Solving high-dimensional Hamilton-Jacobi-Bellman Equations for Optimal Feedback Control via Adaptive Deep Learning Approach

Elisa Giesecke

Supervisors: Prof. Dr. Falk Hante, Prof. Dr. Daniel Walter

Humboldt-Universität zu Berlin
Mathematisch-Naturwissenschaftliche Fakultät
Institut für Mathematik

February 22, 2024

Outline

Outline

Motivation

Goal: designing feedback controllers for OCPs with non-linear ODEs

$\rightarrow$ solving Hamilton-Jacobi-Bellman equations

Motivation

Goal: designing feedback controllers for OCPs with non-linear ODEs
$\rightarrow$ solving Hamilton-Jacobi-Bellman equations

"curse of dimensionality"

Motivation

Goal: designing feedback controllers for OCPs with non-linear ODEs
$\rightarrow$ solving Hamilton-Jacobi-Bellman equations

"curse of dimensionality"

limited scalibility w. r. t. space dimension	restrictive structure of dynamical system	solution validity in local neighborhood	difficult verification of solution accuracy

Motivation

Goal: designing feedback controllers for OCPs with non-linear ODEs
$\rightarrow$ solving Hamilton-Jacobi-Bellman equations

"curse of dimensionality"

limited scalibility w. r. t. space dimension	restrictive structure of dynamical system	solution validity in local neighborhood	difficult verification of solution accuracy

$\rightarrow$ developing data-driven approaches based on neural networks

Contribution

Advancing adaptive deep learning approach from [Nakamura-Zimmerer et al., 2021]:

Contribution

Advancing adaptive deep learning approach from [Nakamura-Zimmerer et al., 2021]:

- value function approximator with exact final condition

Contribution

Advancing adaptive deep learning approach from [Nakamura-Zimmerer et al., 2021]:

- value function approximator with exact final condition
- residual network architecture

Contribution

Advancing adaptive deep learning approach from [Nakamura-Zimmerer et al., 2021]:

- value function approximator with exact final condition
- residual network architecture
- first-order stochastic optimizer Adam

Contribution

Advancing adaptive deep learning approach from [Nakamura-Zimmerer et al., 2021]:

- value function approximator with exact final condition
- residual network architecture
- first-order stochastic optimizer Adam
- physics-informed gradient regularization and L^2 regularization

Contribution

Advancing adaptive deep learning approach from [Nakamura-Zimmerer et al., 2021]:

- value function approximator with exact final condition
- residual network architecture
- first-order stochastic optimizer Adam
- physics-informed gradient regularization and L^2 regularization
- early stopping technique combined with variance estimation based sample size selection

Contribution

Advancing adaptive deep learning approach from [Nakamura-Zimmerer et al., 2021]:

- value function approximator with exact final condition
- residual network architecture
- first-order stochastic optimizer Adam
- physics-informed gradient regularization and L^2 regularization
- early stopping technique combined with variance estimation based sample size selection

Implementing algorithm applied to controlled reaction-diffusion system for numerical analysis:

`https://github.com/ElisaGiesecke/AdaDL-for-HJB-Equations.`

Outline

Problem Formulation

- initial time $t_0 \in \mathbb{R}$, final time $t_f \in \mathbb{R}$
- control $\mathbf{u} : [t_0, t_f] \to \mathbb{R}^m$, $\mathcal{U} := L^\infty(t_0, t_f; \mathbb{R}^m)$
- state $\mathbf{x} : [t_0, t_f] \to \mathbb{R}^n$, $\mathcal{X} := W^{1,\infty}(t_0, t_f; \mathbb{R}^n)$

Problem (fixed-time free-endpoint optimal control problem)

Problem Formulation

- initial time $t_0 \in \mathbb{R}$, final time $t_f \in \mathbb{R}$
- control $\mathbf{u} : [t_0, t_f] \to \mathbb{R}^m$, $\mathcal{U} := L^\infty(t_0, t_f; \mathbb{R}^m)$
- state $\mathbf{x} : [t_0, t_f] \to \mathbb{R}^n$, $\mathcal{X} := W^{1,\infty}(t_0, t_f; \mathbb{R}^n)$
- dynamics $f : [t_0, t_f] \times \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^n$
- initial condition $x_0 \in \mathbb{R}^n$

Problem (fixed-time free-endpoint optimal control problem)

$$\dot{\mathbf{x}}(t) = f(t, \mathbf{u}(t), \mathbf{x}(t)) \quad \text{for a. a. } t \in [t_0, t_f]; \quad \mathbf{x}(0) = x_0;$$

Problem Formulation

- initial time $t_0 \in \mathbb{R}$, final time $t_f \in \mathbb{R}$
- control $\mathbf{u} : [t_0, t_f] \to \mathbb{R}^m$, $\mathcal{U} := L^\infty(t_0, t_f; \mathbb{R}^m)$
- state $\mathbf{x} : [t_0, t_f] \to \mathbb{R}^n$, $\mathcal{X} := W^{1,\infty}(t_0, t_f; \mathbb{R}^n)$
- dynamics $f : [t_0, t_f] \times \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^n$
- initial condition $x_0 \in \mathbb{R}^n$
- cost function $J : \mathcal{U} \times \mathcal{X} \to \mathbb{R}$
- running cost $\psi : [t_0, t_f] \times \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$, final cost $\phi : \mathbb{R}^n \to \mathbb{R}$

Problem (fixed-time free-endpoint optimal control problem)

$$\min_{(\mathbf{u},\mathbf{x}) \in \mathcal{U} \times \mathcal{X}} J(\mathbf{u}, \mathbf{x}) := \int_{t_0}^{t_f} \psi(t, \mathbf{u}(t), \mathbf{x}(t))\, \mathrm{d}t + \phi(\mathbf{x}(t_f))$$

subject to $\quad \dot{\mathbf{x}}(t) = f(t, \mathbf{u}(t), \mathbf{x}(t)) \quad$ for a. a. $t \in [t_0, t_f]; \quad \mathbf{x}(0) = x_0;$

Problem Formulation

- initial time $t_0 \in \mathbb{R}$, final time $t_f \in \mathbb{R}$
- control $\mathbf{u} : [t_0, t_f] \to \mathbb{R}^m$, $\mathcal{U} := L^\infty(t_0, t_f; \mathbb{R}^m)$
- state $\mathbf{x} : [t_0, t_f] \to \mathbb{R}^n$, $\mathcal{X} := W^{1,\infty}(t_0, t_f; \mathbb{R}^n)$
- dynamics $f : [t_0, t_f] \times \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^n$
- initial condition $x_0 \in \mathbb{R}^n$
- cost function $J : \mathcal{U} \times \mathcal{X} \to \mathbb{R}$
- running cost $\psi : [t_0, t_f] \times \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$, final cost $\phi : \mathbb{R}^n \to \mathbb{R}$
- control set $U \subseteq \mathbb{R}^m$

Problem (fixed-time free-endpoint optimal control problem)

$$\min_{(\mathbf{u},\mathbf{x}) \in \mathcal{U} \times \mathcal{X}} J(\mathbf{u}, \mathbf{x}) := \int_{t_0}^{t_f} \psi(t, \mathbf{u}(t), \mathbf{x}(t)) \, \mathrm{d}t + \phi(\mathbf{x}(t_f))$$

subject to $\dot{\mathbf{x}}(t) = f(t, \mathbf{u}(t), \mathbf{x}(t))$ for a. a. $t \in [t_0, t_f]$; $\mathbf{x}(0) = x_0$;

$\mathbf{u}(t) \in U$ for a. a. $t \in [t_0, t_f]$.

Derivation of Conditions for Optimality

variational approach

necessary conditions via Pontryagin's Minimum Principle based on adjoint state as Lagrange multiplier
$\rightarrow$ open-loop control

Derivation of Conditions for Optimality

variational approach

necessary conditions via Pontryagin's Minimum Principle based on adjoint state as Lagrange multiplier
$\rightarrow$ open-loop control

dynamic programming approach

sufficient conditions based on value function as solution of Hamilton-Jacobi-Bellman equation
$\rightarrow$ closed-loop feedback control

Derivation of Conditions for Optimality

variational approach

necessary conditions via Pontryagin's Minimum Principle based on adjoint state as Lagrange multiplier
$\rightarrow$ open-loop control

dynamic programming approach

sufficient conditions based on value function as solution of Hamilton-Jacobi-Bellman equation
$\rightarrow$ closed-loop feedback control

two-point boundary value problem for supervised learning

candidate open-loop solutions for data generation
value function learned by neural network

Open-loop Control via Pontryagin's Minimum Principle

Definition (Hamiltonian function)

$$H : \mathbb{R} \times \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}, \quad H(t,u,x,p) := \psi(t,u,x) + p \cdot f(t,u,x)$$

Open-loop Control via Pontryagin's Minimum Principle

Definition (Hamiltonian function)

$$H : \mathbb{R} \times \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}, \quad H(t,u,x,p) := \psi(t,u,x) + p \cdot f(t,u,x)$$

Theorem (Pontryagin's minimum principle) [Schättler et al., 2012]

Let $\mathbf{u}^* \in \mathcal{U}$ be a globally optimal control of the OCP with initial condition x_0. Furthermore, let $\mathbf{x}^* \in \mathcal{X}$ be the corresponding optimal state trajectory and $\mathbf{p}^* \in \mathcal{X}$ the associated costate, i. e., the solutions to the Hamiltonian system

$$\begin{cases} \dot{\mathbf{x}}(t) = H_p(t, \mathbf{u}^*(t;x_0), \mathbf{x}(t), \mathbf{p}(t)) & \text{for a. a. } t; \quad \mathbf{x}(t_0) = x_0, \\ \dot{\mathbf{p}}(t) = -H_x(t, \mathbf{u}^*(t;x_0), \mathbf{x}(t), \mathbf{p}(t)) & \text{for a. a. } t; \quad \mathbf{p}(t_f) = \phi_x(\mathbf{x}(t_f)). \end{cases}$$

