Pullback score function for AMS: Double-Well potential

Fabian Linsenmeier, Valérian Jacques-Dumas
Supervision: Prof. Dr. Henk Dijkstra
September 2024 - January 2025

1 Double-Well model and AMS

1.1 Double-well potential

We use the time-dependent stochastic Double-Well potential given by:

$$V(x,t;\mu) = 0.25x^4 - 0.5x^2 - \mu x t \tag{1}$$

The coupling parameter μ determines how quickly the potential changes over time. The time dependence causes a saddle-node bifurcation starting at $\mu \cdot t \approx 0.25$. Figure 1 displays the potential at different times t. The differential equation reads:

$$dx_t = -\nabla V(x,t;\mu) + g \cdot dW_t \tag{2}$$

Here, g determines the level of noise and dW denotes the Wiener process.

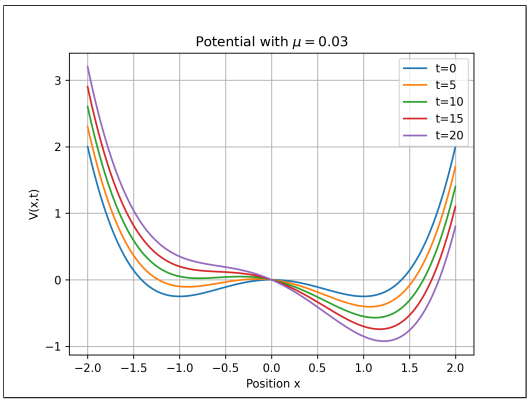


Figure 1: Time-dependent Double-Well potential. Plotted for different times: $t = [0, 5, 10, 15, 20]$

We are interested in transition probabilities from Onstate to the Offstate. The boundaries of On- and Offstate are time-dependent. Figure 2 shows the state boundaries over time.

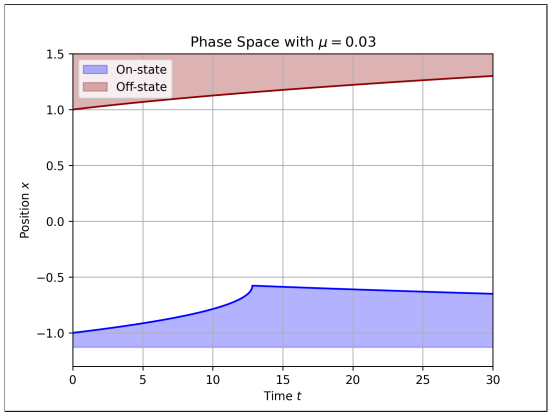


Figure 2: Phase space diagram of the potential. On- and offstate are coloured in blue and red, respectively.

We simulate trajectories starting from the edge of the Onstate and count the number of trajectories which transition to the Offstate. We simulate each trajectory until it either returns to the Onstate (after previously having left the Onstate) or reaches the Offstate. We do not distinguish between temporary and permanent transitions.

1.2 AMS

In recent years, adaptive multilevel-splitting algorithms (AMS, TAMS) have successfully estimated transition probabilities for rare events, such as those present in conceptual ocean models [1, 2, 3, 4]. AMS and the similar TAMS algorithm work by iteratively rebranching trajectories until a certain number of transitions has happened. For more details regarding these algorithms, see e.g. [5] or [6].

In this study, we only focus on AMS (not on TAMS). The reason is, that we expect AMS to be more compatible with the Pullback attractor. We denote the transition time to the off-state as t_{off} and the transition time of returning to the on-state as t_{on}. TAMS estimates $P(t_{off} < T_{max})$, where the maximum transition time T_{max} is a free parameter. In contrast to that, AMS estimates $P(t_{off} < t_{on})$. This is the probability of reaching the offstate before returning to the onstate and its definition does not contain a fixed time limit. Since the Pullback attractor does not contain a maximum transition time, we expect it to match better with AMS.

1.3 AMS: Score function

To determine which trajectory to rebranch at which point, AMS needs a measure of proximity to the target set. This function needs to be defined over the full phase space and is called score function $\phi(x,t)$. The ideal score function is the so-called committor function, which returns the transition probability to the target state for each point in phase-space. This function is usually not available. Therefore, the success of AMS depends on the choice of a score function which succesfully approximates the committor function. There have been different ways in trying to get reliable score functions for AMS/TAMS [3, 4]. In this study, we apply a score function inspired by the pullback attractor.

