Control Consistency Losses for Diffusion Bridges

Samuel Howard
Department of Statistics
University of Oxford

Nikolas Nüsken
Department of Mathematics
King's College London

Jakiw Pidstrigach
Department of Statistics
University of Oxford

Abstract

Simulating the conditioned dynamics of diffusion processes, given their initial and terminal states, is an important but challenging problem in the sciences. The difficulty is particularly pronounced for rare events, for which the unconditioned dynamics rarely reach the terminal state. In this work, we leverage a self-consistency property of the conditioned dynamics to learn the diffusion bridge in an iterative online manner, and demonstrate promising empirical results in a range of settings.

1 Introduction

Diffusion bridges Diffusion processes are ubiquitous throughout mathematics, science, and, more recently, machine learning. A diffusion process $(X_t)_{t\in[0,T]}$ in $\mathbb{R}^d$ is governed by a stochastic differential equation

$$\mathrm{d}X_t = b(t, X_t)\,\mathrm{d}t + \sigma(t, X_t)\,\mathrm{d}B_t, \qquad X_0 = x_0, \tag{1}$$

for drift $b : [0,T] \times \mathbb{R}^d \to \mathbb{R}^d$ and diffusion coefficient $\sigma : [0,T] \times \mathbb{R}^d \to \mathbb{R}^{d\times d}$. In this work we will consider elliptic diffusions, so $\sigma\sigma^\top$ is of full-rank, and make the standard assumptions that b, σ are Lipschitz continuous and of linear growth.

Given the SDE dynamics (1), one can use time-discretisation schemes such as the Euler-Maruyama method to simulate samples of the diffusion process. However, often in applications it is required to simulate the *conditioned* dynamics of this underlying reference process, conditional on terminating at a point $X_T = x_T$. The sampling of such *diffusion bridges* has applications in chemistry (Bolhuis et al., 2002; E and Vanden-Eijnden, 2010), finance (Elerian et al., 2001; Durham and Gallant, 2002), and shape evolution (Arnaudon et al., 2022) to name a few. This is typically a challenging problem, particularly when the conditioned event in question occurs rarely under the base dynamics (1).

Existing approaches Traditional approaches for sampling diffusion bridges have commonly used MCMC or SMC-based methods (Beskos et al., 2008; Fearnhead et al., 2008). More recently, neural approximations have been used to learn the control drift of the conditioned dynamics, leveraging tools from score-matching (Heng et al., 2025; Baker et al., 2025) or Malliavin calculus (Pidstrigach et al., 2025). However, these methods learn the neural approximation using samples from uncontrolled dynamics, limiting their success for rare conditioning events as the training samples may not adequately cover the support of the diffusion bridge solution. Motivated by this, Yang et al. (2025) learn a neurally-guided bridge by minimising the KL-divergence to the true conditioned dynamics, though they require backpropagation through the simulated trajectories which can impact scalability.

Contributions In this work, we derive a *self-consistency property* of the diffusion bridge solution in terms of the control drift $u(t, X_t)$ and the Jacobian process $J_{t|s}$ along the solution trajectories. We consider a class of neural-controlled diffusion processes that ensures transportation to the conditioned endpoint x_T, and train according to a proposed family of *self-consistency losses* based on the above property of the solution. These fixed-point-style losses are trained in an online manner, and provide scalable training without the need to backpropagate through the simulated trajectories. We demonstrate the validity of our proposed approach in a range of settings, showing competitive performance with existing methods at a lower computational cost.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Frontiers in Probabilistic Inference: Sampling Meets Learning.

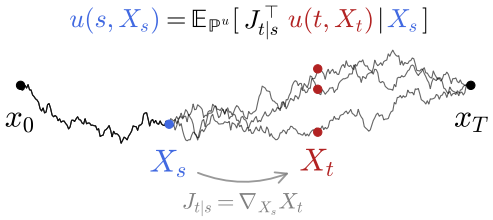


Figure 1: The self-consistency property $u(s, X_s) = \mathbb{E}_{\mathbb{P}^u}[u(t, X_t)J_{t|s}|X_s]$ relates the control value at an earlier time s to its value at a later time t, along the trajectories.

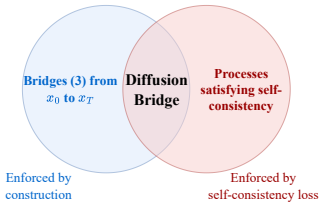


Figure 2: The diffusion bridge is the unique process of form (3) bridging from x_0 to x_T with u of gradient form, that satisfies the self-consistency property (6).

2 Background

Controlled diffusions Let us denote by $\mathbb{P}$ the path measure induced by the unconditioned reference process X_t in (1), and consider a 'target' path measure $\mathbb{Q}$ induced by a reweighting density function $F(X_T)$ defined on the terminal state,

$$\frac{\mathrm{d}\mathbb{Q}}{\mathrm{d}\mathbb{P}}(X) \propto F(X_T). \tag{2}$$

It is a well-known result from stochastic analysis (Eberle, 2019, Section 1.3) that samples from $\mathbb{Q}$ can be obtained by simulating the *controlled* diffusion process X_t^u, whose dynamics are governed by the SDE

$$\mathrm{d}X_t^u = \big[b(t, X_t^u) + (\sigma\sigma^\top)(t, X_t^u)u(t, X_t^u)\big]\mathrm{d}t + \sigma(t, X_t^u)\,\mathrm{d}B_t, \qquad X_0^u = x_0, \tag{3}$$

where the control drift is given by Doob's h-transform, $u(t, x) = \nabla_x \log \mathbb{E}_{\mathbb{P}}[F(X_T)|X_t = x]$. Observe that diffusion bridges are recovered from this framework by using the singular reweighting function $F(X_T) = \delta_{x_T}(X_T)$, which intuitively corresponds to 'retaining only trajectories that terminate at x_T'. In this case, Doob's h-transform takes the more familiar form $u(t, x) = \nabla_x \log p(T, x_T|t, x)$, where $p(t, y|s, x)$ denotes the transition densities of the base process X_t.

Self-consistency of the control In order to explain our approach, we now introduce the *Jacobian*, or *derivative*, process $J_{t|s} = \nabla_{X_s} X_t$ (Rogers and Williams, 2000, Chapter V.13), which evolves for $t > s$ according to the SDE

$$\mathrm{d}J_{t|s} = \nabla b(t, X_t)J_{t|s}\,\mathrm{d}t + \nabla\sigma(t, X_t)J_{t|s}\,\mathrm{d}B_t, \qquad J_{s|s} = \mathrm{Id}. \tag{4}$$

Intuitively, this measures the sensitivity of the value of X_t to perturbations in the earlier value of X_s, along trajectories of the uncontrolled process X. Importantly, the Jacobian process satisfies the semigroup property: for $s < r < t$ we have $J_{t|s} = J_{t|r}J_{r|s}$. Using this object, we can now state the following key result, which is proved in Appendix D.

Theorem 2.1. *The control drift u^* of the solution satisfies*

$$u^*(t, x) = \mathbb{E}_{\mathbb{P}^{u^*}}[J_{T|t}^\top \nabla \log F(X_T)|X_t = x]. \tag{5}$$

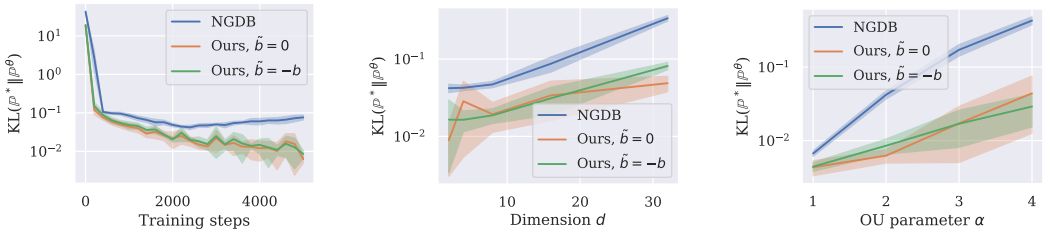
Moreover, the control drift satisfies the following self-consistency property,

$$u^*(s, x) = \mathbb{E}_{\mathbb{P}^{u^*}}[J_{t|s}^\top \, u^*(t, X_t)|X_s = x], \qquad \text{for } s < t. \tag{6}$$

This property characterises how the control at an earlier time s should relate to its value at later times t along the solution trajectories (see Figure 1), and it will form the basis for our approach.

3 Control consistency losses for diffusion bridges

Method The result above shows that self-consistency is a *necessary* property of the optimal control u^*, but it falls short of sufficiency; indeed, it holds for any choice of F. For sufficiency, we must enforce an additional *terminal condition* that incorporates information of the solution's behaviour at the terminal time. The following theorem shows that the diffusion bridge control can be characterised as the unique gradient-form control that forces the dynamics to end at x_T while satisfying the self-consistency property.



(a) KL to solution during training. (b) Effect of dimension, for $\alpha = 2$ (c) Effect of OU parameter, $d = 2$.

Figure 3: Comparison of our proposed approach with Neural Guided Diffusion Bridge (NGDB) (Yang et al., 2025) in the Ornstein-Uhlenbeck bridge experiment (mean± std, over 5 runs).

Theorem 3.1. *Within the class of controlled diffusion processes of the form (3) that terminate at x_T, there is a unique process X_t^u that satisfies the self-consistency property $u(s, X_s) = \mathbb{E}_{\mathbb{P}^u}[J_{t|s}^\top u(t, X_t)|X_s]$ and whose control is of gradient form for some time t, and this process is the diffusion bridge $X_t^u = X_t^*$.*

We provide the proof of the theorem in Appendix D. This result provides the theoretical justification for our proposed approach. Namely, we propose to (see Figure 2)

- enforce the *bridging* and *gradient-form* properties by construction.
- optimise for the *self-consistency property* using a fixed-point style self-consistency loss.

Below, we outline how these two aspects can be implemented to give a practical algorithm.

Bridging property and drift parameterisation Since the termination point is known, in the final discretisation step we can jump directly to x_T. This directly enforces the bridging property, and the control function $u(T - \delta t, \cdot)$ at the penultimate timestep is completely determined by construction.

There is considerable freedom in the drift parameterisation and the exact form can be chosen by the user. A simple and natural choice is to parameterise the drift as

$$f_\theta(t, X_t) = \frac{x_T - X_t}{T - t} + \sigma(t, X_t)\, \eta_\theta(t, X_t). \tag{7}$$

where η_θ is a neurally-parameterised vector field. The first term in the parameterisation is the drift of a Brownian bridge, which guides the controlled process towards the terminal point x_T, and the second is an 'adjustment' term that we learn. In many applications, the reference drift b in (1) is of gradient form and the diffusion term σ is independent of x. As the Brownian drift in (7) is of gradient form, we can directly enforce the same property for u, for all t, by taking $\eta = \nabla\psi_\theta$ for a scalar-valued neural network ψ. See Appendix B for more details regarding other cases.

Self-consistency losses To encourage satisfying the consistency property from Theorem 2.1, $u(s, X_s) = \mathbb{E}_{\mathbb{P}^u}[u(t, X_t) J_{t|s}|X_s]$, we wish to minimise the expression

$$\mathbb{E}_{\mathbb{P}^u}\left[\|u(s, X_s) - \mathbb{E}_{\mathbb{P}^u}[J_{t|s}^\top u(t, X_t)|X_s]\|^2\right] = \mathbb{E}_{\mathbb{P}^u}\left[\|u(s, X_s) - J_{t|s}^\top u(t, X_t)\|^2\right] + C. \tag{8}$$

Note that there are two dependencies on u in this expression, namely through the control values themselves, but also implicitly through the simulated trajectories of X_t^u. We thus propose to optimise a fixed-point style loss, akin to similar fixed-point losses used in Domingo-Enrich et al. (2025). Namely, we regress the control values towards the current estimates of the targets, obtained using samples of the current controlled process $X_t^\theta = X_t^{u_\theta}$, resulting in a 'self-play' style algorithm. Importantly, we do not need to propagate the gradients through the simulated trajectories, which provides significant scalability benefits. Note too that this relation holds for any $s < t$, so one can integrate over the values of t to obtain the following family of losses,

$$\mathcal{L}(\theta) = \mathbb{E}_{\substack{X \sim \mathbb{P}^{u_{\bar{\theta}}} \\ s \sim \mathcal{U}[0,T]}}\left[\left\| u_\theta(s, X_s) - \tfrac{1}{A_s} \int_s^T \alpha_t J_{t|s}^\top u_{\bar{\theta}}(t, X_t)\, \mathrm{d}t \right\|^2\right]. \tag{9}$$

Here, $\bar{\theta}$ denotes the stopgrad operator, α_t can be any weighting function, and $A_s = \int_s^T \alpha_t \, \mathrm{d}t$ is a normalisation factor. We provide a full description of the resulting procedure in Appendix B, along with a generalised version of the algorithm that can improve performance in some settings.

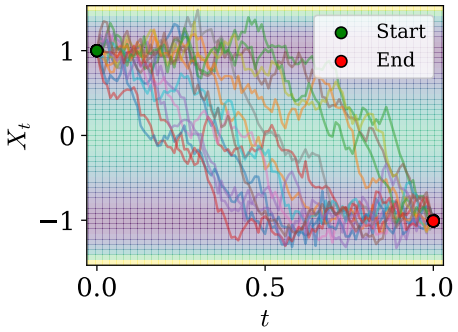


Figure 4: Visualisation of obtained bridge trajectories for the 1d double-well potential.

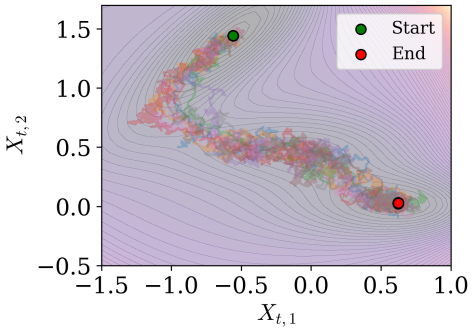


Figure 5: Visualisation of obtained bridge trajectories for the $2d$ Müller-Brown potential.