Open-loop Control via Pontryagin's Minimum Principle

Definition (Hamiltonian function)

$$H : \mathbb{R} \times \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}, \quad H(t,u,x,p) := \psi(t,u,x) + p \cdot f(t,u,x)$$

Theorem (Pontryagin's minimum principle) [Schättler et al., 2012]

Let $\mathbf{u}^* \in \mathcal{U}$ be a globally optimal control of the OCP with initial condition x_0. Furthermore, let $\mathbf{x}^* \in \mathcal{X}$ be the corresponding optimal state trajectory and $\mathbf{p}^* \in \mathcal{X}$ the associated costate, i. e., the solutions to the Hamiltonian system

$$\begin{cases} \dot{\mathbf{x}}(t) = H_p(t, \mathbf{u}^*(t; x_0), \mathbf{x}(t), \mathbf{p}(t)) & \text{for a. a. } t; \quad \mathbf{x}(t_0) = x_0, \\ \dot{\mathbf{p}}(t) = -H_x(t, \mathbf{u}^*(t; x_0), \mathbf{x}(t), \mathbf{p}(t)) & \text{for a. a. } t; \quad \mathbf{p}(t_f) = \phi_x(\mathbf{x}(t_f)). \end{cases}$$

Then the Hamiltonian minimization condition below holds:

$$\mathbf{u}^*(t) \in \operatorname*{argmin}_{u \in U} H(t, u, \mathbf{x}^*(t), \mathbf{p}^*(t)) \quad \text{for a. a. } t.$$

Feedback Control via Hamilton-Jacobi-Bellman Equation

Theorem [Liberzon, 2012]

Consider the OCP and suppose that a continuously differentiable function $V : [t_0, t_f] \times \mathbb{R}^n \to \mathbb{R}$ is a classical solution to the HJB equation with its final condition, i. e., satisfying

$$\begin{cases} -V_t(t,x) = \inf_{u \in U} H(t,u,x,V_x(t,x)) & \text{for all } t \text{ and } x, \\ V(t_f,x) = \phi(x) & \text{for all } x. \end{cases}$$

Theorem [Liberzon, 2012]

Consider the OCP and suppose that a continuously differentiable function $V : [t_0, t_f] \times \mathbb{R}^n \to \mathbb{R}$ is a classical solution to the HJB equation with its final condition, i. e., satisfying

$$\begin{cases} -V_t(t,x) = \inf_{u \in U} H(t,u,x,V_x(t,x)) & \text{for all } t \text{ and } x, \\ V(t_f, x) = \phi(x) & \text{for all } x. \end{cases}$$

Furthermore, suppose that there exists a control $\mathbf{u}^* \in \mathcal{U}$ and a state $\mathbf{x}^* \in \mathcal{X}$ fulfilling the state equation with initial condition, for which the Hamiltonian minimization condition

$$\mathbf{u}^*(t) \in \operatorname*{argmin}_{u \in U} H(t,u,\mathbf{x}^*(t),V_x(t,\mathbf{x}^*(t))) \quad \text{for all } t$$

holds. Then $\mathbf{u}^*$ is a globally optimal control.

Relation between PMP and HJB approach

- costate as gradient of the value function

$$\mathbf{p}^*(t) = V_x(t, \mathbf{x}^*(t)), \quad \text{assuming } V \in \mathcal{C}^2$$

Relation between PMP and HJB approach

- costate as gradient of the value function

$$\mathbf{p}^*(t) = V_x(t, \mathbf{x}^*(t)), \quad \text{assuming } V \in \mathcal{C}^2$$

- Hamiltonian system as characteristics of the HJB equation

$$\begin{cases} \dot{\mathbf{x}}(t) = H_p(t, \mathbf{u}^*(t; x_0), \mathbf{x}(t), \mathbf{p}(t)) & \text{for a. a. } t; \quad \mathbf{x}(t_0) = x_0, \\ \dot{\mathbf{p}}(t) = -H_x(t, \mathbf{u}^*(t; x_0), \mathbf{x}(t), \mathbf{p}(t)) & \text{for a. a. } t; \quad \mathbf{p}(t_f) = \phi_x(\mathbf{x}(t_f)), \end{cases}$$

Relation between PMP and HJB approach

- costate as gradient of the value function

$$\mathbf{p}^*(t) = V_x(t, \mathbf{x}^*(t)), \quad \text{assuming } V \in \mathcal{C}^2$$

- Hamiltonian system as characteristics of the HJB equation

$$\begin{cases} \dot{\mathbf{x}}(t) = H_p(t, \mathbf{u}^*(t; x_0), \mathbf{x}(t), \mathbf{p}(t)) & \text{for a. a. } t; \quad \mathbf{x}(t_0) = x_0, \\ \dot{\mathbf{p}}(t) = -H_x(t, \mathbf{u}^*(t; x_0), \mathbf{x}(t), \mathbf{p}(t)) & \text{for a. a. } t; \quad \mathbf{p}(t_f) = \phi_x(\mathbf{x}(t_f)), \\ \dot{\mathbf{v}}(t) = -\psi(t, \mathbf{u}^*(t; x_0), \mathbf{x}(t)) & \text{for a. a. } t; \quad \mathbf{v}(t_f) = \phi(\mathbf{x}(t_f)), \end{cases}$$

extended by ODE describing the evolution of the value function along trajectory, denoting $\mathbf{v}(t) := V(t, \mathbf{x}^*(t))$, see [Kang and Wilcox, 2015]

Outline

Value Function Approximation for NN Feedback Control

- value function approximator V^{NN} with neural network function F^{NN}
  $\rightarrow$ guaranteeing satisfaction of final condition

$$V^{\mathsf{NN}}(t,x) := F^{\mathsf{NN}}(t,x) - F^{\mathsf{NN}}(t_f,x) + \phi(x) \quad \text{for all } t \text{ and } x$$

Value Function Approximation for NN Feedback Control

- value function approximator V^{NN} with neural network function F^{NN}
  $\rightarrow$ guaranteeing satisfaction of final condition

$$V^{\mathsf{NN}}(t,x) \coloneqq F^{\mathsf{NN}}(t,x) - F^{\mathsf{NN}}(t_f,x) + \phi(x) \quad \text{for all } t \text{ and } x$$

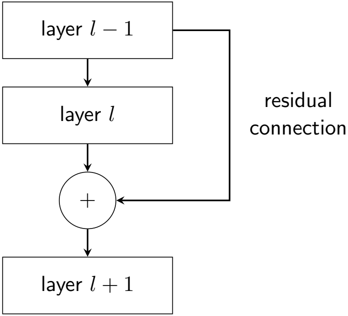
- NN feedback controller $\mathbf{u}^{\mathsf{NN}}$
  $\rightarrow$ exploiting Hamiltonian minimization condition

$$\mathbf{u}^{\mathsf{NN}}(t,x) \in \operatorname*{argmin}_{u \in \mathbb{R}^m} H(t,u,x,V_x^{\mathsf{NN}}(t,x)) \quad \text{for all } t \text{ and } x$$

Residual Network Architecture

$$F^{\mathsf{NN}} = f^{[L]} \circ \cdots \circ f^{[0]}, \quad \theta = \{W^{[l]}, b^{[l]}\}_{l=0}^{L}$$

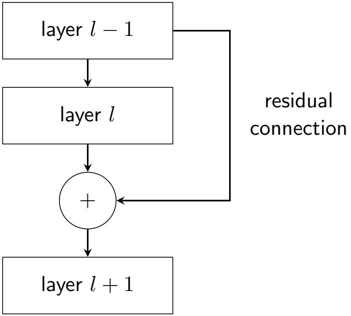
- residual layers $f^{[l]}(y) = y + \sigma\left(W^{[l]}y + b^{[l]}\right)$
- weights $W^{[l]} \in \mathbb{R}^{d_{l+1}\times d_l}$ and biases $b^{[l]} \in \mathbb{R}^{d_{l+1}}$
- softplus activation function $\sigma(x) = \log(1+\exp(x))$



Residual Network Architecture

$$F^{\mathsf{NN}} = f^{[L]} \circ \cdots \circ f^{[0]}, \quad \theta = \{W^{[l]}, b^{[l]}\}_{l=0}^{L}$$

- residual layers $f^{[l]}(y) = y + \sigma\left(W^{[l]}y + b^{[l]}\right)$
- weights $W^{[l]} \in \mathbb{R}^{d_{l+1} \times d_l}$ and biases $b^{[l]} \in \mathbb{R}^{d_{l+1}}$
- softplus activation function $\sigma(x) = \log(1 + \exp(x))$



$\rightarrow$ avoiding vanishing gradients and representational bottlenecks
$\rightarrow$ interpreting as explicit forward Euler discretization of neural ODE

Physics-Informed Training with Adam

- training data $D_{\text{train}} = \{(t^{(k)}, x^{(k)}), V^{(k)}\}_{k=1}^{K}$

Problem (physics-informed deep learning problem) [Raissi et al., 2019]

Physics-Informed Training with Adam

- training data $D_{\text{train}} = \{(t^{(k)}, x^{(k)}), V^{(k)}\}_{k=1}^{K}$
- mean squared error loss
  $\mathrm{MSE}_V(\theta; D_{\text{train}}) \coloneqq \frac{1}{K} \sum_{k=1}^{K} \left| V^{\text{NN}}(t^{(k)}, x^{(k)}; \theta) - V^{(k)} \right|^2$

Problem (physics-informed deep learning problem) [Raissi et al., 2019]

$$\min_{\theta} \mathrm{Cost}(\theta; D_{\text{train}}) \coloneqq \mathrm{MSE}_V(\theta; D_{\text{train}})$$

