Variational Optimization Based Reinforcement Learning for Infinite Dimensional Stochastic Systems

Ethan N. Evans*
Department of Aerospace Engineering
Georgia Institute of Technology
eevans41@gatech.edu

Marcus A. Periera*
Institute of Robotics and Intelligent Machines
Georgia Institute of Technology
mpereira30@gatech.edu

George I. Boutselis
Department of Aerospace Engineering
Georgia Institute of Technology
gbouts@gatech.edu

Evangelos A. Theodorou
Department of Aerospace Engineering
Georgia Institute of Technology
evangelos.theodorou@gatech.edu

Abstract: Systems involving Partial Differential Equations (PDEs) have recently become more popular among the machine learning community. However prior methods usually treat infinite dimensional problems in finite dimensions with Reduced Order Models. This leads to committing to specific approximation schemes and subsequent derivation of control laws. Additionally, prior work does not consider spatio-temporal descriptions of noise that realistically represent the stochastic nature of physical systems. In this paper we suggest a new reinforcement learning framework that is mostly model-free for Stochastic PDEs with additive spacetime noise, based on variational optimization in infinite dimensions. In addition, our algorithm incorporates sparse representations that allow for efficient learning of feedback policies in high dimensions. We demonstrate the efficacy of the proposed approach with several simulated experiments on a variety of SPDEs.

Keywords: Reinforcement Learning, Planning and Control

1 Introduction and Related Work

Stochastic systems that evolve spatio-temporally can have degrees of freedom at every point on a spatial continuum. Such systems are often described by an infinite dimensional stochastic system that is represented by Stochastic Partial Differential Equations (SPDEs). SPDE systems appear in many areas of sciences and engineering such us fluid mechanics, plasma physics, partially observable stochastic control, quantum mechanics, and continuum mechanics. Examples of such systems are the stochastic Navier-Stokes equation which governs fluid flow and turbulence, the stochastic Nonlinear Schrödinger (NLS) equation, which governs the dynamics of the wavefunction of subatomic particles [1], the stochastic Nagumo equation which governs how voltage travels across a neuron in a brain [2, Chapter 10], and the stochastic Kuramoto-Sivashinsky (KS) equation which governs flame front propagation in combustion [3]. These fall into a category which covers a broad class of Partial Differential Equations (PDEs) known as *semi-linear* PDEs. A detailed exposition of certain examples in this category is given in table 1.

Beyond their role in applied physics, several PDE models have been applied in challenging robotics problems. Burgers-like reaction-advection-diffusion PDEs have been utilized to model dynamics of agents as a continuum on a 2D cylindrical surface for multi-agent formation boundary control [4]. Nagumo-like coupled reaction-diffusion models have been used for robot navigation in crowded environments [5]. The heat equation has similarly been used in robotic motion planning [6] and has been shown to have equivalence to multi-agent consensus-based control laws for robot deployment problems [7].

*Authors contributed equally

3rd Conference on Robot Learning (CoRL 2019), Osaka, Japan.

	Operators		
Partial Differential Equation	**Linear $\mathscr{A}$**	**Non-linear $F(t,X)$**	**State (or field)**
Nagumo: $u_t = \varepsilon u_{xx} + u(1-u)(u-\alpha)$	u_{xx}	$u(1-u)(u-\alpha)$	Voltage
Heat: $u_t = \varepsilon u_{xx}$	u_{xx}		Heat/temperature
Burgers (viscous): $u_t + uu_x = \varepsilon u_{xx}$	u_{xx}	uu_x	Velocity
Allen-Cahn: $u_t = \varepsilon u_{xx} + u - u^3$	u_{xx}	$u - u^3$	Phase of a material
NS: $u_t = \varepsilon \Delta u - \nabla p - (u \cdot \nabla)u$	Δu	$(u \cdot \nabla)u$	Velocity
NLS: $iu_t + \frac{1}{2}u_{xx} + \lvert u\rvert^2 u = 0$	u_{xx}	$\lvert u\rvert^2 u$	Wavefunction
KdV: $u_t + 6uu_x + u_{xxx} = 0$	u_{xxx}	uu_x	Plasma wave
KS: $u_t + uu_x + u_{xx} + u_{xxxx} = 0$	$u_{xx} + u_{xxxx}$	uu_x	Flame front

Table 1: Examples of commonly known semi-linear PDEs in a *fields representation* with x representing spatial dimensions and t representing time. NS = Navier–Stokes and KdV = Korteweg-de Vries.

Despite their ubiquity, the theory of control of SPDE systems was only introduced in the last few decades [8, 9] and remains incomplete especially for stochastic boundary control. Numerical results and algorithms for distributed control of SPDEs are limited and typically require some model reduction approach [10, 3]. In [11], the authors approach the control of the stochastic Burgers equation through the Hamilton-Jacobi-Bellman theory by applying the linear Feynman-Kac lemma; nevertheless, it lacks numerical results. In [12], the authors treat optimal control of linear deterministic PDEs by applying linear control theory, however this work is limited to linear PDEs. The book [9] gives a complete understanding of our ability so far, to apply optimal control theory to these systems.

Contrasting with the work from the controls community are recent methods founded on machine learning techniques. Despite having convincing results, these approaches commonly treat deterministic PDEs as a finite set of Ordinary Differential Equations (ODEs) through the use of Reduced Order Model (ROM) type methods. Most recent work includes [13], where the authors find reduced order Koopman-like local linear models and perform convex optimization for control using off-the-shelf solvers. However, this requires solving a least squares problem online which does not guarantee stabilizability of the resulting linear system. In [14] the authors successfully control a Navier-Stokes system with reinforcement learning on policy networks in a deterministic, finite ODE setting. Similarly, [15] presents a Deep RNN framework with MPC to control a finite, deterministic ODE representation (CFD solver) of a Navier-Stokes system.

In contrast to recent work which first require developing deterministic ROMs and then using standard approaches from Reinforcement Learning (RL) or Model Predictive Control (MPC), we treat the SPDE system directly in Hilbert spaces, and derive a variational optimization framework for episodic reinforcement of policy networks as highlighted in red in Fig.1.

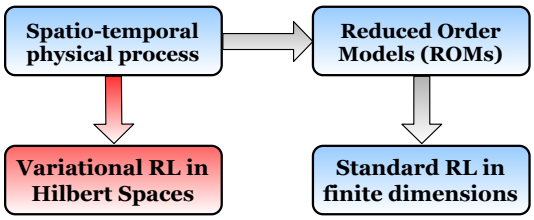


Figure 1: Our proposed approach versus traditional approaches.

We take inspiration from a general principle stemming from statistical physics and thermodynamics that has been shown to have applicability in stochastic optimal control [16]:

$$\text{Free Energy} \leq \text{Work} - \text{Temperature} \times \text{Entropy} \tag{1}$$

Optimization of this relation from a measure theoretic perspective gives rise to the well known Gibbs measure which is used in variational inference problems [17]. This perspective enables us to seek a middle ground between recent results in Deep Learning (DL) and traditional stochastic optimal control: We approach SPDEs with infinite dimensional stochastic calculus, yet apply highly successful DL techniques. We develop a new method fusing together variational optimization, episodic reinforcement learning, and measure theoretic stochastic calculus in infinite dimensions.

Our specific contributions are as follows:

1) We present a reinforcement learning framework in Hilbert spaces based on variational optimization and importance sampling for SPDEs. The resulting algorithm, called Infinite Dimensional Variational Reinforcement Learning (IDVRL), incorporates explicit feedback of the entire SPDE and allows for arbitrary non-linear policies such as Feed-forward Neural Networks (FNNs), Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs).

2) We introduce a technique to handle numerical integration of policy networks over spatial domains which we call *SparseForwardPass* for FNN and CNN policies, enabling scalability to 2D and 3D problems.

3) Since the algorithm is derived in infinite-dimensional space, any choice of numerical approximation scheme such as finite difference, spectral Galerkin or finite-element can be used to approximate trajectory samples. In addition, as a result of performing optimization in infinite dimensional space, the derivation is valid for the stochastic versions of all PDEs included in table 1 and therefore is general.