4 Experiments

We now provide experiments to validate our approach, in a range of different settings. We present a selection below, with additional experiments and comparisons to other diffusion bridge methods in Appendix C. We first provide a comparison with the neural-guided bridge approach (NGDB) of Yang et al. (2025) for sampling Ornstein-Uhlenbeck bridges, a standard benchmark in the literature. This is the most directly comparable method for simulating diffusion bridges, as it also learns a neural control term in an online manner.

Ornstein-Uhlenbeck bridges To provide a quantitative assessment of performance, we consider finding bridges of an Ornstein-Uhlenbeck process $\mathrm{d}X_t = -\alpha X_t \,\mathrm{d}t + \sigma \mathrm{d}B_t$, as in this case the true bridge can be calculated in closed form (see Appendix C.1). In Figure 3a, we report how the KL divergence relative to the ground truth evolves throughout training, and see that our method displays improved convergence to the solution in this example. Moreover, running 5000 steps took approximately 230s with our method, compared to 800s for NGDB; this speed-up can be attributed to the fact that our method does not require backpropagation through the simulated trajectories. In Figures 3b and 3c we also include plots to illustrate the dependence on dimension and the OU parameter α. Full experimental details are provided in Appendix C.1. In Appendix C, we also provide further results for other experiments used in the diffusion bridge literature.

Double-well potential A prominent application of diffusion bridge algorithms is for sampling transition paths between metastable states in molecular dynamics (Bolhuis et al., 2002). As a simple synthetic example, we consider dynamics evolving as $\mathrm{d}X_t = -\nabla U(X_t)\,\mathrm{d}t + \sigma \mathrm{d}B_t$, and first use a $1d$ double-well potential $U(x) = v(x^2 - 1)^2$ for $v = 3.0$ and $\sigma = 1.0$. We visualise obtained trajectories in Figure 4. Experimental details and quantitative results are included in Appendix C.3.

Müller-Brown potential The results in Appendix C and above demonstrate that our method shows strong performance on the standard experiments from the diffusion bridge literature, comparable with the strongest existing method NGDB at a significantly reduced computational cost. To conclude, we consider a harder example inspired by applications in computational chemistry. In Figure 5, we plot a selection of obtained paths for the Müller-Brown potential, a commonly-used benchmark for transition path sampling. The steep slopes of the potential make this a challenging setting, so we use the generalised form of our algorithm presented in Appendix B.5, taking $\tilde{b} = -b$ to avoid large Jacobian terms. We use the experimental setting of Du et al. (2024); a full description of the problem and implementation details are given in Appendix C.5, along with a comparison with the methods of Du et al. (2024) and Yang et al. (2025).

5 Discussion and future directions

In this work, we have presented a novel algorithm for sampling diffusion bridges based on a self-consistency property of the optimal control u^*. The experiments presented here validate our proposed method, and demonstrate strong performance in a range of settings. Limitations of our approach include the ellipticity requirement, and the need for simulating the current process during training. Future work could include investigating extensions to hypoelliptic diffusions, and targeting specific applications in computational chemistry for which we report promising initial findings.

Acknowledgements

SH is supported by the EPSRC CDT in Modern Statistics and Statistical Machine Learning [grant number EP/S023151/1]. JP was supported by the Engineering and Physical Sciences Research Council [grant number EP/Y018273/1].

References

Alexis Arnaudon, Frank van der Meulen, Moritz Schauer, and Stefan Sommer (2022). "Diffusion Bridges for Stochastic Hamiltonian Systems and Shape Evolutions". In: *SIAM Journal on Imaging Sciences* 15.1, pp. 293–323.

Elizabeth Louise Baker, Moritz Schauer, and Stefan Sommer (2025). "Score matching for bridges without time-reversals". In: *The 28th International Conference on Artificial Intelligence and Statistics*.

Alexandros Beskos, Gareth O. Roberts, A. M. Stuart, and Jochen Voss (2008). "MCMC methods for diffusion bridges". In: *Stochastics and Dynamics* Vol.8.No.3, pp. 319–350.

Joris Bierkens, Sebastiano Grazzi, Frank van der Meulen, and Moritz Schauer (Apr. 2021). "A piecewise deterministic Monte Carlo method for diffusion bridges". In: *Statistics and Computing* 31.3.

Mogens Bladt, Samuel Finch, and Michael Sørensen (2015). "Simulation of Multivariate Diffusion Bridges". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 78.2, pp. 343–369.

Mogens Bladt and Michael Sørensen (2014). "Simple simulation of diffusion bridges with application to likelihood inference for diffusions". In: *Bernoulli* 20.2, pp. 645 –675.

Peter G. Bolhuis, David Chandler, Christoph Dellago, and Phillip L. Geissler (2002). "TRANSITION PATH SAMPLING: Throwing Ropes Over Rough Mountain Passes, in the Dark". In: *Annual Review of Physical Chemistry* 53, pp. 291–318.

J.M.C. Clark (1990). "The simulation of pinned diffusions". In: *29th IEEE Conference on Decision and Control*, 1418–1420 vol.3.

John C. Cox, Jonathan E. Ingersoll, Jr., and Stephen A. Ross (1985). "A Theory of the Term Structure of Interest Rates". In: *Econometrica* 53.2, pp. 385–408.

Valentin De Bortoli, James Thornton, Jeremy Heng, and Arnaud Doucet (2021). "Diffusion Schrödinger Bridge with Applications to Score-Based Generative Modeling". In: *Advances in Neural Information Processing Systems*.

Bernard Delyon and Ying Hu (2006). "Simulation of conditioned diffusion and application to parameter estimation". In: *Stochastic Processes and their Applications* 116.11, pp. 1660–1675.

Carles Domingo-Enrich (2024). *A Taxonomy of Loss Functions for Stochastic Optimal Control*. arXiv: 2410.00345 [cs.LG].

Carles Domingo-Enrich, Michal Drozdzal, Brian Karrer, and Ricky T. Q. Chen (2025). "Adjoint Matching: Fine-tuning Flow and Diffusion Generative Models with Memoryless Stochastic Optimal Control". In: *International Conference on Learning Representations*.

Carles Domingo-Enrich, Jiequn Han, Brandon Amos, Joan Bruna, and Ricky T. Q. Chen (2024). "Stochastic Optimal Control Matching". In: *Advances in Neural Information Processing Systems*.

Yuanqi Du, Michael Plainer, Rob Brekelmans, Chenru Duan, Frank Noe, Carla P Gomes, Alan Aspuru-Guzik, and Kirill Neklyudov (2024). "Doob's Lagrangian: A Sample-Efficient Variational Approach to Transition Path Sampling". In: *Conference on Neural Information Processing Systems*.

Garland B Durham and A. Ronald Gallant (2002). "Numerical Techniques for Maximum Likelihood Estimation of Continuous-Time Diffusion Processes". In: *Journal of Business & Economic Statistics* 20.3, pp. 297–338.

Weinan E and Eric Vanden-Eijnden (2010). "Transition-Path Theory and Path-Finding Algorithms for the Study of Rare Events". In: *Annual Review of Physical Chemistry* 61, pp. 391–420.

Andreas Eberle (2019). *Stochastic Analysis*.

Ola Elerian, Siddhartha Chib, and Neil Shephard (2001). "Likelihood Inference for Discretely Observed Nonlinear Diffusions". In: *Econometrica* 69.4, pp. 959–993.

K. D. Elworthy, Y. Le Jan, and X-M. Li (1999). *On the geometry of diffusion operators and stochastic flows*. eng. Lecture notes in mathematics, 1720. Berlin ; Springer.

Paul Fearnhead, Omiros Papaspiliopoulos, and Gareth O Roberts (2008). "Particle filters for partially observed diffusions". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 70.4, pp. 755–777.

A. Golightly and D.J. Wilkinson (2008). "Bayesian inference for nonlinear multivariate diffusion models observed with error". In: *Computational Statistics & Data Analysis* 52.3, pp. 1674–1693.

Jeremy Heng, Valentin De Bortoli, Arnaud Doucet, and James Thornton (2025). "Simulating diffusion bridges with score matching". In: *Biometrika*.

Diederick P Kingma and Jimmy Ba (2015). "Adam: A method for stochastic optimization". In: *International Conference on Learning Representations*.

H. Kunita (1984). "Stochastic differential equations and stochastic flows of diffeomorphisms". In: *École d'Été de Probabilités de Saint-Flour XII - 1982*. Ed. by P. L. Hennequin. Springer Berlin Heidelberg, pp. 143–303.

Hiroshi Kunita (1986). *Lectures on Stochastic Flows and Applications*.

Ming Lin, Rong Chen, and Per Mykland (2010). "On Generating Monte Carlo Samples of Continuous Diffusion Bridges". In: *Journal of the American Statistical Association* 105.490, pp. 820–838.

Christian Léonard (2013). *A survey of the Schrödinger problem and some of its connections with optimal transport*. arXiv: `1308.0215 [math.PR]`.

Frank van der Meulen and Moritz Schauer (2017). "Bayesian estimation of discretely observed multi-dimensional diffusion processes using guided proposals". In: *Electronic Journal of Statistics* 11.1, pp. 2358 –2396.

Marcin Mider, Moritz Schauer, and Frank van der Meulen (2021). "Continuous-discrete smoothing of diffusions". In: *Electronic Journal of Statistics* 15.2, pp. 4295 –4342.

Bernt Øksendal (2003). *Stochastic Differential Equations: An Introduction with Applications*. 6th ed. Springer.

Omiros Papaspiliopoulos and Gareth O. Roberts (2012). "Importance sampling techniques for estimation of diffusion models". In: *Statistical methods for stochastic differential equations*. Vol. 124. CRC Monographs on Statistics & Applied Probability, pp. 311–340.

Amnon Pazy (2012). *Semigroups of linear operators and applications to partial differential equations*. Vol. 44. Springer Science & Business Media.

Asger Roer Pedersen (1995). "Consistency and asymptotic normality of an approximate maximum likelihood estimator for discretely observed diffusion processes". In: *Bernoulli* 1.3, pp. 257 –279.

Jakiw Pidstrigach, Elizabeth Louise Baker, Carles Domingo-Enrich, George Deligiannidis, and Nikolas Nüsken (2025). "Conditioning Diffusions Using Malliavin Calculus". In: *International Conference on Machine Learning*.

G. O. Roberts and O. Stramer (2001). "On inference for partially observed nonlinear diffusion models using the Metropolis–Hastings algorithm". In: *Biometrika* 88.3, pp. 603–621.

Geoffrey Roeder, Yuhuai Wu, and David Duvenaud (2017). "Sticking the landing: simple, lower-variance gradient estimators for variational inference". In: *Conference on Neural Information Processing Systems*.

Sylvie Rœlly (2013). "Reciprocal processes: a stochastic analysis approach". In: *Modern stochastics and applications*. Springer, pp. 53–67.

L. C. G. Rogers and David Williams (2000). *Diffusions, Markov Processes, and Martingales*. 2nd ed. Cambridge Mathematical Library. Cambridge University Press.

Moritz Schauer, Frank van der Meulen, and Harry van Zanten (2017). "Guided proposals for simulating multi-dimensional diffusion bridges". In: *Bernoulli* 23.

Osnat Stramer and Jun Yan (2007). "On Simulated Likelihood of Discretely Observed Diffusion Processes and Comparison to Closed-Form Approximation". In: *Journal of Computational and Graphical Statistics* 16.3, pp. 672–691.

Andrew M. Stuart, Jochen Voss, and Petter Wilberg (2004). "Conditional Path Sampling of SDEs and the Langevin MCMC Method". In: *Communications in Mathematical Sciences* 2.4, pp. 685 –697.

Francisco Vargas, Shreyas Padhy, Denis Blessing, and Nikolas Nüsken (2024). "Transport meets Variational Inference: Controlled Monte Carlo Diffusions". In: *International Conference on Learning Representations*.

Jin Wang, Kun Zhang, Li Xu, and Erkang Wang (2011). "Quantifying the Waddington landscape and biological paths for development and differentiation". In: *Proceedings of the National Academy of Sciences* 108.20, pp. 8257–8262.

Gefan Yang, Frank van der Meulen, and Stefan Sommer (2025). "Neural Guided Diffusion Bridges". In: *International Conference on Machine Learning*.

Appendix

The Appendix is structured as follows. In Appendix A, we discuss related work. In Appendix B, we provide more details regarding the method and present a generalised version of our algorithm. In Appendix C, we provide experimental details and additional results. In Appendix D we provide the proofs of the results in the paper. Finally, in Appendix E we provide the licenses and links to the assets used in the work.

A Related work

Diffusion bridge simulation has been the subject of extensive study over recent decades; we highlight a selection of works from the literature here. A common approach is to use simulations from tractable dynamics, and correct using importance-sampling (Papaspiliopoulos and Roberts, 2012) or Metropolis-Hastings steps. Pedersen (1995) use the unconditional dynamics, and Clark (1990) and Delyon and Hu (2006) utilise the Brownian bridge to ensure termination at the correct state. Schauer et al. (2017) consider a class of guided diffusions that includes an additional drift term while preserving tractability. We also highlight the works Stuart et al. (2004) and Beskos et al. (2008) who consider Langevin-type algorithms on the space of paths, Bierkens et al. (2021) who utilise a piecewise deterministic Monte-Carlo approach, and Lin et al. (2010) who use an SMC-based approach. Often, diffusion bridges approaches are used for inferring the parameters of a discretely-observed diffusion process (Elerian et al., 2001; Roberts and Stramer, 2001; Durham and Gallant, 2002; Stramer and Yan, 2007; Golightly and Wilkinson, 2008; Bladt and Sørensen, 2014; Bladt et al., 2015; Meulen and Schauer, 2017).

