$ is the probability that Ψ_{ψ}^{t} agents starting in state ψ will collectively perform the vector of actions $T_{\psi *}^{t}$. The probability that a single agent performs action a is given by the agent timestep $\pi_{\theta}(\psi, \Psi', a)$ (where we've added the subscript θ to emphasise its possible dependence on the parameters). Since agent actions are simultaneous, the collective behaviour is given by the multinomial distribution

$$P(T_{\psi *}^{t} \,|\, \Psi_{*}^{t}, \psi, \theta) = \begin{cases} \Psi_{\psi}^{t}! \prod_{a} \frac{\pi_{\theta}(\psi, \Psi_{*}^{t}, a)^{T_{\psi a}^{t}}}{T_{\psi a}^{t}!} & \text{if } T_{\psi a}^{t}\mathbf{1}^{a} = \Psi_{\psi}^{t} \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

If the trajectory is continuous then $\Psi_{*}^{t} = T_{*a}^{t}\mathbf{1}^{a}$ so the model forecast is given by

$$P(T \,|\, \theta) = \begin{cases} P(\Psi_{*}^{0} = T_{*c}^{0}\mathbf{1}^{c} \,|\, \theta) \prod_{t,\psi} \left(T_{\psi b}^{t}\mathbf{1}^{b}\right)! \prod_{a} \frac{\pi_{\theta}(\psi, T_{*d}^{t}\mathbf{1}^{d}, a)^{T_{\psi a}^{t}}}{T_{\psi a}^{t}!} & \text{if } T \in \mathcal{T}_{SA}^{N}(F) \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

The likelihood function, $P(\Omega|T)$, can also usually 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

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

[4]This does not mean that agents cannot leave or enter the system, only that if they do then that change must be the result of an action or boundary condition.

[5]for notational clarity we assume that all injections are at time $t = 0$, but this can easily be extended to injections at any or all timesteps.

and the posterior can be written as

$$P(T,\theta|\Omega) \propto \begin{cases} P(\theta)P(\Psi^0_* = T^0_{*c}\mathbf{1}^c|\theta)\prod_{(\omega,v)\in\Omega}P(\omega(T)=v)\prod_{t,\psi}\left(T^t_{\psi b}\mathbf{1}^b\right)!\prod_a \frac{\pi_\theta(\psi,T^t_{*d}\mathbf{1}^d,a)^{T^t_{\psi a}}}{T^t_{\psi a}!} & \text{if } T\in\mathcal{T}^N_{SA}(F) \\ 0 & \text{otherwise.} \end{cases} \quad (12)$$

2.2 Approximating the support of the posterior

As we saw in the introduction, sampling from equation (12) is difficult because it is often very hard to find proposal trajectories that have non-zero posterior probability. Our strategy to solve this problem is to first derive an expression for a volume of trajectory/parameter space that bounds the support of the posterior, supp($P(T, \theta|\Omega)$) (*i.e.* the set of $\langle T, \theta\rangle$ pairs that have non-zero posterior probability). If we choose a bounding volume that it is easy to sample from and is a relatively tight approximation of supp($P(T|\Omega)$) then there's a good chance that a sample drawn from this volume will have non-zero probability. The bounding volume doesn't need to be exact, it just needs to be tight enough to give us a good chance of finding a non-zero probability sample.

From equation (12)

$$\begin{aligned} \text{supp}(P(T,\theta|\Omega)) = & \bigcap_{(\omega,v)\in\Omega,t,\psi,a} \mathcal{T}^N_{SA} \cap \text{supp}(P(\theta)) \\ & \text{supp}(P(\Psi^0_* = T^0_{*c}\mathbf{1}^c)|\theta) \cap \\ & \text{supp}(P(\omega(T)=v)) \cap \\ & (\text{supp}(\pi_\theta(\psi,T^t_{*b}\mathbf{1}^b,a)) \cup \{T : T^t_{\psi a} = 0\}) \end{aligned} \quad (13)$$

i.e. in order for $\langle T, \theta\rangle$ to have non-zero posterior probability, θ must have non-zero prior probability and T must be a trajectory of the ABM that has a possible start state, all observations must have non-zero likelihood and each non-zero element of T must denote an agent action with non-zero probability.

2.2.1 Convex mixed-integer polyhedra. We now approximate equation (13) with a convex, mixed-integer polyhedron, $\mathcal{P}$ (see, for example, Conforti *et al.* (2010)), which we define to be a set of vectors, some of whose elements are integer and some real-valued, that satisfy a set of linear inequalities:

$$\mathcal{P} = \{X \in \mathbb{Z}^N \times \mathbb{R}^M : L \le CX \le U\}$$

for some matrix C and some vectors L and U.

Any linear inequality on T and θ can be expressed in this form by letting X consist of the elements of T and θ flattened into one dimension so that

$$X^i = W_t{}^{\psi ai} T^t_{\psi a} + G^i_j \theta^j$$

and W and G have inverses

$$\hat{W}^t_{\psi ai} = W_{t'}{}^{\psi' a' i'} \delta_{\psi'\psi}\delta_{a'a}\delta^{t't}\delta_{i'i}$$

and

$$\hat{G}^j_i = G^{i'}_{j'}\delta_{i'i}\delta^{j'j}$$

such that

$$T^t_{\psi a} = \hat{W}^t_{\psi ai} X^i$$

and

$$\theta^j = \hat{G}^j_i X^i$$

where δ^{ij} and δ_{ij} are 1 if $i = j$ and 0 otherwise. In this way we can switch unambiguously between X and $\langle T, \theta\rangle$.

We now consider how to express each term in equation (13) as a mixed integer polyhedron. From equation (9) we can see immediately that the set of all trajectories, $\mathcal{T}_{SA}^{N}$, is already defined as a set of linear constraints on T. The support of the prior $P(\theta)$ can usually be easily bounded by a hyper-rectangle giving the range of each variable. The supports of the start state, $P(\Psi_*^0|\theta)$, the observations, $P(\omega(T) = v)$, and the agent actions, $\pi_\theta(\psi, T_{*b}^t \mathbf{1}^b, a)$, can often be easily expressed as linear constraints. However, if this is not the case, each of the probability distributions can be expressed as a computer program. 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 (Becchi & Zaffanella, 2018; Cousot & Halbwachs, 1978; Fukuda, 2020) to efficiently calculate a convex polyhedron that bounds the support of the program. Software to perform abstract interpretation using convex polyhedra already exists (Bagnara *et al.*, 2008; Gurfinkel & Navas, 2021; Henry *et al.*, 2012; Jeannet & Miné, 2009) and the technique has been used in applications such as compiler optimization (Sjödin *et al.*, 2009) and verification of safety-critical systems (Halbwachs *et al.*, 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 an real valued auxiliary variable, r, that satisfies $0 \leq r < 1$ for each call to `Random()`. These auxiliary variables can be removed immediately using the abstract interpretation software to project the polyhedron back into the space of X. This relies on converting the polyhedron to its set of vertices (Motzkin *et al.*, 1953) and projecting the vertices, so the number of vertices must be of a tractable size. Alternatively, the auxiliary variables can be added to X and marginalised out when we take the samples.

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.* $\forall \psi, a, t : T_t^{\psi a} \in \{0, 1\}$). Under this assumption, which we'll call the "Fermionic assumption", the "Fermionic trajectories", $\mathcal{F}_{SA}^{N} = \left\{T \in \mathcal{T}_{SA}^{N} : \forall \psi, a, t : T_t^{\psi a} \in \{0,1\}\right\}$ all lie on the corners of the unit hypercube. So the intersection of $\mathcal{F}_{SA}^{N}$ with any set is a convex polyhedron and supp($P(T|\Omega)$) can always be exactly represented as a mixed-integer polyhedron (finding that polyhedron isn't always tractable, though, in which case abstract interpretation software can be used to find a larger polyhedron that contains it).

So, if we let $\mathcal{P}(f)$ be a mixed integer polyhedron that contains supp(f) then from equation (13) we can approximate the support of the posterior as

$$\begin{aligned}\operatorname{supp}(P(T,\theta|\Omega)) \subseteq \mathcal{P}(P(W_t^{\psi a i} T_{\psi a}^t + G_j^i \theta^j|\Omega)) = & \bigcap_{(\omega,v)\in\Omega,t,\psi,a} \mathcal{T}_{SA}^{N} \cap \mathcal{P}(P(\theta)) \\ & \mathcal{P}(P(\Psi_*^0 = T_{*c}^0 \mathbf{1}^c|\theta)) \cap \\ & \mathcal{P}(P(\omega(T) = v)) \cap \\ & (\mathcal{P}(\pi_\theta(\psi, T_{*b}^t \mathbf{1}^b, a)) \cup \{T : T_{\psi a}^t = 0\}).\end{aligned} \tag{14}$$

The intersection of two mixed-integer polyhedra is easy to express as another mixed integer polyhedron by just concatenating the constraints

$$\begin{aligned}\{X \in \mathbb{Z}^N \times \mathbb{R}^M : L \leq CX \leq U\} \cap \{X \in \mathbb{Z}^N \times \mathbb{R}^M : L' \leq C'X \leq U'\} & \\ = \left\{X \in \mathbb{Z}^N \times \mathbb{R}^M : \begin{pmatrix} L \\ L' \end{pmatrix} \leq \begin{pmatrix} C \\ C' \end{pmatrix} X \leq \begin{pmatrix} U \\ U' \end{pmatrix}\right\} &\end{aligned} \tag{15}$$

so the only difficulty in calculating $\mathcal{P}(P(X|\Omega))$ is the union in the final term of equation (14). We transform this into an intersection by introducing an auxiliary variable z and using the following identity: if the hyper-rectangle $0 \leq X \leq H$ bounds $L \leq CX \leq U$ for some vector $H \in \mathbb{Z}^N \times \mathbb{R}^M$, then for any given i

$$\begin{aligned}\{X \in \mathbb{Z}^N \times \mathbb{R}^M : L \leq CX \leq U\} \cup \{X : X^i = 0, \mathbf{0} \leq X \leq H\} = \{X \in \mathbb{Z}^N \times \mathbb{R}^M, z \in \{0,1\} : & \\ CX + (\overline{B} - U)z \leq \overline{B}, & \\ \underline{B} \leq CX + (\underline{B} - L)z, & \\ 0 \leq H^i z - X^i, & \\ z - X^i \leq 0\} &\end{aligned} \tag{16}$$

where $\overline{B}$ are upper bounds on the values of CX in the hyper-rectangle, defined as

$$\overline{B} = \frac{(C + |C|)H}{2}$$

where $|C|$ here denotes an element-wise absolute value of a matrix and B are lower bounds on the values of CX defined as

$$\underline{B} = \frac{(C - |C|)H}{2}.$$

To see why this identity holds, note first that the constraints make z into an indicator variable that is 0 if $X^i = 0$ or 1 otherwise. When $z = 1$ the first constraint on the right hand side of (16) is equal to $CX \leq U$ and the second is equal to $L \leq CX$ so their intersection is $L \leq CX \leq U$ as required, whereas when $z = 0$ the constraints are equal to $\underline{B} \leq CX \leq \overline{B}$. But $\underline{B}$ and $\overline{B}$ are lower and upper bounds on the values of CX in the hyper-rectangle so this is satisfied for all values $0 \leq X \leq H$ as required.