2 Problem Formulation

This work proposes control of a large class of infinite-dimensional systems described by SPDEs that are of *semi-linear* form. There are other ways to express such systems, however here we take the approach of expressing the system as evolving on time-indexed separable Hilbert spaces in order to leverage several mathematical tools developed in such spaces. Consider the general semi-linear controlled SPDE given by

$$\mathrm{d}X = \big(\mathscr{A}X + F(t,X)\big)\mathrm{d}t + G(t,X)\big(\Phi(t,X,\mathbf{x};\Theta^{(k)})\mathrm{d}t + \frac{1}{\sqrt{\rho}}\mathrm{d}W(t)\big), \tag{2}$$

where $X(t) \in \mathscr{H}$ is the state of the system which evolves on the Hilbert space $\mathscr{H}$, the linear and nonlinear operators $\mathscr{A} : \mathscr{H} \to \mathscr{H}$ and $F(t,X) : \mathbb{R} \times \mathscr{H} \to \mathscr{H}$ (resp.) are uncontrolled drift terms, $\Phi(t,X,\mathbf{x};\Theta^{(k)}) : \mathbb{R} \times \mathscr{H} \times \mathbb{R}^3 \to \mathscr{H}$ is the nonlinear control policy paramterized by $\Theta^{(k)}$ at the k^{th} iteration, $\mathrm{d}W(t) : \mathbb{R} \to \mathscr{H}$ is a Cylindrical spatio-temporal noise process (i.e. space-time white noise), and $G(t,X)$ is nonlinearity that affects both the Cylindrical noise and the control. It is used to incorporate the effects of actuation on either the field (distributed) or at the boundaries. Referring back to table 1, the generality of the *Hilbert spaces formulation* becomes clear as any semi-linear PDE can be handled by appropriately choosing $\mathscr{A}$ and F. For a more complete introduction, including some mild but necessary assumptions and clear definitions of the Cylindrical process, see the supplementary material and the references therein.

Define the uncontrolled and controlled probability measures associated with eq. (2) as $\mathscr{L}$ and $\tilde{\mathscr{L}}$, respectively. These measures roughly describe the probabilistic evolution of the system, with the probability density function as a finite dimensional analog. In this case, eq. (1) takes the form [18]

$$-\frac{1}{\rho}\log\mathbb{E}_{\mathscr{L}}\Big[\exp(-\rho J)\Big] = \min_{\mathscr{U}(\cdot,\cdot)}\Big[\mathbb{E}_{\tilde{\mathscr{L}}}(J) + \frac{1}{\rho}D_{KL}(\tilde{\mathscr{L}}\,\big|\big|\,\mathscr{L})\Big], \tag{3}$$

where $J = J(X)$ can be viewed as an arbitrary state cost function. The associated "Work" and "Entropy" terms that minimize this expression describe a minimum "energy"[2] measure. Sampling from this measure would simultaneously minimize state cost and the KL-divergence between the controlled and uncontrolled distributions, which in this case is roughly interpreted as control effort. The measure that optimizes eq. (3) is the so-called Gibbs measure

$$\mathrm{d}\mathscr{L}^* = \frac{\exp(-\rho J)\mathrm{d}\mathscr{L}}{\mathbb{E}_{\mathscr{L}}\big[\exp(-\rho J)\big]}. \tag{4}$$

While it is not known how to sample directly from eq. (4), the goal of variational optimization methods is to incrementally reduce the distance (defined in the Kullback–Leibler divergence sense) between the controlled distribution $\tilde{\mathscr{L}}$ and the optimal measure eq. (4). We formulate our variational

[2]The term energy here is used loosely to describe the landscape for work and entropy

minimization problem as

$$\begin{aligned}\Theta^* &= \underset{\Theta}{\operatorname{argmin}} D_{KL}(\mathscr{L}^* || \tilde{\mathscr{L}}) \\ &= \underset{\Theta}{\operatorname{argmin}} \left[\int \log\left(\frac{\mathrm{d}\mathscr{L}}{\mathrm{d}\tilde{\mathscr{L}}}\right) \frac{\mathrm{d}\mathscr{L}^*}{\mathrm{d}\mathscr{L}} \frac{\mathrm{d}\mathscr{L}}{\mathrm{d}\tilde{\mathscr{L}}} \mathrm{d}\tilde{\mathscr{L}} \right] = \underset{\Theta}{\operatorname{argmin}}\ L \end{aligned} \tag{5}$$

A more detailed derivation can be found in the supplementary. Finally, we introduce a version of Girsanov's Change of Measure theorem (found in supplementary) between the uncontrolled and controlled processes, resulting in the so-called Radon-Nikodym derivative given as

$$\frac{\mathrm{d}\mathscr{L}}{\mathrm{d}\tilde{\mathscr{L}}} = \exp\left\{ -\sqrt{\rho} \int_0^T \left\langle \Phi(t,X,\mathbf{x};\Theta^{(k)}), \mathrm{d}W(t) \right\rangle - \rho \frac{1}{2} \int_0^T \left\langle \Phi(t,X,\mathbf{x};\Theta^{(k)}), \Phi(t,X,\mathbf{x};\Theta^{(k)}) \right\rangle \mathrm{d}t \right\}. \tag{6}$$

Plugging in eq. (4) and eq. (6) (for importance sampling), the loss-function L becomes

$$L = \mathbb{E}_{\tilde{\mathscr{L}}} \left[\underbrace{\frac{\exp(-\rho \tilde{J})}{\mathbb{E}_{\tilde{\mathscr{L}}}\left[\exp(-\rho \tilde{J})\right]}}_{ImportanceWeight} \left(-\sqrt{\rho} \int_0^T \underbrace{\left\langle \Phi(t,X,\mathbf{x};\Theta^{(k)}), \mathrm{d}W(t) \right\rangle}_{NoiseInnerProduct} \right. \right.$$

$$\left. \left. - \frac{1}{2} \rho \int_0^T \underbrace{\left\langle \Phi(t,X,\mathbf{x};\Theta^{(k)}), \Phi(t,X,\mathbf{x};\Theta^{(k)}) \right\rangle}_{PolicyInnerProduct} \mathrm{d}t \right) \right], \tag{7}$$

where $\tilde{J}$ is defined by

$$\tilde{J} = \underbrace{J}_{StateCost} + \frac{1}{\sqrt{\rho}} \int_0^T \underbrace{\left\langle \Phi(t,X,\mathbf{x};\Theta^{(k)}), \mathrm{d}W(t) \right\rangle}_{NoiseInnerProduct} + \frac{1}{2} \int_0^T \underbrace{\left\langle \Phi(t,X,\mathbf{x};\Theta^{(k)}), \Phi(t,X,\mathbf{x};\Theta^{(k)}) \right\rangle}_{PolicyInnerProduct} \mathrm{d}t. \tag{8}$$

The intermediate steps that lead to the above final forms of eq. (5) and eq. (7) can be found in the supplementary material. The loss-function L exponentiates the cost of the system trajectories, evaluated by $\tilde{J}$, to produce a weighted average of the mixed control-noise term and the quadratic control term. We minimize this loss via Stochastic Gradient Descent (SGD). The resulting Variational RL with learn rate γ is an incremental update of the form

$$\Theta^{(k+1)} = \Theta^{(k)} - \gamma \nabla_{\Theta} L. \tag{9}$$

We contrast this work to prior work that also use variational optimization to approximate optimal probability measures, as in [19]. There, the authors obtain a time-varying policy of step-functions that results in parameter update-rules requiring inversion of a jacobian. Our proposed approach instead uses an arbitrary non-linear feedback policy and produces a SGD-based minimization that can leverage well-known backprop-based algorithms such as ADA-Grad [20] and ADAM [21].

Although the state may be described by an infinite-dimensional vector in a Hilbert space, many physical realizations of actuation are defined on finite subspaces. The above derivation keeps Φ as mapping into the Hilbert space, insinuating that the actuation may be distributed everywhere and infinite-dimensional. However, the goal of this work is to ultimately use finite-action policy networks to control eq. (2). As such, we refine Φ as

$$\Phi(t,X,\mathbf{x};\Theta^{(k)}) = \mathbf{m}(\mathbf{x})^{\top} \varphi(X;\Theta^{(k)}), \tag{10}$$

where $\varphi(X;\Theta^{(k)}) : \mathscr{H} \to \mathbb{R}^N$ is a finite policy network with N control outputs representing N distributed (or boundary) actuators. The function $\mathbf{m}(\mathbf{x}) : D \to \mathbb{R}^N \times \mathscr{H}$ represents the effect of the finite actuation on the infinite-dimensional field, where D is the domain of the finite spatial region. Some examples of $\mathbf{m}(\mathbf{x})$ are Gaussians-like exponential functions with mean centered at the actuator location (for distributed control) and indicator functions (for boundary control).

3 Algorithm and Network Architecture

The above derivation provides a mathematical framework for updating the weights of a policy network in a RL setting. In order to implement it as an algorithm, data must be generated either from a physics-based or data-based model, or from interactions with a real system. Notice that since the only term

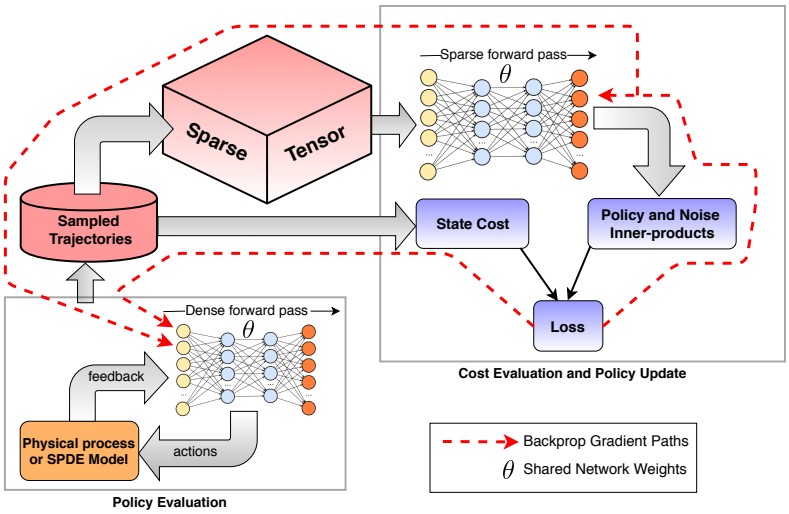


Figure 2: Block diagram of computational graph for the IDVRL algorithm.

from the dynamics to appear in eqs. (7) and (8) is the Cylindrical noise term $\mathrm{d}W$, there is no need to have an explicit SPDE model. As a result, any black-box methods that incorporate spatio-temporal stochasticity can be used to generate sample trajectories of the system.