More recently, neural approximations have been used to learn the control term $u(t, x)$. Heng et al. (2025) utilised the expression of the control drift in the time-reversed bridge dynamics as a score function, and trained the drift using score-matching on simulated trajectories of the unconditioned forwards process. The conditioned forward dynamics can subsequently be learned via a time-reversal. This approach was extended in Baker et al. (2025) to learn the forward dynamics directly, using trajectories of an adjoint process simulated backwards in time. Pidstrigach et al. (2025) uses a novel characterisation of the control based on Malliavin calculus, which is learned from simulations of the unconditioned process. Such approaches enable direct simulation of the conditioned dynamics using the learned drifts. However, as they rely on simulations of the unconditioned dynamics, they are limited in their ability to learn accurate dynamics for *rare* events, as these will hardly ever arise in the trajectories used for training. Recent work Yang et al. (2025) aims to address this by using a neural guiding drift in addition to the linear drift from Schauer et al. (2017), and optimising a KL-based objective to encourage movement towards the true conditioned dynamics. While exhibiting good performance, the need for backpropagation through simulated trajectories can pose challenges for scalability. We also highlight the recent work Du et al. (2024), which introduces a variational formulation to learn a Gaussian mixture path parameterisation. Of these, the most related to our approach is the recent work of Pidstrigach et al. (2025), as the self-consistency property that we use can also be derived from their Malliavin calculus framework.

Finally, we highlight that the self-consistency framework that we use in this work has many connections to recent advances in the stochastic optimal control literature (Domingo-Enrich et al., 2024, 2025; Domingo-Enrich, 2024), though their focus is on optimal control for rewards with gradient information rather than on diffusion bridges. We provide a discussion of these connections in Appendix B.5.

B Method details and discussion

A complete description of the method outlined in Section 3 is provided in Algorithm 1. We now discuss further implementation details regarding the proposed approach. We also discuss potential issues that might arise in certain settings, and present a generalised version of our algorithm which can help to mitigate them.

B.1 Choice of neural parameterisation

Recall from the discussion around equation (7) that we use the neural parameterisation to enforce the gradient property. Aside from this consideration, there is considerable freedom in the choice of neural parameterisation, though it is clearly beneficial to direct the samples towards the endpoint through this choice. As discussed in Section 3, in cases where the base drift b is of gradient form, we

Algorithm 1: Control Consistency Diffusion Bridges

Input: Drift parameterisation $f_\theta = b + (\sigma\sigma^\top)u_\theta$, number of iterations N, batch size B, weighting schedule α_t.
for $n = 0$ **to** $N - 1$ **do**
Simulate B paths of $(X_t^\theta)_{t\in[0,T]}$ using the current control,

$$\mathrm{d}X_t^\theta = \big(b + (\sigma\sigma^\top)u_\theta\big)(t, X_t^\theta)\mathrm{d}t + \sigma(t, X_t^\theta)\mathrm{d}B_t, \tag{10}$$

Solve for training targets $\mathcal{S}_s$ backwards along the obtained paths, using (12).
Perform a gradient step on θ using the self-consistency loss function (9).
end for

use the parameterisation

$$f_\theta(t, X_t) = \frac{x_T - X_t}{T - t} + \sigma(t, X_t)\,\nabla\psi_\theta(t, X_t), \tag{11}$$

for a scalar-valued network ψ_θ. For non-conservative base drifts b, we include the drift in the base process and use the parameterisation

$$f_\theta(t, X_t) = b(t, X_t) + \frac{x_T - X_t}{T - t} + \sigma(t, X_t)\,\nabla\psi_\theta(t, X_t).$$

Certain choices of spatially-dependent diffusion coefficients σ may require further modifications, and the appropriate adjustments can again be made to ensure that u remains of gradient form in these cases too. There are also further possibilities for the parameterisation, such as incorporating the linearised guiding drift used in Mider et al. (2021) and Yang et al. (2025).

We remark here that enforcing the gradient property directly is not strictly necessary for a theoretically-grounded algorithm. Recalling the result in Theorem 3.1, we in fact only need to enforce the gradient property at a single point in time. Note that in cases with base drifts of gradient form, this is ensured by our construction where we directly jump to the terminal state in the final step; this enforces that the control $u(T - \delta t, \cdot)$ is conservative. Therefore, one can use a general neural parameterisation rather than enforcing gradient form directly. However, in practice, rotational terms may accrue if the optimisation is not performed perfectly; we found improved performance by enforcing the gradient property throughout time as described above.

B.2 Solving for the training targets

Recall from Section 3 that we require to compute the training targets $\mathcal{S}_s := \frac{1}{A_s}\int_s^T \alpha_t J_{t|s}^\top\, u(t, X_t)\,\mathrm{d}t$, which can be done by solving backwards along the simulated trajectories. In particular, we can avoid computing the full Jacobian matrices $J_{t|s}^\top$ by using vector-Jacobian products. To performs the computations, we can write the target $\mathcal{S}_s$ in terms of the next-step target $\mathcal{S}_{s+\delta t}$ according to

$$\begin{aligned}
\mathcal{S}_s &= \tfrac{1}{A_s}\int_s^T \alpha_t J_{t|s}^\top\, u(t, X_t)\,\mathrm{d}t \\
&= \tfrac{1}{A_s}\int_{s+\delta t}^T \alpha_t J_{t|s}^\top\, u(t, X_t)\mathrm{d}t + \tfrac{1}{A_s}\int_s^{s+\delta t} \alpha_t J_{t|s}^\top\, u(t, X_t)\,\mathrm{d}t \\
&= \tfrac{1}{A_s}\int_{s+\delta t}^T \alpha_t (J_{t|s+\delta t} J_{s+\delta t|s})^\top\, u(t, X_t)\mathrm{d}t + \tfrac{1}{A_s}\int_s^{s+\delta t} \alpha_t J_{t|s}^\top\, u(t, X_t)\,\mathrm{d}t \\
&= \tfrac{A_{s+\delta t}}{A_s} J_{s+\delta t|s}^\top\, \mathcal{S}_{s+\delta t} + \tfrac{1}{A_s}\int_s^{s+\delta t} \alpha_t J_{t|s}^\top\, u(t, X_t)\,\mathrm{d}t \\
&\approx J_{s+\delta t|s}^\top \left[\tfrac{A_{s+\delta t}}{A_s}\mathcal{S}_{s+\delta t} + (\delta t)\tfrac{\alpha_{s+\delta t}}{A_s} u(s + \delta t, X_{s+\delta t})\right],
\end{aligned} \tag{12}$$

where in the final line we approximate the second integral using value of the integrand at the larger timestep. By relabelling $\lambda_s := \frac{A_{s+\delta t}}{A_s}$, this can be written as

$$J_{s+\delta t|s}^\top \left[\lambda_s \mathcal{S}_{s+\delta t} + (1 - \lambda_s) u_{s+\delta t}(X_{s+\delta t})\right]. \tag{13}$$

This is intuitive; each target $\mathcal{S}_s$ is constructed as a weighted combination of the 'running' next-step target $\mathcal{S}_{s+\delta t}$ and the next-step control $u(s+\delta t, X_{s+\delta t})$, and is then transformed by the single-step Jacobian $J_{s+\delta t|s} \approx \mathrm{Id} + \nabla b(s, X_s)(\delta t) + \nabla \sigma(s, X_s) \cdot \delta B_s$. The weighting is determined by the choice of α-schedule, which therefore determines how 'far into the future' the algorithm looks to enforce the self-consistency property. Following Pidstrigach et al. (2025), we consider three different choices of α-schedule: *next-step*, *average*, and *endpoint* prediction. For clarity, we explicitly write out the recursions for these three α-schedules below.

- *Next-step prediction:* This choice of schedule chooses to only enforce the self-consistency property at consecutive timesteps; that is, it tries to enforce $u(s, X_s) = \mathbb{E}_{\mathbb{P}^u}[J^\top_{s+\delta t|s}\, u(s+\delta t, X_{s+\delta t})|X_s]$. This corresponds to taking $\lambda_s = 0$ in the above expression (13), so the targets are calculated as
$$\mathcal{S}_s = J^\top_{s+\delta t|s}\, u_{s+\delta t}(X_{s+\delta t}). \tag{14}$$
This choice results in small target variance — $u(s, X_s)$ is likely close to $\mathcal{S}_s$, as there is only a single step for variance to accrue. As we enforce the correct control at the final step, this choice can be thought of as 'passing the information along the trajectory one-by-one'. As a result, this can cause slower learning, but can give more stable training in settings where the training target variance might become large due to large Jacobians (for example, see the double-well experiments in Appendix C.6).

- *Average prediction:* This choice instead aims to enforce the self-consistency uniformly across the remaining time $t \in [s, T]$, so $\mathcal{S}_s = \frac{1}{T-s}\int_s^T J^\top_{t|s}\, u(t, X_t)\,\mathrm{d}t$. In this case, the recursive formula becomes
$$\mathcal{S}_s = J^\top_{s+\delta t|s}\left[\tfrac{T-(s+\delta t)}{T-s}\mathcal{S}_{s+\delta t} + \tfrac{(\delta t)}{T-s}u(s+\delta t, X_{s+\delta t})\right]. \tag{15}$$
We found this choice to provide a good balance between training target variance and propagation of the terminal information, and it worked consistently well across the experimental settings (see results in Appendix C.6). For this reason, the results reported in the main body use the average α-schedule.

- *Endpoint prediction:* This choice chooses to only enforce the self-consistency property to the terminal step; that is, it tries to enforce $u(s, X_s) = \mathbb{E}_{\mathbb{P}^u}[J^\top_{T-\delta t|s}\, u(T-\delta t, X_{T-\delta t})|X_s]$. This corresponds to taking $\lambda_s = 1$ in (13), so the targets are calculated as
$$\mathcal{S}_s = J^\top_{s+\delta t|s}\, \mathcal{S}_{s+\delta t}. \tag{16}$$
While this choice directly fits to the known terminal information, variance can accrue in the many intermediate steps. Moreover, target variance is exacerbated in the case of diffusion bridges, as the control grows sharply towards the terminal time. In the experiments in Appendix C.6, we consistently found this choice to perform worse than average prediction.

We remark that other choices of α-schedule are possible and can be chosen by the practitioner. The method for computing the training targets for a single trajectory in the generalised case is presented in Algorithm 3.

B.3 Sticking-the-landing adjustments

Here, we explain how 'Sticking-the-Landing' (STL) adjustments (Roeder et al., 2017) can be included in the training target recursions, similar to the adjustments discussed in Domingo-Enrich (2024). The idea of the STL adjustments is to remove the variance in the training targets at the solution, by leveraging the knowledge that the corresponding potential must solve the appropriate HJB equation. Below, we provide a sketch proof showing how the STL adjustments arise in our framework, and show how they can be incorporated into our algorithm. For full details, see the proof in Appendix D. Note that this presentation is for a scalar noise term σ; the general case follows similarly by making the appropriate changes.

We know that the control u^* of the solution is of the form $u^* = \nabla\phi$, where ϕ solves the HJB equation,
$$\partial_t\phi + \mathcal{L}\phi + \tfrac{1}{2}|\sigma\nabla\phi|^2 = 0. \tag{HJB}$$
Taking gradients of the HJB equation and using $u = \nabla\phi$, we see that
$$0 = \partial_t u + \nabla\mathcal{L}\phi + \tfrac{\sigma^2}{2}\nabla|u|^2 = \partial_t u + \nabla(b\cdot u) + \tfrac{\sigma^2}{2}\Delta u + \tfrac{\sigma^2}{2}\nabla|u|^2. \tag{Grad-HJB}$$

By applying the product rule of Itô stochastic calculus to $J_{t|s}^\top u(t, X_t^u)$ (for full details see the proofs in Appendix D), we have

$$\mathrm{d}(J_{t|s}^\top u(t, X_t^u)) = J_{t|s}^\top \left(\partial_t u + \nabla(b \cdot u) + \tfrac{\sigma^2}{2}\Delta u + \tfrac{\sigma^2}{2}\nabla|u|^2\right)(t, X_t^u)\,\mathrm{d}t + \sigma J_{t|s}^\top \nabla u(t, X_t^u)\cdot \mathrm{d}B_t.$$

Observe that the deterministic term contains precisely the right-hand side of (Grad-HJB), so is zero at the solution. Integrating over time, the optimal control u^* therefore satisfies the following *path-wise* self-consistency property, which holds almost surely along trajectories of the solution,

$$u(s, X_s^u) = J_{t|s}^\top u(t, X_t^u) - \sigma \int_s^t J_{r|s}^\top \nabla u(r, X_r^u)\cdot \mathrm{d}B_r. \tag{17}$$

By taking expectations, we recover the standard self-consistency property (6) used in the standard version of our algorithm. However, numerically speaking, working with the expectation introduces variance, as the stochastic term must be averaged out across samples. Rather than working with the standard self-consistency property, 'sticking-the-landing' means instead working with this *path-wise* property directly, which at optimality will be satisfied almost surely.

