SAMPLING FROM THE BAYESIAN POSTER OF AN AGENT-BASED MODEL GIVEN PARTIAL OBSERVATIONS

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

Nicholas Malleson
Leeds Institute for Data Analytics University of Leeds, UK

September 9, 2021

ABSTRACT

The discipline of data assimilation (DA) addresses the problem of how to make use of partial, noisy experimental observations to provide information about the time evolution of unobserved properties of a dynamical system. DA has developed rapidly in applications such as weather forecasting (Kalnay, 2003) but relatively little progress has been made in developing techniques that are applicable when the dynamical system is an agent based model. ABMs consist of 'agents' which often make discrete choices from a number of possible actions, meaning the space of model trajectories is not continuous and we cannot use assimilation techniques that require the gradient of the posterior in this space. In addition, a set of observations will typically refute a large proportion of model trajectories, making it difficult to even identify trajectories that have non-zero probability given the observations, and making it challenging to use algorithms that rely on sampling.

Here we present an algorithm that generates samples of the time evolution of an agent based model, given a set of noisy, incomplete experimental observations of the system. The algorithm approximates the set of possible trajectories as a linear program and uses an extension of the simplex algorithm to provide a proposal function for Markov-Chain-Monte-Carlo sampling.

We demonstrate the algorithm by performing data assimilation in an agent-based, spatial predator-prey model.

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

1 Introduction

1.1 Background

Data assimilation (DA) is a technique that has been widely used in the physical sciences such as meteorology (Kalnay, 2003), oceanography (Bertino, Evensen, & Wackernagel, 2003) and the earth sciences more broadly (Reichle, 2008). Specific applications of DA to ABMs are much rarer. Examples of data assimilation using well known DA methods such as Particle Filters and variants of the Kalman Filter applied to ABMs include models of crime (Lloyd, Santitissadeekorn, & Short, 2016), bus routes (Kieu, Malleson, & Heppenstall, 2020), pedestrian dynamics (Wang & Hu, 2015; Ward, Evans, & Malleson, 2016; Clay, Kieu, Ward, Heppenstall, & Malleson, 2020; Malleson et al., 2020); and population movement (Lueck, Rife, Swarup, & Uddin, 2019). Many DA algorithms rely on the differentiability of the model (Lewis, Lakshmivarahan, & Dhall, 2006).

2 Formulation of the problem

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

*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)

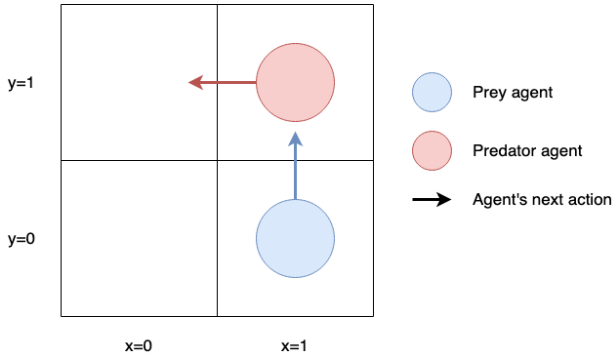

Figure 1: A simple predator-prey example.

- A domain of agent actions $\mathcal{A} = \{a_0 ... a_n\}$
- A domain of agent states $\mathcal{S} = \{\psi_0 ... \psi_m\}$
- An *agent timestep*, which is a computer program, $\pi(\psi, \Psi)$, where $\psi \in \mathcal{S}$ the state of an agent and $\Psi \in \mathbb{Z}^m$ is a (usually sparse) vector whose i^{th} element is the number of agents in state ψ_i at the start of the timestep. The program returns an action $a \in \mathcal{A}$ which defines the behaviour of an agent in state ψ and environment Ψ in a timestep. The program may make calls to a random number generator that returns a value $0 \leq r < 1$ with uniform probability, so its returned value may be non-deterministic and we write $P(\pi(\psi, \Psi) = a)$ to be the probability that the program will return action a, given inputs ψ and Ψ.
- An *action function*, $F : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{Z}^m$, which defines the effect of an action on the world, where $F(\psi, a)$ returns a (sparse) vector, Φ, whose i^{th} element gives the number of agents in state i that result from an agent in state ψ performing act a (including the final state of the acting agent).

As a simple demonstration, consider a spatial predator-prey model where agents are either predator or prey moving about a 2-dimensional grid. In this case, the domain of actions are $\mathcal{A} = \{a_0 = \text{move-left}, a_1 = \text{move-right}, a_2 = \text{move-up}, a_3 = \text{move-down}, a_4 = \text{give-birth}, a_5 = \text{die}\}$, and the domain of agent states, $\mathcal{S}$, can be thought of as the class

```
class Agent {
        Boolean isaPredator
        Integer yPosition
        Integer xPosition
}
```

It will be useful in the following to define an ordering of agent states. The ordering we choose isn't important but we choose to place agent `a` in the $(\texttt{a.isaPredator*G}^2 \texttt{ + a.yPositition*G + a.xPosition})^{th}$ position, where `G` is the number of positions along one side of the grid. So, for example, in the 2×2 grid of Figure 1 there are a total of 8 `Agent` states:

$$\begin{aligned}
\psi_0 &= [0, 0, 0] \\
\psi_1 &= [0, 0, 1] \\
\psi_2 &= [0, 1, 0] \\
\psi_3 &= [0, 1, 1] \\
\psi_4 &= [1, 0, 0] \\
\psi_5 &= [1, 0, 1] \\
\psi_6 &= [1, 1, 0] \\
\psi_7 &= [1, 1, 1]
\end{aligned}$$

where $[p, y, x]$ denotes an `Agent` in state (`isaPredator=p, yPosition=y, xPosition=x`).

Let a model timestep consist of a matrix E whose elements $e_{\psi a}$ are the number of agents in state ψ that perform act a in this timestep. For example, the timestep shown in Figure 1 would be

$$E = \begin{array}{c} \\ \psi_0 \\ \psi_1 \\ \psi_2 \\ \psi_3 \\ \psi_4 \\ \psi_5 \\ \psi_6 \\ \psi_7 \end{array} \begin{array}{c} \begin{array}{cccccc} a_0 & a_1 & a_2 & a_3 & a_4 & a_5 \end{array} \\ \left[\begin{array}{cccccc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \end{array}$$

where all elements are zero except those representing agent ψ_1 performing action a_2 and agent ψ_7 performing action a_0. Note that this matrix will generally be very sparse.