Physics-Informed Training with Adam

- training data $D_{\text{train}} = \{(t^{(k)}, x^{(k)}), (V^{(k)}, p^{(k)})\}_{k=1}^{K}$
- mean squared error loss
  $\mathrm{MSE}_V(\theta; D_{\text{train}}) \coloneqq \frac{1}{K} \sum_{k=1}^{K} \left| V^{\text{NN}}(t^{(k)}, x^{(k)}; \theta) - V^{(k)} \right|^2$
- gradient regularization
  $\mathrm{MSE}_p(\theta; D_{\text{train}}) \coloneqq \frac{1}{K} \sum_{k=1}^{K} \left\| V_x^{\text{NN}}(t^{(k)}, x^{(k)}; \theta) - p^{(k)} \right\|_2^2, \quad \lambda > 0$

Problem (physics-informed deep learning problem) [Raissi et al., 2019]

$$\min_{\theta} \mathrm{Cost}(\theta; D_{\text{train}}) \coloneqq \mathrm{MSE}_V(\theta; D_{\text{train}}) + \lambda \cdot \mathrm{MSE}_p(\theta; D_{\text{train}})$$

Physics-Informed Training with Adam

- training data $D_{\text{train}} = \{(t^{(k)}, x^{(k)}), (V^{(k)}, p^{(k)})\}_{k=1}^{K}$
- mean squared error loss
  $\mathrm{MSE}_V(\theta; D_{\text{train}}) \coloneqq \frac{1}{K}\sum_{k=1}^{K} \left| V^{\text{NN}}(t^{(k)}, x^{(k)}; \theta) - V^{(k)} \right|^2$
- gradient regularization
  $\mathrm{MSE}_p(\theta; D_{\text{train}}) \coloneqq \frac{1}{K}\sum_{k=1}^{K} \left\| V_x^{\text{NN}}(t^{(k)}, x^{(k)}; \theta) - p^{(k)} \right\|_2^2, \quad \lambda > 0$

Problem (physics-informed deep learning problem) [Raissi et al., 2019]

$$\min_{\theta} \mathrm{Cost}(\theta; D_{\text{train}}) \coloneqq \mathrm{MSE}_V(\theta; D_{\text{train}}) + \lambda \cdot \mathrm{MSE}_p(\theta; D_{\text{train}})$$

$\rightarrow$ complementing gradient regularization with weight decay
$\rightarrow$ optimizing via batch gradient descent with adaptive learning rates

Progressive and Adaptive Data Generation

Employing spatially causality-free solver for BVP

$$\begin{cases} \dot{\mathbf{x}}(t) = H_p(t, \mathbf{u}^*(t; x_0), \mathbf{x}(t), \mathbf{p}(t)) & \text{for a. a. } t; \quad \mathbf{x}(t_0) = x_0, \\ \dot{\mathbf{p}}(t) = -H_x(t, \mathbf{u}^*(t; x_0), \mathbf{x}(t), \mathbf{p}(t)) & \text{for a. a. } t; \quad \mathbf{p}(t_f) = \phi_x(\mathbf{x}(t_f)), \\ \dot{\mathbf{v}}(t) = -\psi(t, \mathbf{u}^*(t; x_0), \mathbf{x}(t)) & \text{for a. a. } t; \quad \mathbf{v}(t_f) = \phi(\mathbf{x}(t_f)), \end{cases}$$

for generating training, validation and test data sets

Progressive and Adaptive Data Generation

Employing spatially causality-free solver for BVP

$$\begin{cases} \dot{\mathbf{x}}(t) = H_p(t, \mathbf{u}^*(t; x_0), \mathbf{x}(t), \mathbf{p}(t)) & \text{for a. a. } t; \quad \mathbf{x}(t_0) = x_0, \\ \dot{\mathbf{p}}(t) = -H_x(t, \mathbf{u}^*(t; x_0), \mathbf{x}(t), \mathbf{p}(t)) & \text{for a. a. } t; \quad \mathbf{p}(t_f) = \phi_x(\mathbf{x}(t_f)), \\ \dot{\mathbf{v}}(t) = -\psi(t, \mathbf{u}^*(t; x_0), \mathbf{x}(t)) & \text{for a. a. } t; \quad \mathbf{v}(t_f) = \phi(\mathbf{x}(t_f)), \end{cases}$$

for generating training, validation and test data sets

- ability to generate data in selected regions
  $\rightarrow$ uniform or adaptive sampling

Progressive and Adaptive Data Generation

Employing spatially causality-free solver for BVP

$$\begin{cases} \dot{\mathbf{x}}(t) = H_p(t, \mathbf{u}^*(t; x_0), \mathbf{x}(t), \mathbf{p}(t)) & \text{for a. a. } t; \quad \mathbf{x}(t_0) = x_0, \\ \dot{\mathbf{p}}(t) = -H_x(t, \mathbf{u}^*(t; x_0), \mathbf{x}(t), \mathbf{p}(t)) & \text{for a. a. } t; \quad \mathbf{p}(t_f) = \phi_x(\mathbf{x}(t_f)), \\ \dot{\mathbf{v}}(t) = -\psi(t, \mathbf{u}^*(t; x_0), \mathbf{x}(t)) & \text{for a. a. } t; \quad \mathbf{v}(t_f) = \phi(\mathbf{x}(t_f)), \end{cases}$$

for generating training, validation and test data sets

- ability to generate data in selected regions
  $\rightarrow$ uniform or adaptive sampling
- sensitivity to initialization of $\mathbf{x}$, $\mathbf{p}$ and $\mathbf{v}$
  $\rightarrow$ time-marching or NN warm start

Initialization and Sampling Techniques

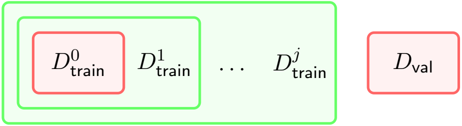
Before training: initial training set and validation set

D^0_{train} D_{val}

- uniform sampling: drawing initial conditions uniformly at random
- time-marching: solving over time intervals of increasing length

Initialization and Sampling Techniques

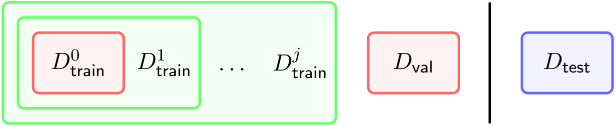
During training: adaptively augmented training sets



- adaptive sampling: selecting initial conditions where predicted value function gradient is large
- NN warm start: simulating with partially trained neural network

Initialization and Sampling Techniques

After training: test set



- sampling as desired: drawing initial conditions randomly or in region of interest
- NN warm start: simulating with fully trained neural network

Empirical Validation of Model Accuracy

Error metrics: for some data set $D = \{(t^{(k)}, x^{(k)}), (V^{(k)}, p^{(k)})\}_{k=1}^{K}$

- $\mathrm{RMAE}_V(\theta; D) := \frac{\sum_{k=1}^{K} \left| V^{\mathsf{NN}}(t^{(k)}, x^{(k)}; \theta) - V^{(k)} \right|}{\sum_{k=1}^{K} \left| V^{(k)} \right|}$

Empirical Validation of Model Accuracy

Error metrics: for some data set $D = \{(t^{(k)}, x^{(k)}), (V^{(k)}, p^{(k)})\}_{k=1}^{K}$

- $\mathrm{RMAE}_V(\theta; D) := \frac{\sum_{k=1}^{K} \left| V^{\mathsf{NN}}(t^{(k)}, x^{(k)}; \theta) - V^{(k)} \right|}{\sum_{k=1}^{K} \left| V^{(k)} \right|}$
- $\mathrm{RM}L^1{}_p(\theta; D) := \frac{\sum_{k=1}^{K} \left\| V_x^{\mathsf{NN}}(t^{(k)}, x^{(k)}; \theta) - p^{(k)} \right\|_1}{\sum_{k=1}^{K} \left\| p^{(k)} \right\|_1}$

Empirical Validation of Model Accuracy

Error metrics: for some data set $D = \{(t^{(k)}, x^{(k)}), (V^{(k)}, p^{(k)})\}_{k=1}^{K}$

- $\mathrm{RMAE}_V(\theta; D) := \frac{\sum_{k=1}^{K} \left| V^{\mathsf{NN}}(t^{(k)}, x^{(k)}; \theta) - V^{(k)} \right|}{\sum_{k=1}^{K} \left| V^{(k)} \right|}$
- $\mathrm{RM}L^1{}_p(\theta; D) := \frac{\sum_{k=1}^{K} \left\| V_x^{\mathsf{NN}}(t^{(k)}, x^{(k)}; \theta) - p^{(k)} \right\|_1}{\sum_{k=1}^{K} \left\| p^{(k)} \right\|_1}$
- $\mathrm{RMAE}_u(\theta; D) := \frac{\sum_{k=1}^{K} \left| u^{\mathsf{NN}}(t^{(k)}, x^{(k)}; \theta) - u^{(k)} \right|}{\sum_{k=1}^{K} \left| u^{(k)} \right|}, \quad u^{(k)} = \mathbf{u}(t^{(k)})$

Empirical Validation of Model Accuracy

Error metrics: for some data set $D = \{(t^{(k)}, x^{(k)}), (V^{(k)}, p^{(k)})\}_{k=1}^{K}$

- $\mathrm{RMAE}_V(\theta; D) := \frac{\sum_{k=1}^{K} \left| V^{\mathsf{NN}}(t^{(k)}, x^{(k)}; \theta) - V^{(k)} \right|}{\sum_{k=1}^{K} \left| V^{(k)} \right|}$
- $\mathrm{RM}L^1{}_p(\theta; D) := \frac{\sum_{k=1}^{K} \left\| V_x^{\mathsf{NN}}(t^{(k)}, x^{(k)}; \theta) - p^{(k)} \right\|_1}{\sum_{k=1}^{K} \left\| p^{(k)} \right\|_1}$
- $\mathrm{RMAE}_u(\theta; D) := \frac{\sum_{k=1}^{K} \left| u^{\mathsf{NN}}(t^{(k)}, x^{(k)}; \theta) - u^{(k)} \right|}{\sum_{k=1}^{K} \left| u^{(k)} \right|}, \quad u^{(k)} = \mathbf{u}(t^{(k)})$