The above derivation introduces a unique problem for our proposed reinforcement learning framework that has not been addressed in prior work. Each inner product in Hilbert space in eqs. (7) and (8) represents a spatial integration over a finite region D. To the knowledge of the authors, integration over a policy network has not been attempted to date. However in this work, we integrate spatially over the input to the network. Consider the inner product indicated as *PolicyInnerProduct*. The representation of this inner product as a spatial integration takes the form

$$\begin{aligned}\int_0^T \left\langle \Phi(X,\mathbf{x};\Theta^{(k)}),\Phi(X,\mathbf{x};\Theta^{(k)})\right\rangle \mathrm{d}t &= \int_0^T \iint_D \varphi(X(t,x,y);\Theta^{(k)})^\top M(x,y)\varphi(X(t,x,y);\Theta^{(k)})\mathrm{d}x\mathrm{d}y\mathrm{d}t \\ &= \int_0^T \sum_{j=1}^{\infty} \varphi(X(e_j);\Theta^{(k)})^\top M(e_j)\varphi(X(e_j);\Theta^{(k)})\mathrm{d}t, \end{aligned} \tag{11}$$

where $D \subseteq \mathbb{R}^2$ is the problem domain, $\{e_j \in \mathscr{H} : j = 0,1,2,\dots\}$ forms an orthonormal basis over $\mathscr{H}$, and $M(\mathbf{x}) = \mathbf{m}(\mathbf{x})\mathbf{m}(\mathbf{x})^\top$. After discretization on a 2D grid, the basis becomes a finite set $\{e_j \in \mathbb{R}^{J^2} : j = 0,1,2,\dots\}$, where each element is a one-hot vector. Thus, evaluating the spatial integral is reduced to summing up forward passes through the policy network with each pixel considered individually. Note that this spatial integration approach is agnostic to choice of discretization scheme.

Spatially integrating over the policy network is a memory intensive task, where the storage becomes (J^2,J,J) for each sample over the time horizon. However, given that the basis elements of each (J,J) "image" have only one activated "pixel", the resulting tensor is tremendously sparse. As such, each layer's activation can be computed with a sparse matrix multiplication, resulting in what we call a *SparseForwardPass* method that is not memory intensive for relatively large 2D problems. This can be applied to both FNNs and CNNs. For CNNs, activation can be achieved by matrix multiplication with a Toeplitz matrix constructed from the filter coefficients [22].

A summary of our architecture is depicted in fig. 2. A policy network with initialized weights is passed through a model or physical realization of the system to produce state trajectories, which are used to compute a state cost as well as a sparse tensor that is used to compute the inner products in eqs. (7) and (8) in a memory and time-efficient manner. Finally the loss is computed and passed to a gradient-based optimization algorithm. This approach is independent of specific policy network architecture used, which can often be problem dependent. In this work we used two different networks: a FNN for 1-dimensional (1D) SPDE and a CNN for 2-dimensional (2D) SPDE.

The resulting IDVRL algorithm is shown in algorithm 1, wherein subscript implies an element of the corresponding vector. The input terms are time horizon (T), number of iterations (K), number of rollouts (R), initial state (X_0), number of actuators (N), noise variance (ρ), time discretization

Algorithm 1 Infinite Dimensional Variational Reinforcement Learning

Function: $\Theta^* = \textbf{\textit{OptimizePolicyNetwork}}(T, K, R, X_0, N, \rho, \Delta t, \mu, \sigma_\mu, \Theta^{(0)})$
Compute $\mathbf{m}(\mathbf{x}), M(\mathbf{x}) \; \forall \; \mathbf{x} \in D$
for $k = 1$ to K **do**
  for $r = 1$ to R **do**
    for $t = 1$ to T **do**
      $\mathrm{d}W_t \leftarrow SampleNoise()$
      $X_t \leftarrow Propagate(X_{t-1}, \Theta^{(k)}, \mathrm{d}W_t)$ via eq. (2)
      $J_r \leftarrow J_r + StateCost(X_t)$
      $S_t \leftarrow SparseForwardPass(\Theta^{(k)}, X_t)$
      $N_t \leftarrow NoiseInnerProduct\big(S_t, \mathrm{d}W_t, \mathbf{m}(\mathbf{x})\big)$
      $P_t \leftarrow PolicyInnerProduct\big(S_t, M(\mathbf{x})\big)$
    end for
    $P,\ N \leftarrow Sum(P_t),\ Sum(N_t)$
    $\tilde{J}_r \leftarrow \tilde{J}(P, N, J_r)$
  end for
  $W \leftarrow ImportanceWeight(\tilde{J})$
  $L \leftarrow ComputeLoss(P, N, W)$ via eq. (8)
  $\Theta^{(k+1)} \leftarrow GradientOptimize(L, \Theta^{(k)})$
end for

(Δt), actuator locations (μ), actuator variance (σ_μ, for distributed control cases), and initial network parameters ($\Theta^{(0)}$). We note that the function $GradientOptimize(L, \Theta^{(k)})$ represents the update from eq. (9). As mentioned above, this is handled by any variant of SGD, which performs backpropagation through the network. The computational graph of the proposed algorithm has multiple backprop paths, as shown by the dotted red line in fig. 2. For more information on $SampleNoise()$, refer to [2, Chapter 10].

4 Simulation Results and Discussion

We applied the IDVRL to reaching tasks for several SPDEs in simulation in both distributed and boundary control settings. In each reaching task, the policy has to control the system to achieve a desired profile in certain parts of the spatial domain. These simulated experiments were developed via computational graphs implemented in Tensorflow [23] to leverage GPU parallelization for training as well as sparse linear algebra operations for *SparseForwardPass*. The data for training the policies was generated by simulating the SPDEs using centered finite-difference approximation for the spatial derivatives on a 1D or 2D grid and a semi-implicit Euler scheme for discretization of the time derivatives. For detailed explanation on these schemes, we refer the reader to [2, Chapters 3 and 10]. For 1D simulations, we used an Alienware laptop with an Intel Core i9-8950HK CPU @ 2.9GHz×12, 32 GB RAM and a NVIDIA GeForce GTX 1080 graphics card. On average, Tensorflow-GPU required around 16 minutes of training time for 1000 iterations. For the 2D simulation, we used Tensorflow-CPU, due to insufficiency of VRAM, which required around 12 hours of training time for 1000 iterations. The code and videos for these experiments are available online [3].

Figure 3 (a) and (d) depict the results of the IDVRL algorithm on a task of controlling the 1D heat SPDE with homogeneous Dirichlet boundary conditions. The goal of the task is to raise and maintain the temperature to $T = 1$ at regions around $x = 0.2$ and $x = 0.8$, and $T = 0.5$ at a region around $x = 0.5$. Figure 3a) shows the temperature contours of a single realization of the completed task and fig. 3d) shows the mean controlled and uncontrolled trajectories at the final time with a 2-σ variance shaded in the corresponding color. The boundary conditions fixed the endpoints to a temperature of $T = 0$, as shown.

Figure 3, (b) and (e) depict the results of the IDVRL algorithm on the task of controlling the 1D Burgers SPDE with non-homogeneous Dirichlet boundary conditions. In this task the goal is to reach a desired velocity in the medium at given locations. This is challenging given the nonlinear advection behavior of the system in addition to the pure diffusion behavior shown in the 1D heat SPDE task.