Computing the STL training targets To incorporate the STL adjustment into our algorithm, we instead fit to the STL training targets

$$\mathcal{S}_s := \frac{1}{A_s}\int_s^T \alpha_t J_{t|s}^\top u(t, X_t^u)\,\mathrm{d}t - \frac{1}{A_s}\int_s^T \alpha_t \left(\sigma \int_s^t J_{r|s}^\top \nabla u(r, X_r^u)\cdot \mathrm{d}B_r\right)\mathrm{d}t. \tag{18}$$

The first term is the same as in the usual case, and the second term constitutes the STL adjustments. It remains to show the recursion that the STL targets satisfy, in order to compute the target backwards along the trajectories. To do so, it is first helpful to rearrange the noise term as follows (using the fact $\int_s^T \mathbf{1}_{r\in[s,t]}\alpha_t\,\mathrm{d}t = \int_r^T \alpha_t\,\mathrm{d}t = A_r$),

$$\begin{aligned}
\frac{1}{A_s}\int_s^T \alpha_t \left(\int_s^t J_{r|s}^\top \nabla u(r, X_r^u)\cdot \mathrm{d}B_r\right)\mathrm{d}t &= \frac{1}{A_s}\int_s^T \int_s^T \mathbf{1}_{r\in[s,t]}\alpha_t J_{r|s}^\top \nabla u(r, X_r^u)\cdot \mathrm{d}B_r\,\mathrm{d}t \\
&= \frac{1}{A_s}\int_s^T \left(\int_s^T \mathbf{1}_{r\in[s,t]}\alpha_t\,\mathrm{d}t\right) J_{r|s}^\top \nabla u(r, X_r^u)\cdot \mathrm{d}B_r \\
&= \frac{1}{A_s}\int_s^T A_r J_{r|s}^\top \nabla u(r, X_r^u)\cdot \mathrm{d}B_r.
\end{aligned}$$

The STL training targets can then be computed according to the following recursion, mirroring the calculation in the standard case.

$$\begin{aligned}
\mathcal{S}_s &= \tfrac{1}{A_s}\int_s^T \alpha_t J_{t|s}^\top u(t, X_t)\mathrm{d}t + \tfrac{\sigma}{A_s}\int_s^T A_r J_{r|s}^\top \nabla u(r, X_r^u)\cdot \mathrm{d}B_r \\
&= \left[\tfrac{1}{A_s}\int_{s+\delta t}^T \alpha_t J_{t|s}^\top u(t, X_t)\mathrm{d}t + \tfrac{\sigma}{A_s}\int_{s+\delta t}^T A_r J_{r|s}^\top \nabla u(r, X_r^u)\cdot \mathrm{d}B_r\right] \\
&\qquad + \left[\tfrac{1}{A_s}\int_s^{s+\delta t} \alpha_t J_{t|s}^\top u(t, X_t)\mathrm{d}t + \tfrac{\sigma}{A_s}\int_s^{s+\delta t} A_r J_{r|s}^\top \nabla u(r, X_r^u)\cdot \mathrm{d}B_r\right] \\
&= J_{s+\delta t|s}^\top \left[\tfrac{1}{A_s}\int_{s+\delta t}^T \alpha_t J_{t|s+\delta t}^\top u(t, X_t)\mathrm{d}t + \tfrac{\sigma}{A_s}\int_{s+\delta t}^T A_r J_{t|s+\delta t}^\top \nabla u(r, X_r^u)\cdot \mathrm{d}B_r\right] \\
&\qquad + \left[\tfrac{1}{A_s}\int_s^{s+\delta t} \alpha_t J_{t|s}^\top u(t, X_t)\mathrm{d}t + \tfrac{\sigma}{A_s}\int_s^{s+\delta t} A_r J_{r|s}^\top \nabla u(r, X_r^u)\cdot \mathrm{d}B_r\right] \\
&\approx \sigma\nabla u(s, X_s)\cdot \delta B_s + J_{s+\delta t|s}^\top \left[\tfrac{A_{s+\delta t}}{A_s}\mathcal{S}_{s+\delta t} + (\delta t)\tfrac{\alpha_{s+\delta t}}{A_s}u(s+\delta t, X_{s+\delta t})\right].
\end{aligned}$$

The only change to the standard case is the inclusion of the additional STL adjustment $\sigma\nabla u(s, X_s)\cdot \delta B_s$ in the recursion; note that this agrees with the calculations in Domingo-Enrich (2024). In the case of a possibly spatially-dependent diffusion coefficient σ, performing similar calculations gives the path-wise self-consistency property

$$u(s, X_s^u) = J_{t|s}^\top u(t, X_t^u) - \int_s^t J_{r|s}^\top (\nabla u\,\sigma + u\,\nabla\sigma)(r, X_r^u)\cdot \mathrm{d}B_r, \tag{19}$$

and the STL adjustment is $(\nabla u\,\sigma + u\,\nabla\sigma)(r, X_r^u)\cdot \delta B_r$.

Performance using STL adjustments We compare performance with and without the STL adjustments in Appendix C.7, and observe that it generally improves performance by a small amount, with the largest improvements seen in cases where large Jacobian terms arise (that is, the double-well and Müller-Brown experiments). The additional terms $(\nabla u\,\sigma + u\,\nabla\sigma)(r, X_r^u) \cdot \delta B_r$ can be computed efficiently using Jacobian-vector products, however this still adds an additional computational cost; generally we found that including the STL adjustments made the training steps approximately 1.5 times slower.

B.4 A generalised self-consistency property

The presentation so far provides a clean and intuitive self-consistency property (6) that can be used to learn diffusion bridges, and in our experiments we find that the resulting algorithm often performs extremely well in practice. Here, we shed light on potential issues that can arise in certain settings, and present a generalisation of the self-consistency property which can help to mitigate them.

Behaviour of the Jacobian process Recall that the sensitivity process $J_{t|s}$ evolves as

$$\mathrm{d}J_{t|s} = \nabla b(t, X_t) J_{t|s}\mathrm{d}t + \nabla\sigma(t, X_t) J_{t|s}\mathrm{d}B_t, \quad J_{s|s} = \mathrm{Id}.$$

Note that the incremental values depend on the current value of $J_{t|s}$. In particular, when ∇b or $\nabla\sigma$ have positive eigenvalues then the value will increase along those directions, and if the eigenvalues are negative then they will decrease. Compounding of any such increases over time can lead to explosion of the value of $J_{t|s}$ and subsequently in the control estimates $J_{t|s}^\top\, u(t, X_t)$, potentially causing numerical issues during training.

To illustrate this effect conceptually, it is instructive to consider overdamped Langevin dynamics $\mathrm{d}X_t = -\nabla U(X_t)\mathrm{d}t + \sigma\mathrm{d}B_t$ for a scalar potential U, which is a frequently occurring scenario in applications. The process traverses the potential randomly, with a tendency to move 'downhill'. In such a setting, negative eigenvalues of ∇b correspond to a convex 'bowl' shape in the potential, while positive eigenvalues occur where the shape is concave, such as passing over a peak or ridge. Recalling the definition $J_{t|s} = \nabla_{X_s} X_t$, this is indeed intuitive—a perturbation of X_s within a convex valley has little effect on the later state X_t, but a perturbation made while crossing a peak can compound to cause large deviations in the subsequent trajectory.

Generalising the self-consistency property To combat such troublesome behaviour in the Jacobian $J_{t|s}$, we can derive a generalisation of the self-consistency property that retains the same theoretical guarantees as the standard case.

Recall from the derivation in (50) that the Jacobian $J_{t|s}$ arises from taking the derivative through the $\mathbb{P}$-expectation. We can instead consider a change of measure before performing this operation. Let $\tilde{\mathbb{P}}$ be an 'auxiliary' measure associated to the process $\mathrm{d}\tilde{X}_t = (b + \tilde{b})(t, \tilde{X}_t)\mathrm{d}t + \sigma(t, \tilde{X}_t)\mathrm{d}B_t$, in which we include an additional drift term $\tilde{b}$. In the derivation, we can instead perform the change of measure

$$\nabla_x \mathbb{E}_{\mathbb{P}}[F(X_T)|X_t = x] = \nabla_x \mathbb{E}_{\tilde{\mathbb{P}}}[F(X_T)\tfrac{\mathrm{d}\mathbb{P}}{\mathrm{d}\tilde{\mathbb{P}}}|X_t = x],$$

where the Radon-Nikodym derivative is given by the Girsanov weights $\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\tilde{\mathbb{P}}} = \exp(-\int_s^T \sigma^{-1}\tilde{b}(r, X_r)^\top \mathrm{d}B_r - \int_s^T \frac{(\sigma\sigma^\top)^{-1}}{2}\|\tilde{b}(r, X_r)\|^2\mathrm{d}r)$ (Øksendal, 2003, Theorem 8.6.8). Continuing the calculation as previously, we obtain a *generalised self-consistency property* which instead includes Jacobian terms $\tilde{J}_{t|s}$ associated to the auxiliary process $\tilde{X}$ rather than the base process X, while also picking up additional 'running cost' terms.

Theorem B.1 (Generalised self-consistency property). *For any differentiable $\tilde{b}$ satisfying the conditions of Girsanov's theorem, the control drift u^* of the solution satisfies*

$$u^*(s, x) = \mathbb{E}_{\mathbb{P}^{u^*}}\Big[-\int_s^t \tilde{J}_{r|s}^\top \nabla\tilde{b}(r, X_r)^\top u^*(r, X_r)\mathrm{d}r + \tilde{J}_{t|s}^\top\, u(t, X_t)|X_s = x\Big]. \tag{20}$$

Importantly, we retain the same theoretical guarantees when using the generalised self-consistency property (20) as we do in the standard case.

Theorem B.2. *The result of Theorem 2.1 holds with the generalised self-consistency property (20) in place of the usual self-consistency property (6).*

Algorithm 2: Control Consistency Diffusion Bridges (Generalised version)

Input: Additional drift $\tilde{b}$ for auxiliary process, drift parameterisation $f_\theta = b + (\sigma\sigma^\top)u_\theta$, number of iterations N, batch size B, weighting schedule α_t.

for $n = 0$ **to** $N - 1$ **do**

Simulate B paths of $(X_t^\theta)_{t\in[0,T]}$ using the current control,

$$\mathrm{d}X_t^\theta = \big(b + (\sigma\sigma^\top)u_\theta\big)(t, X_t^\theta)\mathrm{d}t + \sigma(t, X_t^\theta)\mathrm{d}B_t, \tag{21}$$

Solve for training targets $\mathcal{S}_s$ (23) backwards along the obtained paths (X_t^θ), using Algorithm 3. Perform a gradient step on θ using the squared loss function

$$\mathcal{L}(\theta) = \mathop{\mathbb{E}}_{\substack{X\sim\mathbb{P}^{u_{\tilde{\theta}}} \\ s\sim\mathcal{U}_{[0,T]}}} \Big[\big\|u_\theta(s, X_s) - \mathcal{S}_s\big\|^2\Big]. \tag{22}$$

end for

We are free to choose the additional drift $\tilde{b}$ in the auxiliary process. A natural choice is to use $\tilde{b} = -b$, so that the auxiliary process $\tilde{X}$ has no drift term—in cases where the diffusion coefficient is constant, this will mean $\tilde{J}_{t|s}$ is always the identity. In practice, we find that this gives a significant improvement in settings where the growth of the base Jacobian $J_{t|s}$ would cause numerical issues.

Solving for the training targets Unlike in the standard case, the generalised self-consistency property includes 'running costs'. As before, we can construct the targets by weighting according to a α-schedule. The resulting training targets are therefore of the form

$$\mathcal{S}_s = \tfrac{1}{A_s}\int_s^T \alpha_t\big(-\int_s^t \tilde{J}_{r|s}^\top \nabla\tilde{b}(r, X_r)^\top u(r, X_r)\mathrm{d}r + \tilde{J}_{t|s}^\top u(t, X_t)\big)\mathrm{d}t. \tag{23}$$

To compute these targets backwards along the trajectories, we can modify the reverse recursions as follows,

$$\mathcal{S}_s = \tfrac{1}{A_s}\int_s^T \alpha_t\big(-\int_s^t \tilde{J}_{r|s}^\top \nabla\tilde{b}(r, X_r)^\top u(r, X_r)\mathrm{d}r + \tilde{J}_{t|s}^\top u(t, X_t)\big)\mathrm{d}t \tag{24}$$

$$= \tfrac{1}{A_s}\int_s^{s+\delta t} \alpha_t\big(-\int_s^t \tilde{J}_{r|s}^\top \nabla\tilde{b}(r, X_r)^\top u(r, X_r)\mathrm{d}r + \tilde{J}_{t|s}^\top u(t, X_t)\big)\mathrm{d}t$$
$$+ \tfrac{1}{A_s}\int_{s+\delta t}^T \alpha_t\big(-\int_s^t \tilde{J}_{r|s}^\top \nabla\tilde{b}(r, X_r)^\top u(r, X_r)\mathrm{d}r + \tilde{J}_{t|s}^\top u(t, X_t)\big)\mathrm{d}t \tag{25}$$

$$= \tfrac{1}{A_s}\int_s^{s+\delta t} \alpha_t\big(-\int_s^t \tilde{J}_{r|s}^\top \nabla\tilde{b}(r, X_r)^\top u(r, X_r)\mathrm{d}r + \tilde{J}_{t|s}^\top u(t, X_t)\big)\mathrm{d}t$$
$$+ \tfrac{1}{A_s}\int_{s+\delta t}^T \alpha_t\big(-\int_s^{s+\delta t} \tilde{J}_{r|s}^\top \nabla\tilde{b}(r, X_r)^\top u(r, X_r)\mathrm{d}r\big)\mathrm{d}t \tag{26}$$
$$+ \tfrac{1}{A_s}\int_{s+\delta t}^T \alpha_t\big(-\int_{s+\delta t}^t \tilde{J}_{r|s}^\top \nabla\tilde{b}(r, X_r)^\top u(r, X_r)\mathrm{d}r + \tilde{J}_{t|s}^\top u(t, X_t)\big)\mathrm{d}t \tag{27}$$

$$= \tfrac{1}{A_s}\int_s^{s+\delta t} \alpha_t\big(-\int_s^t \tilde{J}_{r|s}^\top \nabla\tilde{b}(r, X_r)^\top u(r, X_r)\mathrm{d}r + \tilde{J}_{t|s}^\top u(t, X_t)\big)\mathrm{d}t$$
$$+ \tfrac{1}{A_s}\int_{s+\delta t}^T \alpha_t\big(-\int_s^{s+\delta t} \tilde{J}_{r|s}^\top \nabla\tilde{b}(r, X_r)^\top u(r, X_r)\mathrm{d}r\big)\mathrm{d}t \tag{28}$$
$$+ \tfrac{1}{A_s}\tilde{J}_{s+\delta t|s}^\top \int_{s+\delta t}^T \alpha_t\big(-\int_{s+\delta t}^t \tilde{J}_{r|s+\delta t}^\top \nabla\tilde{b}(r, X_r)^\top u(r, X_r)\mathrm{d}r + \tilde{J}_{t|s+\delta t}^\top u(t, X_t)\big)\mathrm{d}t \tag{29}$$

$$\approx -(\delta t)\nabla\tilde{b}(s, X_s)^\top u(s, X_s) + J_{s+\delta t|s}^\top \left[\tfrac{(\delta t)\alpha_{s+\delta t}}{A_s} u(s+\delta t, X_{s+\delta t}) + \tfrac{A_{s+\delta t}}{A_s}\mathcal{S}_{s+\delta t}\right] \tag{30}$$

Algorithm 3: Computing training targets along a single trajectory (generalised version)

Input: Trajectory $X = \{X_0, X_{\delta t}, X_{2\delta t}, ..., X_T\}$, Noise increments $\delta B = \{\delta B_0, \delta B_{\delta t}, \delta B_{2\delta t}, ..., \delta B_{T-\delta t}\}$, additional drift $\tilde{b}$, weighting schedule α_t, optionally include STL adjustments.

do: Initialise at known final control

$$\mathcal{S}_{T-\delta t} = u(T-\delta t, X_{T-\delta t}) = (\sigma\sigma^\top)(T-\delta t, X_{T-\delta t})^{-1}\Big(\frac{X_T - X_{T-\delta t}}{\delta t} - b(T-\delta t, X_{T-\delta t})\Big) \tag{31}$$

for $s = T-2\delta t, T-3\delta t, ..., \delta t, 0$ **do**

Compute next training target $\mathcal{S}_s$ as

$$\mathcal{S}_s = -(\delta t)\nabla\tilde{b}(s, X_s)^\top u(s, X_s) + \tilde{J}^\top_{s+\delta t|s}\left[\tfrac{(\delta t)\alpha_{s+\delta t}}{A_s} u(s+\delta t, X_{s+\delta t}) + \tfrac{A_{s+\delta t}}{A_s}\mathcal{S}_{s+\delta t}\right] \tag{32}$$

$$+(\nabla u\,\sigma + u\,\nabla\sigma)(t, X^u_t)\cdot\delta B_t \qquad \text{(Optional STL adjustment)}$$

using the approximation $\tilde{J}_{s+\delta t|s} \approx \mathrm{Id} + \nabla(b+\tilde{b})(s, X_s)(\delta t) + \nabla\sigma(s, X_s)\cdot\delta B_s$.

end for

Algorithm A description of the resulting generalised version of the algorithm is given in Algorithm 2. The method for solving for the training targets $\mathcal{S}_s$ backwards along the trajectories in the generalised case is given in Algorithm 3.

