Stochastic Modelling and Random Processes

Hand-out 4
Random sequential update, Gillespie algorithm

The properties of Poisson processes (see hand-out 3) can be used to set up efficient sampling algorithms for stochastic particle systems. Here we focus on a system with state space $S = \{0,1\}^\Lambda$ with lattice/population set Λ and flip dynamics (for example the contact process), and sketch two algorithms: the random sequential update, and the rejection-free Gillespie algorithm.

Gillespie algorithm (direct simulation).
Pro: is rejection-free, i.e. a transition occurs in every step
Con: can be computationally heavy since rates and probabilities are computed in each step
$\rightarrow$ works best for processes with very heterogeneous transition rates.

The sampling rate $R(\eta) = \sum_{x \in \Lambda} c(\eta, \eta^x)$ is state-dependent and needs updating in each time step. Pick η_0 from the initial distribution and set $t = 0$. Then repeat iteratively:

(0) compute/update the sampling rate $R(\eta)$ and probabilities $p_x(\eta)$,
(1) update the time counter by $t+ = \text{Exp}(R(\eta))$,
(2) pick a site x with probability $p_x(\eta) = c(\eta, \eta^x)/R(\eta)$,
(3) update (flip) site x.

So in total, the transition $\eta \to \eta^x$ happens with the correct rate $R(\eta) \frac{c(\eta,\eta^x)}{R(\eta)} = c(\eta, \eta^x)$.

Random sequential update (RSU).
Pro: works with fixed sampling rates and probabilities that can be pre-computed and stored, simple to implement and can be computationally cheap
Con: if transition rates are heterogeneous, proposed updates may be rejected with high probability which leads to oversampling and waste of computational time
$\rightarrow$ works best for processes with homogeneous, similar transition rates.

To resolve the full dynamics on site $x \in \Lambda$, the (fixed) sampling rate should be $r_x \geq \max_{\eta \in S} c(\eta, \eta^x)$, determined by the fastest process. The independent PPs on each site add up, and the next possible event in the whole system is sampled at rate $R = \sum_{x \in \Lambda} r_x$. By the thinning property, the probability that it happens on site x is given by $p_x = r_x/R$. This leads to the following algorithm:
Pick η_0 from the initial distribution and set $t = 0$. Then repeat iteratively:

(1) update the time counter by $t+ = \text{Exp}(R)$,
(2) pick a site x with probability p_x,
(3) update (flip) site x with probability $c(\eta, \eta^x)/r_x$.

So in total, the transition $\eta \to \eta^x$ happens with the correct rate $R \frac{r_x}{R} \frac{c(\eta,\eta^x)}{r_x} = c(\eta, \eta^x)$.

For example, for the 1D contact process on $\Lambda = \{1, \ldots, L\}$ with periodic boundaries and rates

$$c(\eta, \eta^x) = \delta_{1,\eta(x)} + \lambda \delta_{0,\eta(x)} \big(\delta_{1,\eta(x-1)} + \delta_{1,\eta(x+1)} \big)$$

we have $r_x = r = \max\{1, 2\lambda\}$, and thus $p_x = 1/L$ choosing sites uniformly and $R = rL$.

Computational complexity.
To simulate one unit of real time of the process the update loops in both algorithms are called $O(\mathbf{L})$ times, assuming that $R(\eta)$ or R are of order $|\Lambda| = L$. Steps **(1)** and **(3)** need $O(\mathbf{1})$ steps to complete. Implementing a **binary decision tree** for the probabilities $p_x(\eta)$ or p_x step **(2)** can be completed in $O(\log \mathbf{L})$ operations. For **Gillespie**, the decision tree depends on the current state η and has to be updated. Since transitions are only local in nature (involving a fixed number of sites independently of L), this can be completed again in $O(\log L)$ steps, so

$$\text{computational complexities are} \quad O(L \log L) \quad \text{for both algorithms .}$$

Initialization of η_0 and the decision tree takes of order L steps in both cases. By symmetries of the model **(2)** is often a uniform choice for **RSU** which needs only $O(1)$ operations, improving the complexity to $O(L)$. But in practice, the acceptance probability $c(\eta, \eta^x)/r_x$ of proposed moves determines whether this is more efficient than Gillespie.

Simplified time counter.
In both algorithms, $R, R(\eta) = O(L)$ are usually of order of the system size, so the increments $\tau_i \sim \text{Exp}(R)$ of the time counter are of order $1/L$. By the **scaling property** $\alpha \text{Exp}(\beta) \sim \text{Exp}(\beta/\alpha)$ of exponential rv's (check!), we have

$$\tau_i \sim Exp(R) \sim \frac{1}{R} \tilde{\tau}_i \quad \text{with normalized} \quad \tilde{\tau}_i \sim Exp(1) .$$

To simulate up to a time $T = O(1)$ we therefore need of order $RT = O(L)$ sampling increments τ_i. The time counter of the simulation is then

$$t = \sum_{i=1}^{RT} \tau_i = \frac{1}{R} \sum_{i=1}^{RT} \tilde{\tau}_i = T + O(L^{-1/2}) \to T \quad \text{as } L \to \infty ,$$

by the law of large numbers. So if we just replace the increments τ_i by their mean $1/R$, i.e. use

(1)' update the time counter by $t+ = 1/R$

instead of the computationally more expensive (1), the error in t is of order $L^{-1/2}$ by the central limit theorem. This is usually negligible for large L unless one is interested in very precise time statistics.