Actor-Critic Methods using Physics-Informed Neural Networks: Control of a 1D PDE Model for Fluid-Cooled Battery Packs

Amartya Mukherjee [*1] **Jun Liu** [1]

Abstract

This paper proposes an actor-critic algorithm for controlling the temperature of a battery pack using a cooling fluid. This is modeled by a coupled 1D partial differential equation (PDE) with a controlled advection term that determines the speed of the cooling fluid. The Hamilton-Jacobi-Bellman (HJB) equation is a PDE that evaluates the optimality of the value function and determines an optimal controller. We propose an algorithm that treats the value network as a Physics-Informed Neural Network (PINN) to solve the continuous-time HJB equation rather than a discrete-time Bellman optimality equation, and we derive a control function from the HJB equation. Our experiments show that a hybrid-policy method that updates the value network using the HJB equation and updates the policy network identically to PPO achieves the best results in the control of this PDE system.

1. Introduction

In recent years, there has been a growing interest in Reinforcement Learning (RL) for continuous control problems. RL has shown promising results in environments with unknown dynamics through a balance of exploration in the environment and exploitation of the learned policies. Since the advent of REINFORCE with Baseline, the value network in RL algorithms has shown to be useful towards finding optimal policies as a critic network (Sutton & Barto, 2018). This value network continues to be used in state-of-the-art RL algorithms today.

Proximal Policy Optimization (PPO) is an actor-critic method introduced by Schulman et al. (2017). It limits the update of the policy network to a trust region at every iteration. This ensures that the objective function of the policy network is a good approximation of the true objective function and forces smooth and reliable updates to the value network as well.

In discrete-time RL, the value function estimates returns from a given state as a sum of the returns over time steps. This value function is obtained by solving the Bellman Optimality Equation. On the other hand, in continuous-time RL, the value function estimates returns from a given state as an integral over time. This value function is obtained by solving a partial differential equation (PDE) known as the Hamilton-Jacobi-Bellman (HJB) equation (Munos, 1999). Both equations are difficult to solve analytically and numerically, and therefore the RL agent must explore the environment and make successive estimations.

The introduction of physics-informed neural networks (PINNs) by Raissi et al. (2019) has led to significant advancements in scientific machine learning. PINNs leverage auto-differentiation to compute derivatives of neural networks with respect to their inputs and model parameters exactly. This enables the laws of physics (described by ODEs or PDEs) governing the dataset of interest to act as a regularization term for the neural network. As a result, PINNs outperform regular neural networks on such datasets by exploiting the underlying physics of the data.

Control of PDEs is considered to be challenging compared to control of ODEs. Works such as Vazquez & Krstic (2017) introduced the backstepping method for the boundary control of reaction-advection-diffusion equations using kernels. For PDE control problems where the control input is encoded in the PDE, the HJB equation has been used ((Sirignano & Spiliopoulos, 2018),(Kalise & Kunisch, 2017)). Works from control of ODEs have been used by writing the PDE as an infinite-dimensional ODE.

To the best of our knowledge, this paper is the first to explore the intersection between PINNs and RL in a PDE control problem. We discretize the PDE as an ODE to derive an HJB equation. In order to facilitate the convergence of the value network in PPO towards the solution of the HJB equation, we utilize PINNs to encode this PDE and train the value

[*]Equal contribution [1]Department of Applied Mathematics, University of Waterloo, Waterloo, Ontario, Canada. Correspondence to: Amartya Mukherjee <a29mukhe@uwaterloo.ca>.

Workshop on New Frontiers in Learning, Control, and Dynamical Systems at the International Conference on Machine Learning (ICML), Honolulu, Hawaii, USA, 2023.

network. Upon deriving the HJB equation, we also derive an optimal controller. We introduce two algorithms: HJB value iteration and Hamilton-Jacobi-Bellman Proximal Policy Optimization (HJBPPO), that train the value function using the HJB equation and use the optimal controller. The HJBPPO algorithm shows superior performance compared to PPO and HJB value iteration on the PackCooling environment.

2. Preliminaries

2.1. The 1D pack cooling problem

The 1D system for fluid-cooled battery packs was introduced by Kato & Moura (2021) and is modeled by the following coupled PDE:

$$u_t(x,t) = -D(x,t)u_{xx}(x,t) + h(x,t,u(x,t)) + \frac{1}{R(x,t)}(w-u) \tag{1}$$

$$w_t = -\sigma(t)w_x + \frac{1}{R(x,t)}(u-w), \tag{2}$$

with the following boundary conditions:

$$u_x(0,t) = u_x(1,t) = 0 \tag{3}$$

$$w(0,t) = U(t) \tag{4}$$

where $u(x,t)$ is the heat distribution across the battery pack, $w(x,t)$ is the heat distribution across the cooling fluid, $D(x,t)$ is the thermal diffusion constant across the battery pack, $R(x,t)$ is the heat resistance between the battery pack and the cooling fluid, $h(x,t,u)$ is the internal heat generation in the battery pack, $U(t)$ is the temperature of the cooling fluid at the boundary, and $\sigma(t)$ is the transport speed of the cooling fluid, which will be the controller in this paper.

The objective of the control problem in this paper is to determine $\sigma(t)$ such that $u(x,t)$ is as close to zero as possible. The transport speed $\sigma(t)$ is strictly non-negative so the cooling fluid travels only in the positive x-direction. We restrict $\sigma(t)$ to $[0,1]$.

2.2. Hamilton-Jacobi-Bellman equation

To achieve optimal control for the 1D PDE pack cooling problem, we will utilize works from control theory for ODEs. Consider a controlled dynamical system modeled by the following equation:

$$\dot{x} = f(x,\sigma), \quad x(t_0) = x_0, \tag{5}$$

where $x(t)$ is the state and $\sigma(t)$ is the control input. In control theory, the optimal value function $V^*(x)$ is useful towards finding a solution to control problems (Munos et al., 1999):

$$V^*(x) = \sup_{\sigma} \frac{1}{\Delta t} \int_{t_0}^{\infty} \gamma^{\frac{t}{\Delta t}} L(x(\tau; t_0, x_0, \sigma(\cdot)), \sigma(\tau)) d\tau, \tag{6}$$

where $L(x,\sigma)$ is the reward function, Δt is the time step size for numerical simulation, and γ is the discount factor. The following theorem introduces a criteria for assessing the optimality of the value function ((Liberzon, 2012), (Kamalapurkar et al., 2018)).

Theorem 2.1. *A function $V(x)$ is the optimal value function if and only if:*

1. *$V \in C^1(\mathbb{R}^n)$ and V satisfies the Hamilton-Jacobi-Bellman (HJB) Equation*

$$(\gamma - 1)V(x) + \sup_{\sigma \in U}\{L(x,\sigma) + \gamma \Delta t \nabla_x V^T(x) f(x,\sigma)\} = 0 \tag{7}$$

   for all $x \in \mathbb{R}^n$.