Two things are worth noting here. Firstly if we make the Fermionic assumption then $z = X^i = T^t_{\psi a}$ so the auxiliary indicator variable becomes unnecessary. Secondly, we must bound $\mathcal{P}(\pi_\theta(\psi, T^t_{*b}\mathbf{1}^b, a))$ by a hyper-rectangle with its lower corner at the origin. In practice, this is not a problem as we can just mak a change of variable, imposing upper and lower bounds on each variable such that the probability of being outside those bounds is negligible, then shifting if the lower bound isn't on zero.

So, the support of the posterior can be approximated by a mixed-integer polyhedron, $\mathcal{P}(P(X|\Omega))$, by using equations (14), (15) and (16).

As a simple illustration, consider a two-timestep trajectory of the cat and mouse model described in section 2.1.1. 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 polyhedron, $\mathcal{P}(P(X|\Omega))$, that bounds the support of the posterior.

Working through (14) term by term, the $\mathcal{T}^2_{42}$ term constrains to non-negative, continuous trajectories as defined in equation (9), which is already in linear form. The second term is the support of the prior over the parameters which we ignore here since there are no parameters. The next term is the support of the start state, since we're flipping a coin for each agent state at $t = 0$, each state occupation must be at most 1, which can be expressed as

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

for each ψ. The fourth term is the support of the observation. Since we observe a cat in the left grid-square at time $t = 1$ we 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, so we constrain against these

$$\begin{aligned}
&\text{supp}\,(\pi(2, T^t_{*a}1^a, 0)) = \{T : -T^t_{00} - T^t_{01} \leq -1\} \\
&\text{supp}\,(\pi(3, T^t_{*a}1^a, 0)) = \{T : -T^t_{10} - T^t_{11} \leq -1\} \\
&\text{supp}\,(\pi(2, T^t_{*a}1^a, 1)) = \{T : T^t_{00} + T^t_{01} \leq 0\} \\
&\text{supp}\,(\pi(3, T^t_{*a}1^a, 1)) = \{T : T^t_{10} + T^t_{11} \leq 0\}
\end{aligned} \tag{17}$$

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 (16) to take the union of each constraint in (17) with $\{T : T^t_{\psi a} = 0, \mathbf{0} \leq T \leq \mathbf{1}\}$ 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 $t = 1$.

Taken together, these constraints define a polyhedron that bounds the set of (Fermionic) trajectories for the cat and mouse ABM with the given start state distribution and observation.

2.3 Sampling from the posterior

Having shown how to calculate $\mathcal{P}(P(X|\Omega))$, we now show how to use this to sample from $P(X|\Omega)$.

Various methods of sampling from a mixed-integer polyhedron exist in the literature. Baumert *et al.* (2009) presents a very general algorithm for sampling from any weighted subset of integer points within a bounding hyper-rectangle. The algorithm relies on sampling points on a random walk. However, in our case the number of valid trajectories could be a vanishingly small proportion of the number of points in the smallest bounding hyper-rectangle, so a random walk would be unlikely to come across any valid trajectories. Universal hashing (Meel *et al.*, 2016) provides a promising way of sampling uniformly from the integer points in a convex polyhedron, but we have found that this technique doesn't scale to the number of dimensions needed for our application.

The Metropolis-Hastings algorithm is another well known and widely used sampling algorithm and this is the approach we take here. To do this we need to define

- a set of Markov states, $\mathcal{M}$, along with a mapping, $E : \mathcal{M} \to \mathbb{Z}^N \times \mathbb{R}^M$, which maps Markov states to sample values, X,
- a stochastic proposal function $f : \mathcal{M} \to \mathcal{M}$ from which we can generate transitions between Markov states,
- a probability distribution $P : \mathcal{M} \to \mathbb{R}$ which gives the probability of each Markov state (this need not be normalised as the Metropolis-Hastings algorithm only ever needs probability ratios).

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 a non-zero probability of being proposed and accepted.
- Given a current Markov state, there should be a computationally efficient procedure to generate a proposal and calculate its acceptance probability.

2.3.1 The set of Markov states. Given the polyhedron $\mathcal{P}(P(X|\Omega))$, 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) so that

$$\mathcal{P}(P(X|\Omega)) = \{X \in \mathbb{Z}^N \times \mathbb{R}^M : (L \leq CX \leq U) \cap (DX = E)\}. \tag{18}$$

Our aim will be to use the equality constraints to reduce the dimension of the problem and so improve the efficiency of the algorithm. Note that D has at least $(N - 1)S$ rows since the continuity constraints (8) are equality constraints.

Suppose there are N_e equality constraints and we partition the elements of X into 'basic' and 'non-basic' elements, X_B and X_N respectively, so that there are exactly N_e basic elements.

If we let Q be a permutation matrix that separates the basic from non-basic elements so that

$$QX = \begin{pmatrix} X_B \\ X_N \end{pmatrix}$$

then we can write

$$DX = DQ^{-1}\begin{pmatrix} X_B \\ X_N \end{pmatrix} = (B\,|\,N)\begin{pmatrix} X_B \\ X_N \end{pmatrix} = E$$

so

$$BX_B + NX_N = E. \tag{19}$$

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

$$X_B = B^{-1}(E - NX_N) \tag{20}$$

so letting

$$V = Q^{-1}\begin{pmatrix} B^{-1}E \\ 0 \end{pmatrix}, M = Q^{-1}\begin{pmatrix} -B^{-1}N \\ 1 \end{pmatrix}$$

gives

$$X = V + MX_N. \tag{21}$$

In Section 2.3.5 we'll show how to choose X_B to ensure that B has an inverse and the integer elements of X_B remain integer as long as the integer elements of X_N are integer. Given this, then the original polyhedron is equivalent to the reduced dimension polyhedron

$$\mathcal{P}' = \left\{X_N \in \mathbb{Z}^{N'} \times \mathbb{R}^{M'} : L - CV \le CMX_N \le U - CV\right\} \tag{22}$$

where N' and M' are the number of integer and real-valued elements remaining in X_N.

Since Q is just a permutation matrix, X_N and X_B can be bound within hyper-rectangles by transforming the $\mathcal{P}$-bounding hyper-rectangle, H, from Section 2.2.1

$$QH = \begin{pmatrix} H_B \\ H_N \end{pmatrix}.$$

So, X_N is bound by $0 \le X_N \le H_N$. Given this, we define the set of Markov states to be

$$\mathcal{M} = \{X_N : 0 \le X_N \le H_N,\}.$$

and equation (21) maps each Markov state, $X_N \in \mathcal{M}$, to a unique sample, X. Although this set includes values outside of the polyhedron (22), we'll show a way to deal with that later.

2.3.2 The proposal function. In order to propose a new Markov state given a current state, X_N, first choose an element of X_N and a direction in which to perturb that element. This defines a rounded perturbed state, X'_N, which is equal to X_N but with one of its elements rounded to the nearest integer and perturbed by ±1. Only rounded perturbed states that remain inside the hyper-rectangle H_N are considered. Real valued variables should be scaled so that a perturbation of ±1 usually changes $P(X|\Omega)$ by a small amount. The boundaries, H_N, should be chosen to lie on an integer value plus 0.5 for real-valued variables. So, we can define the set of all valid perturbations

$$\mathcal{D}(X_N) = \{\langle i, \delta\rangle : X'_N = \lfloor X_N \rceil_i + \delta e_i, 0 \le X'_N \le H_N, \delta \in \{+1, -1\}, i \in \mathcal{V}\}$$

where e_i is the unit vector with a 1 in the i^{th} element, $\mathcal{V}$ is the set of indices of X_N and we use the notation $\lfloor . \rceil_i$ to denote the rounding of the i^{th} element of a vector to the nearest integer.

Assume for now that we have an approximation of the target distribution $\tilde{P}(X_N) \approx P(X_N|\Omega)$. Given a current state, X_N, the probability of choosing perturbation $\langle i, \delta \rangle \in \mathcal{D}(X_N)$ is given by

$$P(X_N, i, \delta) = \frac{\min\left(1, \frac{\tilde{P}(\lfloor X_N \rceil_i + \delta e_i)}{\tilde{P}(\lfloor X_N \rceil_i)}\right)}{S(X_N)} \tag{23}$$

where

$$S(X_N) = \sum_{\langle i,\delta \rangle \in \mathcal{D}(X_N)} \min\left(1, \frac{\tilde{P}(\lfloor X_N \rceil_i + \delta e_i)}{\tilde{P}(\lfloor X_N \rceil_i)}\right) \tag{24}$$

Given a perturbation $\langle i, \delta \rangle$, the final proposed state is given by

$$X'_N = \lfloor X_N \rceil_i + (\delta + \epsilon) e_i$$

where ϵ is a fractional perturbation drawn with uniform probability from $[-0.5 : 0.5]$ if the i^{th} element is real-valued or $\epsilon = 0$ if it is integer-valued.

