An Actor Critic Method for Free Terminal Time Optimal Control

Evan Burton * **Tenavi-Nakamura-Zimmerer** *,** **Qi Gong** *
Wei Kang ***,*

* *University of California Santa Cruz, Santa Cruz, CA 95060, USA (e-mail: esburton@ucsc.edu, tenakamu@ucsc.edu, qgong@ucsc.edu).*
** *Flight Dynamics Branch, NASA Langley Research Center, Hampton, VA 23666, USA.*
*** *Naval Postgraduate School, Monterey, CA 93943, USA (email: wkang@nps.edu)*

Abstract: Optimal control problems with free terminal time present many challenges including nonsmooth and discontinuous control laws, irregular value functions, many local optima, and the curse of dimensionality. To overcome these issues, we propose an adaptation of the model-based actor-critic paradigm from the field of Reinforcement Learning via an exponential transformation to learn an approximate feedback control and value function pair. We demonstrate the algorithm's effectiveness on prototypical examples featuring each of the main pathological issues present in problems of this type.

Keywords: **Iterative learning control, Non-smooth and discontinuous optimal control problems**

1. INTRODUCTION

We aim to solve optimal feedback control problems with a free terminal time and fixed target state, which are common in trajectory optimization and path-planning. These problems often have solutions which saturate the control bounds and have value functions which may not even be continuous. We will consider the problem of minimizing a running cost $\ell(\mathbf{x}, \mathbf{u}) \geq 1$ with free terminal time and given terminal state $x(t_f) \in \mathcal{T}$ over a compact set of control values $\mathcal{U} \subset \mathbb{R}^m$. That is, we wish to approximate the solution of

$$V(\mathbf{x}_0) = \min_{\mathbf{u}\in\mathcal{U}, t_f \geq 0} \int_0^{t_f} \ell(\mathbf{x}(t), \mathbf{u}(t))\, dt$$
$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}), \quad \mathbf{x}(0) = \mathbf{x}_0, \quad \mathbf{x}(t_f) \in \mathcal{T} \tag{1}$$

For optimal control problems with a continuous value function, the value function $V : \mathbb{R}^n \to [0, \infty)$ is the unique viscosity solution to the Hamilton-Jacobi-Bellman (HJB) equation Bardi and Dolcetta (1997). However, in many cases the value function may be non-differentiable or discontinuous at or near the target and one must examine the supremum of (viscosity) subsolutions and infimum of supersolutions along with the comparison principle to identify a unique solution. Solving a partial differential equation is difficult in general because of the curse of dimensionality and here it is complicated further by looking for the solution in the viscosity sense.

In the literature, optimal feedback control for this problem has been done via discretization then applying a fixed point operation over a grid, mesh, or tensor decomposition of a grid/mesh as in Bardi and Dolcetta (1997); Cristiani and Martinon (2010); Falcone et al. (2014); Gorodetsky et al. (2018). The classic trade-off for mesh based methods is although accuracy is very high, scaling to dimensions higher than three is severely limited due to the requirement of adding exponentially more grid points per dimension. Some tensor decomposition methods alleviate the curse of dimensionality while retaining high accuracy so long as the tensor-rank of the optimal value function is low. That is, they are capable of trading dimension dependence with an auxillary metric of complexity, but the tensor-rank of arbitrary value functions may not always be low (even in low dimensions) and is hard to verify before performing the computation and observing the rank or time complexity. We aim to reduce dimension/tensor-rank dependence by applying a neural network based Reinforcement Learning approach. In order to do this, we show that it is possible to apply Reinforcement Learning to free terminal time type problems via the Semi-Lagrangian discretization, allowing one to convert the problem of interest (1) into a form resembling an infinite horizon optimal control problem with a varying discount.

1.1 Semi-Lagrangian Discretization

The Semi-Lagrangian method has been successfully used to solve low dimensional feedback control problems via mesh-based solvers for the HJB equation such as in the appendix of Bardi and Dolcetta (1997); Falcone and Ferretti (2013) or by seeding an open-loop method as in Cristiani and Martinon (2010). We will apply the Kružkov transformation, a classical transformation for minimum-time type problems,

$$\tilde{V}(\mathbf{x}) = \begin{cases} 1 - e^{-V(\mathbf{x})}, & \text{if } V(\mathbf{x}) < \infty. \\ 1, & \text{otherwise.} \end{cases} \tag{2}$$