[3] Code: https://github.gatech.edu/eevans41/spde_explicit_feedback_RL, Video: https://youtu.be/6tmky59xhp4

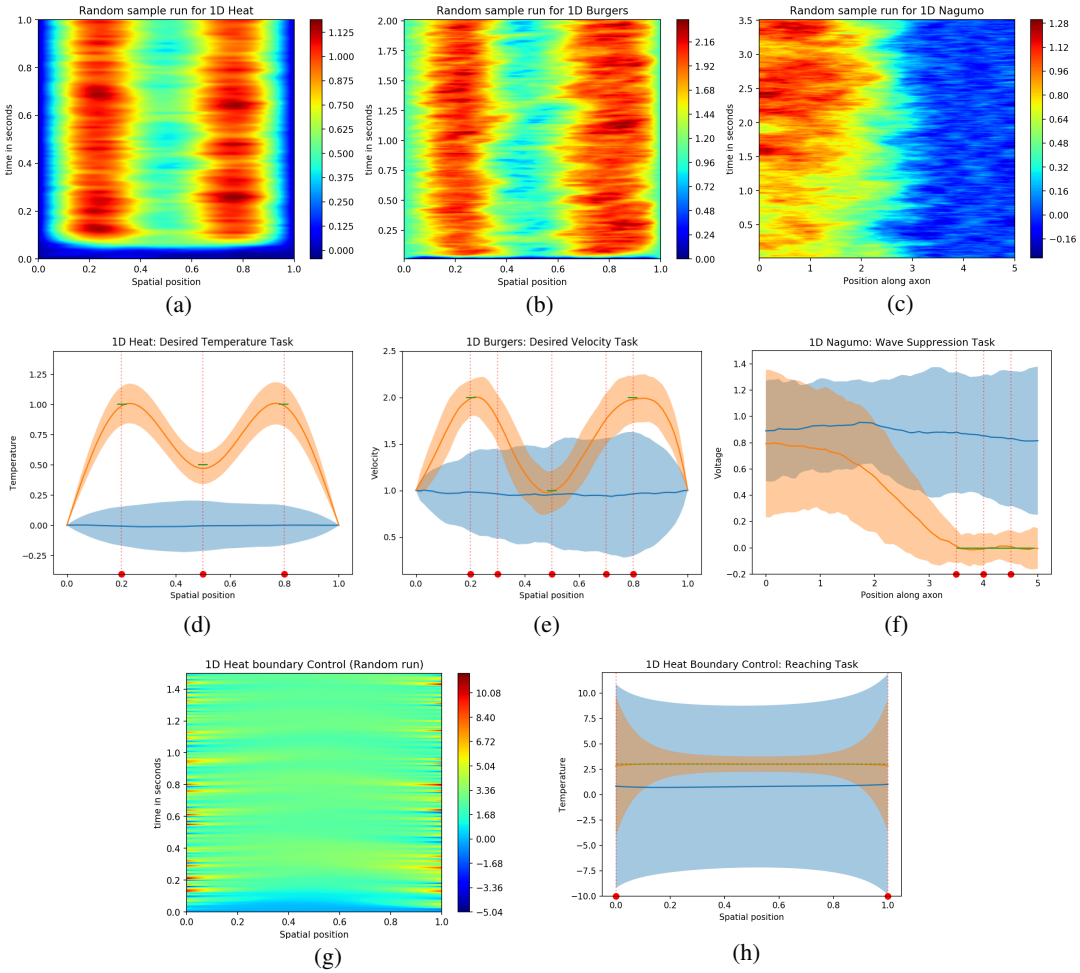
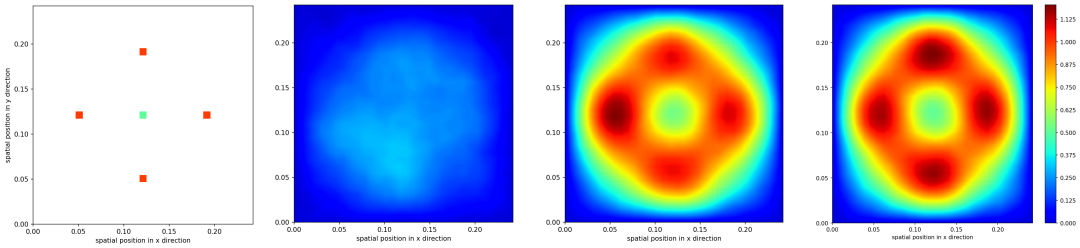


Figure 3: **Control of 1D SPDEs.** (a), (d), (g), (h) correspond to the Heat SPDE, (b), (e) to Burgers SPDE, and (c), (f) to Nagumo SPDE. In (d), (e), (f), (h) blue represents *mean uncontrolled profiles*, orange represents *mean controlled profiles* using the trained policy network, green represents *desired values* in certain spatial regions, and red represents *locations of actuator centers*. The mean and variance statistics are gathered over 200 rollouts. (a), (b), (c), (g) depict a randomly selected trial run to emphasize the presence of spatio-temporal stochasticity. (a-f) depict results for distributed control of SPDEs and (g-h) depict results for boundary control of a SPDE.

The advection-diffusion creates an apparent rightwards wave-front that must be accounted for by the policy network in order to achieve the task. Given the increased difficulty of the problem, we added actuators, as indicated by vertical red dotted lines. Despite the added actuators, the task remains severely under-actuated.

Figure 3, (c) and (f) depict the IDVRL algorithm on the task of controlling the 1D Nagumo SPDE with homogeneous Neumann boundary conditions. As noted earlier, the Nagumo SPDE represents voltage travelling across the axon of a neuron in the brain. The goal of this task is to suppress the voltage from travelling across the axon. Voltage near 1.0 indicates the voltage has travelled across, and in this suppression task, we seek to keep the voltage at the right end of the axon at $V = 0$. As shown in table 1, the Nagumo SPDE has a 3rd order nonlinearity. For this task, we supplied the system with only three actuators near the right end, where voltage must be suppressed.

For the next task, we scaled the IDVRL algorithm to two-dimensional problems. With this task we attempt to control the 2D Heat SPDE with homogeneous Dirichlet boundary conditions with a CNN policy network. The goal of this task it to raise the temperature in five regions. The desired temperature at the four outer regions is $T = 1$ and the desired temperature at the center region is $T = 0.5$. Figure 4 depicts a single realization of the controlled task under a significant amount of noise with five actuators.



(a) (b) (c) (d)

Figure 4: **Control of the 2D Heat SPDE**. (a) shows the desired profile patches and actuator locations for the reaching task. The next three plots show time snapshots from a randomly selected instance of an optimized policy applied to the system. (b) shows the start profile(b), (c) shows half-way through, and (d) shows the end profile. The color-bar depicts the range of temperatures in the simulated field.

In contrast to the previous tasks where actuators are distributed in the field, Figure 3h) depicts a *boundary* control task, where the actuator controls the boundary condition. The Radon-Nikodym derivative exists for the case of boundary control of semi-linear SPDEs with boundary noise [8], and we demonstrate that our method similarly extends to this case. The task here is similar to the first case, where the policy network is tasked with reaching a desired value of $T = 3$.

We invite the interested reader to refer to our supplementary material for specific details on each of our simulations such as cost functions, hyper-parameter values, neural network parameters and videos comparing controlled and uncontrolled SPDEs.

Throughout our simulated experiments, especially for distributed control tasks, we found that the algorithm is not sensitive to the majority of our parameters. We noted that a useful heuristic in applying the algorithm to new problems without having to tune the parameters was to ensure that the starting loss function was not very close to zero (i.e. 1e-10). Despite a large variance of noise that we typically applied to our systems ($\rho = 10$), the optimization algorithm was able to converge in under 1000 iterations for 1D problems and under 2000 iterations for 2D problems.

On the whole, even though injecting higher variance noise into the system inherently makes the control task much more challenging, high variance noise is useful in our algorithm for exploration over rollouts at each iteration. As such, there is an inverse relationship for a given convergence behavior between variance in the noise and number of rollouts.

There are also several interesting behaviors that the IDVRL algorithm demonstrates. First, we noticed that often times throughout optimization, the loss would decrease as desired, but state cost would temporarily increase, before decreasing more dramatically after some number of iterations. This indicates that there may not be a strictly proportional relationship between loss function and state cost. Indeed a lower state cost implies that the task is being accomplished, yet a trend of decreasing loss function indicated that when there was a temporary increase in state cost, the IDVRL algorithm may have been pushing the network parameters out of a local minimum towards better task performance in later iterations. These trends, depicted in fig. A1, indicate that the IDVRL algorithm may perform well on experiments outside the ones considered in this paper.

5 Conclusion and Future Directions

This work presents a variational reinforcement learning algorithm for the distributed and boundary control of infinite dimensional stochastic systems. The optimization method was derived in Hilbert spaces, thereby avoiding the need to depend on specific descretization schemes to realize the algorithm. The resulting algorithm requires only an actuation model and therefore is mostly model-free. The algorithm was demonstrated on five simulated experiments including 1D and 2D with both distributed and boundary type actuation.

In future work the authors will investigate provable convergence properties for IDVRL based on [24], and will implement the algorithm on some SPDEs described in this paper such as the Stochastic Navier-Stokes equation using state-of-the art CFD solvers. The authors also plan to investigate second-order SPDEs such as the Euler-Bernoulli equation which has been used to investigate the dynamics of tentacle-like soft continuum robots [25].

Acknowledgments

This work was supported by Amazon AWS and NSF CMMI #1662523. Ethan N. Evans was supported by the SMART scholarship, Marcus A. Periera was supported by Komatsu, and George I. Boutselis was supported by the A. S. Onasis Foundation.