2 Pullback attractor

Non-autonomous models are dynamical systems with time-dependent mappings. Attractor states from autonomous models are substituted by Pullback and Forward attractors in non-autonomous models [7]. The Pullback attractor A_t is a subset of the phase-space defined as:

$$\lim_{t_0 \to -\infty} dist(\phi(t, t_0), A_t) = 0 \text{ for fixed } t \tag{3}$$

As an intuition, the pullback attractor (PB) represents the trajectory that all trajectories starting at $t_0 \to -\infty$ converge to.

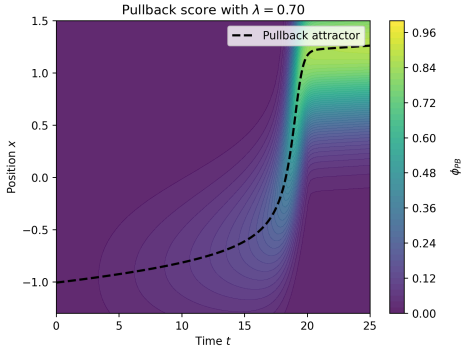
3 Pullback score function

We use the idea of [8] to construct a score function around the Pullback trajectory. Our algorithm proceeds in the following way: First, we assign curvilinear coordinates $s \in [0,1]$ to every point on the PB-trajectory based on arc length. Then, for every point in phase space (x,t), we look for the closest point (x_{ref}, t_{ref}) on the pullback trajectory. We do this by constructing and querying a k-dimensional tree [9, 10] to speed up the computation.

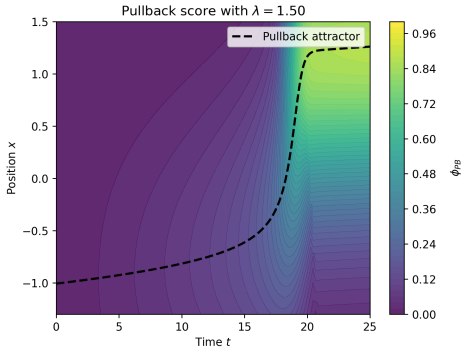
The resulting score $\phi_{PB}(x,t)$ is a combination of the euclidean distance $d = dist\{(x,t),(x_{ref},t_{ref})\}$ and the curvilinear coordinate at our reference point $s(x_{ref}, t_{ref})$:

$$\phi_{PB}(x,t) = s\left(x_{ref}, t_{ref}\right) \cdot exp\left(-\frac{d^2}{\lambda^2}\right) \tag{4}$$

The decay length λ is a free parameter which determines how rapidly the score decays orthogonal to the PB-trajectory. A visualization of ϕ_{PB} for different decay lengths can be seen in Figure 4.



(a) $\phi_{PB}(\lambda = 0.7)$



(b) $\phi_{PB}(\lambda = 1.5)$

Figure 4: Pullback score function for different decay lengths.

4 Results

4.1 Simulation setup

To evaluate the Pullback-inspired score function, we apply different AMS and Monte-Carlo algorithms to our

model. We perform simulations for three different noise levels. At lower noise, the simulated trajectories should follow the pullback attractor more closely, so we would expect a better performance by ϕ_{PB}. Table 1 lists the full simulation setup. The zone boundaries, initial states as well as the Pullback attractor are visualized in Figure 5.