as is done in Bardi and Dolcetta (1997); Cristiani and Martinon (2010) in order to acquire a functional equation which we will iteratively fit a solution to using a neural network. The main idea will be to discretize the dynamics, in this case with forward Euler, to get a discrete optimal control problem which converges to the original problem as the time step Δt decreases to zero. The value function approximation corresponding to the discrete dynamics should satisfy the discrete time dynamic programming principle. Therefore, instead of discretizing the directional derivative via finite differencing, we directly apply the dynamic programming principle for the discrete dynamics in combination with the same order integration method over $[0, \Delta t]$ as in Falcone et al. (2014):

$$\begin{cases} \mathbf{x}_{k+1} = \mathbf{x}_k + \Delta t\, \mathbf{f}(\mathbf{x}_k, \mathbf{u}) \\ \int_0^{\Delta t} \ell(\mathbf{x}, \mathbf{u})\, dt \approx \Delta t\, \ell(\mathbf{x}, \mathbf{u}) \end{cases} \tag{3}$$

$$\implies V(\mathbf{x}_k) = \min_{\mathbf{u} \in \mathcal{U}} \left\{ \Delta t\, \ell(\mathbf{x}_k, \mathbf{u}) + V\left(\mathbf{x}_{k+1}\right) \right\} \tag{4}$$

Upon manipulation of both sides to rewrite the equation in terms of the Kružkov transform of the value function we obtain

$$\begin{aligned} e^{-V(\mathbf{x}_k)} &= e^{-\min_{\mathbf{u}} \{\Delta t \ell(\mathbf{x}_k, \mathbf{u}) + V(\mathbf{x}_{k+1})\}} \\ e^{-V(\mathbf{x}_k)} &= \max_{\mathbf{u}} e^{-\Delta t \ell(\mathbf{x}_k, \mathbf{u}) - V(\mathbf{x}_{k+1})} \\ 1 - e^{-V(\mathbf{x}_k)} &= 1 - \max_{\mathbf{u}} e^{-\Delta t \ell(\mathbf{x}_k, \mathbf{u}) - V(\mathbf{x}_{k+1})} \\ \tilde{V}(\mathbf{x}_k) &= 1 + \min_{\mathbf{u}} \left\{ e^{-\Delta t \ell(\mathbf{x}_k, \mathbf{u})} \left(\tilde{V}(\mathbf{x}_{k+1}) - 1 \right) \right\} \end{aligned}$$

Furthermore, define the discount factor and operator for the right-hand-side of the (semi-discrete) HJB equation as

$$\begin{aligned} \gamma(\mathbf{x}, \mathbf{u}) &= e^{-\Delta t \ell(\mathbf{x}_k, \mathbf{u})} \\ \tilde{H}\left(\mathbf{x}, \tilde{V}, \mathbf{u}\right) &= 1 + \gamma\left(\mathbf{x}, \mathbf{u}\right) \left(\tilde{V}\left(\mathbf{x} + \Delta t\, \mathbf{f}\left(\mathbf{x}, \mathbf{u}\right)\right) - 1 \right) \end{aligned} \tag{5}$$

as well as a bounded subset of $\mathbb{R}^n \setminus \mathcal{T} = \Omega$ to be the computational domain with which to solve over. Then the equation we wish to solve is

$$\begin{aligned} \tilde{V}(\mathbf{x}) &= \min_{\mathbf{u}} \tilde{H}\left(\mathbf{x}, \tilde{V}, \mathbf{u}\right), \ \mathbf{x} \in \Omega \\ \tilde{V}(\mathbf{x}) &= 0, \qquad\qquad \mathbf{x} \in \mathcal{T} \end{aligned} \tag{6}$$

To obtain lower truncation error, one can use a second order differential equation discretization and integration scheme such as second order Runge-Kutta and the Trapezoid Rule. Within this paper, we choose to use Euler discretization in anticipation of high neural network fitting error dominating the overall error in computations and the complexity of backpropagation through the integration terms for $\mathbf{x}_{k+1}$ and the running cost. We formulate the problem in this way in order to force a discount factor γ when viewed as the HJB equation of an infinite horizon problem, prevent gradients from appearing in the semi-discrete HJB equation we wish to solve, and allow for a surrogate model for both value and control functions. Most importantly, if the discount factor γ was a fixed number then this equation would correspond to an infinite horizon discrete optimal control problem where γ is used to make the running cost finite when integrated over an infinite horizon as well as acts as a contraction coefficient when applying a fixed point iteration to 6 provided suitable growth conditions on $\tilde{V}(\mathbf{x})$. Such a problem is amenable to approximate dynamic programming methods as studied in Bertsekas and Tsitsiklis (1996) as well as reinforcement learning methods as in Lewis and Vrabie (2009); Tutsoy and Brown (2016). In some cases, such as in Tutsoy and Brown (2016), the minimum time problem is approximated by setting $\gamma \approx 1$ and applying reinforcement learning techniques such as value iteration or an Actor-Critic method. However, rate of convergence and the conditioning of the problem become worse as $\gamma \to 1$ as the step size and optimal cost are directly related to γ. We aim to make use of this interpretation in order to apply the well known Actor-Critic paradigm, see Lewis and Vrabie (2009); Scherrer et al. (2015); Lillicrap et al. (2016); Zhou et al. (2021), which is typically applied to infinite horizon optimal control problems.