2. *For all $x \in \mathbb{R}^n$, there exists a controller $\sigma^*(\cdot)$ such that:*

$$\begin{aligned}&(\gamma - 1)V(x) + L(x,\sigma^*(x)) + \gamma \Delta t \nabla_x V^T(x) f(x,\sigma^*(x)) \\ &= (\gamma - 1)V(x) + \sup_{\hat{\sigma} \in U}\{L(x,\hat{\sigma}) + \gamma \Delta t \nabla_x V^T(x) f(x,\hat{\sigma})\}.\end{aligned} \tag{8}$$

For completeness, the proof of part 1 of this known result is provided in Appendix B. The HJB equation will be used in this paper to determine a new loss function for the value network $V(x)$ in this pack cooling problem and an optimal controller $\sigma^*(t)$.

3. Related work

The HJB equation we intend to solve is a first-order quasi-linear PDE. The use of HJB equations for continuous RL has sparked interest in recent years among the RL community as well as the control theory community and has led to promising works. Kim et al. (2021) introduced an HJB equation for Q Networks and used it to derive a controller that is Lipschitz continuous in time. This algorithm has shown improved performance over Deep Deterministic Policy Gradient (DDPG) in three out of the four tested MuJoCo environments without the need for an actor network. Wiltzer et al. (2022) introduced a distributional HJB equation to train the FD-WGF Q-Learning algorithm. This models return distributions more accurately compared to Quantile Regression TD (QTD) for a particle-control task.

Finite difference methods are used to solve this HJB equation numerically. Furthermore, the authors mentioned the use of auto-differentiation for increased accuracy of the distributional HJB equation as a potential area for future research in their conclusion.

The use of neural networks to solve the HJB equation has been an area of interest across multiple research projects. Jiang et al. (2016) uses a structured Recurrent Neural Network to solve the HJB equation and achieve optimal control for the Dubins car problem. Tassa & Erez (2007) uses the Pineda architecture (Pineda, 1987) to estimate partial derivatives of the value function with respect to its inputs. They used the iterative least squares method to solve the HJB equation. This algorithm shows convergence in several control problems without the need for an initial stable policy.

RL for PDE control is a challenging field that has been of interest to the machine learning community lately. Farahmand et al. (2017) introduces the Deep Fitted Q Iteration to solve a boundary control problem for a 2D convection-diffusion equation. The model stabilizes the temperature in the environment without encoding any knowledge of the governing PDE. Sirignano & Spiliopoulos (2018) develops the DGM algorithm to solve PDEs. They use auto-differentiation to compute first-order derivatives and Monte Carlo methods to estimate higher-order derivatives. This algorithm was used to solve the HJB equation to control a stochastic heat equation and achieved an error of 0.1%. Kalise & Kunisch (2017) approximates the solution to the HJB equation using polynomials. This was used to control a semilinear parabolic PDE.

PINNs have been used for the control of dynamical systems in recent works. Antonelo et al. (2021) uses a PINN for model predictive control of a dynamical system over a long time interval. The PINN takes the initial condition, the control input, and the spatial and temporal coordinates as input and estimates the trajectory of the dynamical system while repeatedly shifting the time interval towards zero to allow for long-range interval predictions. Nicodemus et al. (2022) uses a PINN-based model predictive control for the tracking problem of a multi-link manipulator. Djeumou et al. (2022) uses a PINN to incorporate partial knowledge about a dynamical system such as symmetry and equilibrium points to estimate the trajectory of a controlled dynamical system.

The use of a PINN to solve the HJB equation for the value network was done by Nakamura-Zimmerer et al. (2020) in an optimal feedback control problem setting. The paper achieves results similar to that of the true optimal control function in high-dimensional problems.

4. HJB control of the pack cooling problem

In this section, we will connect the pack cooling PDE model with the HJB equation to derive a new loss function for the value network $V(u, w)$ using the HJB equation and an optimal controller. The HJB equation has been useful in finding optimal controllers for systems modeled by ODEs. In (Kalise & Kunisch, 2017), the controlled PDE system has been discretized in space to form an ODE that can be used in the HJB equation. Similarly, to form the HJB equation for this paper, we need to write equations 1 and 2 as an ODE.

4.1. ODE discretization of PDE

We can write equations 1 and 2 as an ODE by discretizing it in the x variable. By letting $\Delta x = \frac{1}{N_x}$ where N_x is the number of points we choose to discretize the system along the x-axis, we arrive at a $2N_x$ dimensional ODE:

$$\dot{\hat{U}} = -DA\hat{U} + h(\hat{U}) + \frac{1}{R}(\hat{W} - \hat{U}) \tag{9}$$

$$\dot{\hat{W}} = -\sigma(t)B\hat{W} + \frac{1}{R}(\hat{U} - \hat{W}), \tag{10}$$

where

$$\hat{W}(t) = \begin{pmatrix} w(x_1, t) \\ \vdots \\ w(x_{N_x}, t) \end{pmatrix}, \hat{U}(t) = \begin{pmatrix} u(x_1, t) \\ \vdots \\ u(x_{N_x}, t) \end{pmatrix},$$

and $A\hat{U}$ is a second-order discretization of u_{xx}, e.g.,

$$[A\hat{U}]_k = \frac{u(x_{k+1}, t) - 2u(x_k, t) + u(x_{k-1}, t)}{\Delta x^2},$$

$B\hat{W}$ is a second-order discretization of w_x, e.g.,

$$[B\hat{W}]_k = \frac{w(x_{k+1}, t) - w(x_{k-1}, t)}{2\Delta x}.$$

4.2. Derivation of the optimal controller

The ODE system derived in section 4.1 can be used in the HJB equation to determine a loss function and an optimal controller.

Theorem 4.1. *Let $u(\cdot, t), w(\cdot, t) \in L_2[0, 1]$. With $\sigma(t) \in [0, 1]$ and the reward function $L(U_t, W_t, \sigma_t) = -||U_{t+1}||_2^2 \Delta x$, the HJB equation for the 1D pack cooling problem is:*

$$\begin{aligned} &(\gamma - 1)V - ||u(\cdot, t + \Delta t)||^2 \\ &+ \langle V_u(u(\cdot, t), w(\cdot, t)), u_t(\cdot, t)\rangle \\ &+ \frac{1}{R}\langle V_w(u(\cdot, t), w(\cdot, t)), u(\cdot, t) - w(\cdot, t)\rangle \\ &+ \max(0, -\langle V_w(u(\cdot, t), w(\cdot, t)), w_x(\cdot, t)\rangle) = 0 \end{aligned} \tag{11}$$

where $||\cdot||$ is the $L_2[0, 1]$ norm and $\langle\cdot, \cdot\rangle$ is the $L_2[0, 1]$ inner product.