$\rightarrow$ performance measure E applied to D_{train}, D_{val}, D_{test}

Empirical Validation of Model Accuracy

Error metrics: for some data set $D = \{(t^{(k)}, x^{(k)}), (V^{(k)}, p^{(k)})\}_{k=1}^K$

- $\mathrm{RMAE}_V(\theta; D) := \frac{\sum_{k=1}^K \left|V^{\mathsf{NN}}(t^{(k)}, x^{(k)}; \theta) - V^{(k)}\right|}{\sum_{k=1}^K \left|V^{(k)}\right|}$
- $\mathrm{RM}L^1{}_p(\theta; D) := \frac{\sum_{k=1}^K \left\|V_x^{\mathsf{NN}}(t^{(k)}, x^{(k)}; \theta) - p^{(k)}\right\|_1}{\sum_{k=1}^K \left\|p^{(k)}\right\|_1}$
- $\mathrm{RMAE}_u(\theta; D) := \frac{\sum_{k=1}^K \left|u^{\mathsf{NN}}(t^{(k)}, x^{(k)}; \theta) - u^{(k)}\right|}{\sum_{k=1}^K \left|u^{(k)}\right|}, \quad u^{(k)} = \mathbf{u}(t^{(k)})$

$\rightarrow$ performance measure E applied to D_{train}, D_{val}, D_{test}

Training and validation error: at epoch i in training round j

$$E_{\mathsf{train}}(i) := E(\theta^i; D^j_{\mathsf{train}}) \quad \text{and} \quad E_{\mathsf{val}}(i) := E(\theta^i; D_{\mathsf{val}})$$

Empirical Validation of Model Accuracy

Error metrics: for some data set $D = \{(t^{(k)}, x^{(k)}), (V^{(k)}, p^{(k)})\}_{k=1}^{K}$

- $\mathrm{RMAE}_V(\theta; D) := \frac{\sum_{k=1}^{K} \left| V^{\mathsf{NN}}(t^{(k)}, x^{(k)}; \theta) - V^{(k)} \right|}{\sum_{k=1}^{K} \left| V^{(k)} \right|}$
- $\mathrm{RM}L^1{}_p(\theta; D) := \frac{\sum_{k=1}^{K} \left\| V_x^{\mathsf{NN}}(t^{(k)}, x^{(k)}; \theta) - p^{(k)} \right\|_1}{\sum_{k=1}^{K} \left\| p^{(k)} \right\|_1}$
- $\mathrm{RMAE}_u(\theta; D) := \frac{\sum_{k=1}^{K} \left| u^{\mathsf{NN}}(t^{(k)}, x^{(k)}; \theta) - u^{(k)} \right|}{\sum_{k=1}^{K} \left| u^{(k)} \right|}, \quad u^{(k)} = \mathbf{u}(t^{(k)})$

$\rightarrow$ performance measure E applied to D_{train}, D_{val}, D_{test}

Training and validation error: at epoch i in training round j

$$E_{\mathsf{train}}(i) := E(\theta^i; D^j_{\mathsf{train}}) \quad \text{and} \quad E_{\mathsf{val}}(i) := E(\theta^i; D_{\mathsf{val}})$$

Test error: for final model parameter θ^*

$$E_{\mathsf{test}} := E(\theta^*; D_{\mathsf{test}})$$

Variance Estimation based Sample Size Selection

Goal: find NN parameter $\theta^* \in \mathbb{R}^R$ minimizing population cost function $\mathrm{Cost}(\cdot; D_\infty)$ where $D_\infty := [t_0, t_f] \times \mathbb{X}$

Variance Estimation based Sample Size Selection

Goal: find NN parameter $\theta^* \in \mathbb{R}^R$ minimizing population cost function $\mathrm{Cost}(\cdot; D_\infty)$ where $D_\infty := [t_0, t_f] \times \mathbb{X}$

But: only sample $D^j_{\text{train}} \subset D_\infty$ of size $K_j := |D^j_{\text{train}}|$ available

Variance Estimation based Sample Size Selection

Goal: find NN parameter $\theta^* \in \mathbb{R}^R$ minimizing population cost function $\mathrm{Cost}(\cdot; D_\infty)$ where $D_\infty := [t_0, t_f] \times \mathbb{X}$

But: only sample $D^j_{\text{train}} \subset D_\infty$ of size $K_j := |D^j_{\text{train}}|$ available

Assumption

All data points $(t^{(k)}, x^{(k)}) \in D^j_{\textit{train}}$ are independent and identically distributed.

Variance Estimation based Sample Size Selection

Goal: find NN parameter $\theta^* \in \mathbb{R}^R$ minimizing population cost function $\mathrm{Cost}(\cdot; D_\infty)$ where $D_\infty := [t_0, t_f] \times \mathbb{X}$

But: only sample $D^j_{\text{train}} \subset D_\infty$ of size $K_j := |D^j_{\text{train}}|$ available

Assumption

All data points $(t^{(k)}, x^{(k)}) \in D^j_{\text{train}}$ are independent and identically distributed.

$\rightarrow \nabla \mathrm{Cost}(\theta; D^j_{\text{train}})$ is unbiased estimator of $\nabla \mathrm{Cost}(\theta; D_\infty)$

Variance Estimation based Sample Size Selection

Goal: find NN parameter $\theta^* \in \mathbb{R}^R$ minimizing population cost function $\mathrm{Cost}(\cdot; D_\infty)$ where $D_\infty := [t_0, t_f] \times \mathbb{X}$

But: only sample $D^j_{\text{train}} \subset D_\infty$ of size $K_j := |D^j_{\text{train}}|$ available

Assumption

All data points $(t^{(k)}, x^{(k)}) \in D^j_{\text{train}}$ are independent and identically distributed.

$\rightarrow \nabla \mathrm{Cost}(\theta; D^j_{\text{train}})$ is unbiased estimator of $\nabla \mathrm{Cost}(\theta; D_\infty)$

Approach inspired by [Byrd et al., 2012]: control root mean squared error

$$\sqrt{\mathrm{MSE}\left(\nabla \mathrm{Cost}(\theta; D^j_{\text{train}})\right)} \leq C \left\| \mathbb{E}_{D^j_{\text{train}}} \left[\nabla \mathrm{Cost}(\theta; D^j_{\text{train}})\right] \right\|_1, \quad C > 0$$

Variance Estimation based Sample Size Selection

Goal: find NN parameter $\theta^* \in \mathbb{R}^R$ minimizing population cost function $\mathrm{Cost}(\cdot; D_\infty)$ where $D_\infty := [t_0, t_f] \times \mathbb{X}$

But: only sample $D^j_{\text{train}} \subset D_\infty$ of size $K_j := |D^j_{\text{train}}|$ available

Assumption

All data points $(t^{(k)}, x^{(k)}) \in D^j_{train}$ are independent and identically distributed.

$\rightarrow \nabla \mathrm{Cost}(\theta; D^j_{\text{train}})$ is unbiased estimator of $\nabla \mathrm{Cost}(\theta; D_\infty)$

Approach inspired by [Byrd et al., 2012]: control root mean squared error

$$\sqrt{\frac{1}{K_j} \left\| \mathrm{Var}_{(t,x)\in D_\infty} \left(\nabla \mathrm{Cost}(\theta; (t,x)) \right) \right\|_1} \le C \left\| \mathbb{E}_{D^j_{\text{train}}} \left[\nabla \mathrm{Cost}(\theta; D^j_{\text{train}}) \right] \right\|_1$$

bias-variance decomposition and assumption of i. i. d. data

Variance Estimation based Sample Size Selection

Goal: find NN parameter $\theta^* \in \mathbb{R}^R$ minimizing population cost function $\mathrm{Cost}(\cdot; D_\infty)$ where $D_\infty := [t_0, t_f] \times \mathbb{X}$

But: only sample $D^j_{\text{train}} \subset D_\infty$ of size $K_j := |D^j_{\text{train}}|$ available

Assumption

All data points $(t^{(k)}, x^{(k)}) \in D^j_{\text{train}}$ are independent and identically distributed.

$\rightarrow \nabla \mathrm{Cost}(\theta; D^j_{\text{train}})$ is unbiased estimator of $\nabla \mathrm{Cost}(\theta; D_\infty)$

Approach inspired by [Byrd et al., 2012]: control root mean squared error

$$\sqrt{\frac{1}{K_j} \left\| \mathrm{Var}_{(t,x)\in D^j_{\text{train}}} \left(\nabla \mathrm{Cost}(\theta; (t,x))\right) \right\|_1} \leq C \left\| \nabla \mathrm{Cost}(\theta; D^j_{\text{train}}) \right\|_1$$

approximation by sample variance and sample gradient

Variance Estimation based Sample Size Selection

Convergence test: with tolerance $C > 0$

$$\frac{\sqrt{\left\| \mathrm{Var}_{(t,x)\in D^j_{\text{train}}} \left(\nabla\, \mathrm{Cost}(\theta; (t,x)) \right) \right\|_1}}{\sqrt{K_j} \left\| \nabla\, \mathrm{Cost}(\theta; D^j_{\text{train}}) \right\|_1} \leq C$$

Variance Estimation based Sample Size Selection

Convergence test: with tolerance $C > 0$

$$\frac{\sqrt{\left\|\operatorname{Var}_{(t,x)\in D^j_{\text{train}}}\left(\nabla \operatorname{Cost}(\theta;(t,x))\right)\right\|_1}}{\sqrt{K_j}\left\|\nabla \operatorname{Cost}(\theta; D^j_{\text{train}})\right\|_1} \leq C$$

Sample size selection rule:

$$K_{j+1} \geq \frac{\left\|\operatorname{Var}_{(t,x)\in D^j_{\text{train}}}\left(\nabla \operatorname{Cost}(\theta;(t,x))\right)\right\|_1}{C^2\left\|\nabla \operatorname{Cost}(\theta; D^j_{\text{train}})\right\|_1^2}$$