2. ACTOR-CRITIC ALGORITHM FOR KRUŽKOV HJB

Motivated by the success of Actor-Critic methods for infinite horizon optimal control, we propose to use a Least Squares Temporal Difference Actor-Critic framework in the case studied here. We implement multilayer neural networks to represent both the actor (controller) $U(\mathbf{x}, W_U)$ and critic (value function) representation, $V(\mathbf{x}, W_V)$. We do this to mitigate the necessity of both choosing good set of basis functions for approximating the true optimal control/cost as well as having enough of such functions to represent the value and control functions adequately. It should be noted that although this is effective in practice, the compatibility of the actor and critic will no longer be guaranteed and the gradients $\partial_{\mathbf{u}} \tilde{H}(\mathbf{x}, \tilde{V}, \mathbf{u})$ may be biased estimates of the true descent direction (Silver et al. (2014)).

The actor critic algorithm presented here in Algorithm 1 is based on value iteration, where the control and value approximators are optimized in an alternating fashion. Upon each iteration k, the critic $\tilde{V}(\mathbf{x}, W_V)$ is initially held fixed while the control approximation $U(\mathbf{x}, W_U)$ minimizes its weights against the expected cost-to-go of one step in time, $\tilde{H}(\mathbf{x}, \tilde{V}, \mathbf{u})$ obtaining W_U^{k+1}. After the control refinement stage, the new value approximation is set to the closest approximation to the new predicted cost-to-go, $\tilde{H}(\mathbf{x}, \tilde{V}, U(\mathbf{x}, W_U^{k+1})$. If this approximation is near enough, we expect both $U(\mathbf{x}, W_U)$ and $V(\mathbf{x}, W_V)$ to converge to the optimal control and value function respectively. This intuition is due to the fact that if we use meshes coupled with the Markov Chain Approximation of Kushner and Dupuis (2001) then provided $1 \leq \ell(\mathbf{x}, \mathbf{u}) \leq M$ on Ω and some technical assumptions which bound $\|\mathbf{x}(t) - \mathbf{x}(t + \Delta t)\|$ and guarantee controllability of states at the boundary of the target as outlined in Bardi and Dolcetta (1997), then this process would converge to the desired approximations.

This algorithm can be efficiently performed in parallel over the states because neural networks can operate on batch inputs efficiently and mature frameworks for their batched gradient computation are readily available. Practically, we implement this algorithm as a stochastic gradient method, only taking a few steps per optimization stage (stages 7

Algorithm 1: Actor-Critic Iteration

Input: A tuple of weights (W_V^0, W_U^0) for the value and actor networks, a time step Δt, a moving average coefficient $\alpha \in (0, 1]$, and the number of samples to draw from each set $N_\Omega, N_\mathcal{T}$.

Output: Neural network approximations of cost-to-go and optimal control.

1 Let $\widehat{W}_V \leftarrow W_V^0$

2 Let $\widehat{W}_U \leftarrow W_U^0$

3 **while** $\widehat{W}_V, \widehat{W}_U$ *not converged* **do**

4 Draw set of samples $X_\Omega \leftarrow \{\mathbf{x}_i\}_{i=1}^{N_\Omega} \sim \mathbb{P}(\Omega)$

5 Draw set of samples $X_\mathcal{T} \leftarrow \{\mathbf{x}_j\}_{j=1}^{N_\mathcal{T}} \sim \mathbb{P}(\mathcal{T})$

6 Define function $\tilde{H}(\mathbf{x}, W) := \tilde{H}\left(\mathbf{x}, \tilde{V}(\mathbf{x}, \widehat{W}_V), U(\mathbf{x}, W)\right)$