The proof of this theorem is in Appendix C. Theorem 2.1 shows that there exists a controller that satisfies equation 8. This allows us to determine an optimal controller, as shown in the following corollary:

Corollary 4.2. *Let $w(\cdot,t) \in L_2[0,1]$. With $\sigma(t) \in [0,1]$ and the reward function $L(U_t, W_t, \sigma_t) = -||U_{t+1}||_2^2 \Delta x$, provided the optimal value function $V^*(u,w)$ with $V_w^*(\cdot,t) \in L_2[0,1]$, the optimal controller for the 1D pack cooling problem is:*

$$\sigma^*(t) = \begin{cases} 1, & \langle V_w^*(u(\cdot,t), w(\cdot,t)), w_x(\cdot,t)\rangle < 0, \\ 0, & \textit{otherwise}, \end{cases} \tag{12}$$

where $\langle \cdot,\cdot \rangle$ is the $L_2[0,1]$ inner product.

The proof of this corollary is in Appendix D. These results will be used in our algorithms to achieve optimal control of the pack cooling problem.

5. Algorithm

For the control of the PDE, we introduce two algorithms. The first algorithm, called HJB Value Iteration, uses only a value network and exploits the HJB equation and optimal controller derived in Theorem 4.1 and Corollary 4.2. The second algorithm, called HJBPPO, is a hybrid-policy method that uses policy network updates from PPO and value network updates from HJB Value Iteration.

To define these algorithms, we first define two loss functions. The first loss function is derived from the proof of theorem 4.1.

$$\begin{aligned} MSE_f = \frac{1}{T}\sum_{t=0}^{T-1} ((\gamma-1)V(\hat{U}_t, \hat{W}_t) \\ - ||\hat{U}_{t+1}||_2^2 \Delta x \\ + \nabla_U V^T(\hat{U}_t, \hat{W}_t)\dot{\hat{U}}_t \Delta t \\ + \frac{1}{R}\nabla_W V^T(\hat{U}_t - \hat{W}_t)\Delta t \\ + \max(0, -\nabla_W V^T B\hat{W}))^2 \Delta t \end{aligned} \tag{13}$$

The second loss function provides an initial condition. At $u(x,T) = 0, w(x,T) = -R(x,t) = -2$, we have: $u(x,T) = 0$ and $u_t(x,T) = 0$. As a result, using the reward function L from theorem 4.1, we have $L(0, -R(x,t)) = 0$ and $L_t(0, -R(x,t)) = 0$. This shows us that $u(x,T) = 0, w(x,T) = -R(x,t)$ is considered a stable point that maximizes the reward. Thus, we choose to let $V(0, -R(x,t)) = 0$ be the Dirichlet boundary condition for the HJB equation. This leads to the second loss function:

$$MSE_u = (V(0, -R(x,t)))^2. \tag{14}$$

Algorithm 1 HJB Value Iteration

1: Initiate value network parameter ϕ
2: Run the control as given in equation (16) in the environment for T timesteps and observe samples $\{(s_t, a_t, R_t, s_{t+1})\}_{t=1}^T$, where s_t is the concatenation of U_t and W_t.
3: Compute the value network loss as: $J(\phi) = MSE_f + MSE_u + MSE_n$ described in equations (13), (14), and (15)
4: Update $\phi \leftarrow \phi - \alpha_2 \nabla_\phi J(\phi)$
5: Run steps 2–4 for multiple iterations

Since the value function achieves its global maximum at $u(x,T) = 0, w(x,T) = -2$, this means the derivatives of V must be zero along all directions. Thus, we choose to let $\frac{\partial V}{\partial n} = 0$ at $u(x,T) = 0, w(x,T) = -R(x,t)$ along every normal be the Neumann boundary condition for the HJB equation. This leads to the third loss function:

$$MSE_n = ||\nabla_U V(0, -R(x,t))||_2^2 + ||\nabla_W V(0, -R(x,t))||_2^2 \tag{15}$$

We derived an optimal controller in corollary 4.2. Gym environments recommend that actions be in the range $[-1, 1]$. We can use the proof of the optimal controller in Appendix D to derive a way of selecting actions:

$$a_t = -\text{sign}(\nabla_W V^T B\hat{W}(t)) \tag{16}$$

The algorithms introduced in this paper will focus on minimizing both of the loss functions defined above and using the optimal controller.

5.1. HJB value iteration

The HJB Value Iteration trains the loss function without the need for an actor-network. We treat the value network as a PINN, using auto-differentiation to estimate gradient vectors to compute the loss in equation 13 and the control in equation 16. At every time step, it uses the controller given in equation 16. It updates the value network using the loss functions as shown above. The method is provided in Algorithm 1.

5.2. HJBPPO

HJBPPO is an algorithm that combines policy optimization from PPO with HJB value iteration. This is implemented by modifying the PPO implementation by Barhate (2021).

To facilitate the exploration of the environment and exploitation of the models, we introduce an action selection method that uses the policy network and equation 16 with equal

Algorithm 2 HJBPPO action selection

Retrieve state s_t, policy network parameter θ and value network parameter ϕ
Sample $i \in \{0, 1\}$
if $i = 0$ **then**
    Select controller based on equation (16)
else
    Run policy π_θ
end

Algorithm 3 HJBPPO

Initiate policy network parameter θ and value network parameter ϕ
Run action selection as given in algorithm 2 in the environment for T timesteps and observe samples $\{(s_t, a_t, R_t, s_{t+1})\}_{t=1}^{T}$, where s_t is the concatenation of U_t and W_t.
Compute the advantage A_t
Compute $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$
Compute the objective function of the policy network:

$$L(\theta) = \frac{1}{T}\sum_{t=0}^{T-1} \min[r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t],$$

Update $\theta \leftarrow \theta + \alpha_1 \nabla_\theta L(\theta)$
Compute the value network loss as: $J(\phi) = MSE_f + MSE_u + MSE_n$ described in equations (13), (14), and (15)
Update $\phi \leftarrow \phi - \alpha_2 \nabla_\phi J(\phi)$
Run steps 2–8 for multiple iterations

probability, as shown in Algorithm 2. Upon running the policy π_θ, we sample from a distribution $N(\mu, s)$ where μ is the output from the policy network. We initiate s to 0.3 and decrease it by 0.01 every 1000 episodes until it reaches 0.1. After sampling an action from the normal distribution, we clip it between -1 and 1.

This action selection method ensures that we select actions that are not only in $\{-1, 1\}$ but also in $[-1, 1]$. It introduces a new method of exploration of the environment by choosing from two different methods of action selection. Actions selected using equation 16 are also stored in the memory buffer and are used to train the policy network π_θ. The method is provided in Algorithm 3.

