CS230

Energy-Based Neural Optimal Control

Kevin Troy*
Department of Mechanical Engineering
Stanford University
ktroy@stanford.edu

Abstract

State of the art optimal control methods such as Model Predictive Control provide robust, optimal solutions to trajectory planning and control. These methods, however are computationally complex and typically involve detailed constrained optimization problems that must be solved in real time. With the rise of AI and prevalence of neural networks, I investigate the use of neural networks to approximate solutions to optimal control problems by re-creating and augmenting existing work in neural energy shaping. During analysis we observe that the neural approach exhibits significant increases in runtime performance, but much poorer performance. Through an error analysis we identify the primary performance driving hyperparameters and conclude that more efficient numerical methods are required for this approach to be successful.

1 Introduction

In this work I build upon existing work unifying energy-based optimal control and deep learning. Energy based control is used commonly with robotic actuators and other nonlinear systems and typically results in a constrained quadratic problem, but in more complex cases yields non-convex optimization problem. While these optimal control methods yield superb results, implementing them in real time can be difficult due to their high computation cost. Neural networks provide a promising avenue to reducing runtime complexity while retaining the quality of the control policy by approximating the solution to the optimal control problem. The proposed neural network architecture is built upon existing work and accepts the current dynamical system state as an input, and outputs an approximation of the optimal control output. Existing work on optimal energy based neural networks demonstrates nonlinearities and non-conservative interactions, but is only limited to a single degree of freedom. In this work I expand the complexity of the optimal control problem to multiple degrees of freedom.

2 Related work

Neural-assisted optimal control has seen a rise in research. Works [1][2] investigate Neural-Model Predictive Control, which uses a neural approximation of system dynamics to formulate the problem. Works [3][4][5] investigate the use of the novel neural architecture "NeuralODEs" to approximate system dynamics and control policies. Finally, the state of the art for energy based control is [6], which I am to re-create the results of and expand to multi degree of freedom systems. All of these approaches are promising in different ways, but the niche of energy based control retains a flair of

*https://github.com/kevin-troy/neural_optimal_control

CS230: Deep Learning, Autumn 2022, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

physical intuition that other methods do not have. At its core, data-driven energy based control is a specific subset of nonlinear control that remains underrepresented relative to other control methods.

In addition to data-driven methods, classic optimal control methods such as Model Predictive Control rely on solving constrained optimization problems with a finite time horizon to yield an optimal action for the current state[7]. These methods are widely used and will be used as a baseline for this effort.

3 Dataset and Features

The dynamical system of choice for this project is a pointable two-axis optic, commonly used in communications systems and satellite imaging. This system was chosen because it is a two degree-of-freedom system, but the dynamics are non-chaotic. Additionally, we choose to represent the systems angular states as they have a bounded domain. The continuous-time Hamiltonian dynamics of the mirror can be shown to be:

$$\begin{bmatrix}\dot{q}_1\\ \dot{q}_2\\ \dot{p}_1\\ \dot{p}_2\end{bmatrix} = \begin{bmatrix}0 & 0 & I_x^{-1} & 0\\ 0 & 0 & 0 & I_y^{-1}\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}q_1\\ q_2\\ p_1\\ p_2\end{bmatrix} + \begin{bmatrix}0 & 0\\ 0 & 0\\ 1 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}u_1\\ u_2\end{bmatrix} \tag{1}$$

Where I_x, I_y are the rotational inertias of the system, and the state variables are defined as follows:

$$q_1 = \theta,\ q_2 = \phi,\ p_1 = I_x\dot{\theta},\ p_2 = I_y\dot{\phi} \tag{2}$$

The training dataset used in this effort differs from a traditional learning dataset. Because we are embedding a control problem into the neural networks, our training data consists of random initial states as the inputs, and the corresponding goal states as the outputs. The bounded state domain of the pointable mirror allows us to effectively discretize the initial state space to generate training inputs from as follows:

$$X_{\text{train}} = \begin{bmatrix}\mathcal{U}(\theta_{\text{min}}, \theta_{\text{max}})\\ \mathcal{U}(\phi_{\text{min}}, \phi_{\text{max}})\\ \mathcal{U}(\dot{\theta}_{\text{min}}, \dot{\theta}_{\text{max}})\\ \mathcal{U}(\dot{\phi}_{\text{min}}, \dot{\phi}_{\text{max}})\end{bmatrix}, \quad Y_{\text{train}} = \begin{bmatrix}\theta_g\\ \phi_g\\ \dot{\theta}_g\\ \dot{\phi}_g\end{bmatrix} \tag{3}$$

Training data is synthetically generated from the above angle-space uniform distributions, and then modified to agree with the Hamiltonian dynamics in accordance with equation 2.