Finally, let a model trajectory be a tensor, T, consisting of a number of model timesteps. We use the notation T^t to denote the t^{th} timestep matrix, T^t_ψ to denote the ψ^{th} row of the t^{th} timestep matrix and $T^t_{\psi a}$ to denote the a^{th} element of the ψ^{th} row of the t^{th} timestep. By convention, we'll let indices begin at 0.

The timestep function for a predator/prey agent is shown in algorithm 1 (where we assume the grid has periodic boundary conditions).

Algorithm 1 Timestep of a predator/prey agent

```
function TIMESTEP(agent, otherAgents)
    if agent.isaPredator then
        if RANDOM( ) < β0 and number of prey on neighbouring gridsquares of otherAgents > 0 then
            return give-birth                ▷ more likely to reproduce when food is available
        end if
        if RANDOM( ) < γ0 then
            return die
        end if
    else
        if RANDOM( ) < β1 then
            return give-birth
        end if
        if RANDOM( ) < γ1 then
            return die
        end if
        if RANDOM( ) < δ and number of predators on this or neighbouring gridsquares of otherAgents > 0 then
            return die                       ▷ been eaten by predator
        end if
    end if
    return move-left, move-right, move-up or move-down based on a call to RANDOM( )
end function
```

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

$$\forall t, \psi, a : T^t_{\psi a} \geq 0, T^t_{\psi a} \in \mathbb{Z} \tag{1}$$

A trajectory, $T^t_{\psi a}$, must also be *continuous* which means that the number of agents in each state at the end of timestep $t-1$ must be the number of agents in each state at the beginning of timestep t. This can be expressed as the *continuity constraints*:

$$\forall t \in 1...n : \forall \phi : \sum_{\psi,a} F(\psi, a)_\phi T^{t-1}_{\psi a} - \sum_a T^t_{\phi a} = 0 \tag{2}$$

where the trajectory runs from $t = 0$ to $t = n$.

So, for a given number of timesteps, N, number of agent states, S, and number of agent actions, A, we can define the set of all trajectories, $\mathcal{T}^N_{SA}$ to be the set of tensors that satisfy the equations 1 and 2.

2.1 The posterior

If we let Ψ^t be the vector whose ψ^{th} element is the number of agents in state ψ at the beginning of timestep t, then the prior probability of a trajectory is

$$P(T) = \begin{cases} P\left(\Psi^0 = T^0\mathbf{1}\right) \prod_{\psi,t} P\left(T^t_\psi \mid \Psi^t = T^t\mathbf{1}\right) & \text{if } T \in \mathcal{T}^N_{SA} \\ 0 & \text{otherwise} \end{cases}$$

where $P(\Psi^0)$ is our prior beliefs about the state at time $t = 0$, and we use $\mathbf{1}$ to denote a vector whose elements are all 1.

The probability that a single agent in a given state will perform an action, given the state of the other agents, Ψ^t, is given by the agent timestep function, $P(\pi(\psi, \Psi^t) = a)$, so the joint probability that Ψ^t_ψ agents will perform actions T^t_ψ in an environment of other agents Ψ^t, is given by the multinomial distribution

$$P\left(T^t_\psi \mid \Psi^t\right) = \Psi^t_\psi! \prod_a \frac{P\left(\pi(\psi, \Psi^t) = a\right)^{T^t_{\psi a}}}{T^t_{\psi a}!}.$$

So the prior probability of a trajectory is

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

Suppose now we have a set of noisy, aggregate observations, Ω, that have a likelihood function $P(\Omega|T)$. By Bayes' rule, we have

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

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 the members $(\omega, v) \in \Omega$ 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.

In the case of the predator prey model, let's suppose that at the end of every day we go out to a fixed set of inspection sites and check for footprints. Suppose that if predators were present in the grid-square containing an inspection site on that day, there's a 50% chance we'll see a footprint, and if prey were present, there's a 25% chance. The observation function for a site is shown in algorithm 2

Algorithm 2 Function for observing predator/prey footprints at a site

```
function ObserveFootprints(T)                          ▷ T is the ABM trajectory
    t ← time of observation
    x ← x-position of observation
    y ← y-position of observation
    G ← size of grid
    ψ ← G² + Gy + x                                    ▷ state of predator in this gridsquare
    ϕ ← Gy + x                                         ▷ state of prey in this gridsquare
    F_pred ← False                                     ▷ did we observe predator footprints?
    F_prey ← False                                     ▷ did we observe prey footprints?
    if Σ_a T^t_ψa > 0 and Random( ) < 0.5 then
        F_pred ← True
    end if
    if Σ_a T^t_ϕa > 0 and Random( ) < 0.25 then
        F_prey ← True
    end if
    return F_pred, F_prey
end function
```

Given this

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

The posterior can now be written as

$$P(T|\Omega) \propto \begin{cases} P(\Psi^0 = T^0\mathbf{1}) \prod_{(\omega,v)\in\Omega} P\left(\omega(T) = v\right) \prod_{t,\psi} \left(T^t_\psi \cdot \mathbf{1}\right)! \prod_a \frac{P(\pi(\psi, T^t\mathbf{1}) = a)^{T^t_{\psi a}}}{T^t_{\psi a}!} & \text{if } T \in \mathcal{T}^N_{SA} \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

In many practical applications, sampling from this posterior is difficult because it has zero probability for the vast majority of tensors (i.e. most tensors are not trajectories, contain an impossible action or are refuted by the observations). Even though we can generate trajectories that fit the prior by simply performing a forward execution of the model from an initial state drawn from the prior, if the trajectory has no predator or prey at a time and place that we observed footprints, then the observations refute the trajectory and the probability falls to zero. It doesn't take many observations until the probability of randomly choosing a trajectory that fits the observations becomes very small indeed. So simple techniques such as rejection sampling, for example, are not practical. Techniques based on particle filtering may have more success but will likely soon reach a state containing a set of particles, none of which can be fit to the observations.