We will train PPO, HJB value iteration, and HJBPPO on the PackCooling environment and compare these algorithms.

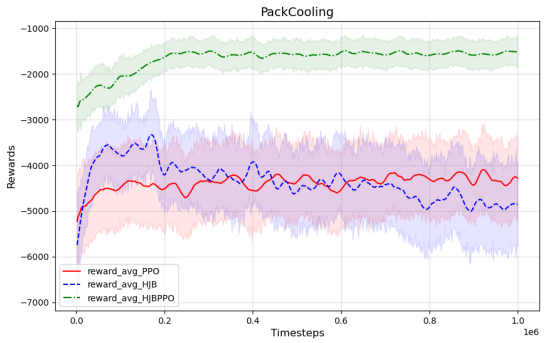

Figure 1. Reward curves of PPO (red), HJB value iteration (blue), and HJBPPO (green) averaged over 5 seeds. Shaded area indicates 0.2 standard deviations.

6. Results

6.1. Training

To ensure the reproducibility of our results, we have posted our code in the following link:

https://github.com/amartyamukherjee/PPO-PackCooling. We posted our hyperparameters in Appendix E. The details of the implementation of the PackCooling gym environment are posted in Appendix A. The code was run using Kaggle CPUs. Each algorithm was trained for a million timesteps. Training each algorithm took approximately 5 hours.

6.2. Reward Curves

The reward curves have been plotted in Figure 1, comparing PPO, HJB value iteration, and HJBPPO. Each algorithm was run for 5 different seeds. We plotted the mean reward over each seed and over 20 consecutive episodes, and shaded the area 0.2 standard deviations from the mean. HJB value iteration shows the worst performance, as its rewards decrease past PPO after training for multiple episodes. PPO shows a rapid increase in average rewards after the first episode and a slow increase in average rewards afterward. HJBPPO shows the best performance in the graph, achieving the highest average reward in each episode and an increase in average rewards after training for multiple episodes.

The significantly higher average reward in HJBPPO in the first episode shows that the action selection method described in Algorithm 2 provides a robust strategy to explore the environment and train the models. The higher average rewards are due to the exploitation of the dynamics of the environment as done by the HJB equation.

6.3. Trajectories

The plots of the trajectories have been posted in Appendix F. After training for a million timesteps, we tested our models on the PackCooling environment and produced the plots. These plots were generated using the rendering feature explained in section A.4.

The trajectory of HJB value iteration shows the worst results. $\sigma(t)$ returns 1.0 only once. It achieves a cumulative reward of -7294.51. Thus, the input of the cooling fluid from the boundary is minimal. As a result of the internal heat generation in the battery pack, $u(x,t)$ reaches high values of roughly 5 at $t = 10$, and as a result, $w(x,t)$ also reaches high values of roughly 4. This shows that the training of the value function in HJB value iteration is inadequate and we have not arrived at an optimal controller for the pack cooling problem. This is because exploration of the environment was at a minimum, as we only exploited equation 16 at each time step.

The trajectory of PPO shows that the values $\sigma(t)$ takes at every timestep have a large variance with its neighboring timesteps. It achieves a cumulative reward of -3970.02. Control of the temperature of the battery pack has been achieved as $u(x,t)$ takes values between -2 and 2 at $t = 10$.

The trajectory of u(x,t) with HJBPPO shows it takes values between -2 and 2 at $t = 10$. The values $\sigma(t)$ takes at every timestep have a lower variance with its neighboring timesteps compared to PPO. It achieves a cumulative reward of -881.55. For $t \in [4,6]$, $u(x,t)$ shows an increasing trend towards $u = 2$. In response, the controller $\sigma(t)$ took values closer to 1.0 to allow for greater input of cooling fluid from the boundary so that $u(x,t)$ decreases towards zero. Due to higher average rewards as shown in Figure 1, this shows that a model that exploits the dynamics of the environment to return a controller shows improved performance compared to a model that returns noisy control centered at $\sigma = 0.5$.

7. Conclusion

In this paper, we have introduced two algorithms that use PINNs to solve the pack cooling problem. This paper combines PINNs with RL in a PDE control setting. In the HJB value iteration algorithm, the HJB equation is used to introduce a loss function and a controller using a value network. The HJBPPO algorithm is a hybrid-policy method that combines the training of the value network from HJB value iteration and the training of the policy network from PPO. HJBPPO shows an overall improvement in performance compared to PPO due to its ability to exploit the physics of the environment to improve the learning curve of the agent.

8. Future research

Despite showing an overall improvement in the reward curves, the HJBPPO algorithm leaves room for improved RL algorithms using PINNs.

In this paper, we computed the HJB equation by expressing the PDE as an ODE by discretizing in x. This was possible because the pack cooling problem was modeled by 1D PDEs. Currently existing works such as (Sirignano & Spiliopoulos, 2018) and (Kalise & Kunisch, 2017) solve the HJB equation for 1D PDEs by discretizing it in x. It will be interesting to see how HJB control can be extended to higher dimensional PDEs.

The goal of PINNs is to solve PDEs without the need for numerical methods. In this paper, we solved the pack cooling problem numerically using the Crank-Nicolson method and the method of characteristics. An area for further research may be the use of PINNs to solve for the HJB equation and the PDE that governs the dynamics of the system.

In the PackCooling environment, the HJBPPO algorithm showed an improvement compared to PPO. But this is due to the fact that we knew the dynamics of the system, thus allowing for the physics of the environment to be exploited. The environments give all the details of the state needed to choose an action. One limitation of HJBPPO is that it may not perform well in partially observable environments because the estimate of the dynamics of the system may be inaccurate. Deep Transformer Q Network (DTQN) was introduced by Esslinger et al. (2022) and achieves state-of-the-art results in many partially observable environments. A potential area for further research may be the introduction of an HJB equation that facilitates partial observability. The DTQN algorithm may be improved by incorporating this HJB equation using PINNs.

References

Antonelo, E. A., Camponogara, E., Seman, L. O., de Souza, E. R., Jordanou, J. P., and Hubner, J. F. Physics-informed neural nets for control of dynamical systems. *arXiv preprint arXiv:2104.02556*, 2021.

Barhate, N. Minimal pytorch implementation of proximal policy optimization. https://github.com/nikhilbarhate99/PPO-PyTorch, 2021.

Djeumou, F., Neary, C., Goubault, E., Putot, S., and Topcu, U. Neural networks with physics-informed architectures and constraints for dynamical systems modeling. In *Learning for Dynamics and Control Conference*, pp. 263–277. PMLR, 2022.

Esslinger, K., Platt, R., and Amato, C. Deep transformer q-

networks for partially observable reinforcement learning. *Preprint arXiv:2206.01078*, 2022.