2.3.3 Approximating the posterior. The proposal function requires an approximation of the target distribution, $P(X|\Omega)$. For this, we use a function of the form

$$\tilde{P}(X) = \prod_i \tilde{P}_i(Z_i X) \tag{25}$$

where Z_i are row vectors and $\tilde{P}_i$ are univariate factors of any form. We call this a "linearly factorized distribution", where the linearity lies in the $Z_i X$ terms while the $\tilde{P}_i$ can be non-linear.

The first thing to note is that we can express $\mathcal{P}(P(X|\Omega))$ as a linearly factorized distribution in the following way: first define the infeasibility function $\iota(l, x, u)$ to be equal to 0 if $l < x < u$ or equal to the distance to the nearest bound otherwise

$$\iota(l, x, u) = \begin{cases} x - u & \text{if } x > u \\ l - x & \text{if } x < l \\ 0 & \text{otherwise.} \end{cases}$$

Now, given a set of constraints $L \leq CX \leq U$, if we define a factor for each constraint

$$\tilde{P}_i(C_i X) = e^{\frac{-\iota(L_i, C_i X, U_i)}{\tau}} \tag{26}$$

then

$$\prod_i \tilde{P}_i(C_i X) = e^{\frac{-\sum_i \iota(L_i, C_i X, U_i)}{\tau}} \tag{27}$$

is a linearly factorized distribution that is 1 inside the polyhedron and that decays exponentially outside at a rate set by τ. So as $\tau \to 0$ equation (27) tends to an indicator function for the set of points in $\mathcal{P}(P(X|\Omega))$.

A linearly factorized distribution in X can easily be transformed into a linearly factorized distribution in X_N by using the linear transform in equation (21). Transformation to X_N ensures that the resulting distribution satisfies the equality constraints $D(V + MX_N) = E$, while the factors in equation (27) can be used to ensure that $0 \leq X_B \leq H_B$ and that the constraints in (22) are satisfied.

Turning our attention now to the distribution inside the polyhedron, if we look back at equation (12) we can see that the posterior of an ABM is already highly factorized and many of the factors are already linearly factorised.

However, when considering the efficiency of the final algorithm, there is a trade off between the closeness of $\tilde{P}(X_N)$ to $P(X_N|\Omega)$ and the number of factors involved in its calculation. In our experiments, we have found we can attain very good acceptance probabilities with relatively simple renderings of $\tilde{P}(X_N)$. The start state, likelihood

function and prior over θ are often already uncorrelated functions of the elements of θ and the state occupation numbers, so are already in the correct form. This leaves only the multinomial term in equation (12) which we approximate with the linearly factorized

$$\prod_{t,\psi}(T^t_{\psi b}\mathbf{1}^b)!\prod_a \frac{\tilde{\pi}^t_{\psi a}{}^{T^t_{\psi a}}}{T^t_{\psi a}!} \tag{28}$$

where

$$\log(\tilde{\pi}^t_{\psi a}) = \frac{\int T^t_{\psi a}\log(\pi_\theta(\psi,\Psi^t,a))\,P(T,\theta)\,dT\,d\theta}{\int T^t_{\psi a}P(T,\theta)\,dT\,d\theta}.$$

This integral can be calculated by drawing samples from the prior $P(T, \theta)$ and using Monte Carlo integration. This approximation marginalises over θ, but if π_θ has a particularly strong dependence on θ then it may be worth capturing some of that dependence by, for example, replacing the $T^t_{\psi a}$ term in (28) with $T^t_{\psi a} + R\theta$ for some row vector R. The $\tilde{\pi}^t_{\psi a}$ can then be calculated by minimising the expectation of the difference between the log probabilities of the normalised approximation and the true multinomial.

So, the final approximation, $\tilde{P}(X_N)$, is given by the product of the factors of the bounding convex polyhedron (27), the multinomial approximation (28) and the approximations of the observations, the start state (boundary conditions) and $P(\theta)$.

Since some of the Markov states are infeasible, calculating $\tilde{P}$ for these states may require the evaluation of a factor at a value that is outside its domain (for example, equation (28) is undefined if $T^t_{\psi a}$ is negative). In this case we choose the value at the nearest point that is within the domain.

2.3.4 The probability of a Markov state. If a Markov state satisfies all the constraints in (22) then its probability is defined to be equal to the true posterior given by (12)

$$P(X_N) = P(X\,|\,\Omega) = P(T,\theta\,|\,\Omega)$$

where

$$X = V + MX_N = W^{\psi a i}_t T^t_{\psi a} + G^i_j\theta^j.$$

If the constraints are not all satisfied, then the probability is defined to be the product of the factors in (12) and the infeasibility factors in (27). If any factor in (12) becomes zero as a consequence of the infeasibility, then it is replaced by its linearly factorized approximation (which must be non-zero otherwise the state wouldn't have been proposed).

This means that, as long as τ is finite, 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 true posterior. 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. When used in the context of a particle filter, it also solves the problem of unexpected observations that can't be generated by any particle, as we can allow the particles to become temporarily infeasible. The price we pay for this is the computational cost of moving through infeasible states that don't generate useful samples.

It's worth noting that equation (27) is a Boltzmann distribution which describes a thermodynamic system whose "energy" is equal to the infeasibility of X that is 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 sets of linear integer inequalities like the one we're dealing with here (Kirkpatrick *et al.*, 1983). Here, we won't be changing 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 and low enough to ensure that a reasonable proportion of samples are feasible. We have found that these two competing effects roughly cancel each other out when measuring the overall efficiency of the algorithm in effective samples per second, so the value of τ is not critical as long as it's within a certain range. We have found that a good rule of thumb is to set the temperature so that 50–80% of the samples in a chain are infeasible.

From the Metropolis-Hastings algorithm, the probability of accepting a proposal $X'_N = \lfloor X_N \rceil_i + (\delta + \epsilon)e_i$ is given by

$$\alpha = \min\left(1, \frac{P(X'_N)P(X'_N, i, -\delta)}{P(X_N)P(X_N, i, \delta)}\right) = \min\left(1, \frac{P(X'_N)}{P(X_N)} \frac{\tilde{P}(\lfloor X_N \rceil_i)\, S(X_N)}{\tilde{P}(\lfloor X'_N \rceil_i)\, S(X'_N)}\right).$$

Since $\tilde{P}(X_N)$ approximates $P(X_N)$, $\frac{P(X'_N)}{P(X_N)} \frac{\tilde{P}(\lfloor X_N \rceil_i)}{\tilde{P}(\lfloor X'_N \rceil_i)}$ should be close to 1. We'll see in the next section that if we choose the basis carefully then $S(X_N)$ will not change very much between transitions so the ratio $\frac{S(X_N)}{S(X'_N)}$ should also be close to 1 meaning that a good proportion of proposals should be accepted. In our experiments we achieved acceptance rates of around 85%.

2.3.5 Choosing a basis. Our definition of a Markov state depends on a partition of X into basic and non-basic variables such that B in equation (20) has an inverse, B^{-1}, and if $X_N \in \mathbb{Z}^{N'} \times \mathbb{R}^{M'}$ then $X \in \mathbb{Z}^N \times \mathbb{R}^M$ (*i.e.* if X_N has all integers in the correct elements then so does X). In general there exist many partitions that satisfy these requirements so it remains to define a method of choosing one. The choice of basis affects the efficiency of the Markov chain via:

- the computational cost of updating the proposal probabilities $P(X_N, i, \delta)$ and the sum $S(X_N)$ after each transition.
- the effect of the ratio of sums $S(X_N)/S(X'_N)$ on the acceptance probability
- the mixing rate of the Markov chain, since the choice of basis affects the available transitions in X space.

From (23) and (24) we can see that in order to calculate $P(X_N, i, \delta)$ and $S(X_N)$ we need to know $\frac{\tilde{P}(\lfloor X_N \rceil_i + \delta e_i)}{\tilde{P}(\lfloor X_N \rceil_i)}$ all $\langle i, \delta \rangle \in \mathcal{D}(X_N)$. Since $\tilde{P}$ is linearly factorised

$$\frac{\tilde{P}(\lfloor X_N \rceil_i + \delta e_i)}{\tilde{P}(\lfloor X_N \rceil_i)} = \frac{\prod_j \tilde{P}_j(Z_j(\lfloor X_N \rceil_i + \delta e_i))}{\prod_j \tilde{P}_j(Z_j(\lfloor X_N \rceil_i))} = \prod_{j: Z_j e_i \neq 0} \frac{\tilde{P}_j(Z_j \lfloor X_N \rceil_i + \delta Z_j e_i)}{\tilde{P}_j(Z_j \lfloor X_N \rceil_i)}.$$

Let

$$\tilde{P}(X_N, i, \delta) = \min\left(1, \prod_{j: Z_j e_i \neq 0} \frac{\tilde{P}_j(Z_j \lfloor X_N \rceil_i + \delta Z_j e_i))}{\tilde{P}_j(Z_j \lfloor X_N \rceil_i)}\right)$$

be the un-normalised proposal probability. If we accept a state transition $\langle k, \delta' \rangle$, $P'(X_N, i, \delta)$ needs to be updated to $P'(\lfloor X_N \rceil_k + (\delta' + \epsilon)_{e_k}, i, \delta)$. However, this update only affects factors $\tilde{P}_j$ such that $\{j : Z_j e_k \neq 0\}$, so a transition $\langle k, \delta' \rangle$ only changes $P'(X_N, i, \delta)$ if $\{j : Z_j e_i \neq 0\}$ intersects with $\{j : Z_j e_k \neq 0\}$. So, if we let Z be the matrix formed from the row vectors Z_i and choose a basis such that Z is sparse, $P'(X_N, i, \delta)$ can be stored for each value of $\langle i, \delta \rangle \in \mathcal{D}(X_N)$, and only values for a few values of i will need to be recalculated on transition. Tang (2022) shows that a binary sum-tree data structure can be used to efficiently update and draw from the un-normalised probability distribution defined by $P'(X_N, i, \delta)$ while also updating $S(X_N)$.

