1 Rare-event sampling with the Giardina-Kurchan-Lecomte-Tailleur algorithm

Let us assume we are interested in a stochastic system $\mathrm{d}X = f(X,t)\mathrm{d}t + \sigma(X,t)\mathrm{d}B(t)$ with $X \in \mathbb{R}^d$, and an observable $O(X(t)) \in \mathbb{R}$. We can then identify a region $\mathcal{A} \subset \mathbb{R}^d$ of the phase space that is not often visited by the system and has particular values of the observable O. We then define a score function $V(\{X\}, t)$ that is a proxy for the probability of trajectory $\{X\}$ reaching $\mathcal{A}$.

For simplicity here, we will assume that the score function is either a simple function v of the observable at time t, i.e. $V(\{X\}, t) = v(O(X(t)))$, or its time integral $V(\{X\}, t) = \int_0^t v(O(X(s)))\mathrm{d}s$. In the present case, a natural observable is the AMOC strength itself $O = \text{AMOC}$, possibly time-averaged. Then, the simplest score function choice is using simply $V(X,t) = -\text{AMOC}(X(t))$. This choice makes the assumption of persistence to approximate probabilities, namely: "the AMOC is more likely to collapse in the future if it is already weak right now".

After we have defined the score function, we initialize N ensemble members with trajectories $\{X_j^{(0)}\}$, $j = 1, \ldots, N$, each ending at time $t_j^{(0)}$. For simplicity and without loss of generality, we can assume $t_j^{(0)} = t_0 = 0$, and, since the dynamics is Markovian, we can ignore everything that happens before t_0. Finally, to complete the initialization of our ensemble we compute the initial scores $V_j^{(0)} = V(\{X_j^{(0)}\})$, and assign to each trajectory an unbiasing weight $\pi_j^{(0)} = 1$. This weight will track the likelihood to observe trajectory j when running the climate model with no REA, and it will be crucial to have access to the unbiased probabilities of events sampled by the biased trajectories.

Once we have our initial ensemble, we need to select a resampling time τ and a selection strength k. Then, for each iteration, we will integrate forward in time each ensemble member producing candidate trajectories (denoted with $\tilde{\bullet}$), and select which ones will survive to the next iteration based on their scores. More precisely, for $i = 1, \ldots, I$:

1. Extend each trajectory integrating forward in time for one resampling step τ, obtaining $\{\tilde{X}_j^{(i)}\}$, $j = 1, \ldots, N$ that run from t_0 to $t_i = i\tau$.

2. Compute for each trajectory its score function at the end of the resampling step:
$$\tilde{V}_j^{(i)} = V(\{\tilde{X}_j^i\}, t_i)$$

3. Compute for each trajectory a weight according to the improvement of its score in the last resampling step:
$$\tilde{w}_j^{(i)} = \exp\left(k\left(\tilde{V}_j^{(i)} - V_j^{(i-1)}\right)\right)$$

and then normalize the weights so that they sum up to N:

$$w_j^{(i)} = \frac{1}{Z^{(i)}}\tilde{w}_j^{(i)}, \quad Z^{(i)} = \frac{1}{N}\sum_{j=1}^{N}\tilde{w}_j^{(i)}$$

4. Resample N new trajectories by cloning or killing according to the weights. More precisely:

   (a) Compute a tentative number of clones for each trajectory: $\tilde{m}_j^{(i)} = \lfloor w_j^{(i)} + u_j^{(i)} \rfloor$, where $u_j^{(i)} \sim \mathcal{U}(0,1)$ are j independent random numbers distributed uniformly between 0 and 1.

   (b) Adjust the number of clones so that they sum to N. In particular if $\Delta N^{(i)} = \sum_{j=1}^{N}\tilde{m}_j^{(i)} - N > 0$, then we randomly select $\Delta N^{(i)}$ values of j among the ones for which $\tilde{m}_j^{(i)} > 0$ and reduce by one the number of clones of these trajectories. On the other hand, if $\Delta N^{(i)} < 0$, then we randomly clone $|\Delta N^{(i)}|$ trajectories selecting again from the ones for which $\tilde{m}_j^{(i)} > 0$. Finally, we are left with definitive number of clones $m_j^{(i)}$ that do add up to N.