References

[1] L. Bouten, M. Guta, and H. Maassen. Stochastic schrödinger equations. *Journal of Physics A: Mathematical and General*, 37(9):3189, 2004.

[2] G. J. Lord, C. E. Powell, and T. Shardlow. *An Introduction to Computational Stochastic PDEs*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2014. doi: 10.1017/CBO9781139017329.

[3] S. N. Gomes, S. Kalliadasis, D. T. Papageorgiou, G. A. Pavliotis, and M. Pradas. Controlling roughening processes in the stochastic kuramoto–sivashinsky equation. *Physica D: Nonlinear Phenomena*, 348:33–43, 2017.

[4] J. Qi, S.-X. Tang, and C. Wang. Parabolic pde-based multi-agent formation control on a cylindrical surface. *International Journal of Control*, 92(1):77–99, 2019.

[5] E. Aidman, V. Ivancevic, and A. Jennings. A coupled reaction-diffusion field model for perception-action cycle with applications to robot navigation. *International Journal of Intelligent Defence Support Systems*, 1(2):93–115, 2008.

[6] J. C. Ryu, F. C. Park, and Y. Y. Kim. Mobile robot path planning algorithm by equivalent conduction heat flow topology optimization. *Structural and Multidisciplinary Optimization*, 45 (5):703–715, 2012.

[7] G. Ferrari-Trecate, A. Buffa, and M. Gati. Analysis of coordination in multi-agent systems through partial difference equations. *IEEE Transactions on Automatic Control*, 51(6):1058–1063, 2006.

[8] G. Da Prato, A. Debussche, and R. Temam. Stochastic burgers' equation. *Nonlinear Differential Equations and Applications NoDEA*, 1(4):389–402, 1994.

[9] G. Fabbri, F. Gozzi, and A. Swiech. *Stochastic Optimal Control in Infinite Dimensions - Dynamic Programming and HJB Equations*. Number 82 in Probability Theory and Stochastic Modelling. Springer, Jan. 2017. URL https://hal-amu.archives-ouvertes.fr/hal-01505767. OS.

[10] Y. Lou, G. Hu, and P. D. Christofides. Model predictive control of nonlinear stochastic pdes: Application to a sputtering process. In *2009 American Control Conference*, pages 2476–2483. IEEE, 2009.

[11] G. D. Prato and A. Debussche. Control of the stochastic burgers model of turbulence. *SIAM Journal on Control and Optimization*, 37(4):1123–1149, 1999. doi:10.1137/S0363012996311307. URL http://dx.doi.org/10.1137/S0363012996311307.

[12] S. J. Moura and H. K. Fathy. Optimal boundary control of reaction–diffusion partial differential equations via weak variations. *Journal of Dynamic Systems, Measurement, and Control*, 135(3): 034501, 2013.

[13] J. Morton, A. Jameson, M. J. Kochenderfer, and F. Witherden. Deep dynamical modeling and control of unsteady fluid flows. In *Advances in Neural Information Processing Systems*, pages 9258–9268, 2018.

[14] J. Rabault, M. Kuchta, A. Jensen, U. Réglade, and N. Cerardi. Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control. *Journal of Fluid Mechanics*, 865:281–302, 2019.

[15] K. Bieker, S. Peitz, S. L. Brunton, J. N. Kutz, and M. Dellnitz. Deep model predictive control with online learning for complex physical systems. *arXiv preprint arXiv:1905.10094*, 2019.

[16] E. Theodorou and E. Todorov. Relative entropy and free energy dualities: Connections to path integral and kl control. In *the Proceedings of IEEE Conference on Decision and Control*, pages 1466–1473, Dec 2012. doi:10.1109/CDC.2012.6426381.

[17] M. J. Wainwright, M. I. Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

[18] E. Theodorou. Nonlinear stochastic control and information theoretic dualities: Connections, interdependencies and thermodynamic interpretations. *Entropy*, 17(5):3352–3375, 2015.

[19] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou. Aggressive driving with model predictive path integral control. *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1433–1440, 2016.

[20] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[21] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[22] K. Chellapilla, S. Puri, and P. Simard. High performance convolutional neural networks for document processing. 2006.

[23] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

[24] E. Zhou and J. Hu. Gradient-based adaptive stochastic search for non-differentiable optimization. *IEEE Transactions on Automatic Control*, 59(7):1818–1832, 2014.

[25] F.-F. Jin and B.-Z. Guo. Lyapunov approach to output feedback stabilization for the euler–bernoulli beam equation with boundary input disturbance. *Automatica*, 52:95–102, 2015.

[26] G. Da Prato and J. Zabczyk. *Stochastic Equations in Infinite Dimensions*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2014. ISBN 9780521385299. URL `https://books.google.com/books?id=Sid6pwAACAAJ`.

[27] G. D. Prato and A. Debussche. Control of the stochastic burgers model of turbulence. *SIAM Journal on Control and Optimization*, 37(4):1123–1149, 1999. doi:10.1137/S0363012996311307. URL `http://dx.doi.org/10.1137/S0363012996311307`.

Appendix A

A1 Q-Wiener and Cylindrical noise

The Q-Wiener noise defines a generalization of the standard Brownian motion for fields. The cylindrical noise which is used in our paper is a specific case of a Q-Wiener process. Below we give an important property of the aforementioned noise profiles which makes a connection to the standard Wiener noise. Formal definitions can be bound in [26, Chapter 4].

Proposition A1.1. *Let $\{e_i\}_{i=1}^{\infty}$ be a complete orthonormal system for the Hilbert Space U such that $Qe_i = \lambda_i e_i$. Here, λ_i is the eigenvalue of Q that corresponds to eigenvector e_i. Then, a Q-Wiener process $W(t) \in U$ has the following expansion:*

$$W(t) = \sum_{j=1}^{\infty} \sqrt{\lambda_j}\beta_j(t)e_j, \tag{A1}$$

where $\beta_j(t)$ are real valued Brownian motions that are mutually independent on $(\Omega, \mathscr{F}, \mathbb{P})$.

We note that when $\lambda_j = 1 \ \forall j \in \mathbb{N}$, $W(t)$ corresponds to a cylindrical Wiener process (space-time white noise). In this case, the series in (A1) converges in another Hilbert space $U_1 \supset U$, when the inclusion $\iota : U \to U_1$ is Hilbert-Schmidt. For more details see [26].

A2 Girsanov's theorem and the Randon-Nikodym derivative

Theorem A2.1 (Girsanov)**.** *Let Ω be a sample space with a σ-algebra $\mathscr{F}$. Consider the following H-valued stochastic processes:*

$$\mathrm{d}X = \big(\mathscr{A}X + F(t,X)\big)\mathrm{d}t + G(t,X)\mathrm{d}W(t), \tag{A2}$$

$$\mathrm{d}\tilde{X} = \big(\mathscr{A}\tilde{X} + F(t,\tilde{X})\big)\mathrm{d}t + \tilde{B}(t,\tilde{X})\mathrm{d}t + G(t,\tilde{X})\mathrm{d}W(t), \tag{A3}$$

where $X(0) = \tilde{X}(0) = x$ and $W \in U$ is a cylindrical Wiener process with respect to measure $\mathbb{P}$. Moreover, let Γ be a set of continuous-time, infinite-dimensional trajectories in the time interval $[0,T]$. Now the probability law of X will be defined as $\mathscr{L}(\Gamma) := \mathbb{P}(\omega \in \Omega | X(\cdot,\omega) \in \Gamma)$. Similarly, the law of $\tilde{X}$ is defined as $\tilde{\mathscr{L}}(\Gamma) := \mathbb{P}(\omega \in \Omega | \tilde{X}(\cdot,\omega) \in \Gamma)$. Then

$$\tilde{\mathscr{L}}(\Gamma) = \mathbb{E}_{\mathbb{P}}\big[\exp\big(\int_0^T \langle \psi(s), dW(s)\rangle_U - \tfrac{1}{2}\int_0^T ||\psi(s)||_U^2 ds\big) | X(\cdot) \in \Gamma\big], \tag{A4}$$

where we have defined $\psi(t) := G^{-1}\big(t,X(t)\big)\tilde{B}\big(t,X(t)\big) \in U_0$ and assumed $\mathbb{E}_{\mathbb{P}}\big[e^{\frac{1}{2}\int_0^T ||\psi(t)||^2 \mathrm{d}t}\big] < +\infty$.

The proof of Girsanov's theorem can be found in [27]. It follows that the *Randon-Nikodym* (RN) derivative between measures $\mathscr{L}(\cdot)$ and $\tilde{\mathscr{L}}(\cdot)$ of the different dynamical systems defined in (A3), is given by

$$\frac{\mathrm{d}\tilde{\mathscr{L}}}{\mathrm{d}\mathscr{L}} = \exp\big(\int_0^T \langle \psi(s), dW(s)\rangle_U - \frac{1}{2}\int_0^T ||\psi(s)||_U^2 ds\big), \tag{A5}$$

In the main paper, we use the RN derivative for the case where $\mathscr{L}(\cdot)$, $\tilde{\mathscr{L}}(\cdot)$ correspond to uncontrolled and controlled trajectories, respectively, with $\psi(\cdot)$ being properly defined.