4 Methods

4.1 Control Policy

As presented in [6], an optimal control policy for an energy shaping problem can be parameterized by neural networks V and K as follows:

$$u_\theta(q,p) = -\nabla_q V_\theta(q) - K_\theta(t,q,p)q \tag{4}$$

4.2 Network Architecture

In this work I use a similar architecture to that presented in [6]: two multi-layer perceptrons are used. Throughout re-creating the original paper results and experimentation activations were changed to ReLU from Softplus, and both the hidden layer size and quantity increased.

Network V has an input layer size of two, three fully connected layers of 128 neurons each with ReLU activations, a final hidden layer with 128 neurons and a tanh activation, and an output layer with two neurons and no activations.

Network K has a input layer size of four, three fully connected layers of 128 neurons each with ReLU activations, and an output layer with two neurons with ReLU activations.

4.3 Loss

The loss of the control policy used aims to maximize the probability of the system reaching the goal states via minimizing the log-likelihood of achieving the goal state given some initial conditions. An additional integral loss that penalizes use of control effort is appended to limit excessive control usage.

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\log(p(x_g|x_0, u_\theta)) + \frac{\gamma}{N}\int_0^T |u_\theta(p(x_g|x_0, u_\theta))|dt \tag{5}$$

We can view this is analogous to a quadratic Q,R cost function commonly used in optimal control. Both formulations consist of running state and control costs with a final terminal goal cost.

4.4 Baseline

The baseline for this effort is a state-of-the-art optimal control method: Model-Predictive Control (MPC). The baseline MPC implementation for this project implements linear dynamics and quadratic cost to mirror that of the neural network control policy. We can loosely state the we are approximating the MPC solution to the control problem with the energy shaping approach. The baseline MPC algorithm implements the below optimization problem.

$$\begin{aligned}
&\underset{s_{0:N}, u_{0:N}}{\text{minimize}} \ (s_N - s_g^*)^T P(s_N - s_g^*) + \sum_{k=0}^{N-1}\left((s_k - s_f^*)^T Q(s_k - s_f^*) + u_k^T R u_k\right) \\
&\text{subject to: } s_0 = s_0^* \\
&\qquad\qquad s_k \in \mathcal{S},\ \forall k \in \{0, 1, ...N-1\} \\
&\qquad\qquad u_k \in \mathcal{U},\ \forall k \in \{0, 1, ...N-1\} \\
&\qquad\qquad s_{k+1} = As_k + Bu_k,\ \forall k \in \{0, 1, ...N-1\}
\end{aligned} \tag{6}$$

Where N is the look-ahead horizon, Q and R are the running state and control costs, and P is the terminal system cost. The problem is constrained to obey the system dynamics and any relevant state constraints.

5 Experiments and Discussion

5.1 Hyperparameters

Hyperparameters were tuned manually with consideration of the existing work, NeuralODE problem best practices, and context of the system dynamics.

Table 1: Hyperparameters

Traditional		ODE-Specific	
Optimizer	Adam	Solver	Rk4
Learning Rate	5e-3	Timespan	[0,10] sec
LR Annealing	0.999	Timesteps	100
Batches	1	-	-
Batch size	2048 Examples	-	-
Epochs	500	-	-

Throughout the tuning process, it was noted that most hyperparameters had minimal effect on final performance. The primary driving hyperparameter is that of the Neural ODE problem: the discretization of the time series. The used values were chosen to align with the project scope. Even a minor increase in timespan and/or discretization fidelity yields a significantly longer training time. All parameters were initialized with best practices (Xavier/He initializations), and in accordance with NeuralODEs initialization practices the final layer weights were set to zero at the start of training [8].