In this paper we'll show how the Metropolis-Hastings algorithm can be used to generate samples from equation 3. The key challenge will be to create a proposal function which randomly generates a proposed next sample given the current one. Given the current sample, finding a new trajectory with non-zero probability is still not easy. For example, a common strategy with Metropolis-Hastings is to generate a new sample by perturbing one or more elements of the previous sample at random. However, if we do this with an ABM trajectory it's very unlikely that the perturbed tensor will be a trajectory that contains only possible actions and satisfies the observations. So, the proposed next sample would almost certainly be rejected and we'd probably end up stuck on the first sample until we grew old.

3 Approximating the support of the posterior

We'll solve this problem by first calculating the support of the posterior, $\mathrm{supp}(P(T^t_{\psi a}|\Omega))$ (i.e. the set of trajectories that have non-zero probability).

From equation 3

$$\mathrm{supp}(P(T|\Omega)) = \mathcal{T}^N_{SA} \cap \mathrm{supp}(P(\Psi^0 = T^0\mathbf{1})) \cap \bigcap_{(\omega,v)\in\Omega} \mathrm{supp}\left(P\left(\omega(T) = v\right)\right) \cap \bigcap_{T^t_{\psi a}\neq 0} \mathrm{supp}\left(P\left(\pi'(t,\psi,T) = a\right)\right) \tag{4}$$

i.e. in order for T to have non-zero posterior probability, it must be a trajectory of the ABM, it must have a start state that has non-zero prior probability, all the observation likelihoods must be non-zero and the probability of each agent's action at every timestep must be non-zero.

3.1 Fermionic Trajectories

In the following we will restrict ourselves to considering *Fermionic trajectories*, which we define to be any trajectory whose elements are all either 0 or 1. i.e. in order to be a Fermionic trajectory a tensor must not only satisfy the constraints of equations 1 and 2 but must also satisfy the *Fermionic constraints*

$$\forall t, \psi, a : T^t_{\psi a} \in \{0, 1\} \tag{5}$$

This can be interpreted as meaning that no two agents in the same state at the same time can perform the same action. Notice that the Fermionic constraints imply the non-negative integer constraints, so a Fermionic trajectory is any tensor that satisfies equations 5 and 2. We'll refer to the set of all Fermionic trajectories as $\mathcal{F}^N_{SA}$.

In practice, the number of agents in an ABM is usually much smaller than the number of agent states, so the switch from $\mathcal{T}^N_{SA}$ to $\mathcal{F}^N_{SA}$ in equations 3 and 5 will often be an acceptable assumption that will have very little effect. However, if the Fermionic constraints are not acceptable, section ** describes how to extend the algorithm to apply to non-Fermionic trajectories.

3.2 Convex polyhedra

Since a trajectory in $\mathcal{F}^N_{SA}$ has NSA elements, we can think of each trajectory as a point in NSA-dimensional space.

The Fermionic constraints are equivalent to requiring that a Fermionic trajectory must lie on a vertex of the unit hypercube defined by

$$\forall t, \psi, a : 0 \leq T^t_{\psi a} \leq 1 \tag{6}$$

Since the vertices of a hypercube describe a convex polyhedron, and any subset of the vertices of a convex polyhedron also describes the vertex set of a convex polyhedron, then, as we add the if we condition the posterior on the Fermionic constraints (in addition to all the other constraints) we ensure that the support of the posterior can be described as the vertices of a convex polyhedron.

If we begin by adding the continuity constraints of equation 2, since they describe a set of flow constraints then, by ****'s theorem, the resulting set of linear constraints describes a convex polyhedron whose vertices are the valid, Fermionic trajectories.

Given a Fermionic trajectory we can easily convert the union term in equation 7

$$\mathcal{P}(\pi'(t,\psi,.),a) \cup \left\{T^s_{\xi b} : T^t_{\psi a} = 0\right\}$$

into a set of linear constraints. Expanding $\mathcal{P}$ we have

$$\left(\bigcap_j \sum_i a_{ij}x_i \leq b_j\right) \cup \{X : x_k = 0\} = \bigcap_j \left(\sum_i a_{ij}x_i \leq b_j \cup \{X : x_k = 0\}\right)$$

where, for conveinence, we've projected the elements of the trajectory $T^t_{\psi a}$ into a vector X. But

$$\sum_i a_{ij}x_i \leq b_j \cup \{X : x_k = 0\} = \sum_i a_{ij}x_i + \left(\sum_i \frac{|a_{ij}| + a_{ij}}{2} - b_j\right) x_k \leq \sum_i \frac{|a_{ij}| + a_{ij}}{2}$$

since all x_i are also either 0 or 1 so $\sum_i a_{ij}x_i$ must be less than $\frac{|a_{ij}|+a_{ij}}{2}$ and the constraint is satisfied if x_k is 0, whereas we have the original constraint if x_k is 1.

TODO: Fermionic trajectory implies convexity... Leads to convex PMF. Also if all supports have only integer vertices on the unit hypercube (i.e. are the convex hull of points on the unit hypercube) then the union is also a convex hull of points on the unit hypercube.

4 Abstract interpretation using convex polyhedra

The first two terms in equation 4 consists of the supports of computer programs whose inputs are ABM trajectories and whose outputs are given, i.e. the set of trajectories that, when passed to a computer program, would produce a given output.