   (c) Create the parent mapping

   $$p^{(i)} : \{1, \dots, N\} \to \{1, \dots, N\},$$
   $$p^{(i)}(j) = \min\{l \text{ such that } \sum_{n=1}^{l} m_n \geq j\}$$

   which links each trajectory to the one it was cloned from.

   (d) Actually perform the cloning and killing: $\{X_j^{(i)}\} = \{\tilde{X}_{p^{(i)}(j)}^{(i)}\}$. Similarly, update the scores and unbiasing weights:

   $$V_j^{(i)} = \tilde{V}_{p^{(i)}(j)}^{(i)}, \quad \pi_j^{(i)} = \frac{\pi_{p^{(i)}(j)}^{(i-1)}}{w_{p^{(i)}(j)}^{(i)}}$$

In the original paper by Giardina et al. **?**, the weights in point 3 are computed not as the improvement of the score function but as the score function itself at the end of the resampling step. However, in **?**, the authors suggest using the improvement of the score function.

We implemented the algorithm both for a fully deterministic ocean-only Veros configuration, as well as for a configuration with a temperature noise component. In the deterministic case, in the cloning step 4.d we introduce a small perturbation to the trajectories added to the temperature T and salinity S fields, which allows them to diverge. The perturbation is defined as

$$\begin{aligned} T &\mapsto T + T\epsilon_T \eta \\ S &\mapsto S + \epsilon_S \eta, \end{aligned} \tag{1}$$

where η is sampled from a normal distribution with mean 0 and variance 1, independently for each field and grid point. The noise amplitudes are chosen to be $\epsilon_T = 0.001$ and $\epsilon_S = 0.002\ \mathrm{g\,kg^{-1}}$. Typical salinity values are around $34.5\ \mathrm{g\,kg^{-1}}$ with fluctuations of the order of $0.2\ \mathrm{g\,kg^{-1}}$, while temperature values have a mean of 5 deg C with a standard deviation of the order of 10 deg C, though the temperature distribution is strongly skewed due to the heating from the top. Thus, in Eq. 1 we employ a perturbation that is multiplicative in temperature, such that the deep ocean temperatures that are close to 0 degrees are perturbed less than the warm surface ocean. The perturbations in salinity are additive, since all values are very close on an absolute scale.

To make inferences on the averages and probabilities of observables in the REA ensemble, we need to account for the fact that the ensemble is biased by the sampling. Given an observable $U(\{X\})$, one can approximate its average over the stationary measure of the system with the empirical average over many realizations, where each trajectory has the same weight. The fact that we biased the trajectories by cloning and killing them is accounted for by the unbiasing weights $\pi_j^{(i)}$:

$$\bar{U}_N = \frac{1}{N}\sum_{j=1}^{N} \pi_j^{(I)} U(\{X_j^{(I)}\}) \sim_{N\to\infty} \mathbb{E}\left[U(\{X\})\right], \tag{2}$$

where $\mathbb{E}$ is the expectation over the stationary measure. Since the trajectories are not independent, the central limit theorem doesn't apply, and thus the typical error can be larger than $1/\sqrt{N}$. In practice, convergence will be faster for observables closely linked to the score function V used for biasing the ensemble, while for others it may be more efficient to run a simple unbiased Monte Carlo estimation.

Note that if we are interested in computing probabilities rather than averages, we can simply think of a probability as the expectation of an indicator function:

$$\mathbb{P}\left(U(\{X\}) > u\right) \equiv \mathbb{E}\left(\mathbb{I}_{U(\{X\})>u}\right) \tag{3}$$

Now, if we unravel the expression for the unbiasing weights, we get

$$\pi_j^{(I)} = \left(\prod_{i=1}^{I} Z^{(i)}\right) \exp\left(-k\left(V(\{X_j^{(I)}\}_{-T_{\text{memory}}\le t\le I\tau}) - V(\{X_j^{(I)}\}_{-T_{\text{memory}}\le t\le 0})\right)\right), \tag{4}$$

where we had a telescopic canceling in the exponential. This shows that the bias we introduced with the algorithm is proportional to the exponential of the improvement of the score over the whole algorithm run for each trajectory.