Sticking-the-Landing Adjustments By examining the proof of Theorem B.1 in Appendix D, we see that the STL adjustments for the generalised version of the algorithm remain the same as the standard case discussed in Appendix B.3. They are included in blue in Algorithm 3.

B.5 Connections to stochastic optimal control methods

We now discuss connections of the self-consistency property with recent methodological developments in the stochastic optimal control literature. In Section 3, we discussed how the self-consistency property is a necessary property for the optimal control, but for sufficiency we must also impose additional information about the behaviour at the terminal time. We remark here that the recently introduced *adjoint matching* objective (Domingo-Enrich et al., 2025), which has proven highly effective for fine-tuning flow-based generative models, can also be recovered from this self-consistency framework by taking $t = T$ in (6) and imposing the terminal condition $u(T, x) = \nabla \log F(x)$. The adjoint matching objective was originally derived through the adjoint equation and removing terms that are zero at the solution; here, we see that it arises naturally from the self-consistency derivation in Theorem 2.1.

We also note that the generalised self-consistency property relates to the family of 'reparameterisation matrices' used in the *SOCM* loss (Domingo-Enrich et al., 2024), by choosing the matrices to be $\tilde{J}_{t|s}$. The resulting generalised version of our algorithm therefore shares connections to the *unweighted SOCM* loss discussed in Domingo-Enrich (2024), but crucially we do not compromise the theoretical guarantees of the method.

Note that the general stochastic optimal control problem also includes a running cost f, while the self-consistency properties we discuss in this work correspond to zero running costs. Analogous self-consistency properties can be derived for non-zero running costs by making the appropriate changes in the derivations. The self-consistency properties hold for general $s < t$, so give rise to a large family of losses by time-weighting according to chosen α-schedules. This is crucial for the diffusion bridge setting studied in this work due to the singular behaviour at the endpoint.

C Experimental details and additional results

We now provide additional experiments to validate our approach in a range of different settings. We will compare the standard form of the algorithm presented in Algorithm 1 (that is, $\tilde{b} = 0$), along with the 'generalised' version in Algorithm 2 in which we take $\tilde{b} = -b$. The main reported results use the *average* α-schedule; in Appendix C.6 we also report results for other α-schedules. We first consider

four standard experimental settings used in the diffusion bridge literature, before finally testing on a more challenging example from the computational chemistry literature.

We focus our comparisons on the recent neural-guided method of Yang et al. (2025) (NGDB); this is the most directly comparable method, as it similarly learns a neural control of the drift in an online manner, enabling applications for rare event scenarios in which the alternative neural methods would struggle. Note that we do not aim to necessarily beat the NGDB method, as this method differentiates through the trajectory which provides an advantage. We use NGDB here as a strong baseline and aim to achieve comparable results at a lower computational cost, since our approach does not differentiate through the trajectory. We also include comparisons with the methods *score-diffusion bridge* (SDB) (Heng et al., 2025) and *forward-bridge* (FB) (Baker et al., 2025), which both learn from uncontrolled simulations. As expected, we find these approaches to generally perform well in settings where the terminal point occurs frequently under the unconditioned dynamics, but to struggle otherwise, which agrees with the findings in Yang et al. (2025).

All experiments were carried out on a Nvidia GeForce RTX 2080Ti GPU. For the neural adjustment term $\eta_\theta = \nabla\psi_\theta$, we parameterise ψ_θ using the MLP architecture with sinusoidal time-embedding used in De Bortoli et al. (2021), with main hidden dimensions [128,128] and time-embedding hidden dimensions [64,64]. For fair comparison, we also used the same conservative architecture for NGDB, which we found performed slightly better than a standard architecture. In all experiments, we use the Adam optimiser (Kingma and Ba, 2015). Below, we describe the individual experiment setups and implementation details used. For the alternative methods, we use the default hyperparameters provided in their respective codebases unless otherwise specified; we provide the links to these codebases in Appendix E.

Metrics In several of the experimental settings, we have access to the ground-truth diffusion bridge solution. In these experiments, we report the KL divergence $\mathrm{KL}(\mathbb{P}^*\|\mathbb{P}^\theta)$ relative to this ground-truth, which can be calculated according to the expression

$$\mathrm{KL}(\mathbb{P}^*\|\mathbb{P}^\theta) = \mathbb{E}_{\mathbb{P}^*}\left[\tfrac{1}{2}\int_0^T \|\sigma(t,X_t)^\top u^*(t,X_t) - \sigma(t,X_t)^\top u_\theta(t,X_t)\|^2 \mathrm{d}t\right]. \tag{33}$$

When we do not have access to the ground-truth, we instead report the KL divergence $\mathrm{KL}(\mathbb{P}^\theta\|\mathbb{P})$ relative to the base process. Amongst stochastic processes bridging between the initial and terminal states, the diffusion bridge solution is the one that minimises this divergence, so provided that the obtained solutions transport to the endpoint correctly then this metric provides an indication of how well the problem has been solved. This metric can be computed as

$$\mathrm{KL}(\mathbb{P}^\theta\|\mathbb{P}) = \mathbb{E}_{\mathbb{P}^\theta}\left[\tfrac{1}{2}\int_0^T \|\sigma(t,X_t)^\top u_\theta(t,X_t)\|^2 \mathrm{d}t\right]. \tag{34}$$

In our experiments, we report values for these metrics computed using 1,000 trajectories. For the SDB method (Heng et al., 2025), we report these values for the forward bridge, which is learned by reversing the learned backwards bridge.

C.1 Ornstein-Uhlenbeck process

We consider sampling bridges of Ornstein-Uhlenbeck processes of the form

$$\mathrm{d}X_t = -\alpha X_t \mathrm{d}t + \sigma \mathrm{d}B_t, \tag{35}$$

with starting state $x_0 = e_1$, termination state $x_1 = e_2$, termination time $T = 1$, and diffusion coefficient $\sigma = 0.1$. Ornstein-Uhlenbeck bridges are a standard setting in the literature for quantitatively evaluating performance (Heng et al., 2025; Baker et al., 2025; Yang et al., 2025), as the ground-truth control $u^*(t,x)$ can be calculated in closed-form as

$$u^*(t,x) = \frac{2\alpha e^{-\alpha(1-t)}}{(1-e^{-2\alpha t})\sigma^2}(x_1 - xe^{-\alpha t}). \tag{36}$$

The training curves shown in Figure 3a are for $\alpha = 2.0$ and $d = 2$. The results in Figure 3b comparing the effect of dimension use $\alpha = 2.0$, and the results comparing the effect of OU parameter in Figure 3c use $d = 2$. To enable a fair comparison between the methods, we use the same conservative neural architecture for both, and use the same learning rate, number of gradient update steps, and number

Table 1: Quantitative results for experiments (mean±std over 5 runs), comparing our method (with both $\tilde{b} = 0$ and $\tilde{b} = -b$ with NGDB (Yang et al., 2025), SDB (Heng et al., 2025), and (Baker et al., 2025). We note that the result marked with an asterisk* is likely unreliable as this result gave an extremely large control at the end of training, meaning that the integration underestimates the correct value.

	CIR model, $\mathrm{KL}(\mathbb{P}^*\|\mathbb{P}^\theta)\downarrow$	Double Well, $\mathrm{KL}(\mathbb{P}^*\|\mathbb{P}^\theta)\downarrow$	Cell Diffusion, $\mathrm{KL}(\mathbb{P}^\theta\|\mathbb{P})\downarrow$		
			Normal	Rare	Multimodal
Ours ($\tilde{b} = 0$)	0.023±0.010	0.148±0.073	6.53±0.14	66.1±0.9	792.8±0.9
Ours ($\tilde{b} = -b$)	0.055±0.009	0.041±0.013	6.53±0.17	66.0±0.5	792.4±1.0
NGDB	0.051±0.012	0.098±0.047	6.48±0.15	65.5±0.6	783.4±0.8
SDB	0.104±0.011	0.229±0.024	5.56±0.39*	75.0±5.1	1845±230
FB	0.074±0.011	-	6.51±0.14	237±98	4836±954

of trajectory simulations per step. We used 500 time-discretisation steps, simulated 64 trajectories at each step, and used a learning rate of 1e-4. For Figures 3b and 3c, we ran the algorithms for 10,000 steps; we sometimes found the value for NGDB to start increasing again later in training, so the reported results are the lowest value for NGDB over the 10,000 training iterations. In all experiments, our method trained significantly faster than NGDB, which can be attributed to not requiring backpropagation through the trajectories.

Both methods learn the true drift to a very high degree of accuracy, and we remark that the difference in KL that is visible between the two methods is predominantly due to the behaviour towards the terminal time T. The methods behave comparably earlier in time, but the error of NGDB significantly increases approaching the endpoint. We hypothesise this is a consequence of the differing types of training objective; NGDB uses a KL-based objective, while our approach fits to the desired endpoint behaviour directly in the loss construction so is more accurate towards the terminal time. We found that the methods SDB and FB did not learn the bridges successfully in these experiments, which is unsurprising given that they involve rare events under the base dynamics.

C.2 Cox–Ingersoll–Ross model

We now consider computing bridges for the Cox-Ingersoll-Ross (CIR) model (Cox et al., 1985), which is used in mathematical finance to model the evolution of interest rates. The CIR model (Cox et al., 1985) is a 1-dimensional stochastic process that evolves according to the SDE

$$\mathrm{d}X_t = a(b - X_t)\mathrm{d}t + \varepsilon\sqrt{X_t}\mathrm{d}B_t. \tag{37}$$

We consider this example as it allows us to verify our method on an SDE with a *spatially-dependent* diffusion coefficient $\sigma(t, x) = \varepsilon\sqrt{x}$, and also because the ground-truth is again known in closed form, as the transition densities can be written as $ce^{-u-v}\left(\frac{v}{u}\right)^{q/2} I_q(2\sqrt{uv})$, where $c = \frac{2a}{(1-e^{-a(T-t)})\sigma^2}$, $q = \frac{2ab}{\sigma^2} - 1$, $u = cxe^{-a(T-t)}$, $v = cx_T$, and I_q is the modified Bessel function of the first kind of order q. This again allows us to report $\mathrm{KL}(\mathbb{P}^*\|\mathbb{P}^\theta)$ and verify that the learned diffusion bridges are correct.

In the experiment, we set $a = b = 1.0$, use a noise parameter of $\varepsilon = 1.0$, and aim to learn the bridge from $x_0 = 2.0$ to $x_1 = 2.0$. For each method, we used 500 time-discretisation steps and 2,000 training steps. For our method and for NGDB, we use a learning rate of 1e-4, and simulated 64 trajectories at each step. For FB and SDB we used a learning rate of 1e-3. When computing the values of $\mathrm{KL}(\mathbb{P}^*\|\mathbb{P}^\theta)$, we truncated the final five steps to avoid numerical issues when computing the ground-truth towards the terminal time.

We find that taking $\tilde{b} = 0$ performs slightly better than $\tilde{b} = -b$, which in turn performs comparably with NGDB. The methods SDB and FB learn reasonably well in this example as the terminal state is well-covered by the unconditioned dynamics, though they are still outperformed by the methods that learn from controlled dynamics.

C.3 Double-well potential

Next, we consider overdamped Langevin dynamics $\mathrm{d}X_t = -\nabla U(X_t)\mathrm{d}t + \sigma\mathrm{d}B_t$ using a classical double-well potential $U(x) = v(x^2 - 1)^2$, and aim to learn the bridge from $x_0 = 1$ to $x_1 = -1$; this

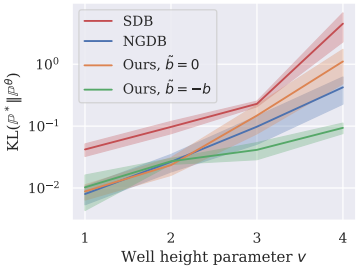


Figure 6: Comparison of performance on the double-well experiment, for different barrier height parameters v.

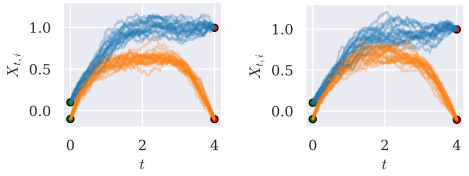


Figure 7: Rare event example for cell diffusion process, from Yang et al. (2025), showing $X_{t,1}$ (•) and $X_{t,2}$ (•). Left: NGDB; Right: Ours.