Calculating the support of a computer program for a given output is, in full generality, NP-complete[2] but it is possible to use a technique known as *abstract interpretation*(Cousot & Cousot, 1977) to efficiently calculate a superset of the support. So, given a computer program ρ, we can calculate a set $\mathcal{P}(\rho, v)$ such that

$$\text{supp}(P(\rho(.) = y)) \subset \mathcal{P}(\rho, v)$$

Tools to perform abstract interpretation already exist (e.g. PAGAI(Henry, Monniaux, & Moy, 2012)) and are used widely in applications such as the verification of safety critical systems(Blanchet et al., 2003) and in practice $\mathcal{P}(\rho, v)$ is often reasonably tight (i.e. most members of $\mathcal{P}(\rho, v)$ are in $\text{supp}(P(\rho(.) = y))$). For our application we choose to express $\mathcal{P}(\rho, v)$ in terms of a set of linear inequalities on ρ's inputs, this corresponds to the abstract domain of convex polyhedra(Cousot & Halbwachs, 1978)(Becchi & Zaffanella, 2018). Calls to the random number generator can be dealt with in the abstract domain by generating a new variable, r, that satisfies $0 \leq r < 1$ for each call to `Random()`. These can either be left in as "auxiliary" variables in the same way as slack variables, or removed as soon as the variable goes out of scope by finding the convex hull of the projection into a lower dimensional space (this can be done using the double description method(Motzkin, Raiffa, Thompson, & Thrall, 1953)).

If we replace the supports in equation 4 with their equivalent supersets we end up with a superset of the posterior

$$\text{supp}(P(.|\Omega)) \subset \bigcap_{(\omega,v)\in\Omega} \mathcal{P}(\omega, v) \cap \bigcap_{t,\psi,a} \left(\mathcal{P}(\pi'(t, \psi, .), a) \cup \left\{T^s_{\xi b} : T^t_{\psi a} = 0\right\}\right) \cap \left\{T^t_{\psi a} \mid T^t_{\psi a} \text{is valid}\right\} \tag{7}$$

Constraining our proposal function to members of the superset in equation 7 instead of the true support won't affect the stationary distribution of the Markov Chain. If the proposal function happens to return a trajectory that isn't in $\text{supp}(P(\rho(.) = y))$ then it will just be rejected. This is fine as long as we generate acceptable proposals at a reasonable rate.

The final term in equation 7, which requires the trajectories in the support to be valid, consists of the continuity constraints of equation 2, which are also linear constraints so can be treated along with the other constraints, and the non-negative integer constraints of equation 1 which we leave as separate constraints.

The union term in equation 7 can also be transformed to a set of linear constraints, but the way we do this depends on the type of ABM we're dealing with, so we defer the explanation until sections 5.1 and ...

Without loss of generality, we express inequalities as equalities by introducing slack variables so that

$$\sum_i c_i x_i \leq y$$

[2]Consider, for example, a program that accepts an assignment of variables to truth values, and returns true if that assignment satisfies a Boolean formula. Deciding whether the support of this program, given that it returns true, is empty or not is equivalent to solving the Boolean satisfiability problem, which is known to be NP-complete(Cook, 1971)

is equivalent to

$$\sum_i c_i x_i + \lambda = y$$
$$\text{subject to } \lambda \geq 0$$

All the constraints can now be gathered into a standard matrix form

$$\begin{aligned} AX &= b \\ \text{subject to } \; l_i \leq x_i &\leq u_i \\ x_i &\in \mathbb{Z} \end{aligned} \tag{8}$$

where the vector X contains the elements of the trajectory tensor $T^t_{\psi a}$ and the slack variables λ.

5 From convex polyhedron to Markov process

Given a set of inequalities that contain the support of the posterior, we next need to define a stochastic proposal function $f : \mathbb{S} \to \mathbb{S}$ on a set of Markov states, $\mathbb{S}$, where each state, S, is associated with a trajectory $T(S)$. The proposal function should have the following properties:

- The graph whose nodes are Markov states and whose edges are possible transitions, should be a connected graph (i.e. there should be a path between any two Markov states). This ensures proper mixing as time tends to infinity.
- At the stationary point, the probability of being in a state associated with trajectory T should be our target posterior probability $P(T \mid \Omega)$.
- For any transition from state $S_a \to S_b$ the probability of transitioning from $S_b \to S_a$ should be non-zero. This allows us to attain detailed balance by using the Metropolis Hastings algorithm.
- The proposal function should be computationally efficient.

5.1 A proposal function for a Fermionic ABM

So, taking our predator prey agent of algorithm 1 as an example, we can see that the move, reproduce and die actions can be returned irrespective of the state of other agents[3], so for these actions the union is also unconditionally true. However, the reproduction actions of a predator are conditional on there being some prey on neighbouring gridsquares. So, suppose we have a predator in state ψ at time t, the support of it's timestep function given that it returns a reproduction action is

$$\sum_{i \in N^t_\psi} -x_i \leq -1$$

where N^t_ψ is the set of indices that correspond to elements of $T^t_{\phi a}$ where ϕ is a prey on a gridsquare neighbouring ψ at time t. So, taking the union with $\{X : x_k = 0\}$, the final constraint for a predator in state ψ at time t is

$$\sum_{i \in N^t_\psi} -x_i + x_k \leq 0$$

Theorem 1. *All trajectories that satisfy equations 1 and 5 are at extreme points. Where an extreme point is one that can't be expressed as a linear combination of two other points that satisfy the constraints.*

Proof. Inequality 5 can only be satisfied by non-negative integers if, for any given t and ψ either all $T^t_{\psi a}$ are zero or one element of $T^t_{\psi a}$ is one and the rest zero. In the first case the solution is touching all $|a|$ of the $T^t_{\psi a} \geq 0$ constraints, in the second case the solution is touching $|a| - 1$ of the $T^t_{\psi a} \geq 0$ constraints and one $\sum_a T^t_{\psi a} \leq 1$ constraint to give a total of $|a|$.

[Conversely, we can show that all extreme points are integer points, so we have the polytope of integer solutions]

[In the case of the action Fermionic constraint, this can be seen immediately since equations 1 and the action Fermionic constraint describe a unit hupercube, so all integer solutions are on vertices.]

□

[3] the Fermionic constraint is dealt with separately, so we don't need to worry about that here

This seemingly abstract fact leads to a computationally very efficient way of proposing new trajectories that have non-zero probability.