7 $W_U^{k+1} \leftarrow \underset{W_U}{\operatorname{argmin}} \underset{\mathbf{x}\in X_\Omega}{\mathbb{E}} \left[\tilde{H}(\mathbf{x}, W_U)\right]$

8 $\widehat{W}_U \leftarrow \alpha W_U^{k+1} + (1-\alpha)\widehat{W}_U$

9 $L_{X_\Omega}^k(W_V) \leftarrow \underset{\mathbf{x}\in X_\Omega}{\mathbb{E}} \left[\left(\tilde{V}(\mathbf{x}, \widehat{W}_V) - \tilde{H}(\mathbf{x}, \widehat{W}_U)\right)^2\right]$

10 $L_{X_\mathcal{T}}^k(W_V) \leftarrow \underset{\mathbf{x}\in X_\mathcal{T}}{\mathbb{E}} \left[\tilde{V}(\mathbf{x}, W_V)^2\right]$

11 $W_V^{k+1} \leftarrow \underset{W_V}{\operatorname{argmin}} \left[L_{X_\Omega}^k(W_V) + L_{X_\mathcal{T}}^k(W_V)\right]$

12 $\widehat{W}_V \leftarrow \alpha W_V^{k+1} + (1-\alpha)\widehat{W}_V$

13 **return** $(\widehat{W}_V, \widehat{W}_U)$

and 11) using an optimizer such as Adam from Kingma and Ba (2017) and increasing the number of steps at stage 11 if the fitting error is large. Then we update a moving average of the model weights $\widehat{W}_V, \widehat{W}_U$to reduce variance in the weight updates. Algorithm (1) can be interpreted as an Actor-Critic type fitted value iteration (or policy gradient when steps 7 and 11 are single gradient descent updates) where each sample from the domain is propagated forward in time by Δt through an actor $U(\mathbf{x}, W_U)$ and the new state is used to update the critic's value estimate. If too few samples are drawn, weight updates will have a high amount of noise and the algorithm may become unstable. One can reduce the update noise through the use of a target network as in Lillicrap et al. (2016) where the weights are slowly averaged into an auxillary network in order to provide stable learning dynamics. We find that this is not necessary for our two examples where the value function is continuous, but helps reduce weight oscillation when there is a discontinuity. Algorithm 1 may be able to be transformed into a Monte Carlo, rollout-based scheme as in Zhou et al. (2021), but due to deterministic dynamics we are able to take advantage of per-sample parallelism without the need for propagation beyond one step. We also note that in the stochastic case, conversion to the form of equation 6 should not be necessary since the system's diffusion should guarantee a unique, sufficiently differentiable solution to the original HJB equation.

Remark 1. This algorithm can be extended to use supervised data as in Nakamura-Zimmerer et al. (2021) by generating a set of known optimal tuples and appending a loss to steps 7 or 11.

Remark 2. A more efficient domain sampling which takes into account model error over the domain instead of uniform random sampling would likely lead to faster convergence. In regards to convergence criteria and iteration count, either a fixed number of iterations is done and the models are evaluated by closed-loop performance or the mean and variance of the moving averages $\widehat{W}_V, \widehat{W}_U$ are used to stop the iteration.

3. EXAMPLES

To demonstrate the algorithm described, we apply it to three qualitatively different examples. The first is the double integrator, a canonical academic example known to many in optimal control, which demonstrates a discontinuous (bang-bang) optimal control and non-differentiable value function. The second is the control-regularized Dubins Vehicle which displays a value function which is both non-differentiable and discontinuous in some regions of its state-space. These properties make optimal open-loop trajectories difficult to obtain without expert initial guesses as there are infinitely many locally optimal controls. The last example is a minimum time attitude control problem for a 7-dimensional rigid body to demonstrate scalability of the algorithm. For each example, we apply the Kružkov transformation to the critic's final output during training as well as enforce control constraints via the actor's final layer.

3.1 Double Integrator

Our first example is that of the time-optimal double integrator:

$$\begin{aligned} &\min_{\mathbf{u}, t_f} \int_0^{t_f} dt \\ &\begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix} = \begin{bmatrix}y\\ u\end{bmatrix}, \quad x(t_f) = y(t_f) = 0, \quad |u| \le 1 \end{aligned} \tag{7}$$

This is a simple example of a problem which gives rise to a non-smooth value function without discontinuity. The switching curve is known to be parabolic and the control is piecewise $+1$ or -1 on each side of this curve, given by $\mathbf{u}(\mathbf{x}) = -\text{sign}(D_y V(\mathbf{x}))$ wherever V is differentiable. However, generating optimal trajectories for this system via open-loop methods presents difficulties due to the discontinuous optimal control. For computation, we aim to solve this problem over the ball $B(r = 5)$ and apply the actor-critic algorithm with networks of 4 layers each comprised of 128 neurons, a time discretization of $dt = 0.05$, no moving average ($\alpha = 0$), and sample sizes of $|X_\Omega| = 500$, $|X_\mathcal{T}| = 1$.