Parameter	**Description**
Time-coupling	$\mu = 0.03$
Integration scheme	Euler-Maruyama with $dt = 0.01$
Initial times	$t_{init} = [2.0, 4.0, 7.0, 10.0]$
Initial positions	Roots @ t_{init}
AMS: Parameters	$N_{traj} = 10.000$, $n_c = 100$
AMS: Score functions	ϕ_{PB} and $\phi_{simple} = \frac{x+1}{2}$
MC: Simulation rule	Simulate until > 30 transitions
Number of runs	$N_{runs} = 20$
Noise factor	$g = [0.1, 0.05, 0.02]$

Table 1: Simulation setup - List of choices and parameters

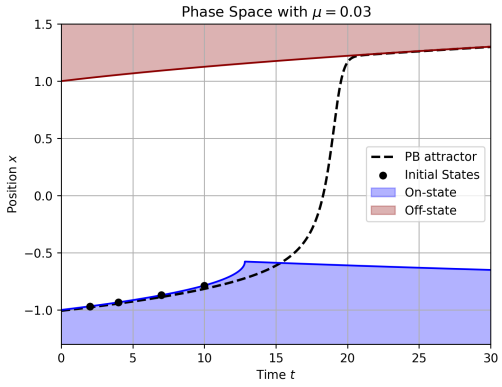


Figure 5: Simulation setup - Phase space

4.2 Simulation results

The results for high, medium and low noise are presented in Tables 2, 3 and 4. We can see that $AMS(\phi_{simple})$ and MC match quite well for all initial conditions and noise levels. However, the results for $AMS(\phi_{PB})$ do not match. They exhibit a clear bias towards higher probabilities for all initial states and noise levels. Interestingly enough, the bias seems to grow with **smaller** noise levels which is contrary to what we expect (see above). Furthermore, we observe that the bias seems to be largest for the later initial states, i.e. largest for $t_{init} = 10.0$.

5 Discussion

In our report, we have discussed the application of AMS with a Pullback-inspired score function to the time-dependent Double-Well potential. We find that $AMS(\phi_{PB})$ performs worse than $AMS(\phi_{simple})$ and exhibits a clear bias towards larger transition probabilities. The bias increases for smaller noise levels and later initial states.

The interpretation of these results is not straight forward since we would expect $AMS(\phi_{PB})$ to work better, especially for low noise levels. Since $AMS(\phi_{simple})$ produced reliable results, the issue must lie in the shape and time-dependence of the Pullback score function. One possible explanation might be visible in Figure 5. We see that the Pullback attractor shows a delayed response to the bifurcation. Trajectories following the Pullback remain inside the onzone longer than the bifurcation point. When the trajectories then leave the onzone, the shape of the potential does not allow them to return to the on-state. Instead, the trajectories reach the off-state. This might lead to an overestimation of the transition probability.

There is room for further improvement with this project. A more detailed analysis is required to find the cause for the bias that $AMS(\phi_{PB})$ exhibits. In addition to that, TAMS compared to AMS seems to be a more applicable algorithm to most practical problems. The Pullback score can be adapted easily to work with TAMS but a reasonable choice must be made for T_{max}.

Code availability
The code for this project is available on Github:
`https://github.com/fabian-lnsm/Pullback-score-AMS`

$\mathbf{g = 0.1}$	$q_{init} \approx (2.0, -0.97)$	$q_{init} \approx (4.0, -0.94)$	$q_{init} \approx (7.0, -0.87)$	$q_{init} \approx (10.0, -0.79)$
ϕ_{simple}	$(6.9 \pm 5.6) \times 10^{-7}$	$(1.1 \pm 0.3) \times 10^{-5}$	$(6.5 \pm 0.5) \times 10^{-4}$	$(9.7 \pm 0.4) \times 10^{-3}$
$\phi_{\text{PB}}(\lambda = 1.5)$	$(8.7 \pm 0.4) \times 10^{-7}$	$(1.6 \pm 0.1) \times 10^{-5}$	$(7.1 \pm 0.3) \times 10^{-4}$	$(10.4 \pm 0.3) \times 10^{-3}$
MC	$(7.0 \pm 1.2) \times 10^{-7}$	$(1.5 \pm 0.3) \times 10^{-5}$	$(6.7 \pm 0.7) \times 10^{-4}$	$(9.5 \pm 0.3) \times 10^{-3}$

Table 2: High noise ($g = 0.1$). The table shows the estimated mean and standard deviation of the transition probability for each initial state and algorithm.