Having been developed to compute large deviation rate functions **?**, there is a large deviation principle for the variation of the score function

$$v_T = V(\{X\}_{t\leq T}) - V(\{X\}_{t\leq 0}), \tag{5}$$

which can be written as in **?**:

$$\mathbb{P}\left(v_T \geq a\right) \asymp_{T\to\infty} e^{-T\mathcal{I}(a)}, \tag{6}$$

where $\asymp$ denotes log-equivalence and $\mathcal{I}$ is the large deviation rate function, which can be computed as the Legendre-Fenchel transform of the scaled cumulant generating function $\lambda(k)$: $\mathcal{I}(a) = \sup_k\{ka - \lambda(k)\}$. The GKTL algorithm provides an efficient estimation of such scaled cumulant generating function at the selection strength k (and its neighborhood **?**):

$$\lambda(k) = \lim_{I\to\infty} \frac{1}{I\tau} \sum_{i=1}^{I} \log Z^{(i)}. \tag{7}$$

We use Eq. 2 to compute the unbiased probabilities of being below threshold a:

$$\mathbb{P}(X(T) \leq a) = \frac{1}{N} \sum_{j=1}^{n} \pi_j^{(I)} \mathbb{I}_{X_j^{(I)}(T)\leq a}. \tag{8}$$

In the main text, we give results for a chosen as the median of the current distribution, as this uses most of the ensemble members and thus has low statistical uncertainty. Given ordered realizations in the ensemble at iteration i, this yields the unbiased probability $p_{1/2}^{(i)}$ of being below the median $a_{1/2}^{(i)}$:

$$p_{1/2}^{(i)} = \mathbb{P}(X(i\tau) < a_{1/2}^{(i)}) = \frac{1}{N} \sum_{l=1}^{N/2} \pi_{j_l}^{(i)}, \quad X_{j_1}^{(i)} \leq X_{j_2}^{(i)} \leq \ldots \leq X_{j_l}^{(i)} \leq \ldots \leq X_{j_N}^{(i)}. \tag{9}$$

The algorithm has four 'hyperparameters': the ensemble size N, the resampling time τ, the selection strength k and the score function V. The ensemble size is a matter of compromise between accuracy and computational cost, while the other parameters require some degree of understanding of the system. The resampling time should be of the order of the Lyapunov time **?**. Long enough to allow clones of the same ensemble member to separate sufficiently, but not so long that they relax back to the attractor. Namely, we don't want segments of trajectories to completely lose memory of their initial condition.

Concerning the selection strength k, it should be chosen based on how extreme of an event we want to observe: the more extreme the event, the higher the selection strength. More precisely, if we assume the random variable v_T defined in Eq. 5 has a normal distribution with mean μ and variance σ^2, then to observe values of the order of a, an indicative value of k should be

$$k \sim \frac{a-\mu}{\sigma^2} \tag{10}$$

Equivalently, if we want to push the system n standard deviations from the mean μ, we should set $k \sim n/\sigma$ **?**. However, the higher k is, the more trajectories will be killed at each iteration, greatly reducing the diversity of the final ensemble. At the extreme this may lead to all trajectories sharing a single 'parent', and we call this *ensemble collapse* or *extinction*. It becomes then clear that the ensemble size limits how heavily we can push the system, and, thus, the values of a that we will be able to sample.

Finally, the score function can be the most crucial parameter of the algorithm. It can make the difference between a more or less efficient sampling of the event of interest or a complete failure to observe the event at all. In fact, it is possible to prove that there is an *optimal* score function **?**, which is closely related to the committor function and gives the most efficient sampling of our rare event of interest. Unfortunately, our ignorance of the committor function is often the reason we want to run the REA in the first place.

To choose the resampling time of the rare event algorithm the autocorrelation time of the AMOC strength in control simulations was considered. At $\tau = 5$ yr the autocorrelation is around 0.4 and cloned trajectories are already starting to branch off. Thereafter the autocorrelation does not fully drop off, and there is still correlation on the centennial timescale. Thus, $\tau = 50$ yr can be chosen, which gives ample time for clones to separate.

The value of k to push the system one standard deviation from the mean (Eq. 10) can be estimated from the control run as a function of the total time for which we want to run the REA. The value turns out not to depend much on the total integration time, and since Eq. 10 only gives an order of magnitude estimate for the selection strength, we can conclude that k should be of order 10 Sv^{-1}. In order to be able to run as many experiments as possible with different hyperparameters we use the relatively small ensemble size of $N = 50$.

The practical implementation of the GKTL algorithm and its application to the Veros model is available at `https://github.com/AlessandroLovo/REA-Veros`.