Variance Estimation based Sample Size Selection

Convergence test: with tolerance $C > 0$

$$\frac{\sqrt{\left\| \mathrm{Var}_{(t,x)\in D^j_{\text{train}}} \left(\nabla \, \mathrm{Cost}(\theta; (t,x)) \right) \right\|_1}}{\sqrt{K_j} \left\| \nabla \, \mathrm{Cost}(\theta; D^j_{\text{train}}) \right\|_1} \leq C$$

Sample size selection rule: with upper bound $M > 1$

$$K_{j+1} := \min \left\{ \left\lceil \frac{\left\| \mathrm{Var}_{(t,x)\in D^j_{\text{train}}} \left(\nabla \, \mathrm{Cost}(\theta; (t,x)) \right) \right\|_1}{C^2 \left\| \nabla \, \mathrm{Cost}(\theta; D^j_{\text{train}}) \right\|_1^2} \right\rceil, \lfloor M K_j \rfloor \right\}$$

Early Stopping

Preventing excessive computation of convergence test and data generation

$\rightarrow$ delay until overfitting is detected by early stopping criterion
$\rightarrow$ promote cheaper heuristics and efficient use of data
$\rightarrow$ reduce computational costs during training phase

Early Stopping

Preventing excessive computation of convergence test and data generation

$\rightarrow$ delay until overfitting is detected by early stopping criterion
$\rightarrow$ promote cheaper heuristics and efficient use of data
$\rightarrow$ reduce computational costs during training phase

Criterion adapted from [Prechelt, 2012]: with generalization loss GL, training progress TP_ι over ι epochs, and threshold $T > 0$

$$\text{stop at epoch } i \geq \iota \text{ if:} \quad \mathrm{TP}_\iota(i) = 0 \quad \text{or} \quad \frac{\mathrm{GL}(i)}{\mathrm{TP}_\iota(i)} > T$$

Algorithm Adaptive Deep Learning

1. Generate D^0_{train} and D_{val}
2. Initialize NN with $\theta \in \mathbb{R}^R$; set $i \leftarrow 0$, $j \leftarrow 0$, $i^* \leftarrow 0$ and $\theta^* \leftarrow \theta$

while not converged **do**

  while not early stopped **do**

    3. Update θ by training NN on D^j_{train} for one epoch; set $i \leftarrow i + 1$
    4. Track $E_{\text{train}}(i)$ and $E_{\text{val}}(i)$; evaluate early stopping criterion

    if $E_{\text{val}}(i) < E_{\text{val}}(i^*)$ **then**

      5. Set $i^* \leftarrow i$ and $\theta^* \leftarrow \theta$

  6. Evaluate convergence criterion

  if not converged **then**

    7. Determine $|D^{j+1}_{\text{train}}|$ by sample size criterion
    8. Augment D^j_{train} adaptively; set $j \leftarrow j + 1$

9. Load NN with θ^*
10. Generate D_{test}
11. Compute E_{test} to evaluate generalization performance

Algorithm Adaptive Deep Learning

1. Generate D^0_{train} and D_{val}
2. Initialize NN with $\theta \in \mathbb{R}^R$; set $i \leftarrow 0$, $j \leftarrow 0$, $i^* \leftarrow 0$ and $\theta^* \leftarrow \theta$

while not converged **do**

    while not early stopped **do**

        3. Update θ by training NN on D^j_{train} for one epoch; set $i \leftarrow i + 1$

        4. Track $E_{\text{train}}(i)$ and $E_{\text{val}}(i)$; evaluate early stopping criterion

        if $E_{\text{val}}(i) < E_{\text{val}}(i^*)$ **then**

            5. Set $i^* \leftarrow i$ and $\theta^* \leftarrow \theta$

    6. Evaluate convergence criterion

    if not converged **then**

        7. Determine $|D^{j+1}_{\text{train}}|$ by sample size criterion

        8. Augment D^j_{train} adaptively; set $j \leftarrow j + 1$

9. Load NN with θ^*
10. Generate D_{test}
11. Compute E_{test} to evaluate generalization performance

Algorithm Adaptive Deep Learning

1. Generate D^0_{train} and D_{val}
2. Initialize NN with $\theta \in \mathbb{R}^R$; set $i \leftarrow 0$, $j \leftarrow 0$, $i^* \leftarrow 0$ and $\theta^* \leftarrow \theta$

while not converged **do**

  while not early stopped **do**

  3. Update θ by training NN on D^j_{train} for one epoch; set $i \leftarrow i + 1$
  4. Track $E_{\text{train}}(i)$ and $E_{\text{val}}(i)$; evaluate early stopping criterion

  if $E_{\text{val}}(i) < E_{\text{val}}(i^*)$ **then**

  5. Set $i^* \leftarrow i$ and $\theta^* \leftarrow \theta$

  6. Evaluate convergence criterion

  if not converged **then**

  7. Determine $|D^{j+1}_{\text{train}}|$ by sample size criterion
  8. Augment D^j_{train} adaptively; set $j \leftarrow j + 1$

9. Load NN with θ^*
10. Generate D_{test}
11. Compute E_{test} to evaluate generalization performance

Algorithm Adaptive Deep Learning

1. Generate D^0_{train} and D_{val}
2. Initialize NN with $\theta \in \mathbb{R}^R$; set $i \leftarrow 0$, $j \leftarrow 0$, $i^* \leftarrow 0$ and $\theta^* \leftarrow \theta$

while not converged **do**

  while not early stopped **do**

    3. Update θ by training NN on D^j_{train} for one epoch; set $i \leftarrow i + 1$
    4. Track $E_{\text{train}}(i)$ and $E_{\text{val}}(i)$; evaluate early stopping criterion

    if $E_{\text{val}}(i) < E_{\text{val}}(i^*)$ **then**

      5. Set $i^* \leftarrow i$ and $\theta^* \leftarrow \theta$

  6. Evaluate convergence criterion

  if not converged **then**

    7. Determine $|D^{j+1}_{\text{train}}|$ by sample size criterion
    8. Augment D^j_{train} adaptively; set $j \leftarrow j + 1$

9. Load NN with θ^*
10. Generate D_{test}
11. Compute E_{test} to evaluate generalization performance

Algorithm Adaptive Deep Learning

1. Generate D^0_{train} and D_{val}
2. Initialize NN with $\theta \in \mathbb{R}^R$; set $i \leftarrow 0$, $j \leftarrow 0$, $i^* \leftarrow 0$ and $\theta^* \leftarrow \theta$

while not converged **do**

  while not early stopped **do**

   3. Update θ by training NN on D^j_{train} for one epoch; set $i \leftarrow i + 1$
   4. Track $E_{\text{train}}(i)$ and $E_{\text{val}}(i)$; evaluate early stopping criterion

   if $E_{\text{val}}(i) < E_{\text{val}}(i^*)$ **then**

    5. Set $i^* \leftarrow i$ and $\theta^* \leftarrow \theta$

  6. Evaluate convergence criterion

  if not converged **then**

   7. Determine $|D^{j+1}_{\text{train}}|$ by sample size criterion
   8. Augment D^j_{train} adaptively; set $j \leftarrow j + 1$

9. Load NN with θ^*
10. Generate D_{test}
11. Compute E_{test} to evaluate generalization performance

Algorithm Adaptive Deep Learning

1. Generate D^0_{train} and D_{val}
2. Initialize NN with $\theta \in \mathbb{R}^R$; set $i \leftarrow 0$, $j \leftarrow 0$, $i^* \leftarrow 0$ and $\theta^* \leftarrow \theta$

while not converged **do**

    while not early stopped **do**

        3. Update θ by training NN on D^j_{train} for one epoch; set $i \leftarrow i + 1$

        4. Track $E_{\text{train}}(i)$ and $E_{\text{val}}(i)$; evaluate early stopping criterion

        if $E_{\text{val}}(i) < E_{\text{val}}(i^*)$ **then**

            5. Set $i^* \leftarrow i$ and $\theta^* \leftarrow \theta$

    6. Evaluate convergence criterion

    if not converged **then**

        7. Determine $|D^{j+1}_{\text{train}}|$ by sample size criterion

        8. Augment D^j_{train} adaptively; set $j \leftarrow j + 1$

9. Load NN with θ^*
10. Generate D_{test}
11. Compute E_{test} to evaluate generalization performance

Outline

Optimal Control of Reaction-Diffusion System

Diffusion equation with state $X : [t_0, t_f] \times \Omega \to \mathbb{R}$ on $\Omega \subseteq \mathbb{R}$, $a > 0$

$$X_t(t,\omega) = a\Delta X(t,\omega) \quad \text{for a. a. } (t,\omega) \in [t_0, t_f] \times \Omega$$

Optimal Control of Reaction-Diffusion System

Reaction term with $b > 0$

$$X_t(t,\omega) = a\Delta X(t,\omega) + bR(X(t,\omega)) \quad \text{for a. a. } (t,\omega)$$

introducing non-linearity via, e. g.,

- hyperbolic growth rate $R_1(X(t,\omega)) := X(t,\omega)^2$
- logistic growth rate $R_2(X(t,\omega)) := X(t,\omega)(1 - X(t,\omega))$

Optimal Control of Reaction-Diffusion System

Control term with control $\mathbf{u} : [t_0, t_f] \to \mathbb{R}$, $c > 0$

$$X_t(t,\omega) = a\Delta X(t,\omega) + bR(X(t,\omega)) + cS(\omega, \mathbf{u}(t), X(t,\omega)) \quad \text{for a. a. } (t,\omega)$$

as source or sink via, e. g.,

- additive control $S_1(\omega, \mathbf{u}(t)) := \mathbf{1}_{\Omega_S}(\omega)\mathbf{u}(t)$, $\Omega_S \subseteq \Omega$
- bilinear control $S_2(\mathbf{u}(t), X(t,\omega)) := \mathbf{u}(t)X(t,\omega)$