The effect of $\frac{S(X_N)}{S(X'_N)}$ on the acceptance probability is also improved if Z is sparse because only a small proportion of the terms in the sum change between $S(X_N)$ and $S(X'_N)$ so we should expect their ratio to be close to 1.

The Markov chain will also mix better if the columns of Z are sparse. Mihelich *et al.* (2018) shows that the Kolmogorov-Sinai entropy of the Markov process is a good proxy for its mixing speed. In the context of our algorithm the entropy is higher if the neighbours of a Markov state have similar probability. If Z is sparse then adjacent Markov states are more likely to have similar probabilities, for a fixed mean element magnitude.

So, our aim will be to find a basis that keeps Z sparse, while maintaining the correct integer signature of X. To do this we use the following algorithm: Start with a linearly factorized distribution $\tilde{P}$ on X (rather than X_N)

$$\tilde{P}(X) = \prod_i \tilde{P}_i(C_i X).$$

and a set of equality constraints that need to be satisfied for all Markov states X_N

$$DX = E.$$

From these, form a mixed-integer polyhedron

$$\mathcal{P} = \{X \in \mathbb{Z}^N \times \mathbb{R}^M : L \leq CX \leq U \bigcap DX = E\}.$$

We assume, without loss of generality, that all constraints involving only integer elements of X have integer coefficients. C is usually sparse to start with, so our aim is to maintain that sparsity while using the equality constraints to reduce X to X_N. If we let

$$W = \begin{pmatrix} D \\ C \end{pmatrix}$$

and

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

then

$$\mathcal{P} = \left\{ X \in \mathbb{Z}^N \times \mathbb{R}^M : \begin{pmatrix} \mathbf{0} \\ L \end{pmatrix} \leq WX - F \leq \begin{pmatrix} \mathbf{0} \\ U \end{pmatrix} \right\} \tag{29}$$

Our aim is to mark variables from X as basic one at a time. Once we're done, the remaining variables will make up the non-basic variables, X_N.

Given a set of constraints in the form (29), we mark the j^{th} element of X as basic and the i^{th} equality constraint as "reduced" by choosing an element W_j^i to be a pivot point and performing Gaussian elimination on the j^{th} column of W so that W_j^i is the only non-zero element in the column W_j. This can be done by pre-multiplying by G

$$\begin{pmatrix} \mathbf{0} \\ L \end{pmatrix} \leq GWX - GF \leq \begin{pmatrix} \mathbf{0} \\ U \end{pmatrix}$$

where G is the matrix

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

Letting $\hat{W} = GW$ and $\hat{F} = GF$ recovers the canonical form

$$\begin{pmatrix} \mathbf{0} \\ L \end{pmatrix} \leq \hat{W}X - \hat{F} \leq \begin{pmatrix} \mathbf{0} \\ U \end{pmatrix} \tag{30}$$

but now the j^{th} element of X is marked as basic and the i^{th} row of $\hat{W}$ is marked as "reduced". We now repeat the process, choosing only pivot points that are non-basic and unreduced, until no more equality constraints can be reduced.

The use of Gaussian elimination here ensures that B^{-1} exists. In order to ensure that X_B has the correct integer structure, given that X_N does, we add the requirement that we can only mark integer elements of X as basic on a pivot point W_j^i such that the row vector W^i is zero on all non-integer columns and W_j^i divides W_k^i for all k (so for rows that have a non-zero element on a real-valued column, we always eliminate on a real-valued column). This requirement ensures that for integer elements, X_B^i, after reduction $X_B^i = F^i - W^i X_N$. But if X_N has the correct integer structure then $W^i X_N$ is a sum of products of integers, which is an integer. So, if X_B^i has any integer solution then F^i must also be an integer. So, if X_N has the correct integer structure, then X_B^i is also an integer.

This defines the set of valid pivot points at each elimination step. In order to decide which point to choose, we note that when we pivot on W_j^i, GW differs from W in at most

$$\mu = (|W_j|_0 - 1)(|W^i|_0 - 1)$$

elements, where $|\cdot|_0$ is the L0-norm (*i.e.* the number of non-zero elements in a vector or matrix). So, this gives an upper bound on $|WG|_0 - |G|_0$, the change in L0-norm (or reduction in sparsity) of W when we pivot on W_j^i. It has been found that choosing W_j^i to be the element that minimises μ (known as the Markowitz criterion) is a computationally efficient way of maintaining the sparsity of a matrix while performing Gaussian elimination (Markowitz, 1957; Maros, 2002; Suhl & Suhl, 1990). So, in order to find a sparse basis, we calculate the set of valid pivot points, choose the point that minimises μ, perform the pivot and repeat until no more valid pivot points remain. Suhl & Suhl (1990) presents a computationally efficient algorithm for doing this.

3 Spatial predator-prey demonstration

We demonstrate the above techniques on a 32x32 grid of "predator" and "prey" agents. The agent acts consist of moving to an adjacent grid-square (*i.e.* up, down, left or right), giving birth to another agent on the an adjacent grid-square or dying. If any agent moves off the edge of the grid, it reappears at the opposite edge (*i.e.* the grid is topologically a torus). Predators only give birth when there is at least one prey on an adjacent grid-square (*i.e.* if there is a source of food close by). Prey may be eaten by predators so their probability of dying is much greater when predators are present on adjacent grid-squares. The probability of each agent's behaviour is shown in Table 1. These values were chosen to minimise the probability that either species becomes extinct.

In order to generate observations to assimilate, we used the following procedure:

1. Generate a start state by drawing the number of predators and prey in each gridsquare at $t = 0$ from a Bernoulli distribution with probability 0.05. *i.e.* for $\Psi_*^0 \in \{0, 1\}^S$

$$P(\Psi_*^0) = \prod_\psi 0.05^{\Psi_\psi^0} 0.95^{1-\Psi_\psi^0} \tag{31}$$

2. Simulate the ABM for 16 timesteps from the start state. If the resulting trajectory isn't Fermionic, repeat from step 1. The resulting trajectory, T_{real}, is considered to be the "real-world" trajectory from which we take observations.

3. For each timestep and gridsquare of T_{real}, take two draws from a Bernoulli distribution with probability 0.05. If the first draw is 1, then observe the number of predators in that gridsquare at that time. If the second draw is 1 then observe the number of prey. The observations were noiseless so the observed count was the true count.

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

Aget type	Adjacent agents	Behaviour	Probability
Prey	No predators	die	0.100
		give birth	0.156
		move	0.744
Prey	Predators > 0	die	0.400
		give birth	0.156
		move	0.444
Predator	No prey	die	0.100
		give birth	0.000
		move	0.900
Predator	Prey > 0	die	0.100
		give birth	0.300
		move	0.600

Given the (Fermionic) model, the start state $P(\Psi_*^0)$ and the observations, an approximation of the posterior was generated as described in section 2.3.3 and used to generate a sparse basis as described in section 2.3.5. The resulting basis had 22.4 non-zero elements per million.

Four initial feasible solutions were generated in the following way:

1. Draw a start state from $P(\Psi_*^0)$.
2. Simulate for 16 timesteps to get a (non-Fermionic) prior trajectory.
3. Starting with the prior trajectory, generate proposals as described in section 2.3.2 and accept with probability 1.0 until a feasible state is reached.

Starting with these four initial solutions, four separate Markov chains were generated using the Matropolis-Hastings algorithm as described in section 2.3. 1,000,000 samples were then taken from each chain and discarded in order to burn them in. Finally, 10,000,000 samples were taken from each chain and split into first and last halves to give eight sample sequences of 5,000,000 samples each. For all samples, the temperature, τ, was set to 0.1.

Feasible samples were generated at a rate of one every 42μs on a single core of a 1.6GHz Intel i5-8250U. 85% of the proposals were accepted and 70% of samples were infeasible.

In order to assess the convergence of the chains, a set of summary statistics were calculated for each sample of each sequence. The summary statistics consisted of the number of agents within each shaded square shown in Figure 2 measured at the end of the last timestep. We'll refer to these as statistic 1 to 4 going from largest to smallest area. This arrangement of regions was chosen in order to capture the convergence at different spatial scales.

For each statistic, let x_{ij} to refer to the value of the statistic for the i^{th} sample of the j^{th} sequence and let

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

be the mean of the j^{th} sequence and

$$\bar{x} = \frac{1}{m}\sum_{j=1}^{m} \bar{x}_j$$

be the mean over all sequences.

Let W be the within-sequence variance

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

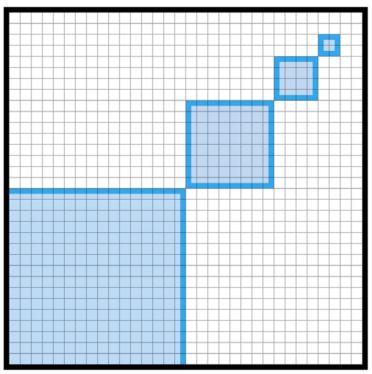


Figure 2. Summary statistics consist of the total number of agents in each of the four shaded regions.

and B be the between-sequence variance

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

Following Gelman *et al.* (2013), we calculated an over-approximation of the true variance of the statistic as

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

and the Gelman-Rubin diagnostic (Gelman & Rubin, 1992) was calculated using

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

This gives a measure of the uncertainty in the standard deviation of the statistic due to the finite length of the sample sequences. If this number is close to 1, then we have some justification in believing that we have taken enough samples. A value of less than 1.1 is often used as a criterion for convergence.

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

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

where

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

at various time lags for each sample sequence. The autocorrelation of a convergent chain should decline to zero long before the lag reaches the total number of samples.

$\hat{\rho}$ can also be used to calculate the effective number of samples (*i.e.* the number of independent and identically distributed draws that would give the same uncertainty in the mean of the statistic) using

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

where the sum runs from $t = 0$ until the first t such that $\hat{\rho}_t \leq 0$.