5.2 Metrics and Results

The implementation for these experiments was performed with PytorchLightining, PyTorch, and TorchDyn[9]. The two primary metrics we compare are 1.) time to compute the optimal control for a scenario and 2.) the RMS final absolute state error for a pose regulation task, denoted as follows:

$$\tilde{x}_{\text{RMS}} = \frac{1}{4}\sum_{i=0}^{4} |x_f^{(i)} - x_g^{(i)}| \tag{7}$$

We begin by showing that the runtime complexity of the neural energy shaping approach is much faster than the MPC implementation, as expected. We note that in more efficient implementations of both methods, the runtime performance is expected to improve for both methods.

Table 2: Runtime Simulation Results

Runtime Data, 256 Simulations			
	Scenario 1 (No initial momenta, p=0)	Scenario 2 (No initial deflection, q=0)	Scenario 3 (Random initial conditions)
MPC Average	5.05 sec	5.62 sec	5.74 sec
MPC Total	21.55 min	23.89 min	24.50 min
Energy Net Avg.	0.0094 sec	0.0095 sec	0.0098 sec
Energy Net Total	2.39 sec	2.44 sec	2.51 sec

Next, we examine the RMS state error as a function of initial state for both metrics. We see that the energy network is able to correctly learn the cardinality of the control inputs due to the visible radial symmetry shown. However when examining the error magnitudes, it is clear that the energy neural net is underperforming.

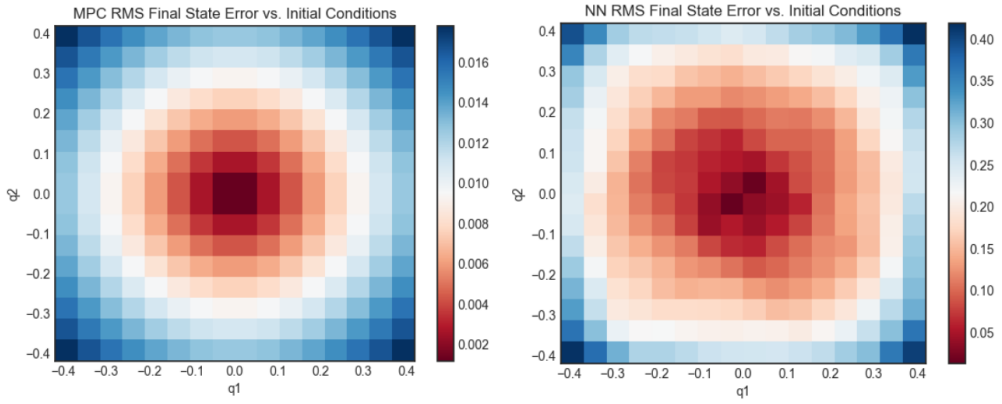

Figure 1: Final RMS Error vs. Initial Conditions

To further investigate we can examine a series of state trajectories from 256 different initial conditions generated with each control method.

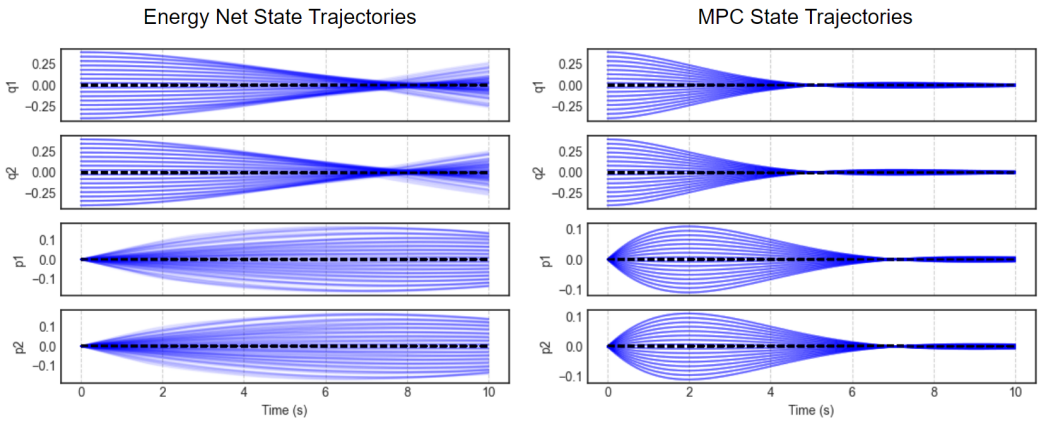

Figure 2: Comparison of State Trajectories (Black=goal state)