A3 Derivation of Variational Minimization and Loss Function

This section explains the steps to arrive at eqs. (5), (7) and (8) from the main paper.

$$
\begin{aligned}
\Theta^* &= \underset{\Theta}{\operatorname{argmin}} D_{KL}(\mathscr{L}^* || \tilde{\mathscr{L}}) \\
&= \underset{\Theta}{\operatorname{argmin}} \left[\int \log\Big(\frac{\mathrm{d}\mathscr{L}^*}{\mathrm{d}\tilde{\mathscr{L}}}\Big) \mathrm{d}\mathscr{L}^* \right] \\
&= \underset{\Theta}{\operatorname{argmin}} \left[\int \log\Big(\frac{\mathrm{d}\mathscr{L}^*}{\mathrm{d}\mathscr{L}} \frac{\mathrm{d}\mathscr{L}}{\mathrm{d}\tilde{\mathscr{L}}}\Big) \mathrm{d}\mathscr{L}^* \right] \\
&= \int \log\Big(\frac{\mathrm{d}\mathscr{L}^*}{\mathrm{d}\mathscr{L}}\Big) \mathrm{d}\mathscr{L}^* + \underset{\Theta}{\operatorname{argmin}} \left[\int \log\Big(\frac{\mathrm{d}\mathscr{L}}{\mathrm{d}\tilde{\mathscr{L}}}\Big) \mathrm{d}\mathscr{L}^* \right] \\
&= \underset{\Theta}{\operatorname{argmin}} \left[\int \log\Big(\frac{\mathrm{d}\mathscr{L}}{\mathrm{d}\tilde{\mathscr{L}}}\Big) \frac{\mathrm{d}\mathscr{L}^*}{\mathrm{d}\mathscr{L}} \frac{\mathrm{d}\mathscr{L}}{\mathrm{d}\tilde{\mathscr{L}}} \mathrm{d}\tilde{\mathscr{L}} \right] = \underset{\Theta}{\operatorname{argmin}}\ L
\end{aligned} \tag{A6}
$$

Now,

$$
L = \mathbb{E}_{\tilde{\mathscr{L}}} \left[\log\Big(\frac{\mathrm{d}\mathscr{L}}{\mathrm{d}\tilde{\mathscr{L}}}\Big) \frac{\mathrm{d}\mathscr{L}^*}{\mathrm{d}\mathscr{L}} \frac{\mathrm{d}\mathscr{L}}{\mathrm{d}\tilde{\mathscr{L}}} \mathrm{d}\tilde{\mathscr{L}} \right]
$$

Substituting eq. (6), the log goes away because of the exponential,

$$
\begin{aligned}
L = \mathbb{E}_{\tilde{\mathscr{L}}} \Bigg[\Bigg(& -\sqrt{\rho} \int_0^T \Big\langle \Phi(t, X, \mathbf{x}; \Theta^{(k)}), \mathrm{d}W(t) \Big\rangle \\
& - \frac{1}{2}\rho \int_0^T \Big\langle \Phi(t, X, \mathbf{x}; \Theta^{(k)}), \Phi(t, X, \mathbf{x}; \Theta^{(k)}) \Big\rangle \mathrm{d}t \Bigg) \frac{\mathrm{d}\mathscr{L}^*}{\mathrm{d}\mathscr{L}} \frac{\mathrm{d}\mathscr{L}}{\mathrm{d}\tilde{\mathscr{L}}} \mathrm{d}\tilde{\mathscr{L}} \Bigg]
\end{aligned}
$$

Evaluating, $\frac{\mathrm{d}\mathscr{L}^*}{\mathrm{d}\mathscr{L}} \frac{\mathrm{d}\mathscr{L}}{\mathrm{d}\tilde{\mathscr{L}}}$ separately, we have,

$$
\begin{aligned}
\frac{\mathrm{d}\mathscr{L}^*}{\mathrm{d}\mathscr{L}} \frac{\mathrm{d}\mathscr{L}}{\mathrm{d}\tilde{\mathscr{L}}} &= \frac{\exp(-\rho J)}{\mathbb{E}_{\mathscr{L}}\big[\exp(-\rho J)\big]} \exp\Bigg(-\sqrt{\rho} \int_0^T \Big\langle \Phi(t, X, \mathbf{x}; \Theta^{(k)}), \mathrm{d}W(t) \Big\rangle \\
&\qquad\qquad - \frac{1}{2}\rho \int_0^T \Big\langle \Phi(t, X, \mathbf{x}; \Theta^{(k)}), \Phi(t, X, \mathbf{x}; \Theta^{(k)}) \Big\rangle \mathrm{d}t \Bigg) \\
&= \frac{\exp(-\rho \tilde{J})}{\mathbb{E}_{\mathscr{L}}\big[\exp(-\rho J)\big]},
\end{aligned}
$$

where $\tilde{J}$ is defined in eq. (8). Similarly, we can use importance sampling for the expectation in the denominator using eq. (6) as,

$$
\begin{aligned}
\mathbb{E}_{\mathscr{L}}\big[\exp(-\rho J)\big] &= \mathbb{E}_{\tilde{\mathscr{L}}}\left[\frac{\mathrm{d}\mathscr{L}}{\mathrm{d}\tilde{\mathscr{L}}} \exp(-\rho J) \right] = \mathbb{E}_{\tilde{\mathscr{L}}}\big[\exp(-\rho \tilde{J})\big] \\
\therefore \frac{\mathrm{d}\mathscr{L}^*}{\mathrm{d}\mathscr{L}} \frac{\mathrm{d}\mathscr{L}}{\mathrm{d}\tilde{\mathscr{L}}} &= \frac{\exp(-\rho \tilde{J})}{\mathbb{E}_{\tilde{\mathscr{L}}}\big[\exp(-\rho \tilde{J})\big]}
\end{aligned}
$$

Putting all of this together, we get the required form of eq. (7) as,

$$
\begin{aligned}
L = \mathbb{E}_{\tilde{\mathscr{L}}} \Bigg[& \frac{\exp(-\rho \tilde{J})}{\mathbb{E}_{\tilde{\mathscr{L}}}\big[\exp(-\rho \tilde{J})\big]} \Bigg(-\sqrt{\rho} \int_0^T \Big\langle \Phi(t, X, \mathbf{x}; \Theta^{(k)}), \mathrm{d}W(t) \Big\rangle \\
& - \frac{1}{2}\rho \int_0^T \Big\langle \Phi(t, X, \mathbf{x}; \Theta^{(k)}), \Phi(t, X, \mathbf{x}; \Theta^{(k)}) \Big\rangle \mathrm{d}t \Bigg) \Bigg]
\end{aligned}
$$

A4 Additional Information on Simulations

Following are some details on each of our simulations which will help in reproducing our results.

A4.1 1D Heat SPDE distributed and boundary control

A4.1.1 Distributed Control

The heat SPDE in 1D is given by

$$\begin{aligned} dh(t,x) &= \varepsilon h_{xx}(t,x)dt + G(t,h)\big(\mathbf{m}(\mathbf{x})^\top \varphi(h;\Theta)dt + \sigma dW(t)\big) \\ h(0,x) &= h_0(x) \end{aligned} \tag{A7}$$

where ε is the thermal diffusivity parameter, which was set to 1 for our experiments. The task is to achieve a desired temperature profile at 3 regions along the spatial domain. At the center of these regions are actuators. The three-actuator-based control is achieved by setting $\mathbf{m}(\mathbf{x})^\top = [m_1(\mathbf{x}), m_2(\mathbf{x}), m_3(\mathbf{x})]^\top$ and $G(t,h)$ to an identity operator. The actuator dynamics $m(\mathbf{x})$ are modelled by Gaussian-like exponential functions with the means co-located with the actuator locations at: $\mu = \big[\mu_1, \mu_2, \mu_3\big] = \big[0.2a, 0.5a, 0.8a\big]$ and the variance of the effect of each actuator on nearby field states given by $\sigma_l^2 = (0.1a)^2, \forall l = 1,2,3$. The cost function considered for the experiments is defined as

$$J := \sum_t \sum_x \kappa\big(h_{\text{actual}}(t,x) - h_{\text{desired}}(t,x)\big)^2 \cdot \mathbb{1}_S(x) \tag{A8}$$

where $S := \cup_{i=1}^3 S_i$ and the indicator function $\mathbb{1}_S(x)$ is defined as

$$\mathbb{1}_S(x) := \begin{cases} 1, & \text{if } x \in S \\ 0, & \text{otherwise} \end{cases} \tag{A9}$$

where,

$$\begin{aligned} &S_1 = \{x \in D \mid x \in [0.18a, 0.22a]\} \text{ is the region of the spatial domain on the left,} \\ &S_2 = \{x \in D \mid x \in [0.48a, 0.52a]\} \text{ is the region of the spatial domain in the center,} \\ &S_3 = \{x \in D \mid x \in [0.78a, 0.82a]\} \text{ is the region of the spatial domain on the right.} \end{aligned} \tag{A10}$$

The non-linear policy $\varphi(h;\Theta)$ was chosen to be a FNN with 2 hidden layers of 64 neurons each and ReLU activations. The network was trained using the ADAM optimizer for 1000 iterations with 200 trajectories sampled from the Heat SPDE model per iteration. Each trajectory was 1.0 seconds long with $\Delta t = 0.01$ seconds.