4 Results

The algorithm described here was validated against two timesteps of the cat-and-mouse model as described in the running example in section 2. This model is small enough to allow the number of samples of each trajectory to be compared to the exact posterior probability. It was found that the proportion of samples of each trajectory did converge towards the exact probability as the number of samples increased, up to a maximum of 10,000,000 samples. In a similar way, we validated the algorithm against 2 timesteps of a 3 × 3 predator-prey model with just one observation of a single prey on the middle gridsquare at time t = 1. This particular case could be validated against a rejection sampler. It was found that the model state at t = 2 converged to the same distribution for both the MCMC and rejection samplers.

The convergence diagnostics for the 32 × 32 predator-prey demonstration are shown in Figure 3 and Table 2. These show that the samples passed all convergence tests. For the purposes of this demonstration, we take the unobserved values, Λ, to be the expected number of agents in each state at the end of the simulation. Figure 4 shows this averaged over all samples, along with the end state of the "real" trajectory T_{real}. This gives a visual demonstration that the observations have given rise to a posterior that provides information about the true positions of the agents.

The code used to generate these results can be found at https://github.com/danftang/AgentBasedMCMC.

5 Discussion

5.1 Sequential MCMC for ABM

The predator-prey example demonstrated assimilation over a relatively short time period. In practice it's likely that assimilation would be required over a longer period, or perhaps over a continuous stream of observations.

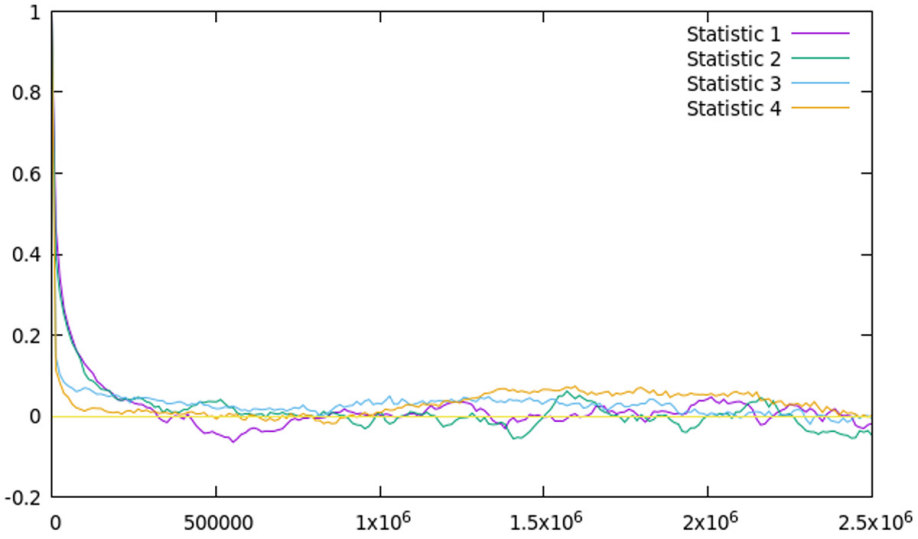


Figure 3. Mean autocorrelation for each of the summary statistics, averaged over all chains, as a function of the lag in number of samples.

Table 2. The Gelman Rubin diagnostic and number of effective samples per sample-sequence for each statistic.

Measure	Statistic 1	Statistic 2	Statistic 3	Statistic 4
Gelman-Rubin	1.00727	1.01167	1.00451	1.00153
Effective samples	420	417	489	1019

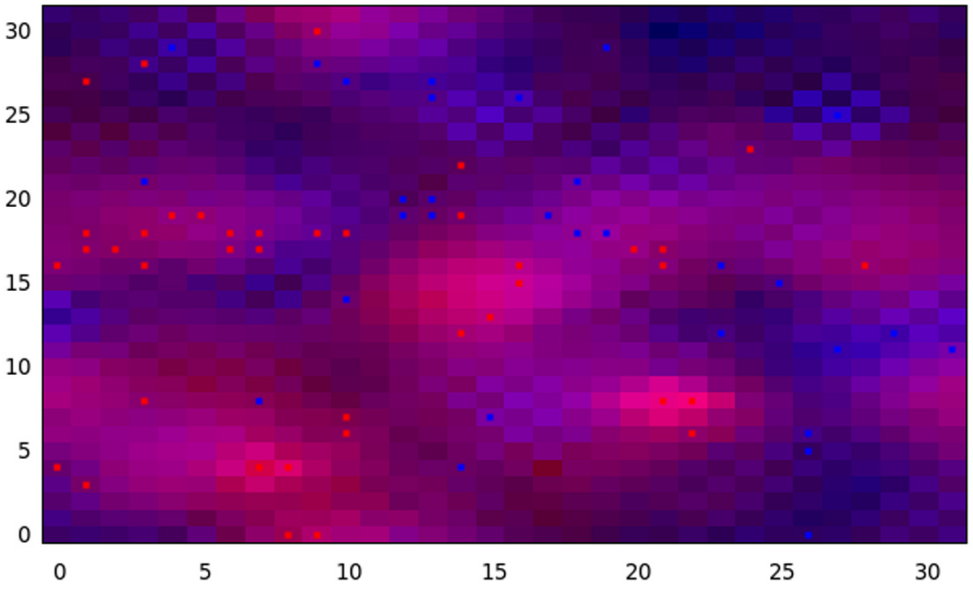


Figure 4. A representation of the positions of the agents at time *t* = 16. The dots represent the positions of agents in the "real" trajectory, T_{real}, from which the observations were generated. Blue dots represent predators, red dots represent prey. The background colour of each square has blue/red intensity proportional to the log of the mean number of predators/prey in that square averaged over all samples.

As discussed earlier, the way to deal with this is to split the longer time period into windows as in equation (5) and treat each window separately. We can make use of the MCMC sampler to implement this broad idea in many different ways.

One way would be the following: given N samples from the posterior of a window $X_t^{1:N} \sim P(X_t|\Omega_{1:t})$, approximate the posterior from some family of simpler, analytical distributions, $P_\Theta(X_t)$, by finding the Θ that minimises the KL-divergence from $P_\Theta(X_t)$ to $P(X_t|\Omega_{1:t})$. The KL-divergence can itself be approximated from the samples $X_t^{1:N}$

with

$$D_{KL}(P(X_t\,|\Omega_{1:t})\,|\,P_\Theta(X_t)) \approx \sum_{s=1}^{N} \frac{1}{N} \log\left(\frac{1}{NP_\Theta(X_t^s)}\right)$$

which is minimised when

$$\Theta \approx \min_\Theta \left(-\sum_{s=1}^{N} \log\left(P_\Theta(X_t^s)\right)\right).$$

The approximation, $P_\Theta(X_t)$, can then be inserted into the recursion equation (5) to generate samples from the next window $X_{t+1}^{1:N} \sim P(X_{t+1}|\Omega_{1:t+1})$, from where we can repeat the cycle indefinitely. The exact form of P_Θ is model dependent. The simplest form would be a product of univariate Poisson distributions, one for each of the integer variables, and Gaussians for the real-valued variables. If we wish to capture correlations between variables, a multivariate Poisson distribution could be used (see for example Muñoz-Pichardo *et al.* (2021), where P_Θ is a product of univariate Poisson distributions whose rate parameters are given by $\Lambda = BX_R + AX_I$, where X_R are the real-valued variables and X_I are the integer valued variables, B and A are matrices, and A is triangular).

An alternative approach is to use the Resample-Move algorithm presented in Gilks & Berzuini (2001). This is a particle filter based on the sequential importance resampling algorithm described earlier. The innovation comes in the form of an additional MCMC *move* step after each resample step. At each move step each particle, $X_t^i = \langle \tau_{1:t}^i, \theta^i \rangle$, is "moved" by initialising the Markov chain with X_t^i and drawing a single new sample $X_t'^i \sim P(X_t|\Omega_{1:t})$ by taking one or more Markov transitions. This solves the problem of particle impoverishment because even if multiple samples collapse to the same state during the resample step, the subsequent move step is likely to move them apart again. The algorithm remains exact because the samples coming out of the m^{th} resample step are already distributed according to the target distribution, $P(X_t|\Omega_{1:t})$, but this is also the stationary distribution of the Markov transition so the moved samples are also distributed according to the target distribution.

The Resample-Move algorithm still uses an importance sampling step with a draw from the prior which, as we have seen, is often a problem for ABMs because drawing from the prior is likely to result in a sample with zero weight. If this is the case, we can replace the draw from the prior with an draw from $P(X_{t+1}|\Omega_{1:t+1}, X_t^i)$ using our MCMC algorithm for each particle X_t^i from the previous timestep. The correct weight of the particle is then proportional to $P(\Omega_{t+1}|X_t^i)$ (Doucet *et al.*, 2009). Unfortunately we don't immediately know this value. If we're doing an assimilation step every timestep, or if the ABM is simple enough, it may be possible to calculate this value analytically, however, often this is not possible. Han & Carlin (2001); Newton & Raftery (1994); Štefankovič *et al.* (2009) give a number of ways of approximating these values, at least up to a multiplicative constant (which is all we need). An alternative is to maintain a population of feasible and infeasible particles. At each window, draw X_t from the prior, extract X_N to give a Markov state then weight by the Markov probability as defined in 2.3.4 divided by the prior probability of drawing X_N. The set of feasible particles are then draws from the posterior.

As the number of assimilated windows increases, there may be a lag L where we are happy to assume that the new observations Ω_{t+1} do not change our beliefs about the model state before the end of the $(t - L)^{th}$ window, i.e. $P(\tau_{1:t-L}|\Omega_{1:t}, \theta) \approx P(\tau_{1:t-L}|\Omega_{1:t+1}, \theta)$. In this case, it is sufficient to perturb the trajectories of only a finite number of windows into the past when performing the MCMC moves, so we perturb $\tau_{t-L+1:t+1}$ and θ while holding $\tau_{1:t-L}$ fixed. This means that the dimension of the MCMC problem does not increase as we assimilate more windows. The early parts of the trajectory are then subject to possible particle impoverishment again if we're using an algorithm with a resampling step, but this time it doesn't matter because it is, by assumption, not affecting the distribution of the current state.