Optimal Control of Reaction-Diffusion System

Controlled reaction-diffusion system

$$X_t(t,\omega) = a\Delta X(t,\omega) + bR(X(t,\omega)) + cS(\omega, \mathbf{u}(t), X(t,\omega)) \quad \text{for a. a. } (t,\omega)$$

Dirichlet boundary conditions with initial condition $X_0 : \Omega \to \mathbb{R}$

$$\begin{cases} X(t,\omega) = 0 & \text{on } [t_0, t_f] \times \partial\Omega \\ X(t_0,\omega) = X_0(\omega) & \text{on } \Omega, \end{cases}$$

Optimal Control of Reaction-Diffusion System

Controlled reaction-diffusion system

$$X_t(t,\omega) = a\Delta X(t,\omega) + bR(X(t,\omega)) + cS(\omega, \mathbf{u}(t), X(t,\omega)) \quad \text{for a. a. } (t,\omega)$$

Dirichlet boundary conditions

$$\begin{cases} X(t,\omega) = 0 & \text{on } [t_0, t_f] \times \partial\Omega \\ X(t_0,\omega) = X_0(\omega) & \text{on } \Omega, \end{cases}$$

Quadratic cost function with $\alpha, \beta, \gamma > 0$

$$J(\mathbf{u}, X) = \int_{t_0}^{t_f} \left[\alpha \left\| X(t,\cdot) - Y(\cdot) \right\|_{L^2(\Omega)}^2 + \beta |\mathbf{u}(t)|^2 \right] \mathrm{d}t + \gamma \left\| X(t_f,\cdot) - Y(\cdot) \right\|_{L^2(\Omega)}^2$$

steering X to target $Y : \Omega \to \mathbb{R}$ expending minimum control effort

Optimal Control of Reaction-Diffusion System

Controlled reaction-diffusion system

$$X_t(t,\omega) = a\Delta X(t,\omega) + bR(X(t,\omega)) + cS(\omega, \mathbf{u}(t), X(t,\omega)) \quad \text{for a. a. } (t,\omega)$$

Dirichlet boundary conditions

$$\begin{cases} X(t,\omega) = 0 & \text{on } [t_0, t_f] \times \partial\Omega \\ X(t_0,\omega) = X_0(\omega) & \text{on } \Omega, \end{cases}$$

Quadratic cost function

$$J(\mathbf{u}, X) = \int_{t_0}^{t_f} \left[\alpha \left\| X(t,\cdot) - Y(\cdot) \right\|_{L^2(\Omega)}^2 + \beta |\mathbf{u}(t)|^2\right] \mathrm{d}t + \gamma \left\| X(t_f,\cdot) - Y(\cdot) \right\|_{L^2(\Omega)}^2$$

→ transforming PDE- into ODE-constrained OCP via spatial discretization

Outline

Implementation and Experimental Set-up

Modular code for discretized OCP of reaction-diffusion system:

PROBLEM

time interval with $t_0 = 0$ and $t_f = 5$,
domain $\Omega = [0, 1]$ discretized with $n = 20$ mesh nodes,
reaction-diffusion system and quadratic cost function with respective coefficients

DATA GENERATION

initial condition domain
$\mathbb{X}_0 = [-1.5, 1.5]^n$,
upper bound $M = 1.25$

MODEL

ResNet of depth $L = 5$ and width $d = 100$

TRAINING

Adam with gradient regularization $\lambda = 100$,
convergence tolerance $C = 10^{-4}$

EVALUATION

monitoring of training and test errors, data set size and runtime,
simulations of NN and LQR controllers in presence of noise

Implementation and Experimental Set-up

Numerical experiments:

1. Noise in the Linear-Quadratic Problem
2. Non-linearities in the System Dynamics
3. Heuristic-Guided Training for Bilinear Control
4. Scalability across System Dimensions
5. BVP Solver Initialization for Data Generation Efficiency
6. Simulation for Out-of-Domain Initial Conditions

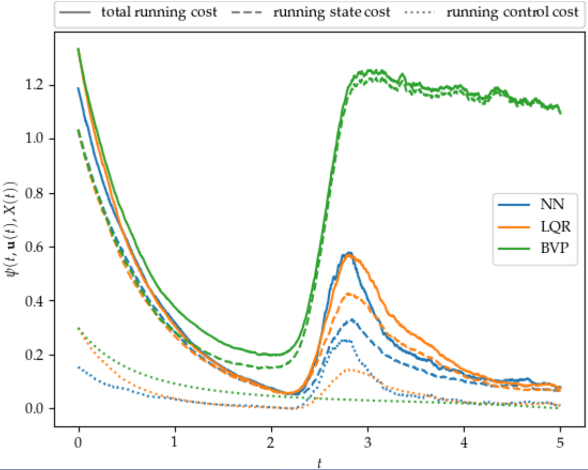
Implementation and Experimental Set-up

Numerical experiments:

1. Noise in the Linear-Quadratic Problem
2. Non-linearities in the System Dynamics
3. Heuristic-Guided Training for Bilinear Control
4. Scalability across System Dimensions
5. BVP Solver Initialization for Data Generation Efficiency
6. Simulation for Out-of-Domain Initial Conditions

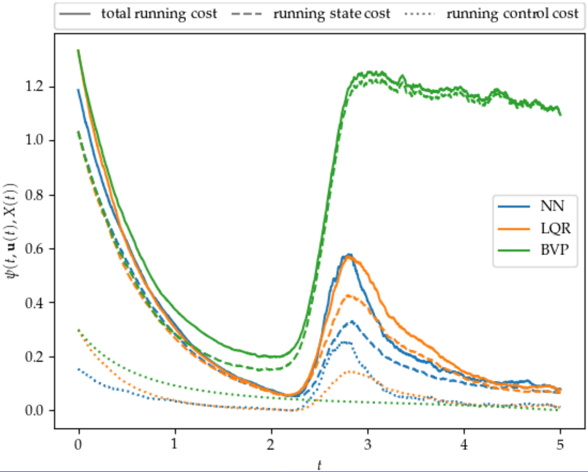
Noise in the Linear-Quadratic Problem

- consider linear-quadratic case, i. e., additive control without reaction
- analyse effect of Gaussian and shock noise
- simulate NN and LQR feedback, as well as open-loop BVP solution



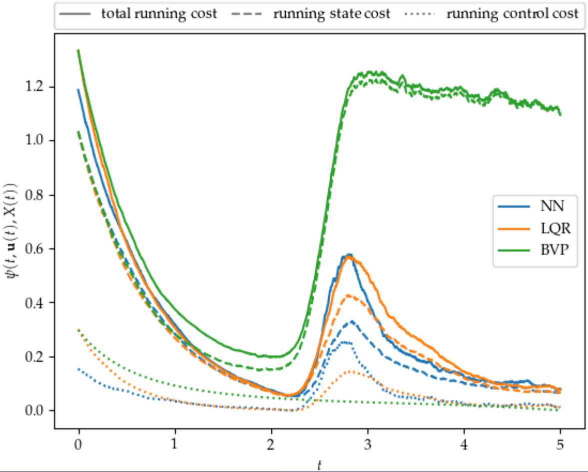
solution technique	total cost
NN	1.56
LQR	1.65
BVP	6.21

→ open-loop control is unable to respond to disturbances



solution technique	total cost
NN	1.56
LQR	1.65
BVP	6.21

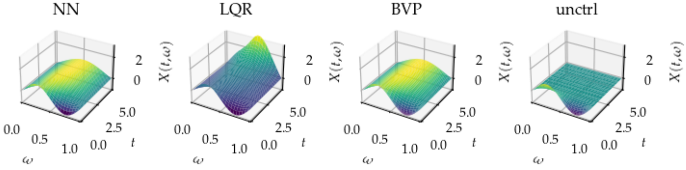
- $\rightarrow$ open-loop control is unable to respond to disturbances
- $\rightarrow$ NN and LQR feedback are robust to stochastic and deterministic noise, NN even more than LQR

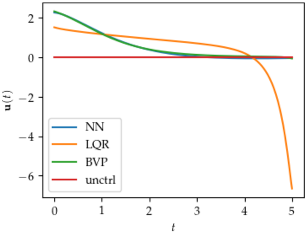


solution technique	total cost
NN	1.56
LQR	1.65
BVP	6.21

Non-linearities in the System Dynamics

- add non-linear reaction term to system, e. g., hyperbolic growth rate
- apply LQR to linearized state dynamics
- benchmark against optimal open-loop solution and uncontrolled system

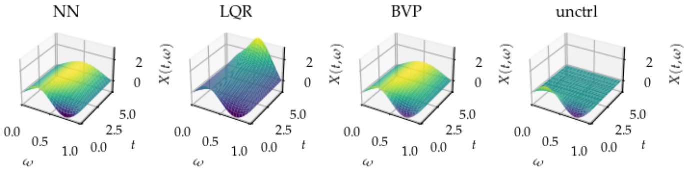


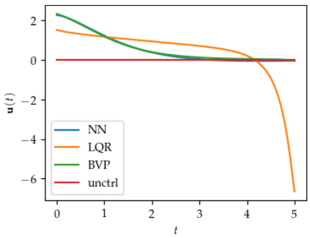


solution technique	total cost
NN	0.734
LQR	6.440
BVP	0.730
unctrl. system	3.687

Non-linearities in the System Dynamics

$\rightarrow$ linearization based LQR may even perform worse than uncontrolled system



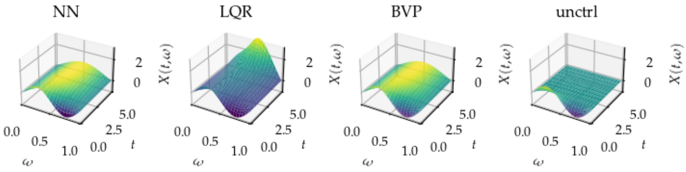


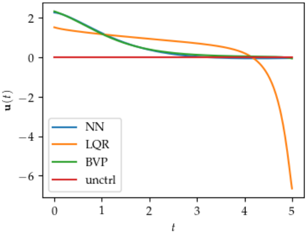
solution technique	total cost
NN	0.734
LQR	6.440
BVP	0.730
unctrl. system	3.687