These parameters were run over 200 trials to obtain the convergence results depicted in fig. A1. These plots demonstrate that even though the state cost is not monotonically decreasing, the loss has a monotonic-like decreasing behavior. As described in the main text, this demonstrates that IDVRL may be pushing the state cost out of local minima. Additionally, the variance in the algorithm decreases over iterations.

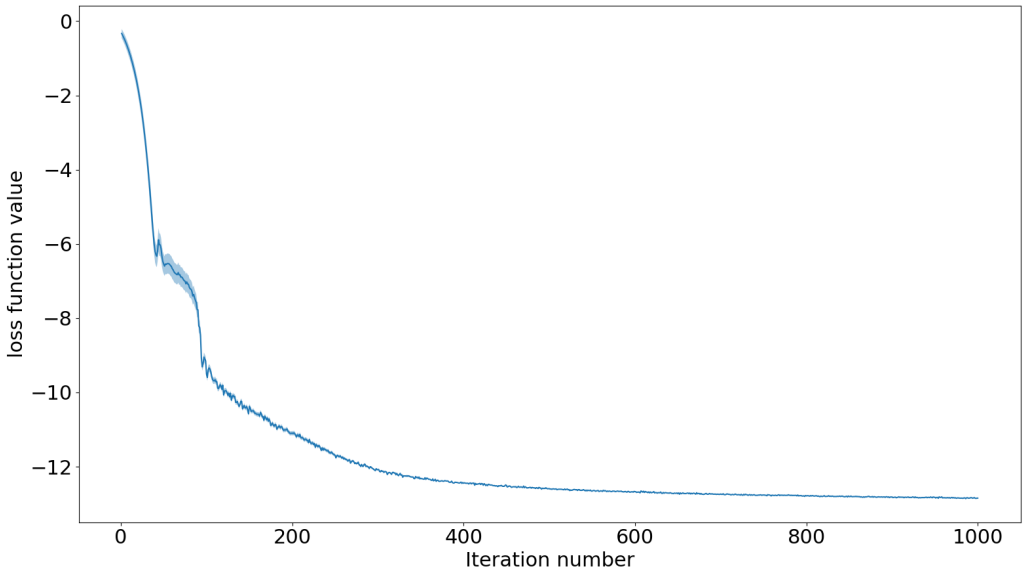
A4.1.2 Boundary Control

In the boundary control case, we make use of the 1D stochastic heat equation

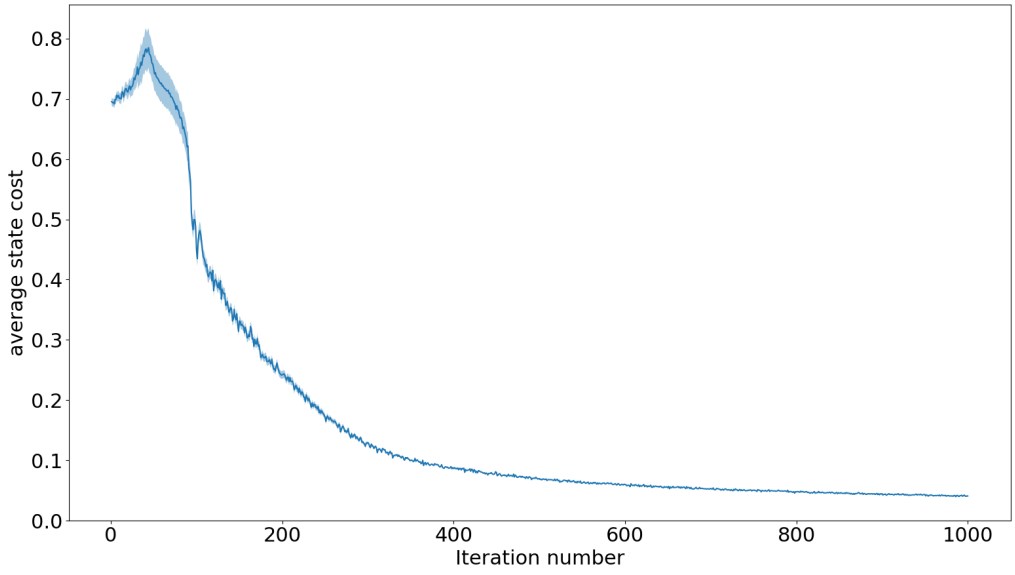
$$\begin{aligned} dh(t,x) &= \varepsilon h_{xx}(t,x)dt + G(t,h)\big(\mathbf{m}(\mathbf{x})^\top \varphi(h;\Theta)dt + \sigma dW(t)\big) \\ h(0,x) &= h_0(x) \end{aligned} \tag{A11}$$

For Dirichlet and Neumann boundary conditions we have $h(t,x) = \gamma(x)$, $\forall x \in \partial O$ and $h_x(t,x) = \gamma(x)$, $\forall x \in \partial O$, respectively. In our 1-D boundary control example, we set $\varepsilon = 1$, $\rho = 10$, $h_x(t,0) = u_1(t) + \frac{1}{\sqrt{\rho}} dW(t)$ and $h_x(t,a) = u_2(t) + \frac{1}{\sqrt{\rho}} dW(t)$. In the infinite-dimensional Hilbert space formulation, these boundary conditions are incorporated into the $G(t,h)\,\varphi(h;\Theta)$ term. In this case, $\mathbf{m}(\mathbf{x})^\top = [m_1(\mathbf{x}), m_2(\mathbf{x})]^\top$, where each $m(\mathbf{x})$ is simply given by an indicator function and $G(t,h)$ is an identity operator. The cost function is given by eq. (A8) with $S = D$ and $h_{desired}(t,x) = 3$.

For 1D boundary control, the non-linear policy $\varphi(h;\Theta)$ was chosen to be a FNN with 2 hidden layers of 64 neurons each and ReLU activations. The network was trained using the ADAM optimizer for 1000 iterations with 200 trajectories sampled from the Heat SPDE model per iteration. Each trajectory was 1.5 seconds long with $\Delta t = 0.01$ seconds.



(a)



(b)

Figure A1: **Convergence of IDVRL Policy for the 1D Heat SPDE**. The plots show (a) convergence of the loss function and (b) convergence of the state cost for the IDVRL algorithm over 200 trials of 1000 iterations each for a FNN network.

A4.2 2D Heat SPDE distributed control

The 2D Heat SPDE with homogeneous Dirichlet boundary conditions given by

$$\begin{aligned} dh(t,x,y) &= \varepsilon h_{xx}(t,x,y)dt + \varepsilon h_{yy}(t,x,y)dt + G(t,h)\left(\mathbf{m}(\mathbf{x})^\top \varphi(h;\Theta)dt + \sigma dW(t)\right), \\ h(t,0,y) &= h(t,a,y) = h(t,x,0) = h(t,x,a) = 0, \\ h(0,x,y) &\sim \mathcal{N}(h_0;0,\sigma_0), \end{aligned} \tag{A12}$$

where the parameter ε is the so called thermal diffusivity, which governs how quickly the initial temperature profile diffuses across the spatial domain. Equation (A12) considers the scenario of controlling a metallic plate to a desired temperature profile using 5 actuators distributed across the plate. The edges of the plate are always held at constant temperature of 0 degrees Celsius. The parameter a is the length of the sides of the square plate, for which we use $a = 0.25$ meters.