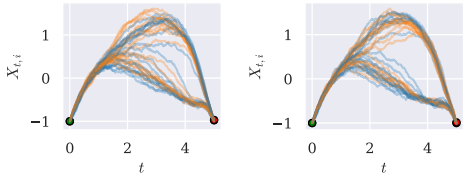


Figure 8: Multi-modal example for cell diffusion process, from Yang et al. (2025), showing $X_{t,1}$ (•) and $X_{t,2}$ (•). Left: NGDB; Right: Ours

bridge problem was previously considered in Pidstrigach et al. (2025). In light of the discussion in Appendix B.4, this presents a potentially more troublesome problem setting for our method, as the concave peak between the two metastable states can lead to large Jacobian terms. We again report $\mathrm{KL}(\mathbb{P}^* \| \mathbb{P}^\theta)$ relative to the ground-truth, which in 1 dimension can be approximated by numerically solving the backwards Kolmogorov equation.

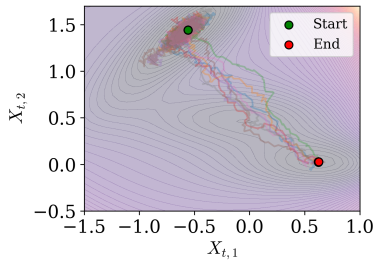
For each method, we used 500 time-discretisation steps and 4,000 training steps. For our method and NGDB we used a learning rate of 1e-3, and simulated 64 trajectories at each step, and we use the default hyperparameters for SDB. We note that we were unable to obtain good results for the double-well experiments using the FB method; we hypothesise that is due to the behaviour of the adjoint process used for the training samples, which Baker et al. (2025) remark can exacerbate the difficulty of learning the correct dynamics for rare events.

We report results in Table 1 for a well height of $v = 3.0$, which results in a strong peak between the states. In Figure 6, we also compare the methods for different heights v of the well barrier. We see that the methods that learn from controlled dynamics (that is, NGDB and our method) outperform SDB which learns from the unconditional dynamics; in particular, the performance of SDB becomes significantly worse for the largest value, which is unsurprising as the uncontrolled simulations will rarely cross between the wells in this case. For lower well heights, our method performs comparably with NGDB for both choices of $\tilde{b}$. As the well height increases, we see that taking $\tilde{b} = -b$ performs better; this reflects the discussion in Appendix B.4 and demonstrates how avoiding large growths in the Jacobian terms can improve performance.

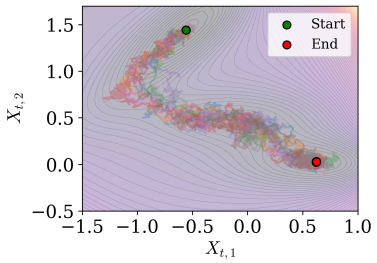
C.4 Cell Diffusion Process

We now consider an example in which the reference drift b is not conservative, so following the discussion in Section 3 we include the reference drift in our parameterisation. We consider the cell differentiation model of Wang et al. (2011), which has previously been used to provide a qualitative evaluation of diffusion bridge methods in Heng et al. (2025), Baker et al. (2025), and Yang et al. (2025). The model dynamics are given by the 2-dimensional SDE

$$\mathrm{d}X_t = \begin{bmatrix} \frac{X_{t,1}^4}{2^{-4}+X_{t,1}^4} + \frac{2^{-4}}{2^{-4}+X_{t,2}^4} - X_{t,1} \\ \frac{X_{t,2}^4}{2^{-4}+X_{t,2}^4} + \frac{2^{-4}}{2^{-4}+X_{t,1}^4} - X_{t,2} \end{bmatrix} \mathrm{d}t + \sigma \, \mathrm{d}B_t. \tag{38}$$



(a) Using $\tilde{b} = 0$ gives unstable learning due to the large Jacobian terms, and thus fails to learn the correct dynamics.



(b) Using $\tilde{b} = -b$ avoids large growth in the Jacobians, enabling successful learning of the correct dynamics.

Figure 9: Comparison of the trajectories obtained by taking auxiliary drifts $\tilde{b} = 0$ and $\tilde{b} = -b$ in our method, in the Müller-Brown example.

Table 2: Comparison of results for the Müller–Brown experiment. The values for $\mathrm{KL}(\mathbb{P}^\theta \| \mathbb{P})$ show mean±std over 5 runs, while the Max Energy values report the average mean and average std over the 5 runs.

	$\mathrm{KL}(\mathbb{P}^\theta \| \mathbb{P}) \downarrow$	Max Energy
Ours ($\tilde{b} = -b$)	27.8±1.7	-33.6±6.1
NGDB (Yang et al., 2025)	28.6±0.4	-35.9±4.9
DL (Du et al., 2024)	38.6±0.2	-32.5±5.9

We consider the three settings used in Yang et al. (2025): a *normal* event, a *rare* event, and a *multi-modal* example. For the normal event, the starting state is $x_0 = [0.1, -0.1]$, the termination state is $x_T = [2.0, -0.1]$, and the termination time is $T = 4$. For the rare event, the starting state is again $x_0 = [0.1, -0.1]$, the termination state is $x_T = [1.0, -0.1]$, and the termination time is again $T = 4$. For the multi-modal example, the starting state is $x_0 = [-1.0, -1.0]$, the termination state is $x_T = [-1.0, -1.0]$, and the termination time is $T = 5$. In all settings, the diffusion coefficient is $\sigma = 0.1$, and we train for 1000 steps. For our approach and NGDB, we use a learning rate of 1e-3, and otherwise use the default parameters for the alternative methods.

There is no ground-truth available in this setting, so to provide a quantitative assessment of performance we report $\mathrm{KL}(\mathbb{P}^\theta \| \mathbb{P})$ relative to the base process. All methods performed similarly well in the 'normal' setting. In the rare and multimodal settings, NGDB appears to perform slightly better than our method, but both performed strongly and significantly outperformed the methods that use unconditioned dynamics. SDB performed reasonably in the rare setting but failed in the multimodal setting, and FB failed to learn the dynamics accurately in both settings. We also provide a qualitative comparison with NGDB in Figures 7 and 8, and observe similar bridge simulations for the two methods.

C.5 Müller-Brown potential

So far, we have demonstrated strong performance of our method in the standard experimental settings considered in the diffusion bridge literature, comparable with the strongest existing method NGDB at a significantly reduced computational cost. We have also observed how taking $\tilde{b} = -b$ in the generalised version of our algorithm can provide improved performance over the standard version in settings where the Jacobian terms become large. To conclude, we consider a harder example in this vein which is chosen to challenge our method and better understand its limitations, inspired by applications in computational chemistry.

The Müller-Brown potential is a classical example of a potential energy surface U, and is widely used as a benchmark for studying transition paths of overdamped Langevin dynamics between two metastable states. To examine performance in this setting, we follow the experimental setup used in

Du et al. (2024). The potential used is defined as

$$U(x,y) = -200\exp\big(-(x-1)^2 - 10y^2\big) \tag{39}$$
$$-100\exp\big(-x^2 - 10(y-0.5)^2\big) \tag{40}$$
$$-170\exp\big(-6.5(0.5+x)^2 + 11(x+0.5)(y-1.5) - 6.5(y-1.5)^2\big) \tag{41}$$
$$+15\exp\big(0.7(1+x)^2 + 0.6(x+1)(y-1) + 0.7(y-1)^2\big), \tag{42}$$

with initial state $x_0 = [-0.559, 1.442]$, terminal state $x_T = [0.624, 0.028]$, diffusion term $\sigma(t,x) = 3\mathrm{Id}_2$, and termination time $T = 0.05$.

We compare against NGDB, and also to the recent Doob's Lagrangian method (DL) (Du et al., 2024); this method uses a Gaussian path parameterisation and is specifically designed for transition path sampling, making it a strong baseline for comparison. The steep slopes of the potential make this a challenging setting, so we use $\tilde{b} = -b$ in our method to avoid large Jacobian terms. Indeed, using $\tilde{b} = 0$ generally failed to learn the correct dynamics, again reflecting the discussion in Appendix B.4. We used 1000 discretisation steps. For our method and for NGDB, we ran for 4000 training steps, simulated 64 trajectories at each outer step, and used a learning rate of 3e-4; for DL (Du et al., 2024) we use the default parameters provided in their codebase. For our method, we also clipped the losses at 1e4, which we found to improve stability.

The results are shown in Table 2, and a visualisation of the obtained trajectories for our method is given in Figure 5. We see that NGDB and our method learn similar values of $\mathrm{KL}(\mathbb{P}^\theta \| \mathbb{P})$ relative to the base process, though our method has a slightly larger variance across runs. These values are lower than for DL, which is likely because the Gaussian path parameterisation cannot capture the true dynamics as accurately. Their parameterisation does however mean that DL can train significantly faster than our method and NGDB, as it can be trained without requiring simulations. All of the methods learn qualitatively similar trajectories, and have similar maximum energies along the transition paths. We emphasise that we do not aim to achieve state-of-the-art results for transition path sampling (TPS) in this work; this is a field with an extensive literature, with many methods specifically designed to tackle this problem. We instead aim to provide a method for general diffusion bridge problems, and use this example to again highlight the improvements that come from the generalised form of the self-consistency property. Nevertheless, our method obtains good performance in this setting, and extending our method to target TPS problems specifically is a promising direction for future work.

This experiment provides another example showcasing the benefit of taking $\tilde{b} = -b$ in the generalised version of our algorithm, compared to taking $\tilde{b} = 0$. This helps to avoid large growth in the Jacobian terms caused by the steep potential. Taking $\tilde{b} = 0$ generally caused learning to be unstable and usually fail, other than for some specific choices of α-schedule (see the trajectories in Figure 9). In contrast, taking $\tilde{b} = -b$ gave consistent and reliable training.

C.6 α-schedule experiments

In Table 3, we perform the same experiments as above, and show the results for the *next-step*, *average*, and *endpoint* α-schedules. We use the same hyperparameters as above (aside from adding loss clipping of 10 for the CIR model, for $\tilde{b} = 0$ with next-step α-schedule). Generally, we find the average schedule to perform strongest so we report those results in the main comparison above; it outperforms the endpoint schedule in all examples, and next-step prediction in the majority. In cases where the Jacobian can lead to large target variance (like in the double-well example, for $\tilde{b} = 0$), next-step prediction performs better than average prediction, which is in line with the discussion in Appendix B.4.

C.7 Sticking-the-landing adjustments

We now compare performance of our algorithm with and without the STL adjustments discussed in Appendix B.3. We use the same hyperparameters as in the previous experiments, and report the results in Table 4. Including the STL adjustments generally appears to improve performance slightly, with the largest improvements seen in cases where large Jacobians arise (that is, the double-well and Müller-Brown experiments). However, as the STL adjustments require computing the terms $(\nabla u\,\sigma + u\,\nabla\sigma)(t, X_t^u) \cdot \delta B_t$, they increase the computational cost of the method and training steps are approximately 1.5 times slower.

Table 3: Quantitative results for experiments, comparing α-schedules (mean $\pm$std, over 5 runs).

	CIR model, $\mathrm{KL}(\mathbb{P}^*\|\mathbb{P}^\theta)\downarrow$	Double Well, $\mathrm{KL}(\mathbb{P}^*\|\mathbb{P}^\theta)\downarrow$	Cell Diffusion, $\mathrm{KL}(\mathbb{P}^\theta\|\mathbb{P})\downarrow$		
			Normal	Rare	Multimodal
$\tilde{b}=0$, next-step	0.021±0.007	0.126±0.069	7.38±0.49	74.6±14.1	959.9±55.3
$\tilde{b}=0$, average	0.023±0.010	0.150±0.078	6.53±0.14	66.1±0.9	792.8±0.9
$\tilde{b}=0$, endpoint	0.026±0.005	3.49±2.40	6.79±0.10	69.1±1.1	793.0±1.0
$\tilde{b}=-b$, next-step	0.042±0.022	0.067±0.044	6.95±0.29	68.6±1.9	1136.2±288.9
$\tilde{b}=-b$, average	0.055±0.009	0.041±0.013	6.53±0.17	66.0±0.5	792.4±1.0
$\tilde{b}=-b$, endpoint	0.068±0.027	0.278±0.101	7.64±0.59	67.4±0.9	795.5±3.2

Table 4: Quantitative results for experiments, comparing performance with and without STL adjustments (mean $\pm$std, over 5 runs).

	CIR model, $\mathrm{KL}(\mathbb{P}^*\|\mathbb{P}^\theta)\downarrow$	Double Well, $\mathrm{KL}(\mathbb{P}^*\|\mathbb{P}^\theta)\downarrow$	Cell Diffusion, $\mathrm{KL}(\mathbb{P}^\theta\|\mathbb{P})\downarrow$		
			Normal	Rare	Multimodal
$\tilde{b}=0$, without STL	0.023±0.010	0.148±0.073	6.53±0.14	66.1±0.9	792.8±0.9
$\tilde{b}=0$, with STL	0.016±0.001	0.042±0.024	6.47±0.16	65.7±0.6	792.4±1.0
$\tilde{b}=-b$, without STL	0.055±0.009	0.041±0.013	6.53±0.17	66.0±0.5	792.4±1.0
$\tilde{b}=-b$, with STL	0.051±0.007	0.022±0.015	6.48±0.14	66.1±0.5	792.6±0.8

D Proofs

In this section, we provide proofs for the results stated in the paper.

Recall that we consider making a change of measure $\frac{\mathrm{d}\mathbb{Q}}{\mathrm{d}\mathbb{P}}(X) \propto F(X_T)$. We first provide a brief lemma concerning conditional expectations under this change of measure, which we will utilise to prove Theorem 2.1.