From the results of the state trajectory analysis we confirm that the energy net is regulating the states in the correct "direction" but fails to dampen the response, leading to the degraded RMS final error. Based on the parameterization of the energy net control law we conclude that the under-performance is dominated by the damping injection network $K(t, q, p)$, rather than the energy shaping network $V(q)$. As mentioned in the hyperparameter section, this approach is heavily sensitive to the time discretization used to solve the ODE, and unfortunately could not be tuned further within the time constraints of this project.

6 Conclusion/Future Work

In summary, a method of controlling systems via a data-driven approach was investigated for rigid multiple-degree-of-freedom systems and compared to Model Predictive Control. We observed significant increases in runtime performance, but a degradation of pose regulation performance. The lack of performance can be attributed to the damping network K, and is dependent on the time discretization of the ODE problem. Unfortunately with the limited scope of this project the discretization fidelity could not be further improved. With more resources and time, I would expect this approach to work as it demonstrates successful learning of a portion of the energy problem. Additionally, new approach such as Hypersolvers (ODE solvers combined with neural nets) and libraries such as TorchODE[10] are in their infancy but show promising results regarding increasing performance of ODE-based neural networks.

7 Contributions

This is my solo project, so I accordingly did all of the presented analysis. A quick list of tasks completed for this project are shown below:

- Literature Search
- Investigation of alternate energy-based neural networks (LNNs, HNNs)
- Re-creation of results from the original paper [6]
- Implementation of baseline dynamics (Newtonian, Hamiltonian)
- Implementation of energy shaping network and experiments
- Manual hyperparameter tuning and error analysis

References

[1] J. Nubert, J. Köhler, V. Berenz, F. Allgöwer, and S. Trimpe, "Safe and fast tracking on a robot manipulator: Robust mpc and neural network control," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3050–3057, 2020.

[2] X. Zhang, M. Bujarbaruah, and F. Borrelli, "Near-optimal rapid mpc using neural networks: A primal-dual policy learning framework," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 5, pp. 2102–2114, 2020.

[3] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," *Advances in neural information processing systems*, vol. 31, 2018.

[4] J. Du, J. Futoma, and F. Doshi-Velez, "Model-based reinforcement learning for semi-markov decision processes with neural odes," *Advances in Neural Information Processing Systems*, vol. 33, pp. 19805–19816, 2020.

[5] I. O. Sandoval, P. Petsagkourakis, and E. A. del Rio-Chanona, "Neural odes as feedback policies for nonlinear optimal control," *arXiv preprint arXiv:2210.11245*, 2022.

[6] S. Massaroli, M. Poli, F. Califano, J. Park, A. Yamashita, and H. Asama, "Optimal energy shaping via neural approximators," *SIAM Journal on Applied Dynamical Systems*, vol. 21, no. 3, pp. 2126–2147, 2022.

[7] J. Rawlings and D. Mayne, *Model Predictive Control: Theory and Design*. Nob Hill Pub., 2009.

[8] P. Kidger, J. Morrill, J. Foster, and T. Lyons, "Neural controlled differential equations for irregular time series," in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds.), vol. 33, pp. 6696–6707, Curran Associates, Inc., 2020.

[9] M. Poli, S. Massaroli, A. Yamashita, H. Asama, J. Park, and S. Ermon, "Torchdyn: Implicit models and neural numerical methods in pytorch,"

[10] M. Lienen and S. Günnemann, "torchode: A parallel ODE solver for pytorch," in *The Symbiosis of Deep Learning and Differential Equations II, NeurIPS*, 2022.