Farahmand, A.-m., Nabi, S., and Nikovski, D. N. Deep reinforcement learning for partial differential equation control. In *2017 American Control Conference (ACC)*, pp. 3120–3127, 2017. doi: 10.23919/ACC.2017.7963427.

Jiang, F., Chou, G., Chen, M., and Tomlin, C. J. Using neural networks for fast reachable set computations. *Preprint arXiv:611.03158*, 2016.

Kalise, D. and Kunisch, K. Polynomial approximation of high-dimensional hamilton–jacobi–bellman equations and applications to feedback control of semilinear parabolic pdes. *SIAM Journal on Scientific Computing*, 40, 02 2017. doi: 10.1137/17M1116635.

Kamalapurkar, R., Rosenfeld, J., Dixon, W. E., and Walters, P. *Reinforcement Learning for Optimal Feedback Control: A Lyapunov-Based Approach*. Springer Cham, 2018.

Kato, D. and Moura, S. J. 1d pde model for thermal dynamics in fluid-cooled battery packs: Numerical methods and sensor placement. In *2021 American Control Conference (ACC)*, pp. 3102–3107, 2021. doi: 10.23919/ACC50511.2021.9483248.

Kim, J., Shin, J., and Yang, I. Hamilton–jacobi deep q-learning for deterministic continuous-time systems with lipschitz continuous controls. *Journal of Machine Learning Research*, 22:1–34, 2021.

Liberzon, D. *Calculus of variations and optimal control theory: a concise introduction*. Princeton University Press, 2012.

Munos, R. A Study of Reinforcement Learning in the Continuous Case by the Means of Viscosity Solutions. *Machine Learning*, 40:265–299, 1999.

Munos, R., Baird, L., and Moore, A. Gradient descent approaches to neural-net-based solutions of the hamilton-jacobi-bellman equation. In *IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No.99CH36339)*, volume 3, pp. 2152–2157 vol.3, 1999.

Nakamura-Zimmerer, T., Gong, Q., and Kang, W. A causality-free neural network method for high-dimensional hamilton-jacobi-bellman equations. In *2020 American Control Conference (ACC)*, pp. 787–793, 2020.

Nicodemus, J., Kneifl, J., Fehr, J., and Unger, B. Physics-informed neural networks-based model predictive control for multi-link manipulators. *IFAC-PapersOnLine*, 55(20): 331–336, 2022. ISSN 2405-8963. 10th Vienna International Conference on Mathematical Modelling MATHMOD 2022.

Pineda, F. Generalization of back propagation to recurrent and higher order neural networks. In *Neural information processing systems*, 1987.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. Reinforcement learning tips and tricks. https://stable-baselines3.readthedocs.io/en/master/guide/rl_tips.html, 2022.

Raissi, M., Perdikaris, P., and Karniadakis, G. 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, 2019.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *Preprint arXiv:1707.06347*, 2017.

Sirignano, J. and Spiliopoulos, K. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018. ISSN 0021-9991.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction (2nd edition)*. Cambridge, MA : The MIT Press, 2018.

Tassa, Y. and Erez, T. Least squares solutions of the hjb equation with neural network value-function approximators. *IEEE Transactions on Neural Networks*, 18(4):1031–1041, 2007.

Vazquez, R. and Krstic, M. Boundary control of coupled reaction-advection-diffusion systems with spatially-varying coefficients. *IEEE Transactions on Automatic Control*, 62(4):2026–2033, 2017. doi: 10.1109/TAC.2016.2590506.

Wiltzer, H. E., Meger, D., and Bellemare, M. G. Distributional Hamilton-jacobi-Bellman equations for continuous-time reinforcement learning. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pp. 23832–23856. PMLR, 2022.

A. The PackCooling gym environment

To run numerical experiments in this paper, we had to create a gym environment that solves the pack cooling model numerically and takes a controller as input every time step.

In gym environments, it is recommended to set the action range to $[-1, 1]$ (Raffin et al., 2022). In our control problem, $\sigma(t)$ returns values in $[0, 1]$. Thus, we let $\sigma(t) = \frac{a_t+1}{2}$.

A.1. Internal heat generation

In this paper, we set the internal heat generation in the battery pack, $h(x, t, u(x, t))$ as:

$$h(x, t, u(x, t)) = e^{0.1u(x,t)}. \tag{17}$$

The objective of the internal heat generation was to increase the temperature distribution across the battery pack at every time step, which serves as a motivation for choosing a strictly positive exponential function. This makes the desired state of the controller, $u = 0$, unstable.

A.2. Initial conditions

For $u(x, t)$ we chose initial conditions that satisfy the boundary conditions 3. A set of solutions that satisfy the boundary conditions are:

$$u(x, 0) = \sum_{n=0}^{\infty} C_n \cos(\pi n x), \tag{18}$$

since the derivative of $\cos(\pi n x)$ is zero at $x = 0$ and $x = 1$ for all integers n. Thus, we set our initial conditions to the following:

$$u(x, 0) = \sum_{n=0}^{N} C_n \cos(\pi n x), \tag{19}$$

where N is the number of Fourier nodes. Note that $N \leq N_x$ where N_x is the number of points we discretize the numerical solution into. In our experiments, we set $N = 9$. The Fourier coefficients C_n are sampled from the uniform distribution $U(-2, 2)$. Adding randomness in our initial conditions leads to more robustness in our controller.

For $w(x, t)$, we set the initial condition to $w(x, 0) = U(0)$ where U is defined in the boundary condition in equation 4. In our experiments, we set $U(t)$ to a constant: $U(t) = -5.0$.

This means if we set the controller to a constant $\sigma(t) = 0$, then $u(x, t)$ will increase to high values due to the forcing term $h(x, t, u(x, t))$. And if we set the controller to a constant $\sigma(t) = 1$, then $u(x, t)$ will decrease to low values due to the cooling fluid. So the objective of the controller is to choose $\sigma(t)$ so that the temperature distribution in the battery pack is as close to zero as possible.

A.3. Numerical solution

In our numerical solutions, we discretize the heat distributions $u(x, t)$ and $w(x, t)$ in x and t as: $\Delta x = 0.01, \Delta t = 0.01$, thus satisfying $\Delta x \geq \Delta t \geq \sigma(t)\Delta t$. We set $D(x, t)$ and $R(x, t)$ from equations 1 and 2 to constants as $D(x, t) = 0.01, R(x, t) = 2.0$.

The numerical solution to the PDE was implemented as explained in (Kato & Moura, 2021). We solve for a system

$$A^+ z(t + \Delta t) - \frac{1}{2} h(z(t + \Delta t)) = Az(t) + \frac{1}{2} h(z(t)), \tag{20}$$

where $z(t)$ is the concatenation of $u(x, t)$ and $w(x, t)$ discretized in space. The matrices A^+ and A are derived using a combination of the Crank-Nicolson method, characteristics, and interpolation. Finally, to solve this non-linear system, we run 10 iterations of the Newton-Raphson method.