Non-linearities in the System Dynamics

→ linearization based LQR may even perform worse than uncontrolled system

→ NN solution closely resembles optimal solution provided by BVP

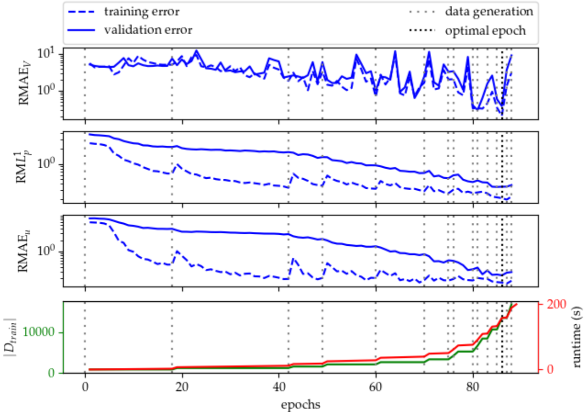




solution technique	total cost
NN	0.734
LQR	6.440
BVP	0.730
unctrl. system	3.687

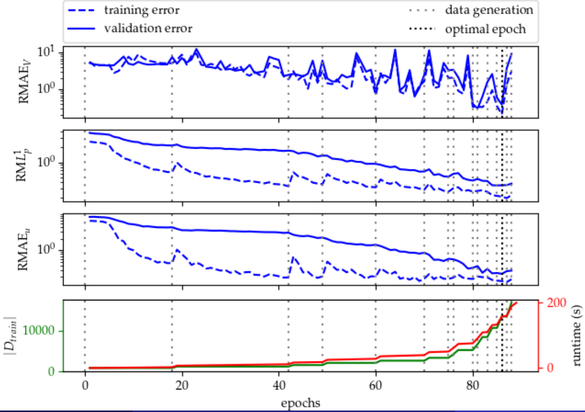
Heuristic-Guided Training for Bilinear Control

- replace additive by bilinear control
- monitor errors, amount of data and runtime during training phase
- evaluate effects of adaptive and progressive data generation



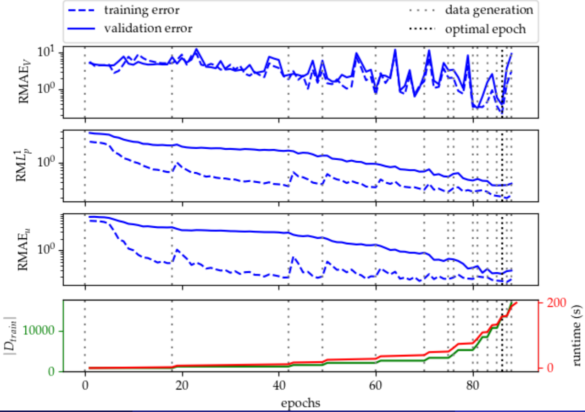
Heuristic-Guided Training for Bilinear Control

→ data expansion is delayed until significant increase of generalization error and becomes more frequent for model refinement



Heuristic-Guided Training for Bilinear Control

- → data expansion is delayed until significant increase of generalization error and becomes more frequent for model refinement
- → combination of stochastic optimization, gradient regularization, early stopping, dynamic data set size and adaptive sampling promises effective training



Scalability across System Dimensions

- refine spatial discretization yielding high dimensional states
- compute test errors for empirical verification of NN accuracy

limited training time of 100 s
(hyperbolic bilinear problem setting with zero target)

n	# data points	RMAE_u
5	11306	0.24
20	6626	0.27
40	4895	2.39

satisfied convergence criterion
(hyperbolic bilinear problem setting with quadratic target)

n	training time	RMAE_u
5	113 s	0.93
20	159 s	0.87
40	502 s	1.17

Scalability across System Dimensions

- refine spatial discretization yielding high dimensional states
- compute test errors for empirical verification of NN accuracy

limited training time of 100 s
(hyperbolic bilinear problem setting with zero target)

n	# data points	RMAE_u
5	11306	0.24
20	6626	0.27
40	4895	2.39

$\rightarrow$ accuracy declines as dimension grows

satisfied convergence criterion
(hyperbolic bilinear problem setting with quadratic target)

n	training time	RMAE_u
5	113 s	0.93
20	159 s	0.87
40	502 s	1.17

Scalability across System Dimensions

- refine spatial discretization yielding high dimensional states
- compute test errors for empirical verification of NN accuracy

limited training time of 100 s
(hyperbolic bilinear problem setting with zero target)

n	# data points	RMAE_u
5	11306	0.24
20	6626	0.27
40	4895	2.39

$\rightarrow$ accuracy declines as dimension grows

$\rightarrow$ less data-intensive due to slower overfitting

satisfied convergence criterion
(hyperbolic bilinear problem setting with quadratic target)

n	training time	RMAE_u
5	113 s	0.93
20	159 s	0.87
40	502 s	1.17

Scalability across System Dimensions

- refine spatial discretization yielding high dimensional states
- compute test errors for empirical verification of NN accuracy

limited training time of 100 s
(hyperbolic bilinear problem setting with zero target)

n	# data points	RMAE_u
5	11306	0.24
20	6626	0.27
40	4895	2.39

$\rightarrow$ accuracy declines as dimension grows

$\rightarrow$ less data-intensive due to slower overfitting

satisfied convergence criterion
(hyperbolic bilinear problem setting with quadratic target)

n	training time	RMAE_u
5	113 s	0.93
20	159 s	0.87
40	502 s	1.17

$\rightarrow$ high-fidelity models across all dimensions

Scalability across System Dimensions

- refine spatial discretization yielding high dimensional states
- compute test errors for empirical verification of NN accuracy

limited training time of 100 s
(hyperbolic bilinear problem setting
with zero target)

n	# data points	RMAE_u
5	11306	0.24
20	6626	0.27
40	4895	2.39

$\rightarrow$ accuracy declines as dimension grows

$\rightarrow$ less data-intensive due to slower overfitting

satisfied convergence criterion
(hyperbolic bilinear problem setting
with quadratic target)

n	training time	RMAE_u
5	113 s	0.93
20	159 s	0.87
40	502 s	1.17

$\rightarrow$ high-fidelity models across all dimensions

$\rightarrow$ computational effort scales reasonably

Scalability across System Dimensions

- refine spatial discretization yielding high dimensional states
- compute test errors for empirical verification of NN accuracy

limited training time of 100 s
(hyperbolic bilinear problem setting with zero target)

n	# data points	RMAE_u
5	11306	0.24
20	6626	0.27
40	4895	2.39

satisfied convergence criterion
(hyperbolic bilinear problem setting with quadratic target)

n	training time	RMAE_u
5	113 s	0.93
20	159 s	0.87
40	502 s	1.17

$\rightarrow$ DL method remains viable at least up to problem dimension 40

Outline

Conclusion and Future Research

Conclusion

- address challenge of feedback design for high-dimensional non-linear optimal control problems
- establish adaptive deep learning framework for solving Hamilton-Jacobi-Bellman equations
- conduct numerical analysis demonstrating scalability and robustness of neural network controllers

Conclusion and Future Research

Conclusion

- address challenge of feedback design for high-dimensional non-linear optimal control problems
- establish adaptive deep learning framework for solving Hamilton-Jacobi-Bellman equations
- conduct numerical analysis demonstrating scalability and robustness of neural network controllers

Future Research

- develop theoretical results to further enhance performance of neural network feedback
- investigate alternative models, optimization techniques and data generation methods
- refine heuristics for progressive data augmentation dependent on specific application
- tackle infinite-horizon or minimum-time problems, with state and control constraints, or infinite dimensional systems

References I

Byrd, R., Chin, G., Nocedal, J., and Wu, Y. (2012).
Sample Size Selection in Optimization Methods for Machine Learning.
Mathematical Programming, 134:127–155.

Kang, W. and Wilcox, L. C. (2015).
Mitigating the Curse of Dimensionality: Sparse Grid Characteristics Method for Optimal Feedback Control and HJB Equations.

Kunisch, Karl and Walter, Daniel (2021).
Semiglobal optimal feedback stabilization of autonomous systems via deep neural network approximation.
ESAIM: COCV, 27:16.

Liberzon, D. (2012).
Calculus of Variations and Optimal Control Theory: A Concise Introduction.
Princeton University Press.

References II

Nakamura-Zimmerer, T., Gong, Q., and Kang, W. (2021).
Adaptive Deep Learning for High-Dimensional Hamilton–Jacobi–Bellman Equations.
SIAM Journal on Scientific Computing, 43(2):A1221–A1247.

Prechelt, L. (2012).
Early Stopping — But When?
Springer Berlin Heidelberg.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2019).
Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations.
Journal of Computational Physics, 378:686–707.

Schättler, H. and Ledzewicz, U. (2012).
Geometric Optimal Control: Theory, Methods and Examples.
Interdisciplinary Applied Mathematics. Springer New York.

Questions

Thank you for your attention!

I am now available to answer any questions you may have.