5.2 Limitations and further work

This algorithm presented here is intended as the proof-of-concept of a novel approach to MCMC sampling from an ABM rather than a production-ready tool for immediate use by practitioners. At present, the bounding

of the supports of the agent time-step, boundary conditions and observation likelihood functions by a convex polyhedron must either be done by hand or by third party abstract interpretation tools. Code for commonly encountered scenarios is available on the GitHub repository but this process could be automated for user supplied functions by taking as input a computer program and outputting a set of linear constraints for the support. Similarly, the temperature setting, τ, at present needs to be set by trial-and-error, whereas an automated warm-up or adaptive scheme would improve the algorithm's efficiency and usability.

As presented, the algorithm is only applicable if the internal state of an agent can be represented in a few bytes. This is because it depends on explicitly representing the probability of each Markov transition at each iteration. But the number of possible transitions is of the order of the number of non-basic variables, and this expands exponentially with the size of the internal state of an agent. So, if agents have a large amount of internal state it will become impractical to represent this distribution explicitly. This problem can be overcome by considering only a subset of non-basic variables for update at each iteration, in a similar way to partial pricing in the simplex algorithm (Maros, 2002). For example, given the current Markov state, only consider updating the non-basic variables that correspond to alternative actions of some agent in the current trajectory at some timestep. In this situation we also can't explicitly store the coefficients of the linear constraints, C, so in order to calculate the effect of a perturbation on the basic variables we'd calculate this on the fly from equations (7) and (8), the agent timestep function and the observation operators. Finding a good basis of non-basic variables could also be done on the fly for some subset of variables, or alternatively equality constraints could be left as-is as inequalities with equal lower and upper bounds. More work needs to be done exploring the relative efficiency of these options.

6 Conclusions

We have described and demonstrated an algorithm to sample model-trajectories and parameters from the posterior distribution of an ABM given a set of observations. We derived an analytical expression for the posterior and showed how to generate a convex polyhedron that bounds the support of this distribution. We then showed how these can be used to construct an approximation of the posterior in the form of a linearly factorized distribution, and how this can be embedded in a Metropolis-Hastings algorithm to generate samples from the posterior. This allowed us to perform Bayesian inference with a spatial predator/prey model and so demonstrate data assimilation on an ABM which would be impossible with other techniques described in the literature.

The current lack of tools to perform data assimilation with ABMs has restricted their use in data analytics. The development of more powerful techniques for performing Bayesian inference with ABM could transform the usefulness and applicability of these models and the algorithm described here will hopefully be a first step towards this transformation.

Data availability

Underlying data

Zenodo: danftang/AgentBasedMCMC: Code used to generate results in paper, https://doi.org/10.5281/zenodo.6553199 Tang & Malleson (2022)

This project contains the data used to test the algorithm.

Extended data

Analysis code available from: https://github.com/danftang/AgentBasedMCMC

Archived analysis code as at time of publication:https://doi.org/10.5281/zenodo.6553199

License: MIT

References

Andrieu C, Doucet A, Singh SS, *et al.*: **Particle methods for change detection, system identification, and control.** *Proceedings of the IEEE.* 2004; **92**(3): 423–438.
Publisher Full Text

Bagnara R, Hill PM, Zaffanella E: **The parma polyhedra library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems.** *Sci Comput Program.* 2008; **72**(1–2): 3–21.
Publisher Full Text

Baumert S, Ghate A, Kiatsupaibul S, *et al.*: **Discrete hit-and-run**

for sampling points from arbitrary distributions over subsets of integer hyperrectangles. *Oper Res.* 2009; **57**(3): 727–739.
Publisher Full Text

Becchi A, Zaffanella E: **An efficient abstract domain for not necessarily closed polyhedra.** *International static analysis symposium.* 2018; 146–165.
Publisher Full Text

Carrassi A, Bocquet M, Bertino L, *et al.*: **Data assimilation in the geosciences: An overview of methods, issues, and perspectives.** *Wiley Interdiscip Rev Clim Change.* 2018; **9**(5): e535.
Publisher Full Text

Chatterjee S, Diaconis P: **The sample size required in importance sampling.** *The Annals of Applied Probability.* 2018; **28**(2): 1099–1135.
Publisher Full Text

Clay R, Kieu LM, Ward JA, *et al.*: **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.* 2020; Cham: Springer International Publishing. **12092**: 68–79.
Publisher Full Text

Conforti M, Cornuéjols G, Zambelli G: **Polyhedral approaches to mixed integer linear programming.** In *50 years of integer programming 1958-2008.* 2010; 343–385.
Publisher Full Text

Cousot P, Cousot R: **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.* 1977; 238–252.
Publisher Full Text

Cousot P, Halbwachs N: **Automatic discovery of linear restraints among variables of a program.** In *Proceedings of the 5th acm sigact-sigplan symposium on principles of programming languages.* 1978; 84–96.
Publisher Full Text

Douc R, Cappé O: **Comparison of resampling schemes for particle filtering.** In *Ispa 2005. proceedings of the 4th international symposium on image and signal processing and analysis, 2005.* 2005; 64–69.
Publisher Full Text

Doucet A, Johansen AM, *et al.*: **A tutorial on particle filtering and smoothing: Fifteen years later.** *Handbook of nonlinear filtering.* 2009; **12**(656–704): 3.

Evensen G: **The ensemble kalman filter: Theoretical formulation and practical implementation.** *Ocean Dyn.* 2003; **53**(4): 343–367.
Publisher Full Text

Finke A, Doucet A, Johansen AM: **Limit theorems for sequential mcmc methods.** *Adv Appl Probab.* 2020; **52**(2): 377–403.
Publisher Full Text

Fukuda K: **Polyhedral computation.** 2020.
Publisher Full Text

Gelman A, Carlin JB, Stern HS, *et al.*: **Bayesian data analysis.** (3rd ed.). 2013.
Publisher Full Text

Gelman A, Rubin DB: **Inference from iterative simulation using multiple sequences.** *Statist Sci.* 1992; **7**(4): 457–472.
Publisher Full Text

Gilks WR, Berzuini C: **Following a moving target-monte carlo inference for dynamic bayesian models.** *J R Stat Soc Series B Stat Methodol.* 2001; **63**(1): 127–146.
Publisher Full Text

Gordon NJ, Salmond DJ, Smith AF: **Novel approach to nonlinear/non-gaussian bayesian state estimation**. In *Iee proceedings f-radar and signal processing*. 1993; **140**(2): 107–113.
Publisher Full Text

Gurfinkel A, Navas JA: **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, 2021.
Publisher Full Text

Halbwachs N, Proy YE, Roumanoff P: **Verification of real-time systems using linear relation analysis.** *Form Methods Syst Des.* 1997; **11**(2): 157–185.
Publisher Full Text

Han C, Carlin BP: **Markov chain monte carlo methods for computing bayes factors: A comparative review.** *J Am Stat Assoc.* 2001; **96**(455): 1122–1132.
Publisher Full Text

Henry J, Monniaux D, Moy M: **Pagai: A path sensitive static analyser.** *Electron Notes Theor Comput Sci.* 2012; **289**: 15–25.
Publisher Full Text

Jeannet B, Miné A: **Apron: A library of numerical abstract domains for static analysis.** In: *International conference on computer aided verification.* 2009; 661–667.
Publisher Full Text

Kalnay E: **Atmospheric Modeling, Data Assimilation and Predictability.** Cambridge University Press, 2003.
Publisher Full Text

Khan Z, Balch T, Dellaert F: **Efficient particle filter-based tracking of multiple interacting targets using an mrf-based motion model.** In: *Proceedings 2003 ieee/rsj international conference on intelligent robots and systems (iros 2003)(cat. no. 03ch37453).* 2003; **1**: 254–259.
Publisher Full Text

Kieu LM, Malleson N, Heppenstall A: **Dealing with uncertainty in agent-based models for short-term predictions.** *R Soc Open Sci.* 2020; **7**(1): 191074.
PubMed Abstract | **Publisher Full Text** | **Free Full Text**

Kirkpatrick S, Gelatt CD, Vecchi MP: **Optimization by simulated annealing.** *Science.* 1983; **220**(4598): 671–680.
PubMed Abstract | **Publisher Full Text**

Lewis JM, Lakshmivarahan S, Dhall S: **Dynamic Data Assimilation: A Least Squares Approach.** Cambridge: Cambridge University Press, 2006.
Publisher Full Text

Li T, Sun S, Sattar TP, *et al.*: **Fight sample degeneracy and impoverishment in particle filters: A review of intelligent approaches.** *Expert Syst Appl.* 2014; **41**(8): 3944–3954.
Publisher Full Text

Liao C, Barooah P: **An integrated approach to occupancy modeling and estimation in commercial buildings.** In *Proceedings of the 2010 american control conference.* 2010; 3130–3135.
Publisher Full Text

Liu J, West M: **Combined parameter and state estimation in simulation-based filtering.** In *Sequential monte carlo methods in practice.* 2001; 197–223.
Publisher Full Text

Lloyd DJB, Santitissadeekorn N, Short MB: **Exploring data assimilation and forecasting issues for an urban crime model.** *Eur J Appl Math.* 2016; **27**: 451–478.
Publisher Full Text

Lueck J, Rife JH, Swarup S, *et al.*: **Who Goes There? Using an Agent-based Simulation for Tracking Population Movement.** In *Winter Simulation Conference.* National Harbor, MD, USA. 2019.
Reference Source

Malleson N, Minors K, Kieu LM, *et al.*: **Simulating crowds in real time with agent-based modelling and a particle filter.** *J Artif Soc Soc Simul.* 2020; **23**(3): 3.
Publisher Full Text

Markowitz HM: **The elimination form of the inverse and its application to linear programming.** *Management Science.* 1957; **3**(3): 255–269.
Reference Source

Maros I: **Computational techniques of the simplex method.** Springer Science & Business Media. 2002; **61**.
Reference Source

Meel KS, Vardi MY, Chakraborty S, *et al.*: **Constrained sampling and counting: Universal hashing meets sat solving.** In *Workshops at the thirtieth aaai conference on artificial intelligence.* 2016.
Reference Source