A.4. Rendering

In the rendering of the environment, we plot the trajectories of $\sigma(t), u(x,t), w(x,t)$ in the environment. We store the numerical solutions of the three variables in a buffer every episode and reset the buffer upon resetting the environment.

Examples of rendered plots are shown in Appendix F. We return a line plot of $\sigma(t)$ along t, and a mesh plot of $u(x,t)$ and $w(x,t)$ along x and t.

B. Proof of theorem 2.1 part 1

Theorem *A function $V(x)$ is the optimal value function if and only if $V \in C^1(\mathbb{R}^n)$ and V satisfies the HJB Equation*

$$(\gamma - 1)V(x) + \sup_{\sigma \in U}\{L(x,\sigma) + \gamma \Delta t \nabla_x V^T(x) f(x,\sigma)\} = 0 \tag{21}$$

for all $x \in \mathbb{R}^n$, assuming for all $x \in \mathbb{R}^n$, there exists a controller $\sigma^*(\cdot)$ such that:

$$\begin{aligned}&(\gamma - 1)V(x) + L(x,\sigma^*(x)) + \gamma \Delta t \nabla_x V^T(x) f(x,\sigma^*(x)) \\ &= (\gamma - 1)V(x) + \sup_{\hat{\sigma} \in U}\{L(x,\hat{\sigma}) + \gamma \Delta t \nabla_x V^T(x) f(x,\hat{\sigma})\}.\end{aligned} \tag{22}$$

Proof We start with equation 6:

$$V^*(x(t_0)) = \sup_{\sigma} \frac{1}{\Delta t} \int_{t_0}^{\infty} \gamma^{\frac{t}{\Delta t}} L(x(\tau), \sigma(\tau)) d\tau \tag{23}$$

Separating the integral term gives:

$$V^*(x(t_0)) = \sup_{\sigma} \left[\frac{1}{\Delta t} \int_{t_0}^{t_0+\Delta t} \gamma^{\frac{t}{\Delta t}} L(x(\tau), \sigma(\tau)) d\tau + \frac{\gamma}{\Delta t} \int_{t_0+\Delta t}^{\infty} \gamma^{\frac{t}{\Delta t}} L(x(\tau), \sigma(\tau)) d\tau \right] \tag{24}$$

$$= \sup_{\sigma} \left[\frac{1}{\Delta t} \int_{t_0}^{t_0+\Delta t} \gamma^{\frac{t}{\Delta t}} L(x(\tau), \sigma(\tau)) d\tau + \gamma V^*(x(t_0 + \Delta t)) \right] \tag{25}$$

As $\Delta t \to 0$, we can rely on the following two assumptions:

$$\frac{1}{\Delta t} \int_{t_0}^{t_0+\Delta t} \gamma^{\frac{t}{\Delta t}} L(x(\tau), \sigma(\tau)) d\tau \approx \frac{1}{\Delta t}(\Delta t L(x(t_0), \sigma(t_0))) = L(x(t_0), \sigma(t_0)) \tag{26}$$

$$V^*(x(t_0 + \Delta t)) \approx V^*(x(t_0)) + \Delta t \nabla_x V^*(x(t_0))^T f(x(t_0)) \tag{27}$$

This gives us the following equation:

$$V^*(x(t_0)) = \sup_{\sigma} \left[L(x(t_0), \sigma(t_0)) + \gamma V^*(x(t_0)) + \gamma \Delta t \nabla_x V^*(x(t_0))^T f(x(t_0)) \right] \tag{28}$$

$$0 = (\gamma - 1)V^*(x(t_0)) + \sup_{\sigma} \left[L(x(t_0), \sigma(t_0)) + \gamma \Delta t \nabla_x V^*(x(t_0))^T f(x(t_0)) \right] \tag{29}$$

C. Proof of theorem 4.1

Theorem *Let $u(\cdot,t), w(\cdot,t) \in L_2[0,1]$. With $\sigma(t) \in [0,1]$ and the reward function $L(U_t, W_t, \sigma_t) = -||U_{t+1}||_2^2 \Delta x$, the HJB equation for the 1D pack cooling problem is:*

$$(1-\gamma)V(u,w) - ||u(\cdot, t+\Delta t)||^2 + \langle V_u, u_t \rangle + \frac{1}{R}\langle V_w, u - w \rangle + \max(0, -\langle V_w, w_x \rangle) = 0 \tag{30}$$

where $||\cdot||$ is the $L_2[0,1]$ norm and $\langle\cdot,\cdot\rangle$ is the $L_2[0,1]$ inner product.

Proof In the pack cooling problem, the value function is a function of $\hat{U}(t)$ and $\hat{W}(t)$, which is $u(x,t)$ and $w(x,t)$ discretized in x. The HJB equation from equation 7 can thus be written as follows:

$$(\gamma-1)V(\hat{U}(t),\hat{W}(t))+\sup_{\sigma\in[0,1]}\{L(\hat{U}(t),\hat{W}(t),\sigma)+\gamma\Delta t\nabla_{\hat{U}}V^T(\hat{U}(t),\hat{W}(t))\dot{\hat{U}}+\gamma\Delta t\nabla_{\hat{W}}V^T(\hat{U}(t),\hat{W}(t))\dot{\hat{W}}\}=0 \quad (31)$$

We know that $L(\hat{U}(t),\hat{W}(t),\sigma)=-||\hat{U}(t+\Delta t)||_2^2\Delta x$, where Δx and Δt are the discretizations in x and t used in the environment respectively. $\hat{U}(t+\Delta t)$ is determined by $\hat{U}(t)$ and $\dot{\hat{U}}(t)$. We know from equation 9 that neither $\hat{U}(t)$ or $\dot{\hat{U}}(t)$ depend on $\sigma(t)$. Thus, we can bring $L(\hat{U}(t),\hat{W}(t),\sigma)$ and $\nabla_{\hat{U}}V^T(\hat{U}(t),\hat{W}(t))\dot{\hat{U}}$ outside the supremum.

$$(\gamma-1)V(\hat{U}(t),\hat{W}(t))+L(\hat{U}(t),\hat{W}(t),\sigma)+\gamma\Delta t\nabla_{\hat{U}}V^T(\hat{U}(t),\hat{W}(t))\dot{\hat{U}}+\sup_{\sigma\in[0,1]}\{\gamma\Delta t\nabla_{\hat{W}}V^T(\hat{U}(t),\hat{W}(t))\dot{\hat{W}}\}=0 \quad (32)$$

We expand $\dot{\hat{W}}$ from equation 10.