Lemma D.1. *Define $F_t = \mathbb{E}_{\mathbb{P}}[F(X_T)|X_t]$. Then for any H, we have*

$$\tfrac{1}{F_t}\mathbb{E}_{\mathbb{P}}[FH|X_t] = \mathbb{E}_{\mathbb{Q}}[H|X_t], \qquad \mathbb{Q}-a.s. \tag{43}$$

Proof. As we are considering the change of measure $\frac{\mathrm{d}\mathbb{Q}}{\mathrm{d}\mathbb{P}}(X) \propto F(X_T)$, a quick calculation shows that the local densities are given by $\frac{\mathrm{d}\mathbb{Q}}{\mathrm{d}\mathbb{P}}|_{\sigma(X_t)} = F_t$. For a test function $g(X_t)$, we therefore have

$$\begin{aligned}
\mathbb{E}_{\mathbb{Q}}\big[g(X_t)\tfrac{1}{F_t}\mathbb{E}_{\mathbb{P}}[FH|X_t]\big] &= \mathbb{E}_{\mathbb{P}}\big[g(X_t)\mathbb{E}_{\mathbb{P}}[FH|X_t]\big] &(44)\\
&= \mathbb{E}_{\mathbb{P}}\big[g(X_t)FH\big] &(45)\\
&= \mathbb{E}_{\mathbb{Q}}\big[g(X_t)H\big] &(46)\\
&= \mathbb{E}_{\mathbb{Q}}\big[g(X_t)\mathbb{E}_{\mathbb{Q}}[H|X_t]\big]. &(47)
\end{aligned}$$

Here, we used properties of conditional expectations in the second and final lines, and in the first and third lines used the definition of the reweighted measure $\mathbb{Q}$. We thus see that $\frac{1}{F_t}\mathbb{E}_{\mathbb{P}}[FH|X_t] = \mathbb{E}_{\mathbb{Q}}[H|X_t]$ $\mathbb{Q}$-almost surely, as required. □

For the next theorem, we assume for the first part that F is differentiable. The second part holds for more general F, including for $F = \delta_{x_T}$ corresponding to diffusion bridges.

Theorem 2.1. *The control drift u^* of the solution satisfies*

$$u^*(t,x) = \mathbb{E}_{\mathbb{P}^{u^*}}[J_{T|t}^\top \nabla \log F(X_T)|X_t = x]. \tag{5}$$

Moreover, the control drift satisfies the following self-consistency property,

$$u^*(s,x) = \mathbb{E}_{\mathbb{P}^{u^*}}[J_{t|s}^\top u^*(t,X_t)|X_s = x], \qquad \text{for } s<t. \tag{6}$$

Proof. We obtain the first statement of the theorem by making the following calculations, starting from Doob's h-transform and using Lemma D.1 with $H = \nabla \log F(X_T)$ in the final line.

$$u^*(t,x) = \nabla_x \log \mathbb{E}_{\mathbb{P}}[F(X_T) \mid X_t = x] \tag{48}$$

$$= \frac{\nabla_x \mathbb{E}_{\mathbb{P}}[F(X_T) \mid X_t = x]}{\mathbb{E}_{\mathbb{P}}[F(X_T) \mid X_t = x]} \qquad \nabla \log f = \tfrac{\nabla f}{f} \tag{49}$$

$$= \frac{\mathbb{E}_{\mathbb{P}}[J_{T|t}^\top \nabla F(X_T) \mid X_t = x]}{\mathbb{E}_{\mathbb{P}}[F(X_T) \mid X_t = x]} \qquad \text{interchange } \nabla_x \text{ and } \mathbb{E}_{\mathbb{P}} \tag{50}$$

$$= \frac{\mathbb{E}_{\mathbb{P}}[J_{T|t}^\top F(X_T) \nabla \log F(X_T) \mid X_t = x]}{\mathbb{E}_{\mathbb{P}}[F(X_T) \mid X_t = x]} \qquad \nabla \log f = \tfrac{\nabla f}{f} \tag{51}$$

$$= \mathbb{E}_{\mathbb{Q}}[J_{T|t}^\top \nabla \log F(X_T) \mid X_t = x] \qquad \text{Lemma C.1} \tag{52}$$

Each of these steps are self-explanatory, with the exception of the interchanging of the derivative and the expectation in the third line. This step is nevertheless intuitive and is well-known in stochastic analysis, and is often referred to as the reparameterisation trick in the machine learning literature. It can be made rigorous using the theory of stochastic flows (Kunita, 1984; Elworthy et al., 1999); namely, if one considers a fixed realisation ω of the Brownian path between t and T, then the terminal state X_T is a function of the value x at time t, which we can denote $x \mapsto \varphi_{t,T}(x,\omega) = X_T(\omega)$. Under mild assumptions on b, σ (see Kunita (1986)), the stochastic flow $\varphi_{t,T}(\cdot,\omega)$ is continuously differentiable. Using Leibniz's rule, for continuously differentiable F we then have

$$\nabla_x \mathbb{E}_{\mathbb{P}}[F(X_T)|X_t = x] = \nabla_x \int F(\varphi_{t,T}(x,\omega)) \mathrm{d}\mu(\omega) \tag{53}$$

$$= \int \nabla_x \varphi_{t,T}(x,\omega)^\top \nabla F(\varphi_{t,T}(x,\omega)) \mathrm{d}\mu(\omega) \tag{54}$$

$$= \mathbb{E}_{\mathbb{P}}[J_{T|t}^\top \nabla F(X_T)|X_t = x] \tag{55}$$

The self-consistency property follows from this expression by utilising the semigroup property of the Jacobian process. For $s < t$, we can 'invert' the property at the later time,

$$u^*(s,x) = \mathbb{E}_{\mathbb{Q}}[J_{T|s}^\top \nabla \log F(X_T)|X_s = x] \tag{56}$$

$$= \mathbb{E}_{\mathbb{Q}}[(J_{T|t} J_{t|s})^\top \nabla \log F(X_T)|X_s = x] \tag{57}$$

$$= \mathbb{E}_{\mathbb{Q}}\left[\mathbb{E}_{\mathbb{Q}}[J_{t|s}^\top J_{T|t}^\top \nabla \log F(X_T)|X_{[s,t]}] \,|X_s = x\right] \tag{58}$$

$$= \mathbb{E}_{\mathbb{Q}}\left[J_{t|s}^\top \mathbb{E}_{\mathbb{Q}}[J_{T|t}^\top \nabla \log F(X_T)|X_t] \,|X_s = x\right] \tag{59}$$

$$= \mathbb{E}_{\mathbb{Q}}\left[J_{t|s}^\top u^*(t,X_t)|X_s = x\right]. \tag{60}$$

Here, we used the tower rule in line 3, and the Markov property of X in line 4.

Finally, we remark that this self-consistency property also holds in the singular case when $F = \delta_{x_T}$, which corresponds to diffusion bridges. To see this, take $\tilde{T} < T$, and apply the above result with $\tilde{T}$ in place of T and with $F(X_{\tilde{T}}) = p(X_T = x_T|X_{\tilde{T}})$, which is smooth. We thus obtain the result for $s < t < \tilde{T}$, and as $\tilde{T}$ was arbitrary we see that the self-consistency property holds for $s < t < T$. □

D.1 Proof of Theorem 3.1

In this section we will prove a slightly stronger result, from which Theorem 3.1 will follow. We first start with a definition.

Definition D.1. *We call a (time-dependent) vector field u*

- bridge-preserving*, if the bridges of* (3) *coincide with those of* (1)*,*
- self-consistent*, if* (6) *is satisfied,*

- of gradient form, *if there exists a time-dependent potential ϕ such that $u(t,x) = \nabla_x \phi(t,x)$,*
- of gradient form at one point in time, *if $u(t,\cdot) = \nabla\phi$ for some $t > 0$ and time-independent potential ϕ.*

The statement of our next result requires the generator of the base dynamics,

$$\mathcal{L} = \sum_i b_i \partial_i + \tfrac{1}{2} \sum_{ij} (\sigma\sigma^\top)_{ij} \partial_{ij}. \tag{61}$$

Theorem D.2. *The following are equivalent:*

(i) The vector field u is bridge-preserving,

(ii) we have $u = \nabla\phi$, and ϕ satisfies the Hamilton-Jacobi-Bellman (HJB) equation

$$\partial_t \phi + \mathcal{L}\phi + \tfrac{1}{2}|\sigma\nabla\phi|^2 = 0, \tag{HJB}$$

(iii) the vector field u is of gradient form and self-consistent.

(iv) the vector field u is of gradient form at one point in time and self-consistent.

Proof. To simplify the notation, we present the proof here for a scalar diffusion term σ, but the proof proceeds similarly for general diffusion coefficients under our ellipticity assumption by making the appropriate modifications.

(i) $\iff$ (ii): This equivalence is well known (and SDEs with bridge preserving drifts are in the reciprocal class of the base process), see for example Léonard (2013) and Rœlly (2013), or Vargas et al. (2024, Section E.5).

(ii) $\iff$ (iii): Taking the gradient of the HJB-equation and using $u = \nabla\phi$, we see that

$$0 = \partial_t u + \nabla\mathcal{L}\phi + \frac{\sigma^2}{2}\nabla|u|^2 = \partial_t u + \nabla(b\cdot u) + \tfrac{\sigma^2}{2}\Delta u + \tfrac{\sigma^2}{2}\nabla|u|^2, \tag{Grad-HJB}$$

using the facts that $\nabla\mathcal{L}\phi = \nabla(b\cdot u) + \frac{\sigma^2}{2}\nabla(\Delta\phi)$ and $\nabla(\Delta\phi) = \Delta u$. Here and throughout, it is understood the Laplacian is taken element-wise in Δu.

On the other hand, we can compute the time evolution of $t \mapsto J_{t|s}^\top u(t, X_t^u)$ along the controlled dynamics, using the product rule of (Itô) stochastic calculus:

$$\begin{aligned}
\mathrm{d}(J_{t|s}^\top u(t,X_t^u)) &= J_{t|s}^\top \,\mathrm{d}u(t,X_t^u) + \mathrm{d}J_{t|s}^\top u(t,X_t^u) && \text{(62a)}\\
&= J_{t|s}^\top \Big(\partial_t u(t,X_t^u)\,\mathrm{d}t + \nabla u(t,X_t^u)\cdot\big(b(t,X_t^u)\,\mathrm{d}t + \sigma^2 u(t,X_t^u)\,\mathrm{d}t + \sigma\mathrm{d}B_t\big) + \tfrac{\sigma^2}{2}\Delta u(t,X_t^u)\,\mathrm{d}t\Big)\\
&\quad + J_{t|s}^\top \nabla b(t,X_t^u)\cdot u(t,X_t^u)\,\mathrm{d}t && \text{(62b)}\\
&= J_{t|s}^\top \underbrace{\Big(\partial_t u + \nabla(b\cdot u) + \tfrac{\sigma^2}{2}\Delta u + \tfrac{\sigma^2}{2}\nabla|u|^2\Big)(t,X_t^u)}_{\text{RHS of (Grad-HJB)}}\,\mathrm{d}t + \sigma J_{t|s}^\top \nabla u(t,X_t^u)\cdot\mathrm{d}B_t. && \text{(62c)}
\end{aligned}$$

The last line of this calculation uses $\frac{1}{2}\nabla|u|^2 = (u\cdot\nabla)u$, which holds for vector fields of gradient form. Note that in the general case with a possibly spatially-dependent diffusion coefficient, performing similar calculations gives the expression

$$\begin{aligned}
\mathrm{d}(J_{t|s}^\top u(t,X_t^u)) = J_{t|s}^\top &\underbrace{\Big(\partial_t u + \nabla(b\cdot u) + \tfrac{\sigma^2}{2}\Delta u + \tfrac{\sigma^2}{2}\nabla|u|^2\Big)(t,X_t^u)}_{\text{RHS of (Grad-HJB)}}\,\mathrm{d}t\\
&+ J_{t|s}^\top \left(\nabla u\,\sigma + u\,\nabla\sigma\right)(t,X_t^u)\cdot\mathrm{d}B_t.
\end{aligned} \tag{63}$$

The equivalence between (ii) and (iii) essentially follows from the observation that the drift in $\mathrm{d}(J_{t|s}^\top u(t, X_t^u))$ coincides with the right-hand side of (Grad-HJB). More precisely, from the above calculation the self-consistency property (6) can be re-expressed in the form

$$u(s,x) = \mathbb{E}_{\mathbb{P}^u}[J_{t|s}^\top u(t, X_t)|X_s = x] \tag{64}$$

$$= \mathbb{E}_{\mathbb{P}^u}\left[J_{s|s}^\top u(s, X_s^u) + \int_s^t \left(\partial_t u + \nabla(b\cdot u) + \tfrac{\sigma^2}{2}\Delta u + \tfrac{\sigma^2}{2}\nabla|u|^2\right)(r, X_r^u)\,\mathrm{d}r \middle| X_s^u = x\right], \tag{65}$$

owing to the fact that the martingale term $\nabla u(t, X_t^u)\cdot \mathrm{d}B_t$ has zero expectation. Using $J_{s|s} = Id$, this is equivalent to

$$\mathbb{E}_{\mathbb{P}^u}\left[\int_s^t \left(\partial_t u + \nabla(b\cdot u) + \tfrac{\sigma^2}{2}\Delta u + \tfrac{\sigma^2}{2}\nabla|u|^2\right)(r, X_r^u)\,\mathrm{d}r \middle| X_s^u = x\right] = 0. \tag{66}$$

Now it is clear that (ii) implies (iii), because (Grad-HJB) implies (66), for all $s \le t$ and $x \in \mathbb{R}^d$. Conversely, we can obtain (Grad-HJB) from (66) by taking the derivative with respect to t of (66) and setting $t = s$; hence (iii) implies (ii).