Given a system of m linear constraints on n variables expressed in the form $AX = B$, let a *pivot state* be a partition of the variables into m *basic variables* and $n - m$ *non-basic variables* so we can express the constraints in the form

$$BX_B + NX_N = b \tag{9}$$

where X_B and X_N are the basic and non-basic variables respectively, B is the matrix formed from the columns in A that correspond to the basic variables and N is the matrix formed from the columns of A that correspond to the non-basic variables. If we now constrain all non-basic variables to be on one of their bounds (in the case of action variables, this is either 0 or 1) then we're left with

$$BX_B = b - NX_N$$

but since B is an $m \times m$ square matrix then

$$X_B = B^{-1}b - B^{-1}NX_N$$

gives a unique solution, as long as B is invertable. If B is invertable and X_B has no negative entries then we say that the pivot state is valid. It can be shown that all valid pivot states of equations 8 correspond to extreme points(Dantzig, Orden, Wolfe, et al., 1955).

We begin by representing the constraints and the current trajectory in the form of equation 9. If we multiply any row of A_b by some constant, c, then the truth of equation 9 can be maintained by multiplying the same row of A_n and of B by c as well. Similarly if we add one row of A_b to another row, the equations can be maintained by performing the same row operation on A_n and B. If the pivot state is valid, then A_b is invertable so we can use Gaussian elimination, by performing a sequence of row additions and multiplications, to transform A_b into a form such that each row and column is zero on all elements except one, which is 1. We'll call this the standard form. Once in this form, if we assume $X_n = 0$ the solution for X_b can be read off directly from B.

Given a valid pivot state in standard form, a perturbed valid solution can be generated by choosing a column, C, of A_n, with at least one positive-valued row, and choosing from among those rows the one, C_i, that minimises B_i/C_i. We then make the variable corresponding to column C a basic variable and make the basic variable corresponding to row i of A_b non-basic by swapping column C with the column on row i of A_b that has value 1. A_b is now not in standard form as column C will, in general, have more than one non-zero value so we perform Gaussian elimination (row additions and multiplications) to return it to standard form. This perturbation is known as a *pivot* operation and is the same method as employed in the Simplex algorithm(Dantzig et al., 1955)(Vanderbei, 2020). This is exactly what we need for our proposal function since it efficiently generates a new valid trajectory from a current one.

TODO: re-do to include upper bounds on variables...

...So, each Markov state consists of a pivot-state (a partition of variables into m basic varaibles and N-m non-basic variables) and an active-bound state (which assigns either upper or lower bound to each non-basic variable).

5.1.1 Dealing with degeneracy

The use of pivoting is complicated somewhat when the solutions are degenerate. A degenerate solution is one that can be generated by more than one pivot state. This can occur when a pivot state has any basic variables that are on their bounds (in which case we say the pivot state is degenerate and call the variables on their bounds *degenerate varaibles* of the pivot state). A degenerate variable can be pivoted-out without changing the solution since during a pivot the outgoing basic variable becomes constrained to one of its bounds; but if the basic-variable's value is already on its bound to start with, constraining it has no effect on the solution (for a thorough theoretical development of degeneracy see (Zörnig, 1993)). This is problematic because we need to assign a probability to each Markov state but under degeneracy the same solution can be associated with a number of Markov states, so if we associate a Markov state with the probability of its trajectory then degenerate trajectories are at risk of being "counted" multiple times, once for each Markov state that has that solution. To maintain the correct distribution, the probabilities of all the Markov states with a given trajectory should sum to the probability of that trajectory.

However, note that in a valid Fermionic trajectory, all the structural variables are binary (i.e. either 0 or 1) so are always on one of their bounds. So, if we split the variables into binary and non-binary variables, and restrict ourselves to the subset of Markov states where all non-basic variables are binary (or fixed, and no fixed variable is basic), then every pivot state can be made to represent any valid solution by activating the appropriate bounds of the non-basic variables. This means that for any given valid trajectory, there exists one Markov state for each pivot state. So, if we divide up the probability equally between these Markov states, then each Markov state, (π, X), has a probability $P(X)/N$ where N is the number of pivot states. However, since N is constant, we need not actually do the division for the purposes of MCMC.

5.1.2 Infeasible states

In order to improve mixing during MCMC we allow the Markov chain to pass through simplex states that are infeasible. In order to make these Markov states, we must assign a probability to each infeasible state and define a way of proposing transitions to and from these states.

A state can become infeasible when one or more of the basic variables goes outside their bounds.

Let

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

be the infeasibility of a variable x in relation to its lower, l and upper u bounds, and let the infeasibility of a basic vector, X_B, be

$$\iota(X_B) = \sum_i \iota_{l_i}^{u_i}(X_{Bi})$$

Now let

$$\rho_l^u(x) = \begin{cases} -1 & \text{if } x < l \\ 1 & \text{if } x > u \\ 0 & \text{otherwise} \end{cases}$$

be the derivative of the infeasibility ι so that

$$\frac{d\iota(X_B)}{dX_{Bi}} = \rho_{l_i}^{u_i}(X_{Bi}) = R(X_B)$$

and $R(X_B)$ can be thought of as an "infeasibility objective function" for a given basic vector X_B, for which the reduced objective, D, is defined as

$$D = -R(X_B)B^{-1}N$$

so that

$$D_j = -R(X_B)B^{-1}N_j = \sum_i \frac{d\iota(X_B)}{dX_{Bi}} \frac{dX_{Bi}}{dX_{Nj}} = \frac{d\iota(X_B)}{dX_{Nj}}$$

is the rate of change of X_B-infeasibility with the j^th non-basic variable.

5.1.3 Proposing a pivot column: Potential energy

When proposing a pivot, we first decide which column to pivot on.

Let the potential energy of the j^{th} non-basic variable in pivot state μ be

$$E_j(\mu) = D_j(X_{Nj} - H(-D_j))$$

where H is the Heaviside step function.

A column is chosen with probability proportional to the exponential of its potential energy

$$P(j) = \frac{e^{k_c E_j}}{\sum_{j'} e^{k_c E_{j'}}}$$

Let

$$E_P = \sum_j E_j$$

be the total potential energy of the Markov state, and let the *energy penalty* be

$$P_E = \frac{\sum_j e^{k_c E_j}}{n \prod_j e^{k_c E_j}} = \frac{\sum_j e^{k_c E_j}}{n} e^{-k_c E_P}$$

where n is the number of non-basic variables. Notice that the energy penalty of all feasible states is 1.