We use tanh activations in this case because the optimal control is bang-bang and the tanh function can smoothly saturate between the bounds +1 and -1. Notably, the actor network obtained is continuous by construction and

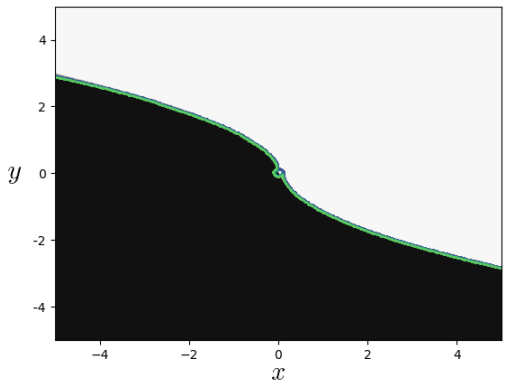

Fig. 1. Switching surface given by neural network actor with tanh activation (white/black) and switching curve computed by grid-based semi-lagrangian iteration (green)

therefore the switching behavior is approximated by a steep, smooth transition between -1 to +1 (and vice-versa) which also happens to avoid the chattering problem faced by bang-bang controllers.

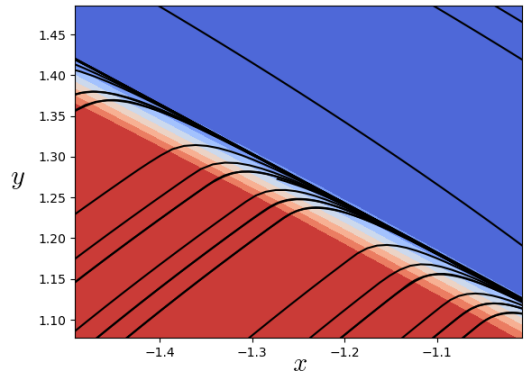

Fig. 2. Close up of switching curve given by neural network actor with sample trajectories overlayed (black).

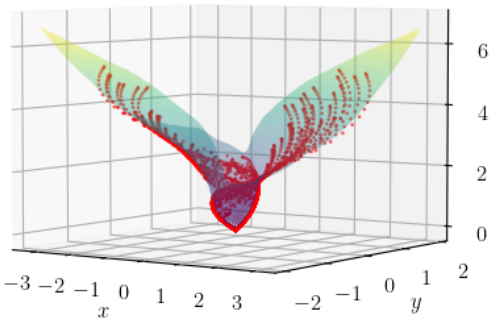

Fig. 3. Comparison between the true value function (red) and the critic network (surface).

Compared to the true solution over a grid of 3200 test points, obtained using a pseudospectral collocation method, see Ross and Karpenko (2012). Note the critic (value approximator) network is very similar geometrically to the optimal value function and has typically differs from the target with mean-squared-error of order 10^{-2}.

3.2 Dubins Vehicle

For our next example we will consider the control-regularized minimum time problem for Dubin's Vehicle:

$$\min_{u,t_f} \int_0^{t_f} \left(1 + \frac{1}{2}\left[u(t)\right]^2\right) dt$$
$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}\cos\theta\\ \sin\theta\\ u\end{bmatrix}, \quad \begin{matrix}|u| \le 6, \quad t_f \ge 0\\ |x(t_f)|^2 + |y(t_f)|^2 \le (0.1)^2\end{matrix} \tag{8}$$

This problem displays a value function with both a non-smooth gradient and a discontinuity at the boundary of the target set. Physically, we aim to steer a vehicle with constant velocity to the origin in the least time and using the least steering input. The discontinuity at the boundary of the target set comes from the fact that the dynamics are not small-time-locally-controllable Bardi and Dolcetta (1997). In other words, starting near the target does not guarantee a small cost-to-go for some states because a lack the controllability leads to a high cost-to-go as the vehicle must follow an arc dependent on its turning radius. For example, starting at position $(x, y, \theta) = (0.1, 0, 0)$ the vehicle must first travel away from $\mathcal{T}$ and circle back since there is no direct control of x, y and the velocity of the vehicle is constant. This problem has a radially symmetric value function and so visualization is done in only the $(x, 0, \theta)$ coordinates. For reference, we present below a plot of a representative slice of the value function as computed via the Markov Chain Approximation of Kushner and Dupuis Kushner and Dupuis (2001).