Mihelich M, Dubrulle B, Paillard D, *et al.*: **Maximum kolmogorov-sinai entropy versus minimum mixing time in markov chains.** *J Stat Phys.* 2018; **170**(1): 62–68.
Publisher Full Text

Motzkin TS, Raiffa H, Thompson GL, *et al.*: **The double description method.** *Contributions to the Theory of Games.* 1953; **2**(28): 51–73.
Reference Source

Muñoz-Pichardo J, Pino-Mejías R, García-Heras J, *et al.*: **A multivariate poisson regression model for count data.** *J Appl Stat.* 2021; **48**(13–15): 2525–2541.
Publisher Full Text

Newton MA, Raftery AE: **Approximate bayesian inference with the weighted likelihood bootstrap.** *J Roy Stat Soc B Met.* 1994; **56**(1): 3–26.
Publisher Full Text

Schelling TC: **Dynamic models of segregation.** *J Math Sociol.* 1971; **1**(2): 143–186.
Publisher Full Text

Septier F, Pang SK, Carmi A, *et al.*: **On mcmc-based particle methods for bayesian filtering: Application to multitarget**

tracking. In *3rd ieee international workshop on computational advances in multi-sensor adaptive processing (camsap).* 2009; 360–363.
Publisher Full Text

Sjödin J, Pop S, Jagasia H, *et al.*: **Design of graphite and the polyhedral compilation package.** In *Gcc developers' summit.* 2009; 113.
Reference Source

Štefankovič D, Vempala S, Vigoda E: **Adaptive simulated annealing: A near-optimal connection between sampling and counting.** *J ACM (JACM).* 2009; **56**(3): 1–36.
Publisher Full Text

Suhl UH, Suhl LM: **Computing sparse lu factorizations for large-scale linear programming bases.** *ORSA Journal on Computing.* 1990; **2**(4): 325–335.
Publisher Full Text

Talagrand O: **Assimilation of Observations, an Introduction.** *J Meteorol Soc Jpn Ser. II.* 1997; **75**(1B): 191–209.
Reference Source

Tang D: **Data assimilation in agent-based models using creation and annihilation operators.** *arXiv preprint arXiv: 1910.09442.* 2019.
Reference Source

Tang D: **Mutable categorical distribution.** GitHub. 2022.
Reference Source

Tang D, Malleson N: **danftang/AgentBasedMCMC: Code used to generate results in paper.** *Zenodo.* 2022.
http://www.doi.org/10.5281/zenodo.6553199

Wan EA, Van Der Merwe R, Haykin S: **The unscented kalman filter.** *Kalman filtering and neural networks.* 2001; **5**(2007); 221–280.
Reference Source

Wang M, Hu X: **Data assimilation in agent based simulation of smart environments using particle filters.** *Simul Model Pract Theory.* 2015; **56**: 36–54.
Publisher Full Text

Ward JA, Evans AJ, Malleson NS: **Dynamic calibration of agent-based models using data assimilation.** *R Soc Open Sci.* 2016; **3**(4): 150703.
PubMed Abstract | **Publisher Full Text** | **Free Full Text**

Open Peer Review

Current Peer Review Status: ? ? ?

Version 1

Reviewer Report 12 October 2023

https://doi.org/10.21956/openreseurope.15985.r34152

© **2023 Monti C.** This is an open access peer review report distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

?

Corrado Monti
CENTAI, Turin, Italy

In this work, authors formalize the problem of data assimilation of agent-based models, and define a solution using MCMC.

ABMs are formalized as defining a distribution over possible trajectories over states, parametrized by some θ, with an associated prior. The problem of data assimilation becomes the problem of estimating the a posteriori distribution of some unobservable quantity, given some observed property of the system.

This paper is undoubtedly of very high quality. The important open problem of data assimilation in ABMs is formalized in a very elegant and compact way, also contributing to the issue of giving a formal definition for agent-based models. The problem at hand is solved in a very principled and rigorous way by taking notions from the field of probabilistic modeling and Bayesian approaches. The exposition of the mathematical details of the algorithm is in general adequately written.

The fundamental issue of this work is that the space devoted to the development of such a sophisticated approach is not matched by an attentive analysis of its results. After reading the paper, the reader does not have a clear, quantitative idea of how much it could trust the result of the estimation in a realistic setting, even in the same case analyzed by authors of a simplified predator-prey model.

1. The most important issue with this work is the lack of a rigorous and formal experimental evaluation of the proposed technique.

   Experimental evaluation is limited to a toy example, diagnostics, and one single instance of estimation of posterior. As a result, the reader is left unaware of how useful is this technique with simple agent-based models. Questions such as "Is Figure 4 a lucky example? What kind of error can I expect?" are left unanswered. I appreciate that the problem of estimating the prior can be solved exactly only for trivial examples, and therefore in meaningful cases it cannot be compared against a ground truth. Anyway, as with any learning or inference task, other approximations can be used. Some general ideas could be

the following. One is repeating the experiment from Figure 4 and showing some statistics about the quality of the output. For instance, one could consider the binary classification task of indicating whether in a given cell there is an agent of type 'predator'; the posterior can be seen as a way to provide a score for this task, which can then be evaluated e.g. with AUC ROC. Another way to evaluate results would be to consider the task to estimate the position of agents; the posterior would allow to give such an estimate, which can then be confronted with the real state of the system e.g. through R2 or RMSE. In general, I believe authors should find a way to systematically assess the quality of their approach.

2. The proposed approach is made feasible, essentially, by the approximation of Eq. (25). While this point is expressed formally, it is unclear why this 'linearly factorized distribution' is particularly suited for ABMs. It would seem like it is a very general solution for the problem of approximating the target distribution, but either it is a known solution that therefore should be properly referenced, or it is an ad-hoc solution for the problem of data assimilation on ABMs. Since this appears to be one of the cornerstones that make the proposed approach feasible, it should be properly explained concerning agent-based models. In other words, authors should answer the important question 'Why this is a good approximation of the target distribution specifically for ABMs?'

3. Authors introduce the concept of 'fermionic [abm] trajectory'. This appears as a justifiable but impacting limitation of the proposed approach. For this reason, it should be explained and analyzed in more depth. Why is it necessary and, more importantly, how frequently does it appear in real scenarios? In step 2 of page 19, it says that in experiments 'if the resulting trajectory isn't Fermionic, repeat from step 1'. How many trajectories are discarded this way?

4. Similarly, the authors make several very interesting qualitative claims for which more details and quantitative analysis should be performed and reported. For example "In our experiments, we have found we can attain very good acceptance probabilities with relatively simple renderings of $P(X|N)$." Or "It was found that the proportion of samples of each trajectory did converge towards the exact probability as the number of samples increased, up to a maximum of 10,000,000 samples"

Overall, the work marks an important advancement in the field, but it would need a more careful experimental design to be impactful in the community.

Is the rationale for developing the new method (or application) clearly explained?

Yes

Is the description of the method technically sound?

Yes

Are sufficient details provided to allow replication of the method development and its use by others?

Partly

If any results are presented, are all the source data underlying the results available to ensure full reproducibility?

Partly

Are the conclusions about the method and its performance adequately supported by the findings presented in the article?
Partly

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Machine learning, computational social science

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Reviewer Report 18 September 2023

https://doi.org/10.21956/openreseurope.15985.r34157

© 2023 Huang Y. This is an open access peer review report distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

?

Yilin Huang
Delft University of Technology, Netherlands, The Netherlands

The paper presents a method to address the sample impoverishment issue in data assimilation for ABMs. I like the proposed approach: limit the support of the posterior, markov chain abstraction of model states, among others; and a simple example (cat-mouse) model is given to illustrate the idea. While I fully recognize the potential of the topic and the contribution of the work, the paper has structural issues that need revision to create a clearer storyline and a more straightforward presentation of the method and its implications. This way, it can benefit the ABM community, and widen the use of DA.

My comments and suggestions: (the page numbers refer to the ones in the pdf version)

- The paper is hard to follow partly due to its highly complex (at places overly complex) math notations. (I deem myself sufficiently well-trained in math as a researcher in computer science. I had a hard time following them.) There are a large number of symbols (and super/subscriptions), some of which are not clearly explained, some are used in a counter-intuitive way, some can be simplified, some interrupt the main storyline and probably can be moved to a more suitable place in the main text, or the appendix, or removed. These are all over the article, I can just give a few examples:

  - in timestep pi, why is small psi a subscript of an agent state (little sigma, which is an element of the state space S), while capital psi stands for agents that is S dimensional? They are not easy to read.

- p7, the cat-mouse illustration: S and A seem to be sets or tuples, but then I realized they are encoded in the timestep expression. It is not clear what capital psi sub 1/2 stands for, etc. It seems the positions are numbered from 0, and agents are numbered from 1? I am not sure.

- p9, the unnumbered eq. under (9): why is proto sign used instead of = ?

- by the way, the set of trajectories is defined as capital Tau, which looks the same as capital T in the digital version.

- A list of symbols can help; before making it, simplify the symbols and expressions when possible.

- It would be better if you bring out the issue to address and the contribution of the paper earlier in the introduction. It is now stated at the end of the introduction which is 4 pages long.

- Separate the explanation of the equations (and rationale of choices), with the equations explaining the (steps of) methods. It will help if you first give an overview of the method / steps, and then get into the explanation. It is your choice to present the steps mathematically and/or algorithmically. In latter case, pseudo-code can help explain the steps.

- Explain why you used MH algm.

- p11 2nd paragraph: "If the number of agents is very much smaller than...(which is often the case with agent-based models)" - you do not need the part in the brackets (is it true?) to state your assumptions.

- I am not clear about the main idea of the last paragraph (boltzman distribution and thermodymanic system) on page 16.

- p20 "starting from these four initial solutions": did I miss something? Which four do you refer to? Explain the outcome of "85%...accepted and 70%...infeasible".

- Section 3/4 explained how you generated observations, assess the convergence of MC. It would be nice to actually demonstrate how to use the method in this small case step by step, and how well the method performed (MCMC for DA with ABM).

