Testing A Particle Filtering Approach as a Means of Data Assimilation for Agent-Based Modelling [⋆]

XXXX Authors

No Institute Given

Abstract. XX ABSTRACT

Keywords: Agent-based modelling · Particle Filter · Uncertainty · Data assimilation · Bayesian inference

1 Introduction

Aim: How good is a particle filter at doing DA in a pedestrian/crowd ABM?

Structure / argument:

- Context: importance modelling people in environments, including whole cities.
- What data assimilation is (**?** have a nice, clear outline).
- Difference between particle filtering approaches (that do not dynamically change the model *state*) and others (e.g. EnKF) that do. Also note that this is not parameter estimation, although it could be.

2 Background

- How people have tried to do state (and parameter?) estimation in ABMs before
- How to evaluate PFs and (broadly) where they have been used
- Difference between normal parameter estimation with a (e.g.) GA and dynamic state estimation
- Data assimilation methods, focussing on Particle Filter
- Data assimilation methods in ABM (will be brief!)

[⋆] This work was supported by a European Research Council (ERC) Starting Grant [number 757455], a UK Economic and Social Research Council (ESRC) Future Research Leaders grant [number ES/L009900/1], and through an internship funded by the UK Leeds Institute for Data Analytics (LIDA).

3 Method

- Intro to station sim. Point to ODD
- Intro to particle filter
- Outline of experiments, including criteria to measure ‘success’ of the PF

Assumptions:

- Assume PF knows initial conditions

Other things to include:

- How to quantify ‘success’?
- Difference between PF and methods that dynamically alter the state. E.g.: “We note that this work is different from the work of dynamically calibrating or modifying the agent-based model based on real time data. In our work, the model is not dynamically changed. Instead, the simulation is dynamically reset to start from initial conditions that are estimated from real time sensor data, and thus the simulation/prediction results are dynamically adjusted in real time” (**?**).
- Difference between state / parameter estimation and traditional calibration. E.g.: “Also note that even though this work does not calibrate the agent-based model over time, it is possible to treat some of the model parameters as part of the system state and dynamically estimate (calibrate) those parameters based on real time sensor data.”(**?**).

3.1 The Agent-Based Model: StationSim

XXXX outline stations sim

3.2 Data Assimilation - Introduction and Definitions

XXXX Outline how data assimilation method works broadly (e.g. *update* and *predict*) and define general concepts / objects.

3.3 The Particle Filter

Here, the *state vector* contains all the information that a transition function needs to iterative the model forward by one step, including all of the agent ($i = \{0, 1, \dots, N\}$) parameters ($\overrightarrow{p_i}$) and variables ($\overrightarrow{v_i}$) as well as global model parameters $\overrightarrow{P}$:

$$S = \left[\, \overrightarrow{p_0}\ \overrightarrow{v_0}\ \overrightarrow{p_1}\ \overrightarrow{v_1} \dots \overrightarrow{p_N}\ \overrightarrow{v_N}\ \overrightarrow{P}\, \right] \tag{1}$$

The *observation vector* contains all of the observations made from the ‘real world’ (in this case the pseudo-truth model) that the particle filter uses to predict the current true state, with the addition of some Gaussian noise, ϵ:

$$O = \left[x_0 \; y_0 \; x_1 \; y_1 \; \ldots \; x_n \; y_n \right] \tag{2}$$

In this paper, the particle filter is not used to estimate the state of the models variables ($\overrightarrow{v_i}$), not any of the parameters ($\overrightarrow{p_i}$ and $\overrightarrow{P}$) – although it is worth noting that parameter estimation is technically feasible and will be experimented with in later iterations of this work. Therefore in the experiments conducted here, all parameters are fixed. Hence a further vector is required to map the observations to the state vector that the particle can actually manipulate. We define the partial state vector S_{partial} to match the shape of O, i.e.:

$$S_{\text{partial}} = \left[x_0 \; y_0 \; x_1 \; y_1 \; \ldots \; x_n \; y_n \right] \tag{3}$$

4 Experiments

4.1 Experiments with Uncertainty

Purpose here is basically to see how the particle filter behaves when we give it:

1. Number of particles (i.e. how many does it need to work reasonably well?)
2. Randomness in particles
3. Measurement noise (external)
4. Internal randomness (e.g. in agent behaviour)
5. (Simultaneous combinations of different randomness)

4.2 Experiments with Measurement Noise

1. Reduce the amount of information given to the particle filter (e.g. only allow it to optimise half of the state vector).
2. Aggregate the measurements (e.g. counts per area rather than individual traces).

5 Conclusion

XXXX Conclusion