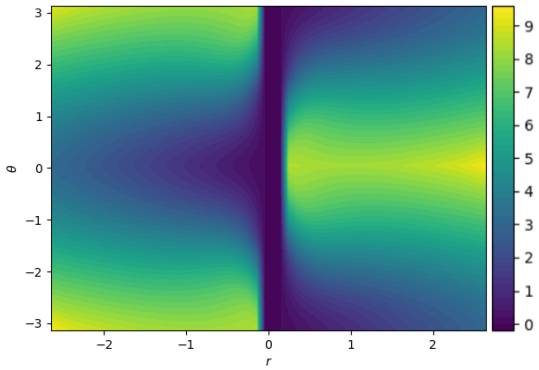

Fig. 4. The reference value function as computed via mesh as in Kushner and Dupuis (2001), shown radially in the (r, θ) plane where r is on the x-axis and θ on the y-axis.

The Actor-Critic method using a weight-averaging update for the target network with $\alpha = 0.1$, ReLU activation with 4 layers of 128 neurons per layer, and a rescaling of the value function by 0.2 produced excellent results. Although the optimal control is discontinuous, owing to the fact that the vehicle must physically turn left or right depending on

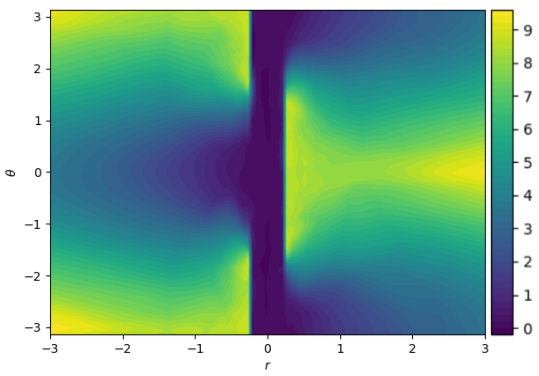

Fig. 5. Contour of the neural network approximation of the value function in the (r, θ) plane.

heading, the neural network controller $U(\mathbf{x}, W_U)$ performs well after tuning the parameter α governing the moving average.

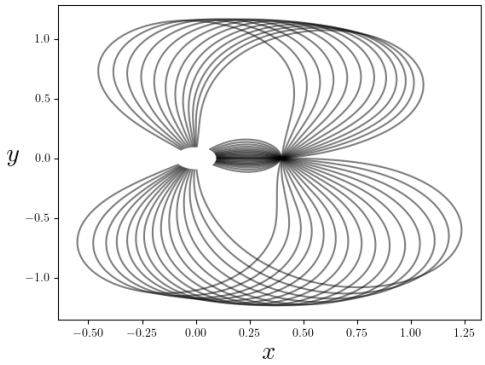

Fig. 6. Actor feedback control over sample trajectories in the (x, y) space near the target. Each trajectory begins with an initial heading $\theta(0) \in [-\pi, \pi)$. There is a paradigm switch between trajectories facing near the target to begin to face about 45 degrees away, corresponding to a kink in the value function. The choice of steering left or right when facing exactly opposite to the target also corresponds to a kink.

We found that it is necessary to rescale either the running cost $\ell(\mathbf{x}, \mathbf{u})$ in order to produce adequate approximations for this problem as the exponential term $\exp(-V(\mathbf{x})) \approx 0$ when $V(\mathbf{x}) \approx 5$ and many states do achieve a cost greater than that limit. Due to limited precision of floating point, the Kružkov transformation truncates the values of all states which would incur a cost of about 5. We currently get around this limitation through either the rescaling $\ell(\mathbf{x}, \mathbf{u}) \to \mu\,\ell(\mathbf{x}, \mathbf{u})$ or a direct rescaling of the dynamic programming equation through $V(\mathbf{x}) \to \mu\, V(\mathbf{x})$ for $0 < \mu \leq 1$.