So, if we multiply the probability of the Markov state by its energy penalty then

$$P_E P(j) = \frac{e^{-k_c(E_P - E_j)}}{n}$$

and the contribution of the column decision to the Metropolis-Hastings acceptance probability becomes

$$\lambda_j = \frac{P'_E P'(j)}{P_E P(j)} = e^{k_c((E_P - E_j) - (E'_P - E'_j))} = e^{-k_c(\Delta E_P - \Delta E_j)}$$

5.1.4 Proposing a pivot row and column offset

Once a pivot-column is chosen, a pivot row on that column is chosen, along with an offset, Δ_j for the column, to give the proposed pivot. In order to be a valid pivot, Δ_j should bring the leaving variable to one of its bounds. If Δ_j brings column j to its opposite bound, we also have the option to swap the active bound of column j without performing any pivot. Let this be signified by a pivot-row of -1. If $\Delta_j = 0$ we also have the option to perform the 'null-pivot' which leaves everything unchanged.

Pivots that make a non-binary variable non-basic are immediately excluded. In order to maintain integer solutions, we also restrict ourselves to pivot points whose tableau coefficient have an absolute value of 1 (TODO: prove that this maintains integer solutions and maintains connectedness of the Markov states). Let α_j be the set of remaining pivots on column j.

An active pivot point is chosen with a probability proportional to the exponential of the infeasibility of the post-pivot solution (note that this includes any infeasibility of the entering variable). i.e.

$$P(\iota \to \iota' | j) = \frac{e^{-k_r \iota'}}{\sum_{\iota''(X_B + \Delta'' T_j):\Delta'', i'' \in \alpha_j} e^{-k_r \iota''}}$$

where $T_j = -B^{-1}N_j$ is the t^{th} column of the tableau.

TODO: Need to include column j itself in this, and its contribution to infeasibility.

After any pivot, the set of available vertices of the reverse pivot is the same as for the forward pivot so the probability of the reverse pivot $P(\iota' \to \iota | j)$ have the same denominator sums. So,

$$\frac{P(\iota \to \iota' | j)}{P(\iota' \to \iota | j)} = \frac{e^{-k_r \iota'}}{e^{-k_r \iota}} = e^{-k_r(\iota' - \iota)}$$

If we let the *infeasibility penalty* of a Markov state be

$$P_\iota = e^{-k_r \iota}$$

and multiply the probability of a Markov state by its infeasibility penalty, the contribution of the choice of pivot row and column offset to the Metropolis-Hastings acceptance is

$$\frac{P'_\iota P(\iota' \to \iota | j)}{P_\iota P(\iota \to \iota' | j)} = \frac{e^{-k_r \iota'} e^{-k_r \iota}}{e^{-k_r \iota} e^{-k_r \iota'}} = 1$$

Notice again that the infeasibility penalty of feasible states is 1, so again it has not effect on feasible states.

5.1.5 Proposal function summary

To summarise, the probability of a Markov state, μ, is the probability of the trajectory times the energy penalty times the infeasibility penalty

$$P(\mu) = P(X(\mu)) \frac{\sum_j e^{k_c E_j(\mu)}}{n} e^{-k_c E_P(\mu)} e^{-k_r \iota(\mu)}$$

The probability of proposing a pivot (i, j, l), where l identifies the bound of the leaving variable (or the j^{th} column if $i = -1$) is given by

$$P(\mu \to \mu') = \frac{e^{k_c E_{j(\mu \to \mu')}(\mu)}}{\sum_{j'} e^{k_c E_{j'}(\mu)}} \frac{e^{-k_r \iota(\mu')}}{\sum_{\mu'' \in a_j} e^{-k_r \iota(\mu'')}}$$

So

$$P(\mu) P(\mu \to \mu') = \frac{P(X(\mu)) e^{-k_c (E_P(\mu) - E_{j(\mu \to \mu')}(\mu))} e^{-k_r (\iota(\mu) + \iota(\mu'))}}{n \sum_{\mu'' \in a_{j(\mu \to \mu')}} e^{-k_r \iota(\mu'')}}$$

So, the MH acceptance probability is

$$\frac{P(\mu') P(\mu' \to \mu)}{P(\mu) P(\mu \to \mu')} = \frac{P(X(\mu')) e^{-k_c (E_P(\mu') - E_{j(\mu' \to \mu)}(\mu'))}}{P(X(\mu)) e^{-k_c (E_P(\mu) - E_{j(\mu \to \mu')}(\mu))}}$$

So the final contribution to the MH acceptance probability is

$$\lambda_j = e^{k_c (\Delta E_P - \Delta E_j)}$$

TODO: should k_c and k_r (or alternatively E_P and ι) be per column and row so that the rate of decay doesn't depend on the size of the problem? Can we approximate the number of infeasible vertices (for given infeasible variable set. For n infeasible vars, the infeasible vertices are described by 2^n nnc polyhedra. Alternatively, if we ensure that $k_r \iota$ dominates over $k_c E_P$ then we should expect the dynamics of a phase 1 potential energy pivot...i.e. ensure that probabilities increase monotonically in the forward phase 1 direction).

An efficient way of calculating the infeasibility of all pivots on a column is as follows:

6 An alternative proposal function

If we restrict ourselves to degenerate pivots (i.e. pivots that don't change state) and bound swaps, then we have the following algorithm:

First choose column with zero probability for columns with zero reduced objective [need to have finite probability to allow swaps that reduce the objective to zero for this column], and with probability equal to the exponential of their potential energy otherwise. [instead, have a finite weight $1/N$ for low energy columns, where N is the total number of columns, and exponentials for high, so prob of choosing a high-potential column is $e^{kE_j}/(\sum e^{kE_l} + w_0)$ and prob of choosing a low-potential column is $w_0/n_0(\sum e^{kE_l} + w_0)$ where n_0 is the number of low energy columns. So, if we make the penalty prob $(\sum e^{kE_l} + w_0)/((w_0/n_0)_0^n e^{kE_P})$ but since this needs to be 1 in the feasible case, $(w_0/n_0)^(n_0 - N)e^{kE_P}$]

Once a column is chosen, we choose to either swap bounds or make a degenerate pivot. A degenerate pivot can be performed on any row that is on one of its bounds (i.e. not one that is currently outside its bound) and that has a non zero pivot element. We choose to bound swap with probability proportional to the exponential of the reduction in infeasibility (how do we normalise this?). If we choose a degenerate pivot, we choose with uniform probability from among the possibilities.