$\mathbf{g = 0.05}$	$q_{init} \approx (2.0, -0.97)$	$q_{init} \approx (4.0, -0.94)$	$q_{init} \approx (7.0, -0.87)$	$q_{init} \approx (10.0, -0.79)$
ϕ_{simple}	$(15.0 \pm 2.8) \times 10^{-6}$	$(15.2 \pm 1.9) \times 10^{-5}$	$(31.2 \pm 1.5) \times 10^{-4}$	$(23.2 \pm 0.8) \times 10^{-3}$
$\phi_{\text{PB}}(\lambda = 1.5)$	$(16.8 \pm 0.6) \times 10^{-6}$	$(20.5 \pm 0.8) \times 10^{-5}$	$(37.7 \pm 0.9) \times 10^{-4}$	$(26.5 \pm 0.6) \times 10^{-3}$
MC	$(14.2 \pm 2.9) \times 10^{-6}$	$(17.7 \pm 2.7) \times 10^{-5}$	$(32.5 \pm 1.7) \times 10^{-4}$	$(23.1 \pm 0.5) \times 10^{-3}$

Table 3: Medium noise ($g = 0.05$). The table shows the estimated mean and standard deviation of the transition probability for each initial state and algorithm.

$\mathbf{g = 0.02}$	$q_{init} \approx (2.0, -0.97)$	$q_{init} \approx (4.0, -0.94)$	$q_{init} \approx (7.0, -0.87)$	$q_{init} \approx (10.0, -0.79)$
ϕ_{simple}	$(34.8 \pm 1.5) \times 10^{-4}$	$(13.2 \pm 0.4) \times 10^{-3}$	$(54.4 \pm 0.9) \times 10^{-3}$	$(12.9 \pm 0.2) \times 10^{-2}$
$\phi_{\text{PB}}(\lambda = 0.7)$	$(37.8 \pm 0.8) \times 10^{-4}$	$(14.9 \pm 0.3) \times 10^{-3}$	$(63.0 \pm 1.1) \times 10^{-3}$	$(14.4 \pm 0.2) \times 10^{-2}$
MC	$(33.8 \pm 1.9) \times 10^{-4}$	$(13.6 \pm 0.3) \times 10^{-3}$	$(56.4 \pm 0.6) \times 10^{-3}$	$(12.9 \pm 0.1) \times 10^{-2}$

Table 4: Low noise ($g = 0.02$). The table shows the estimated mean and standard deviation of the transition probability for each initial state and algorithm.

References

[1] Daniele Castellana, Sven Baars, Fred W Wubs, and Henk A Dijkstra. Transition probabilities of noise-induced transitions of the atlantic ocean circulation. *Scientific Reports*, 9(1):20284, 2019.

[2] Sven Baars, D Castellana, Fred W Wubs, and Henk A Dijkstra. Application of adaptive multilevel splitting to high-dimensional dynamical systems. *Journal of Computational Physics*, 424:109876, 2021.

[3] Valérian Jacques-Dumas, René M van Westen, Freddy Bouchet, and Henk A Dijkstra. Data-driven methods to estimate the committor function in conceptual ocean models. *Nonlinear Processes in Geophysics*, 30(2):195–216, 2023.

[4] Valérian Jacques-Dumas, René M van Westen, and Henk A Dijkstra. Estimation of amoc transition probabilities using a machine learning–based rare-event algorithm. *Artificial Intelligence for the Earth Systems*, 3(4):e240002, 2024.

[5] Frédéric Cérou and Arnaud Guyader. Adaptive multilevel splitting for rare event analysis. *Stochastic Analysis and Applications*, 25(2):417–443, 2007.

[6] Thibault Lestang, Francesco Ragone, Charles-Edouard Bréhier, Corentin Herbert, and Freddy Bouchet. Computing return times or return periods with rare event algorithms. *Journal of Statistical Mechanics: Theory and Experiment*, 2018(4):043213, 2018.

[7] Peter Kloeden and Meihua Yang. *An introduction to nonautonomous dynamical systems and their attractors*, volume 21. World Scientific, 2020.

[8] Pascal Wang, Daniele Castellana, and Henk Dijkstra. Improvements to the use of the trajectory-adaptive multilevel sampling algorithm for the study of rare events. *Nonlinear Processes in Geophysics Discussions*, 2020:1–24, 2020.

[9] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[10] Scipy.spatial: Kd-tree. `https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.KDTree.html`. Accessed: 2025-01-04.