3.3 Attitude Control of Spacecraft

For our next example, we chose a higher dimensional satellite system to show that this algorithm has the potential to push to higher dimensions. The optimal control problem is given as

$$\begin{gathered}\min_{u,t_f} \quad t_f \\ \begin{bmatrix}\dot{q}_0 \\ \dot{\mathbf{q}} \\ \mathbf{J}\dot{\boldsymbol{\omega}}\end{bmatrix} = \begin{bmatrix} -\frac{1}{2}\boldsymbol{\omega}^T\mathbf{q} \\ \frac{1}{2}\left(-\omega_\times \mathbf{q} + q_0\boldsymbol{\omega}\right) \\ -\boldsymbol{\omega}_\times \mathbf{J}\boldsymbol{\omega} - \mathbf{u}\end{bmatrix} \\ -0.3 \leq \mathbf{u} \leq 0.3 \\ \mathbf{q}(t_f) = \boldsymbol{\omega}(t_f) = 0, \quad q_0(t_f) = 1 \\ J = \begin{bmatrix} 59.22 & -1.14 & -0.8 \\ -1.14 & 40.56 & 0.1 \\ -0.8 & 0.1 & 57.60 \end{bmatrix}\end{gathered} \tag{9}$$

The state space sampling is done by sampling Euler angles in $[-\frac{\pi}{2}, \frac{\pi}{2}]$ uniformly then converting to quaternions, the angular velocities are sampled uniformly from the sphere $10^{-4} \leq \|\boldsymbol{\omega}\|^2 \leq 0.3$, and the inertia matrix is that of the TRACE spacecraft (see Zimbelman et al. (1995)). We used the Adagrad optimizer, sample sizes of $|X_\Omega| = |X_\mathcal{T}| = 1500$, $\Delta t = 0.3$, and ran the algorithm for 2000 iterations for a total of 73 seconds. The large Δt is used to speed up convergence as the time-scale of trajectories is comparatively large. The neural networks used are sequential networks with hidden layers, $L_i(x)$, of sizes $[7 \to 200 \to 200]$ where the hidden layers are residual blocks of the form $x \to L_{i-1}(x) + x$, and ReLU activation.

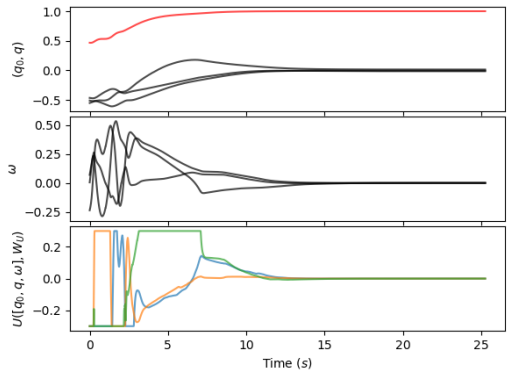

Fig. 7. An example trajectory for the TRACE system using the actor network as the controller. The value of q_0 is in red and $\mathbf{q}, \boldsymbol{\omega}$ in black. Note that the control inputs resemble that of a smoothed, bang-bang controller which one expects to observe for minimum time solutions.

It is important to note here that because of the generality of the model, the terminal condition $\|\mathbf{q}\| = 0$ may not exactly correspond to $\tilde{V}(q(t_f), W_V) = 0$ and therefore the steady-state may be $\|\mathbf{q}\| = \epsilon$ since the model is minimizing its squared deviation from 0 over the target. Experimentally, we have observed that for nearly all cases that the steady state satisfies $\|\mathbf{q}\|_\infty \leq 0.05$. Modifications to achieve neural network controller stability can be found Nakamura-Zimmerer et al. (2022). Updating the model architecture to automatically accommodate the terminal condition would be ideal and enforce $\mathbf{q}(t_f) = 0$ exactly. A collection of 100 sample trajectories are shown in figure 8, note that they are stable. In future work, we intend to compare these trajectories with open-loop solutions generated via pseudospectral methods.

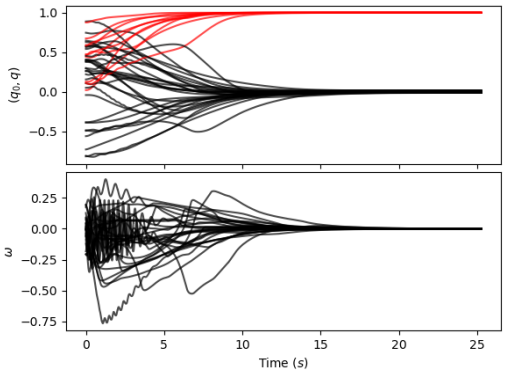

Fig. 8. Ensemble trajectories displaying stability of the neural network controller.