(iii) $\iff$ (iv): We first define the vorticity

$$\Omega_{ij}(t,x) := \partial_i u_j(t,x) - \partial_j u_i(t,x), \qquad t > 0, \quad x \in \mathbb{R}^d. \tag{67}$$

The Poincaré lemma implies that $\Omega(t,\cdot) = 0$ if and only if $u(t,\cdot)$ is of gradient form. As above, self-consistency yields the equation

$$\partial_t u + \tfrac{\sigma^2}{2}\Delta u + (\nabla b)u + (b\cdot\nabla)u + (u\cdot\nabla)u = 0. \tag{68}$$

Note that this does not simplify to (Grad-HJB), since we do not make the assumption $u = \nabla\phi$. Taking another spatial derivative and anti-symmetrising, we see that Ω solves the linear parabolic PDE

$$\left(\partial_t + (b+u)\cdot\nabla + \tfrac{1}{2}\Delta\right)\Omega + (\nabla(b+u))\,\Omega + \Omega\,(\nabla(b+u))^\top = 0, \tag{69}$$

which clearly admits $\Omega \equiv 0$ as a solution. Under standard regularity assumptions, solutions to (69) are unique once $\Omega(t,\cdot)$ is prescribed for some t (both forwards and backwards in time), see for example Pazy, 2012. From this, it follows that (iv) implies (iii), and the converse implication follows directly from the definition. □

Theorem 3.1. *Within the class of controlled diffusion processes of the form (3) that terminate at x_T, there is a unique process X_t^u that satisfies the self-consistency property $u(s, X_s) = \mathbb{E}_{\mathbb{P}^u}[J_{t|s}^\top u(t, X_t)|X_s]$ and whose control is of gradient form for some time t, and this process is the diffusion bridge $X_t^u = X_t^*$.*

Proof of Theorem 3.1. The result follows as a corollary from Theorem D.2. More precisely, we use the implication (iv) $\iff$ (ii). Under the gradient form and consistency assumptions, the addition of the control u does not change the process when conditioned on $X_T = x_T$. □

D.2 Generalised self-consistency property

In this section, we provide the proofs of the analogous results for the generalised self-consistency property (20) presented in Appendix B.4. We first fill in the details of the change-of-measure argument outlined in Appendix B.4. Recall that $\tilde{J}_{t|s}$ is the Jacobian process associated to the auxiliary process $\tilde{X}$, which evolves according to the SDE $\mathrm{d}\tilde{X}_t = (b+\tilde{b})(t,\tilde{X}_t)\mathrm{d}t + \sigma(t,\tilde{X}_t)\mathrm{d}B_t$ that includes an additional drift function $\tilde{b}$.

Theorem B.1 (Generalised self-consistency property). *For any differentiable $\tilde{b}$ satisfying the conditions of Girsanov's theorem, the control drift u^* of the solution satisfies*

$$u^*(s,x) = \mathbb{E}_{\mathbb{P}^{u^*}}\Big[-\int_s^t \tilde{J}_{r|s}^\top \nabla\tilde{b}(r, X_r)^\top u^*(r, X_r)\mathrm{d}r + \tilde{J}_{t|s}^\top u(t, X_t)|X_s = x\Big]. \tag{70}$$

Proof. Define $\tilde{\mathbb{P}}$ to be the path measure associated to the auxiliary process $\mathrm{d}\tilde{X}_t = (b + \tilde{b})(t, \tilde{X}_t)\mathrm{d}t + \sigma(t, \tilde{X}_t)\mathrm{d}B_t$. For ease of notation, we will present the result for scalar constant diffusion coefficient σ; the proof of the general case follows with the appropriate changes. By Girsanov's theorem (Øksendal, 2003, Theorem 8.6.8), the Radon-Nikodym derivative is

$$\frac{\mathrm{d}\mathbb{P}}{\mathrm{d}\tilde{\mathbb{P}}}(X_{[s,T]}) = \exp\Big(- \int_s^T \sigma^{-1}\tilde{b}(r, X_r)^\top \mathrm{d}B_r - \int_s^T \tfrac{\sigma^{-2}}{2}\|\tilde{b}(r, X_r)\|^2 \,\mathrm{d}r\Big). \tag{71}$$

As in the proof of Theorem 2.1, we begin with Doob's h-transform, then perform the following calculations (where $B^{\mathbb{P}}, B^{\tilde{\mathbb{P}}}, B^{\mathbb{Q}}$ denote standard Brownian motions under their respective measures).

$$\begin{aligned}
u^*(s,x) &= \nabla_x \log \mathbb{E}_{\mathbb{P}}[F(X_T)|X_s = x] && (72)\\
&= \frac{1}{\mathbb{E}_{\mathbb{P}}[F(X_T)|X_s = x]}\nabla_x \mathbb{E}_{\mathbb{P}}[F(X_T)|X_s = x] && (73)\\
&= \frac{1}{\mathbb{E}_{\mathbb{P}}[F(X_T)|X_s = x]}\nabla_x \mathbb{E}_{\tilde{\mathbb{P}}}[F(\tilde{X}_T)\tfrac{\mathrm{d}\mathbb{P}}{\mathrm{d}\tilde{\mathbb{P}}}|\tilde{X}_s = x] && (74)\\
&= \frac{1}{\mathbb{E}_{\mathbb{P}}[F(X_T)|X_s = x]}\mathbb{E}_{\tilde{\mathbb{P}}}\Big[\tilde{J}_{T|s}^\top \nabla F(\tilde{X}_T)\tfrac{\mathrm{d}\mathbb{P}}{\mathrm{d}\tilde{\mathbb{P}}}\\
&\qquad + F(\tilde{X}_T)\tfrac{\mathrm{d}\mathbb{P}}{\mathrm{d}\tilde{\mathbb{P}}}\Big(- \int_s^T \tfrac{1}{\sigma^2}\tilde{J}_{r|s}^\top \nabla\tilde{b}(r,\tilde{X}_r)^\top \tilde{b}(r,\tilde{X}_r)\mathrm{d}r \int_s^T \tfrac{1}{\sigma}\tilde{J}_{r|s}^\top \nabla\tilde{b}(r,\tilde{X}_r)^\top \mathrm{d}B_r^{\tilde{\mathbb{P}}}\Big)|\tilde{X}_s = x\Big] && (75)\\
&= \frac{1}{\mathbb{E}_{\mathbb{P}}[F(X_T)|X_s = x]}\mathbb{E}_{\mathbb{P}}\Big[\tilde{J}_{T|s}^\top \nabla F(X_T)\\
&\qquad + F(X_T)\Big(- \int_s^T \tfrac{1}{\sigma^2}\tilde{J}_{r|s}^\top \nabla\tilde{b}(r,X_r)^\top \tilde{b}(r,X_r)\mathrm{d}r\\
&\qquad\quad - \int_s^T \tfrac{1}{\sigma}\tilde{J}_{r|s}^\top \nabla\tilde{b}(r,X_r)^\top (\mathrm{d}B_r^{\mathbb{P}} - \tfrac{1}{\sigma}\tilde{b})\Big)|X_s = x\Big] && (76)\\
&= \frac{1}{\mathbb{E}_{\mathbb{P}}[F(X_T)|X_s = x]}\mathbb{E}_{\mathbb{P}}\Big[\tilde{J}_{T|s}^\top \nabla F(X_T) + F(X_T)\Big(- \int_s^T \tfrac{1}{\sigma}\tilde{J}_{r|s}^\top \nabla\tilde{b}(r,X_r)^\top \mathrm{d}B_r^{\mathbb{P}}\Big)|X_s = x\Big] && (77)\\
&= \mathbb{E}_{\mathbb{Q}}\Big[\tilde{J}_{T|s}^\top \nabla \log F(X_T) - \int_s^T \tfrac{1}{\sigma}\tilde{J}_{r|s}^\top \nabla\tilde{b}(r,X_r)^\top (\mathrm{d}B_r^{\mathbb{Q}} + \sigma u^*\mathrm{d}t)|X_s = x\Big] && (78)\\
&= \mathbb{E}_{\mathbb{Q}}\Big[\tilde{J}_{T|s}^\top \nabla \log F(X_T) - \int_s^T \tilde{J}_{r|s}^\top \nabla\tilde{b}(r,X_r)^\top u^*(r,X_r)\mathrm{d}r|X_s = x\Big]. && (79)
\end{aligned}$$

Much of this derivation follows similarly to the standard case. When we perform the change of measures, we include the appropriate modifications to the Brownian motions; namely, we have $\mathrm{d}B_r^{\mathbb{P}} = \mathrm{d}B_r^{\tilde{\mathbb{P}}} + \frac{1}{\sigma}\tilde{b}(r,X_r)\mathrm{d}t$, and $\mathrm{d}B_r^{\mathbb{P}} = \mathrm{d}B_r^{\mathbb{Q}} + \sigma u^*\mathrm{d}t$.

The remainder of the proof follows similarly to the proof of Theorem 2.1; as before, we 'invert' the property at a later time t, utilising the tower rule and the semigroup property of the Jacobian $\tilde{J}$.

$$u^*(s,x) = \mathbb{E}_{\mathbb{Q}}[\tilde{J}_{T|s}^\top \nabla \log F(X_T) - \int_s^T \tilde{J}_{r|s}^\top \nabla \tilde{b}(r, X_r)^\top u^*(r, X_r)\mathrm{d}r | X_s = x] \tag{80}$$

$$= \mathbb{E}_{\mathbb{Q}}[- \int_s^t \tilde{J}_{r|s}^\top \nabla \tilde{b}(r, X_r)^\top u^*(r, X_r)\mathrm{d}r - \int_t^T \tilde{J}_{r|s}^\top \nabla \tilde{b}(r, X_r)^\top u^*(r, X_r)\mathrm{d}r + \tilde{J}_{T|s}^\top \nabla \log F(X_T) | X_t = x] \tag{81}$$

$$= \mathbb{E}_{\mathbb{Q}}[- \int_s^t \tilde{J}_{r|s}^\top \nabla \tilde{b}(r, X_r)^\top u^*(r, X_r)\mathrm{d}r - \int_t^T (J_{r|t} J_{t|s})^\top \nabla \tilde{b}(r, X_r)^\top u^*(r, X_r)\mathrm{d}r + (J_{T|t} J_{t|s})\top \nabla \log F(X_T) | X_t = x] \tag{82}$$

$$= \mathbb{E}_{\mathbb{Q}}\big[\mathbb{E}_{\mathbb{Q}}[- \int_s^t \tilde{J}_{r|s}^\top \nabla \tilde{b}(r, X_r)^\top u^*(r, X_r)\mathrm{d}r - \int_t^T (\tilde{J}_{r|t} \tilde{J}_{t|s})^\top \nabla \tilde{b}(r, X_r)^\top u^*(r, X_r)\mathrm{d}r + (\tilde{J}_{T|t} \tilde{J}_{t|s})^\top \nabla \log F(X_T) | X_{[s,t]}] \, | X_s = x\big] \tag{83}$$

$$= \mathbb{E}_{\mathbb{Q}}\big[- \int_s^t \tilde{J}_{r|s}^\top \nabla \tilde{b}(r, X_r)^\top u^*(r, X_r)\mathrm{d}r + \tilde{J}_{t|s}^\top \mathbb{E}_{\mathbb{Q}}[- \int_t^T \tilde{J}_{r|t}^\top \nabla \tilde{b}(r, X_r)^\top u^*(r, X_r)\mathrm{d}r + \tilde{J}_{T|t}^\top \nabla \log F(X_T) | X_t] | X_s = x\big] \tag{84}$$

$$= \mathbb{E}_{\mathbb{P}^{u^*}}\big[- \int_s^t \tilde{J}_{r|s}^\top \nabla \tilde{b}(r, X_r)^\top u^*(r, X_r)\mathrm{d}r + \tilde{J}_{t|s}^\top u^*(t, X_t) | X_s = x\big] \tag{85}$$

□

To conclude, we show that the generalised self-consistency property retains the same theoretical results as in the standard case.

Theorem B.2. *The result of Theorem 2.1 holds with the generalised self-consistency property (20) in place of the usual self-consistency property (6).*

Proof. In light of the proof of Theorem D.2, we need only to show that the equivalence $(ii) \iff (iii)$ still holds. To do so, we mirror the proof of the standard case, and apply an Itô expansion of $\tilde{J}_{t|s}^\top u(t, X_t^u)$ along the controlled dynamics,

$$\mathrm{d}(\tilde{J}_{t|s}^\top u(t, X_t^u)) = \tilde{J}_{t|s}^\top \mathrm{d}u(t, X_t^u) + \mathrm{d}\tilde{J}_{t|s}^\top u(t, X_t^u) \tag{86}$$

$$= \tilde{J}_{t|s}^\top \Big(\partial_t u(t, X_t^u)\,\mathrm{d}t + \nabla u(t, X_t^u) \cdot \big(b(t, X_t^u)\,\mathrm{d}t + \sigma^2 u(t, X_t^u)\,\mathrm{d}t + \sigma \mathrm{d}B_t\big) + \tfrac{\sigma^2}{2} \Delta u(t, X_t^u)\,\mathrm{d}t \Big) + \tilde{J}_{t|s}^\top \nabla(b + \tilde{b})(t, X_t^u) \cdot u(t, X_t^u)\mathrm{d}t \tag{87}$$

$$= \tilde{J}_{t|s}^\top \underbrace{\Big(\partial_t u + \nabla(b \cdot u) + \tfrac{\sigma^2}{2} \Delta u + \tfrac{\sigma^2}{2} \nabla |u|^2 \Big)(t, X_t^u)}_{\text{(Grad-HJB)}}\,\mathrm{d}t + J_{t|s}^\top \nabla \tilde{b}(t, X_t^u) \cdot u(t, X_t^u)\,\mathrm{d}t + \sigma \tilde{J}_{t|s}^\top \nabla u(t, X_t^u) \cdot \mathrm{d}B_t. \tag{88}$$

The equivalence then follows by the same argument as in the proof of Theorem D.2. □

E Licenses

The following assets were used in this work.

- Neural Guided Diffusion Bridges (NGDB) (Yang et al., 2025), `https://github.com/gefanyang/neuralbridge`
- Simulating Diffusion Bridges with Score Matching (SDB) (Heng et al., 2025), `https://github.com/jeremyhengjm/DiffusionBridge`
- Score matching for bridges without learning time-reversals (FB) (Baker et al., 2025), MIT License `https://github.com/libbylbaker/forward_bridge`
- Doob's Lagrangian (DL) (Du et al., 2024), MIT License `https://github.com/plainerman/Variational-Doob`