The 5 actuator-based control is achieved by setting $\mathbf{m}(\mathbf{x})^\top = [m_1(\mathbf{x}), m_2(\mathbf{x}), m_3(\mathbf{x}), m_4(\mathbf{x}), m_5(\mathbf{x})]^\top$ and $G(t,h)$ to an identity operator. The actuator dynamics $m(\mathbf{x})$ are modelled by Gaussian-like exponential functions with the means co-located with the actuator locations at: $\mu = \big[\mu_1, \mu_2, \mu_3, \mu_4, \mu_5\big] = \big[(0.2a, 0.5a), (0.5a, 0.2a), (0.5a, 0.5a), (0.5a, 0.8a), (0.8a, 0.5a)\big]$ and the variance of the effect of each actuator on nearby field states given by $\sigma_l^2 = (0.1a)^2$, $\forall l = 1, \ldots, 5$. The spatial domain is discretized by dividing the x and y domains into $J = 32$ points each creating a grid of 32×32 spatial locations on the plate surface. The resulting $m_l(\mathbf{x})$ has the form

$$m_{l,j}\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \exp\left\{-\frac{1}{2}\left(\begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} \mu_{l,x} \\ \mu_{l,y} \end{bmatrix}\right)^\top \begin{bmatrix} \sigma_l^2 & 0 \\ 0 & \sigma_l^2 \end{bmatrix} \left(\begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} \mu_{l,x} \\ \mu_{l,y} \end{bmatrix}\right)\right\},$$
$$\forall\, j = 1, \ldots, J,\ \ l = 1, \ldots, 5$$

For our simulations, we use a semi-implicit forward Euler discretization scheme for time and central difference for the 2^{nd} order spatial derivatives h_{xx} and h_{yy}. We used time discretization $\Delta t = 0.02s$, simulation time horizon $T = 1.0s$ and thermal diffusivity $\varepsilon = 1.0$. The cost function considered for the experiments is defined as

$$J := \sum_t \sum_x \sum_y \kappa\big(h_{\text{actual}}(t,x,y) - h_{\text{desired}}(t,x,y)\big)^2 \cdot \mathbb{1}_S(x,y)$$

where $S := \cup_{i=1}^5 S_i$ and the indicator function $\mathbb{1}_S(x,y)$ is defined similar to eq. (A9) as

$$\mathbb{1}_S(x,y) := \begin{cases} 1, & \text{if } (x,y) \in S \\ 0, & \text{otherwise,} \end{cases}$$

where,

$S_1 = \{(x,y) \in D \mid x \in [0.48a, 0.52a] \text{ and } y \in [0.48a, 0.52a]\}$ is in the central region,
$S_2 = \{(x,y) \in D \mid x \in [0.22a, 0.18a] \text{ and } y \in [0.48a, 0.52a]\}$ is the left-mid region,
$S_3 = \{(x,y) \in D \mid x \in [0.82a, 0.78a] \text{ and } y \in [0.48a, 0.52a]\}$ is the right-mid region,
$S_4 = \{(x,y) \in D \mid x \in [0.48a, 0.52a] \text{ and } y \in [0.18a, 0.22a]\}$ is in the top-central region,
$S_5 = \{(x,y) \in D \mid x \in [0.48a, 0.52a] \text{ and } y \in [0.78a, 0.82a]\}$ is in the bottom-central region.

In addition $h_{\text{desired}}(t,x,y) = 0.5^\circ C$ for $(x,y) \in S_1$ and $h_{\text{desired}}(t,x,y) = 1.0^\circ C$ for $(x,y) \in \cup_{i=2}^5 S_i$ and the scaling parameter $\kappa = 10^{-3}$.

Since the domain is 2D, the inputs to the non-linear policy $\varphi(h;\Theta)$ are image-like data after discretization, and therefore the policy was chosen to be a CNN. The description of the network architecture is given in table A1. The network was trained using the ADAM optimizer for 1000 iterations with 50 trajectories sampled from the 2D Heat SPDE model per iteration. Each trajectory was 1.0 seconds long with $\Delta t = 0.02$.

Layer name	**Kernel size**	**# Filters (output size)**	**Stride**	**Padding type**	**Activation**
Input	-	1	-	-	-
Conv-1	4	5	2	VALID	ReLU
Max-pool-1	2	-	2	-	-
Conv-2	2	16	1	SAME	ReLU
Max-pool-2	2	-	2	-	-
Dense	-	5	-	-	Linear

Table A1: Description of CNN policy network for 2D Heat SPDE.

A4.3 1D Burgers SPDE distributed control

The 1D Burgers SPDE with non-homogeneous Dirichlet boundary conditions is given by

$$\begin{aligned} dh(t,x) + hh_x(t,x)dt &= \varepsilon h_{xx}(t,x)dt + G(t,h)\big(\mathbf{m}(\mathbf{x})^\top \varphi(h;\Theta)dt + \sigma dW(t)\big) \\ h(t,0) &= h(t,a) = 1.0 \\ h(0,x) &= 0,\ \forall x \in (0,a) \end{aligned} \tag{A13}$$

where the parameter ε is the viscosity of the medium. Equation (A13) considers a simple model of a 1D flow of a fluid in a medium with non-zero flow velocities at the two boundaries. The goal is to achieve and maintain a desired flow velocity profile at certain points along the spatial domain. As seen in the desired profile in fig. 3e in the main paper, there are 3 areas along the spatial domain with desired flow velocity such that the flow has to be accelerated, then decelerated, and then accelerated again while trying to overcome the stochastic forces and the dynamics governed by the Burgers SPDE. Similar to the experiments for the Heat SPDE, we consider $\mathbf{m}(\mathbf{x})^\top = [m_1(\mathbf{x}), m_2(\mathbf{x}), m_3(\mathbf{x}), m_4(\mathbf{x})]$ and $G(t,h)$ as an identity operator with the actuators behaving as Gaussian-like exponential functions with the means co-located with the actuator locations at: $\mu = \big[0.2a, 0.3a, 0.5a, 0.7a, 0.8a\big]$ and the spatial effect (variance) of each actuator given by $\sigma_l^2 = (0.1a)^2$, $\forall l = 1,\ldots,5$. The parameter $a = 1.0\,m$ is the length of the channel along which the fluid is flowing.

This spatial domain was discretized using a grid of 64 points. The numerical scheme used semi-implicit forward Euler discretization for time and central difference approximation for both the 1^{st} and 2^{nd} order derivatives in space. The 1^{st} order derivative terms in the advection term uu_x were evaluated at the current time instant while the 2^{nd} order spatial derivatives in the diffusion term u_{xx} were evaluated at the next time instant, hence the scheme is semi-implicit. Following are values of some other parameters used in our experiments: time discretization $\Delta t = 0.01$, total simulation time = $1.0\,s$, and the scaling parameter $\kappa = 100$. The cost function considered for the experiments is given by eq. (A8), where $S := \cup_{i=1}^{3} S_i$ and the indicator function $\mathbb{1}_S(x)$ is given by eq. (A9) with regions S_1, S_2, S_3 given by eq. (A10). In addition, $h_{\text{desired}}(t,x) = 2.0\,m/s$ for $x \in S_1 \cup S_3$, which is at the sides, and $h_{\text{desired}}(t,x) = 1.0\,m/s$ for $x \in S_2$, which is in the central region.

The non-linear policy $\varphi(h;\Theta)$ was chosen to be a FNN with 2 hidden layers of 64 neurons each and ReLU activations. The network was trained using the ADAM optimizer for 1000 iterations with 100 trajectories sampled from the Burgers SPDE model per iteration. Each trajectory was 2.0 seconds long with $\Delta t = 0.01$ seconds.

A4.4 1D Nagumo SPDE distributed control (Suppression Task)

The stochastic Nagumo equation with Neumann boundary conditions is given by

$$\begin{aligned} dh(t,x) &= \varepsilon h_{xx}(t,x)dt + h(t,x)\big(1-h(t,x)\big)\big(h(t,x)-\alpha\big)dt + G(t,h)\big(\mathbf{m}(\mathbf{x})^\top \varphi(h;\Theta)dt + \sigma dW(t)\big) \\ h_x(t,0) &= h_x(t,a) = 0 \\ h(0,x) &= \left(1 + \exp\left(-\frac{2-x}{\sqrt{2}}\right)\right)^{-1} \end{aligned} \tag{A14}$$

The parameter α determines the speed of a wave traveling down the length of the axon and ε the rate of diffusion. By simulating the deterministic Nagumo equation with $a = 5.0$, $\varepsilon = 1.0$ and $\alpha = -0.5$, we observed that after about 3.5 seconds, the wave completely propagates to the end of the axon. We consider $\mathbf{m}(\mathbf{x})^\top = [m_1(\mathbf{x}), m_2(\mathbf{x}), m_3(\mathbf{x})]$ and $G(t,h)$ as an identity operator with the actuators dynamics $m(\mathbf{x})$ modelled as Gaussian-like exponential functions with actuator centers (mean values) at $\mu = \big[0.7a, 0.8a, 0.9a\big]$ and the spatial effect (variance) of each actuator given by $\sigma_l^2 = (0.1a)^2$, for $l = 1,2,3$. The spatial domain was discretized using a grid of 64 points. The cost function for this experiment is defined as

$$J = \sum_t \sum_x \kappa\big(h_{\text{actual}}(t,x)\big)^2 \cdot \mathbb{1}_S(x)$$

where κ was chosen as 10^{-3}, and the indicator function $\mathbb{1}_S(x)$ is defined as in eq. (A9) with $S = [0.7a, 0.99a]$. The non-linear policy $\varphi(h;\Theta)$ was chosen to be a FNN with 2 hidden layers of 64 neurons each and ReLU activations. The network was trained using the ADAM optimizer for 1000 iterations with 50 trajectories sampled from the Nagumo SPDE model per iteration. Each trajectory was 3.5 seconds long and $\Delta t = 0.01$ seconds.