4. CONCLUSION

In this paper we demonstrated the use of a neural network Actor-Critic algorithm for solving optimal control problems with free terminal time that may have nonsmooth solutions in the value or control. We find that it is a promising approach towards alleviating the curse of dimensionality and as a possible companion method to open-loop solvers that struggle with local optima or computing a solution near kinks and corners of the value function. We find that the method has promise even for problems with discontinuous optimal cost, such as in the Dubins Vehicle example. In future work, the efficacy of coupling the proposed algorithm to direct methods could be interesting as well as analysis on the convergence of such methods.

ACKNOWLEDGEMENTS

This work was supported by the Air Force Office of Scientific Research (AFOSR) under grant no. FA9550-21-1-0113, and the National Science Foundation (NSF) under grant no. 2134235.

REFERENCES

Bardi, M. and Dolcetta, I.C. (1997). *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equations*. Modern Birkhauser Classics. Birkhauser Basel. doi:10.1007/978-0-8176-4755-1.

Bertsekas, D.P. and Tsitsiklis, J.N. (1996). *Neuro-Dynamic Programming*. Athena Scientific, 1st edition.

Cristiani, E. and Martinon, P. (2010). Initialization of the Shooting Method via the Hamilton-Jacobi-Bellman Approach. *Journal of Optimization Theory and Applications*, 146(2), 321–346. doi:10.1007/s10957-010-9649-6.

Falcone, M. and Ferretti, R. (2013). *Semi-Lagrangian Approximation Schemes for Linear and Hamilton Jacobi Equations*. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics. doi:10.1137/1.9781611973051.

Falcone, M., Kalise, D., and Kröner, A. (2014). A semi-Lagrangian scheme for Lp-penalized minimum time problems. In *21st International Symposium on Mathematical Theory of Networks and Systems*. Groningen, Netherlands.

Gorodetsky, A., Karaman, S., and Marzouk, Y. (2018). High-dimensional stochastic optimal control using continuous tensor decompositions. *The International Journal of Robotics Research*, 37(2-3), 340–377. doi:10.1177/0278364917753994. Publisher: SAGE Publications Ltd STM.

Kingma, D.P. and Ba, J. (2017). Adam: A method for stochastic optimization.

Kushner, H. and Dupuis, P.G. (2001). *Numerical Methods for Stochastic Control Problems in Continuous Time*. Stochastic Modelling and Applied Probability. Springer-Verlag, New York, 2 edition. doi:10.1007/978-1-4613-0007-6.

Lewis, F.L. and Vrabie, D. (2009). Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Circuits and Systems Magazine*, 9(3), 32–50. doi:10.1109/MCAS.2009.933854. Conference Name: IEEE Circuits and Systems Magazine.

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2016). Continuous control with deep reinforcement learning. In Y. Bengio and Y. LeCun (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

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. Publisher: SIAM.

Nakamura-Zimmerer, T., Gong, Q., and Kang, W. (2022). Neural Network Optimal Feedback Control with Guaranteed Local Stability. doi:10.48550/arXiv.2205.00394. ArXiv:2205.00394 [cs, eess, math].

Ross, I.M. and Karpenko, M. (2012). A review of pseudospectral optimal control: From theory to flight. *Annual Reviews in Control*, 36(2), 182–197. doi:10.1016/j.arcontrol.2012.09.002.

Scherrer, B., Ghavamzadeh, M., Gabillon, V., Lesner, B., and Geist, M. (2015). Approximate Modified Policy Iteration and its Application to the Game of Tetris. *Journal of Machine Learning Research*, 16(49), 1629–1676.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. (2014). Deterministic Policy Gradient Algorithms. In *Proceedings of the 31st International Conference on Machine Learning*, 387–395. PMLR. ISSN: 1938-7228.

Tutsoy, O. and Brown, M. (2016). Reinforcement learning analysis for a minimum time balance problem. *Transactions of the Institute of Measurement and Control*, 38(10), 1186–1200. doi:10.1177/0142331215581638. Publisher: SAGE Publications Ltd STM.

Zhou, M., Han, J., and Lu, J. (2021). Actor-Critic Method for High Dimensional Static Hamilton–Jacobi–Bellman Partial Differential Equations based on Neural Networks. *SIAM Journal on Scientific Computing*, 43(6), A4043–A4066. doi:10.1137/21M1402303. Publisher: Society for Industrial and Applied Mathematics.

Zimbelman, D., Wilmot, J., and Evangelista, S. (1995). The Attitude Control System Design for the Transition Region and Coronal Explorer Mission. *Small Satellite Conference*.