The proposal function for ordered pivot states is shown in algorithm 3

Algorithm 3 Proposal function for ordered pivot states

function PROPOSAL(A_b, A_n, B) ▷ Standard representation of the current ordered pivot state
  $A'_b, A'_n, B' \leftarrow A_b, A_n, B$
  $\alpha, \beta \leftarrow$ constant parameters
  $p_0 \leftarrow$ the set of degenerate pivots of (A_b, A_n, B) (i.e. the ones that do not change the solution)
  $p_1 \leftarrow$ the set of non-degenerate pivots of (A_b, A_n, B) (i.e. the ones that change the solution)
  $P_{f/b} \leftarrow 1$ ▷ the ratio of probabilities of making this transition forwards/backwards
  if BERNOULLI(α) **then**
    $r \leftarrow$ choose member of p_1 with uniform probability
    perform pivot r on (A'_b, A'_n, B')
    $p'_1 \leftarrow$ number of non-degenerate pivots of (A'_b, A'_n, B')
    $P_{f/b} \leftarrow \frac{|p'_1|}{|p_1|}$
  else if BERNOULLI(β) **then**
    $r \leftarrow$ choose member of p_0 with uniform probability
    perform pivot r on (A'_b, A'_n, B')
    $p'_0 \leftarrow$ number of degenerate pivots of (A'_b, A'_n, B')
    $P_{f/b} \leftarrow \frac{|p'_0|}{|p_0|}$
  else
    choose two rows a and b with uniform probability
    swap rows a and b of (A'_b, A'_n, B')
  end if
  return $(A'_b, A'_n, B', P_{f/b})$
end function

We now show that this assignment of probability to ordered pivot states has the property that the sum of the probabilities of ordered pivot states associated with a trajecotry is the probability of that trajectory.

First some notation: let $\mathbb{E}(T)$ be the set of all ordered pivot states with solution T, let $T(S)$ be the trajectory associated with pivot state S, let $D(S)$ be the tuple of degenerate variables in ordered pivot state S, let $D_{\leq n}(S)$ be the n-tuple consisting of the first n elements of $D(S)$ and let

$$C(\mathbb{S} \mid V) = \left\| \left\{ V^+ : S \in \mathbb{S}, V^+ = D_{\leq |V|+1}(S), V^+_{\leq |V|} = V \right\} \right\| \tag{10}$$

i.e. among the members of $\mathbb{S}$ that have the degenerate variable prefix V, $C(\mathbb{S}|V)$ counts how many distinct degenerate variable prefixes of length $|V| + 1$ there are.

Theorem 2. *If we assign the probability*

$$P(S) = \frac{\sigma_{T(S)}!}{m!} \frac{P_{T(S)}}{\prod_{q=0}^{\sigma_{T(S)}-1} C(\mathbb{E}(T(S)) \mid D_{\leq q}(S))} \tag{11}$$

to pivot state S, where m is the number of constraints and σ_T is the degeneracy of trajectory T (i.e. m minus the number of non-zero variables in T) then

$$\sum_{S\in\mathbb{E}(T)} P(S) = P_T$$

Proof. Expanding the sum over degenerate states

$$\sum_{S\in\mathbb{E}(T)} P(S) = P_T \frac{\sigma_T!}{m!} \sum_{S\in\mathbb{E}(T)} \frac{1}{\prod_{q=0}^{\sigma_T-1} C(\mathbb{E}(T) \mid D_{\leq q}(S))}$$

Notice first that all pivot states with the same σ_T-tuple of degenerate variables, $D(S)$, have the same probability. There are exactly $\frac{m!}{\sigma_T!}$ ordered pivot states with a given $D(S)$, corresponding to the different placings of the $m-\sigma_T$ non-degenerate basic variables among the m possible positions. So, if we let

$$\mathbb{D}(T) = \{D(S) : S \in \mathbb{E}(T)\}$$

then

$$\sum_{S\in\mathbb{E}(T)} P(S) = P_T \sum_{D\in\mathbb{D}(T)} \frac{1}{\prod_{q=0}^{\sigma_T-1} C(\mathbb{E}(T) \mid D_{\leq q})}$$

Since $C(\mathbb{S} \mid V)$ depends only on the order of the degenerate variables in the members of $\mathbb{S}$

$$C(\mathbb{E}(T) \mid V) = C(\mathbb{D}(T) \mid V)$$

so

$$\sum_{S\in\mathbb{E}(T)} P(S) = P_T \sum_{D\in\mathbb{D}(T)} \frac{1}{\prod_{q=0}^{\sigma_T-1} C(\mathbb{D}(T) \mid D_{\leq q})} \tag{12}$$

if we separate the terms in the sum into groups that share the same prefix apart from the last variable then we get

$$\sum_{S\in\mathbb{E}(T)} P(S) = P_T \sum_{V:D\in\mathbb{D}(T),V=D_{\leq(\sigma_T-1)}} \left(\frac{1}{\prod_{q=0}^{\sigma_T-2} C(\mathbb{D}(T|V_{\leq q}))} \sum_{V^+:V^+\in\mathbb{D}(T),V=V^+_{\leq(\sigma_T-1)}} \frac{1}{C(\mathbb{D}(T|V))} \right)$$