Early Stopping

- performance measure E
  for tracking training and validation error at epoch i in training round j

  $$E_{\text{train}}(i) := E(\theta^i; D^j_{\text{train}}) \quad \text{and} \quad E_{\text{val}}(i) := E(\theta^i; D_{\text{val}}),$$

  determining generalization loss

  $$\mathrm{GL}(i) := \frac{E_{\text{val}}(i)}{\min_{1 \le i' \le i} E_{\text{val}}(i')} - 1$$

  and training progress for training strip of length ι

  $$\mathrm{TP}_\iota(i) := \frac{\sum_{i'=i-\iota+1}^{i} E_{\text{train}}(i')}{\iota \cdot \min_{i-\iota+1 \le i' \le i} E_{\text{train}}(i')} - 1$$

- trigger with threshold $T > 0$
  for detecting overfitting and interrupting training

  $$\text{stop at epoch } i \ge \iota \text{ if:} \quad \mathrm{TP}_\iota(i) = 0 \quad \text{or} \quad \frac{\mathrm{GL}(i)}{\mathrm{TP}_\iota(i)} > T$$

Key Strengths of AdaDL Algorithm

- exploiting prior information:
  value function BC, NN warm start, physics-informed learning
- using data efficiently:
  costate data usage, adaptive sampling, data generation delay
- promoting generalization:
  early stopping, gradient and L2 regularization, stochastic optimization
- reducing the sensitivity to hyperparameter tuning:
  Adam, ResNet, data set size heuristic

Spatial Discretization

- using central difference scheme on equidistant mesh $\{\omega_j\}_{j=1}^{n} \subset \Omega$ of size h

$$\dot{\mathbf{x}}_j(t) = \frac{a}{h^2}(\mathbf{x}_{j-1}(t) - 2\mathbf{x}_j(t) + \mathbf{x}_{j+1}(t)) + bR(\mathbf{x}_j(t)) + cS(\omega_j, \mathbf{u}(t), \mathbf{x}_j(t))$$

for a. a. t and all $j = 1, \ldots, n$

$\rightarrow$ dimension n of state $\mathbf{x} := (\mathbf{x}_1, \ldots, \mathbf{x}_n)^\top$ is scalable

- transforming into ODE-constrained OCP

$$\min_{(\mathbf{u},\mathbf{x}) \in \mathcal{U} \times \mathcal{X}} \int_{t_0}^{t_f} \left[\alpha h \left\|\mathbf{x}(t) - y\right\|_2^2 + \beta |\mathbf{u}(t)|^2\right] \, \mathrm{d}t + \gamma h \left\|\mathbf{x}(t_f) - y\right\|_2^2$$

subject to $\dot{\mathbf{x}}(t) = aQ\mathbf{x}(t) + bR(\mathbf{x}(t)) + cS(\mathbf{u}(t), \mathbf{x}(t))$ for a. a. t;

$\mathbf{x}(t_0) = x_0$

Boundary Value Problem and Neural Network Control

Boundary value problem specified for discretized OCP

$$
\begin{cases}
\dot{\mathbf{x}}(t) = aQ\mathbf{x}(t) + bR(\mathbf{x}(t)) + cS(\mathbf{u}(t), \mathbf{x}(t)) \\
\quad \text{for a. a. } t; \quad \mathbf{x}(t_0) = x_0, \\
\dot{\mathbf{p}}(t) = -2\alpha h(\mathbf{x}(t) - y) - aQ^\top \mathbf{p}(t) - bR_x^\top(\mathbf{x}(t))\mathbf{p}(t) - cS_x^\top(\mathbf{x}(t))\mathbf{p}(t) \\
\quad \text{for a. a. } t; \quad \mathbf{p}(t_f) = 2\gamma h(\mathbf{x}(t_f) - y), \\
\dot{\mathbf{v}}(t) = -\alpha h \left\|\mathbf{x}(t) - y\right\|_2^2 - \beta|\mathbf{u}(t)|^2 \\
\quad \text{for a. a. } t; \quad \mathbf{v}(t_f) = \gamma h \left\|\mathbf{x}(t_f) - y\right\|_2^2
\end{cases}
$$

Boundary Value Problem and Neural Network Control

Boundary value problem specified for discretized OCP

$$\begin{cases} \dot{\mathbf{x}}(t) = aQ\mathbf{x}(t) + bR(\mathbf{x}(t)) + cS(\mathbf{u}(t), \mathbf{x}(t)) \\ \quad \text{for a. a. } t; \quad \mathbf{x}(t_0) = x_0, \\ \dot{\mathbf{p}}(t) = -2\alpha h(\mathbf{x}(t) - y) - aQ^\top \mathbf{p}(t) - bR_x^\top(\mathbf{x}(t))\mathbf{p}(t) - cS_x^\top(\mathbf{x}(t))\mathbf{p}(t) \\ \quad \text{for a. a. } t; \quad \mathbf{p}(t_f) = 2\gamma h(\mathbf{x}(t_f) - y), \\ \dot{\mathbf{v}}(t) = -\alpha h \left\|\mathbf{x}(t) - y\right\|_2^2 - \beta|\mathbf{u}(t)|^2 \\ \quad \text{for a. a. } t; \quad \mathbf{v}(t_f) = \gamma h \left\|\mathbf{x}(t_f) - y\right\|_2^2 \end{cases}$$

Candidate open-loop solution by Hamiltonian minimization condition

$$\mathbf{u}^*(t; x_0) \in \underset{u \in \mathbb{R}}{\operatorname{argmin}} \{\beta u^2 + c\mathbf{p}^*(t) \cdot S(u, \mathbf{x}^*(t))\} \quad \text{for a. a. } t$$

Boundary Value Problem and Neural Network Control

Boundary value problem specified for discretized OCP

$$
\begin{cases}
\dot{\mathbf{x}}(t) = aQ\mathbf{x}(t) + bR(\mathbf{x}(t)) + cS(\mathbf{u}(t), \mathbf{x}(t)) \\
\quad \text{for a. a. } t; \quad \mathbf{x}(t_0) = x_0, \\
\dot{\mathbf{p}}(t) = -2\alpha h(\mathbf{x}(t) - y) - aQ^\top \mathbf{p}(t) - bR_x^\top(\mathbf{x}(t))\mathbf{p}(t) - cS_x^\top(\mathbf{x}(t))\mathbf{p}(t) \\
\quad \text{for a. a. } t; \quad \mathbf{p}(t_f) = 2\gamma h(\mathbf{x}(t_f) - y), \\
\dot{\mathbf{v}}(t) = -\alpha h \left\| \mathbf{x}(t) - y \right\|_2^2 - \beta |\mathbf{u}(t)|^2 \\
\quad \text{for a. a. } t; \quad \mathbf{v}(t_f) = \gamma h \left\| \mathbf{x}(t_f) - y \right\|_2^2
\end{cases}
$$

Neural Network feedback controller

$$
\mathbf{u}^{\mathsf{NN}}(t, x) \in \operatorname*{argmin}_{u \in \mathbb{R}} \{\beta u^2 + cV_x^{\mathsf{NN}}(t, x) \cdot S(u, x)\} \quad \text{for all } t \text{ and } x
$$

Linear-Quadratic Regulator

LQR feedback controller computed via Riccati differential equation
$\rightarrow$ as baseline for evaluating NN

Linear-Quadratic Regulator

LQR feedback controller computed via Riccati differential equation
$\rightarrow$ as baseline for evaluating NN

- consider $R \equiv 0$ and $S = S_1$, i. e., linear system

$$\dot{\mathbf{x}}(t) = aQ\mathbf{x}(t) + c\mathbb{1}_{\Omega_S}\mathbf{u}(t)$$

$\rightarrow$ analytical solution for linear-quadratic problem

Linear-Quadratic Regulator

LQR feedback controller computed via Riccati differential equation
$\rightarrow$ as baseline for evaluating NN

- consider $R \equiv 0$ and $S = S_1$, i. e., linear system

$$\dot{\mathbf{x}}(t) = aQ\mathbf{x}(t) + c\mathbb{1}_{\Omega_S}\mathbf{u}(t)$$

  $\rightarrow$ analytical solution for linear-quadratic problem

- consider non-linear $R = R_1$ or R_2 and $S = S_1$, i. e.,

$$\dot{\mathbf{x}}(t) = aQ\mathbf{x}(t) + b\mathbf{x}(t) \odot \mathbf{x}(t) + c\mathbb{1}_{\Omega_S}\mathbf{u}(t) \quad \text{or}$$
$$\dot{\mathbf{x}}(t) = aQ\mathbf{x}(t) + b\mathbf{x}(t) \odot (\mathbb{1} - \mathbf{x}(t)) + c\mathbb{1}_{\Omega_S}\mathbf{u}(t)$$

Linear-Quadratic Regulator

LQR feedback controller computed via Riccati differential equation
$\rightarrow$ as baseline for evaluating NN

- consider $R \equiv 0$ and $S = S_1$, i. e., linear system

$$\dot{\mathbf{x}}(t) = aQ\mathbf{x}(t) + c\mathbb{1}_{\Omega_S}\mathbf{u}(t)$$

  $\rightarrow$ analytical solution for linear-quadratic problem

- consider linearized $R = R_1$ or R_2 and $S = S_1$, i. e.,

$$\begin{aligned}\dot{\mathbf{x}}(t) &= aQ\mathbf{x}(t) + c\mathbb{1}_{\Omega_S}\mathbf{u}(t) \quad \text{or}\\ \dot{\mathbf{x}}(t) &= (aQ + bI)\mathbf{x}(t) + c\mathbb{1}_{\Omega_S}\mathbf{u}(t)\end{aligned}$$

  $\rightarrow$ approximate solution via linearization of state dynamics

BVP Solver Initialization for Data Generation Efficiency

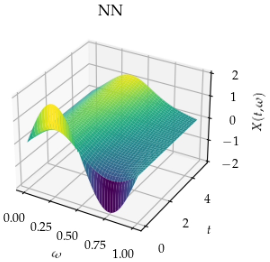
- investigate dependency of time-marching on time sequence and NN warm start on model accuracy
- determine initialization strategy, for problems of varying difficulty and at different training stages, yielding best trade-off of convergence rate and speed

→ improved convergence attained through sophisticated techniques, i. e., time-marching with tuned adaptive intervals and NN warm start even with briefly trained low-fidelity models

→ speed of NN warm start scales much better with problem dimension than time-marching

Simulation for Out-of-Domain Initial Conditions

- confront NN with data outside its training domain by upscaling the initial state value



solution technique	total cost
NN	3.02
BVP	1.66

$\rightarrow$ NN model produces acceptable feedback controls, preventing blow-ups but with increased costs

$\rightarrow$ DL method may handle scenarios in which disturbances propel state beyond learned domain