$$\sup_{\sigma\in[0,1]}\{\nabla_{\hat{W}}V^T(\hat{U},\hat{W})\dot{\hat{W}}\}=\sup_{\sigma\in[0,1]}\{\nabla_{\hat{W}}V^T(\hat{U},\hat{W})(-\sigma(t)B\hat{W}+\frac{1}{R}(\hat{U}-\hat{W}))\} \quad (33)$$

$$=\nabla_{\hat{W}}V^T(\hat{U},\hat{W})(\frac{1}{R}(\hat{U}-\hat{W}))+\sup_{\sigma\in[0,1]}\{\nabla_{\hat{W}}V^T(\hat{U},\hat{W})(-\sigma(t)B\hat{W})\} \quad (34)$$

For the expression to achieve its supremum, we let $\sigma(t)$ be defined as the following:

$$\sigma(t)=\begin{cases}1 & \nabla_{\hat{W}}V^T(\hat{U},\hat{W})B\hat{W}<0\\ 0 & \text{otherwise}\end{cases} \quad (35)$$

We get:

$$\sup_{\sigma\in[0,1]}\{\nabla_{\hat{W}}V^T(\hat{U},\hat{W})(-\sigma(t)B\hat{W})\}=\max(0,-\nabla_{\hat{W}}V^T(\hat{U},\hat{W})B\hat{W}) \quad (36)$$

Thus, we get the following HJB equation in discretized x:

$$(\gamma-1)V(\hat{U},\hat{W})+L(\hat{U},\hat{W},\sigma)+\gamma\Delta t\nabla_{\hat{U}}V^T(\hat{U},\hat{W})\dot{\hat{U}}+\nabla_{\hat{W}}V^T(\hat{U},\hat{W})(\frac{1}{R}(\hat{U}-\hat{W}))+\max(0,-\gamma\Delta t\nabla_{\hat{W}}V^T(\hat{U},\hat{W})B\hat{W})=0 \quad (37)$$

Let $\Delta x=\Delta t$ to satisfy the condition for numerical stability. As we let $\Delta x\to 0$, we see a Riemann sum forming.

$$(\gamma-1)V-\int_0^1(u(\cdot,t+\Delta t))^2dx+\int_0^1 V_u u_t dx+\int_0^1 V_w(\frac{1}{R}(u-w))dx+\max(0,-\int_0^1 V_w w_x dx)=0 \quad (38)$$

We can write this as inner products and norms in the $L_2[0,1]$ space.

$$(\gamma-1)V-||u(\cdot,t+\Delta t)||^2+\langle V_u,u_t\rangle+\langle V_w,\frac{1}{R}(u-w)\rangle+\max(0,-\langle V_w,w_x\rangle)=0 \quad (39)$$

D. Proof of corollary 4.2

Corollary *Let $u(\cdot,t), w(\cdot,t) \in L_2[0,1]$. With $\sigma(t) \in [0,1]$ and the reward function $L(U_t, W_t, \sigma_t) = -||U_{t+1}||_2^2 \Delta x$, provided the optimal value function $V^*(u,w)$ the optimal controller for the 1D pack cooling problem is:*

$$\sigma^*(t) = \begin{cases} 1, & \langle V_w^*(\cdot,t), w_x(\cdot,t)\rangle < 0, \\ 0, & \text{otherwise}, \end{cases} \tag{40}$$

where $||\cdot||$ is the $L_2[0,1]$ norm and $\langle\cdot,\cdot\rangle$ is the $L_2[0,1]$ inner product.

Proof The optimal controller was derived in the proof of the HJB equation in equation 35. As we let $\Delta x \to 0$, we get:

$$\sigma(t) = \begin{cases} 1 & \int_0^1 V_w w_x dx < 0 \\ 0 & \text{otherwise} \end{cases} \tag{41}$$

We can write this as inner products and norms in the $L_2[0,1]$ space.

$$\sigma(t) = \begin{cases} 1 & \langle V_w, w_x\rangle < 0 \\ 0 & \text{otherwise} \end{cases} \tag{42}$$

E. Hyperparameters

Hyperparameter	Value
Horizon (T)	1024
Actor learning rate	3e-04
Critic learning rate	1e-03
Num. epochs	10
Minibatch size	64
Discount (γ)	0.99
GAE parameter (λ)	0.95