So

$$\sum_{S\in\mathbb{E}(T)} P(S) = P_T \sum_{V:D\in\mathbb{D}(T),V=D_{\leq(\sigma_T-1)}} \frac{1}{\prod_{q=0}^{\sigma_T-2} C(\mathbb{D}(T|V_{\leq q}))} \frac{\left\| \left\{ V^+ : V^+ \in \mathbb{D}(T), V = V^+_{\leq(\sigma_T-1)} \right\} \right\|}{C(\mathbb{D}(T|V_{\leq(\sigma_T-1)}))}$$

but the final fraction here is 1 by equation 10, so

$$\sum_{S\in\mathbb{E}(T)} P(S) = P_T \sum_{V:D\in\mathbb{D}(T),V=D_{\leq(\sigma_T-1)}} \frac{1}{\prod_{q=0}^{\sigma_T-2} C(\mathbb{D}(T|V_{\leq q}))}$$

we can now use the same argument again to reduce the upper bound of q iteratively until we show that the sum on the right hand equals one, so

$$\sum_{S\in\mathbb{E}(T)} P(S) = P_T$$

□

Theorem 3. *Algorithm* **??** *assigns a probability*

$$P(S) = \frac{\sigma_{T(S)}!}{m!} \frac{P_{T(S)}}{\prod_{q=0}^{\sigma_{T(S)}-1} C(\mathbb{E}(T(S)) \mid D_{\leq q}(S))}$$

to pivot state S

Proof. By inspection of the algorithm, it can be seen that the algorithm assigns probability

$$P(S) = \frac{\sigma_{T(S)}!}{m!} \frac{P_{T(S)}}{\prod_{r=0}^{\sigma_{T(S)}-1} (\|C'(S,r)\| + r + 1)}$$

where

$$C'(S,r) = C'(S,r-1) \cup N(S,r)$$
$$C'(S,-1) = \emptyset$$

and $N(S, r)$ is the set of indices of the non-zero entries in the r-from-last degenerate row of A_n. So, $C'(S, r)$ is the set of columns that have at least one non-zero entry on a degenerate row on or below the r-from-last.

However, $C(\mathbb{E}(T(S)) \mid D_{\leq q}(S))$ is, by definition, the number of different degenerate variables that can take the $(q+1)^{th}$ place in the list of degenerate variables, given that the first q are given by $D_{\leq q}(S)$.

Now, given an ordered pivot state, S in standard form; if there exists a column, j, with a non-zero entry in a degenerate row, i, of A_n or A_b below the q^{th} degenerate row, then we can generate a valid member of $\mathbb{E}(T(S))$ with j the $(q+1)^{th}$ variable by pivoting on column j and row i (if not already pivoted in), then swapping row i with whichever row makes it the $(q+1)^{th}$ degenerate variable. Conversely, if there exists a member of $\mathbb{E}(T(S))$ with j as the $(q+1)^{th}$ variable, it must be possible to pivot on the j^{th} column without pivoting out any of the first q degenerate variables or the non-degenerate variables. The only way this is possible is if there is either a non-zero entry in the j^{th} column of A_n or A_b below the q^{th} degenerate row. So, the set of degenerate variables that can take the $(q+1)^{th}$ place is the set of columns that have non-zero entries in degenerate rows below the q^{th} in A_n or A_b. From the form of A_b we know immediately that there are $r+1$ columns with non-zero degenerate entries on or below the r-to-last degenerate row. So

$$C(\mathbb{E}(T(S)) \mid D_{\leq \sigma_{T(S)}-1-r}(S)) = \|C'(S, r)\| + r + 1$$

□

6.0.1 Metropolis-Hastings on ordered pivot states

All this can now be assembled into a sampling algorithm based on Metropolis-Hastings. Starting with the constraints in the form of equation 8, recursively choose a non-zero element on an un-pivoted row and pivot on that element, until we reach an ordered pivot state. In general, B will contain negative elements which means that the solution will not conform to the non-negative constraint. In order to find an initial positive solution let $I^- = \{i : B_i < 0\}$ be the set of rows of B that are negative and pivot on the column, j, that minimises $\sum_{i \in I^-} A_{ij}$. Repeat until a positive solution is found (or no column has a negative sum, in which case no positive solution exists).

From the initial valid pivot state, S, generate a proposal pivot state S' and a transition probability ratio $\frac{P(S \to S')}{P(S' \to S)}$ using algorithm 3 and accept with probability

$$\alpha = \frac{P(S')}{P(S)} \frac{P(S' \to S)}{P(S \to S')}$$

where the probability of a pivot state is given by algorithm **??**.

Given this, it's clear that all the necessary properties of the proposal function are fulfilled.

7 Results

8 Conclusion

References

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

Bertino, L., Evensen, G., & Wackernagel, H. (2003, August). Sequential Data Assimilation Techniques in Oceanography. *International Statistical Review*, *71*(2), 223–241. doi: 10.1111/j.1751-5823.2003.tb00194.x

Blanchet, B., Cousot, P., Cousot, R., Feret, J., Mauborgne, L., Miné, A., … Rival, X. (2003). A static analyzer for large safety-critical software. In *Proceedings of the acm sigplan 2003 conference on programming language design and implementation* (pp. 196–207).

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

Cook, S. A. (1971). The complexity of theorem-proving procedures. In *Proceedings of the third annual acm symposium on theory of computing* (pp. 151–158).

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

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

Dantzig, G. B., Orden, A., Wolfe, P., et al. (1955). The generalized simplex method for minimizing a linear form under linear inequality restraints. *Pacific Journal of Mathematics*, *5*(2), 183–195.

Evans, A. (2012). Uncertainty and Error. In A. J. Heppenstall, A. T. Crooks, L. M. See, & M. Batty (Eds.), *Agent-Based Models of Geographical Systems* (pp. 309–346). Dordrecht: Springer Netherlands.

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

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

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

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

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

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

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

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

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

Swarup, S., & Mortveit, H. S. (2020). Live simulations. In *Proceedings of the 19th international conference on autonomous agents and MultiAgent systems* (pp. 1721–1725). Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems.

Talagrand, O. (1991). The Use of Adjoint Equations in Numerical Modelling of the Atmospheric Circulation. In A. Griewank & G. F. Corliss (Eds.), *Automatic Differentiation of Algorithms: Theory, Implementation, and Application* (pp. 169–180). Philadelphia, PA: SIAM.

Vanderbei, R. J. (2020). *Linear programming* (Vol. 5). Springer.

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

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

Zörnig, P. (1993). A theory of degeneracy graphs. *Ann Oper Res*, *46*, 541–556. doi: 10.1007/BF02023113