- Section 5 suggests 2 approaches for DA over longer time windows. It would be nice to discuss your method, its performance and implications. E.g., among others, what do you think of the fermionic assumption: does this generally hold in ABMs? If not, what changes are needed. This section can also be combined with the conclusion which is now mainly a very short summery of the paper (not a conclusion).

- The github repo seems also need more docs if you wish to help others reuse / reproduce / understand your method - I just had a quick browse, maybe I missed something.

The use of MCMC makes a nice addition to DA applied to ABM. Since many ABM modellers are not

familiar with DA, probably also MCMC, it is important that you clearly present the methods and its step by step instructions to the community and explain your choices.

Is the rationale for developing the new method (or application) clearly explained?
Partly

Is the description of the method technically sound?
Yes

Are sufficient details provided to allow replication of the method development and its use by others?
Partly

If any results are presented, are all the source data underlying the results available to ensure full reproducibility?
Yes

Are the conclusions about the method and its performance adequately supported by the findings presented in the article?
No

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Computer science, simulation theory and method

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Reviewer Report 19 August 2022

https://doi.org/10.21956/openreseurope.15985.r29598

© 2022 Kolic B. This is an open access peer review report distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

?

Blas Kolic
Institute for New Economic Thinking at the Oxford Martin School, University of Oxford, Oxford, UK

The authors present a novel data assimilation method for sampling the trajectory-parameter posterior distribution of agent-based models (ABM). The method approximates the posterior's support by constructing a convex polyhedron and uses a Metropolis-Hastings algorithm to explore such a polyhedron and obtain samples of the posterior. They validate their method in a small, discrete, agent-based predator-prey model.

I believe this paper's methodological contribution can have a very significant impact on the ABM community, particularly for latent state estimation. However, it needs an enormous amount of polishing as, in its current state, it is tough to follow, and many of the main ideas are buried deep within the text.

Find my comments in the list below. My intention with them is to try to remove the clutter and make your main ideas stand out. Of course, feel free to disagree with me if you feel as so.

- Notation and language seem all over the place. Sometimes I feel I'm reading a Physics piece, sometimes a Computer science one, sometimes a pure Mathematics one. My suggestion: think of your audience and pitch your notation and language to that audience.

- Put only the main concepts in the text and send everything else to appendices. Your method uses very non-standard terminology (at least for the ABM community) - e.g., mixed integer polyhedra, Fermionic trajectories - so keeping track of all the details in one go was nearly impossible for me. I suggest the following refurbishing of section 2, which I think captures the main ideas of your methodology:
  - Section 2.1 is mostly good. However, I would add a subsection explicitly listing the main assumptions you make. The assumptions I noticed are: 1) agent updates are synchronous, 2) trajectories are Fermionic (only one agent per state-action pair), 3) states and actions are finite, 4) agents are indistinguishable (timesteps count the number of agents in which state performing which action), 4) observations are i.i.d, 5) ABM dynamics are Markovian (as of eq.(8)). Moreover, I would really appreciate a table with notation and terminology.

  - Section 2.2 The main idea is that you can approximate supp(posterior) with polyhedron(posterior). I suggest you introduce convex mixed-integer polyhedra with a formal definition. I would move the details of the mapping between X and (T,\theta) to an appendix. I also suggest expressing polyhedron(posterior) as a single set of equality and inequality constraints just after you present eq. (14) and send all the details and proofs of eqs. (15) and (16) (transforming a union into an intersection) to an appendix. The discussion of Fermionic trajectories should be almost at the end of the section. From what I understand, their only use is removing the auxiliary variable "z", which is the last part of the construction of the polyhedron. I liked that you included the explicit example of the cat-mouse ABM.

  - Section 2.3. I'll go by sub-subsections
    - 2.3.1. The main idea is that you can choose a state from the Markov states M = {X_N <= H_N}. State what X_N and H_N are and move details to an appendix. Mention that X_N depends on the basis you choose, and direct the reader to section 2.3.4. Maybe you can put the set of Markov states as a formal proposition. Also, the hyper-rectangle H is an important quantity of your method, but it is buried deep in section 2.2.1. Make it stand out more.

    - 2.3.2. I think this section is good, but I didn't understand your discussion of real-valued variables if, by assumption, everything is discrete. I would include the approximation of the target distribution (section 2.3.3) in this section as you need it in eqs. (23) and (24).

- 2.3.3. See above.
- 2.3.4. I like this section. I would move the paragraph on the Boltzmann distribution closer to eq. (27) in the section above.
- 2.3.5. What's the main idea of the basis selection? Is it to separate X into X_B and X_N and make the P(X_N, i, \delta) matrix sparse so you can store it? If so, say it from the beginning and send all details (e.g., the transformation of polyhedra, Gaussian elimination) to an appendix.

- Simplify overcomplicated concepts and equations. I understand the race for mathematical rigor, but sometimes it blurs the main ideas. For instance, eqs. (7), (8), and (9) define the set of continuous trajectories in 4 steps. First, you say that trajectories, T, live in real space (R^N_{SA}). Second, you constrain T to the natural numbers (eq. (7)). Third, you define the set of continuous trajectories (eq. (8)). Finally, you intersect the set of natural numbers with the continuous trajectories. Instead, say that trajectories must be natural numbers in the first place (N^N_{SA}), and then define the continuous trajectories using eq. (8). Done. By the way, I would change the wording from "continuous" to "feasible" trajectories; saying continuous makes it sound that your agents evolve in continuous time. Another example of overcomplicating things is the approximation of the target distribution in section 2.3.3. What does discussing linearly factorized distributions add to the main argument? Wouldn't it be better to start with what you said just after eq. (27): "We want to approximate the target distribution in a way that it is maximal inside the polyhedron and decays outside of it at a certain rate," and present eq. (27) directly (without eqs. (25) and (26))? In general, look for overcomplicated cases in the main text and fix them.

- The results section is poor. While the description of the predator-prey model is good, most of what you compute are just convergence metrics of the algorithm; the only time you show the method's performance is in Fig 4. which is more qualitative than quantitative. There are many things that you can improve. See the following list of suggestions:
  - You should benchmark your methodology with other Bayesian inference/estimation methods. Check out Grazzini 2017[1], Platt 2022[2], Ward 2016[3] for some references (I'm sure you guys know of the Leeds group, ha).
  - Design an accuracy metric. Maybe you can compute the distance of samples from the posterior with the real trajectory. The idea is to quantify if a resulting distribution is better or worse than another.
  - How does the number of observations increase or reduce the power of your method? How well can you perform if you only have 1) the last observation of the trajectory, 2) the last two, 3) the last half, 4) all of it? In Kolic 2022[4], the authors discuss the issue of how the number of observations affects the accuracy of initialized latent states (as well as the information lost by aggregating them).
  - How good is your method in making predictions? Take samples at t=N & simulate up to t=N+n. How do the simulated and real trajectories look?
  - Make the dots in Figure 4 bigger and clearer.

- In Figure 4's caption, say that these positions and distribution are for the final state.

Other minor things:

- When referencing an equation, say what that equation represents (e.g., the posterior of eq. (12) rather than just eq. (12)).
- Discuss how tight the approximation of eq. (14) is and/if you can control it.
- Maybe I missed it, but how do you choose H?
- Letter "N" is used to denote number of timesteps (as in T \in R^N_{SA}) and to denote non-basic elements (as in X_N); change one of them.
- Make the domain of the support in eq. (13) explicit in the intersections (e.g., observations (w,v) is wrt \Omega).
- Change word contractions to full form (e.g., isn't -> is not)
- Use active rather than passive writing (e.g., we performed X rather than X was performed.)
- Explain what Metropolis-Hastings is and how it works (on a basic level) in section 1.4, not only that you will use it.
- First paragraph, page 11: for me it's not so clear why it is obvious that trajectories are a set of linear constraints by looking at eq. (9).
- First paragraph, page 11 (again). How can you compute the intersection of polyhedra (eq. (15)) if the components of the posterior (eq.(12)) are not linear constraints? The discussion about abstract representations puzzles me.
- Section 2.3.3. Do you have any idea of how the difficulty of computing log(pi) after eq. (28) increases with the dimension of the ABM? This looks like a major bottleneck to the algorithm.
- I didn't find the argument related to your method only working for agents with few bytes of internal state. Is it because of the desired sparsity of Z in the basis selection?
- Just to be sure, when you say that feasible samples are generated at every 42\mu s, with 85% accepted states and 70% of them feasible, does this mean that you have a total of 5.1 million feasible samples and that it took 214 seconds to compute everything? Are these timings after you constructed the polyhedron and the basis? If yes, how much time does computing these take?

Sorry for taking so long to review your article and for making the list of comments so huge. I think your paper will be great if you can make the main ideas stand out and prove the efficacy of your method better.

References
1. Grazzini J, Richiardi M, Tsionas M: Bayesian estimation of agent-based models. *Journal of Economic Dynamics and Control*. 2017; **77**: 26-47 Publisher Full Text
2. Platt D: Bayesian Estimation of Economic Simulation Models Using Neural Networks. *Computational Economics*. 2022; **59** (2): 599-650 Publisher Full Text
3. Ward JA, Evans AJ, Malleson NS: Dynamic calibration of agent-based models using data assimilation.*R Soc Open Sci*. 2016; **3** (4): 150703 PubMed Abstract | Publisher Full Text
4. Kolic B, Sabuco J, Farmer J: Estimating initial conditions for dynamical systems with incomplete information. *Nonlinear Dynamics*. 2022; **108** (4): 3783-3805 Publisher Full Text

Is the rationale for developing the new method (or application) clearly explained?
Partly

Is the description of the method technically sound?
Yes

Are sufficient details provided to allow replication of the method development and its use by others?
Yes

If any results are presented, are all the source data underlying the results available to ensure full reproducibility?
Yes

Are the conclusions about the method and its performance adequately supported by the findings presented in the article?
Partly

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: I'm a Physicist doing a Ph.D. in Mathematics. I have expertise in Dynamical Systems, Agent-Based Modelling, and Latent State Estimation using Data Assimilation.

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.