Table 1. Hyperparameters

F. Plots of trajectories

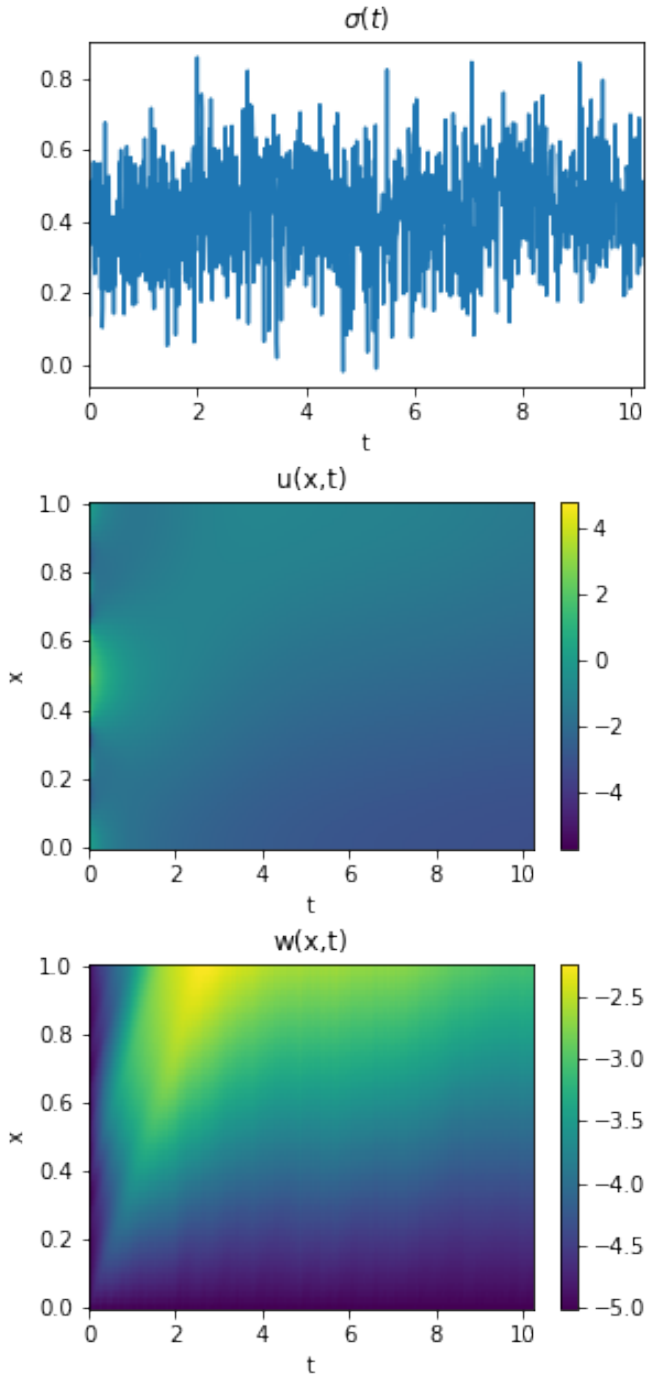

Figure 2. Trajectory of PPO. Cumulative reward: -3970.02

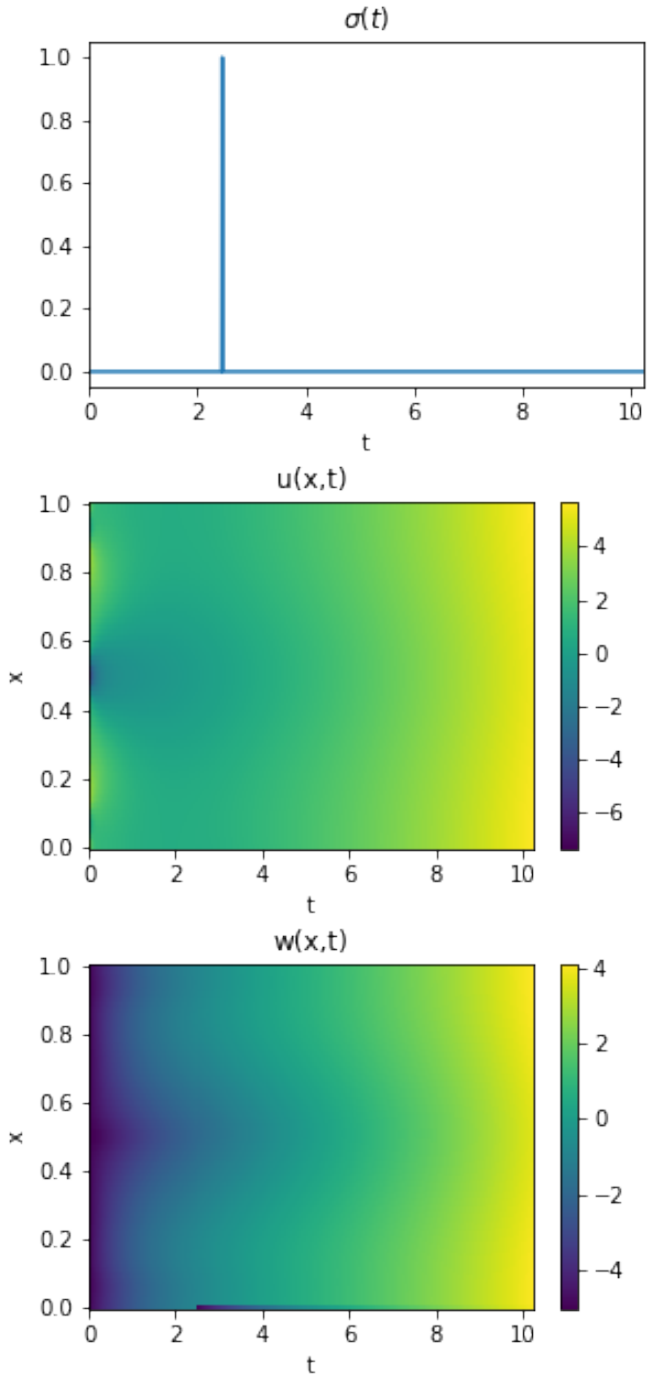

Figure 3. Trajectory of HJB value iteration. Cumulative reward: -7294.51

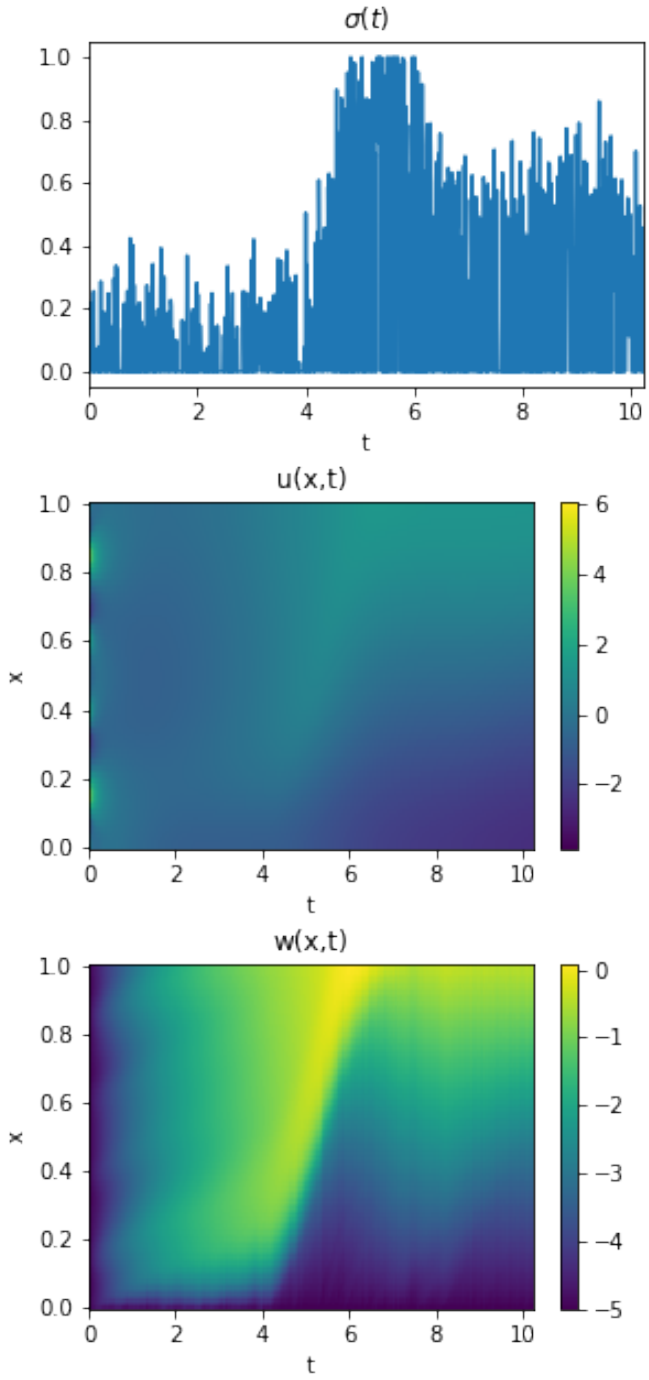

Figure 4. Trajectory of HJBPPO. Cummulative reward: -881.55