Adaptive Neural Gradient Fields for Robot Planning and Control With Hardware In The Loop

Hussain Bhavnagarwala, Mechanical Engineering

Mentor: Wanxin Jin, Assistant Professor

School for Engineering of Matter, Transport and Energy

QR CODE

INTRODUCTION

- Gradient-based optimization for robot planning and control with hardware of unknown dynamics in the loop requires differentiation through the hardware.
- Differentiation through the hardware is possible using numerical-differentiation-like techniques like score function gradient estimators [1] and policy gradient reinforcement learning [2]. However, those lead to significant data inefficiency when incorporating into optimization-based planning and control frameworks.
- We propose an adaptive neural gradient field method for hardware differentiation. We focus on application of the proposed method in optimal control of a robotic system with unknown dynamics in the loop.

PROBLEM FORMULATION

Optimal control and planning of a robot with unknown hardware dynamics given an initial condition:

$$\min_{x,u} \; J(x,u) = \sum_{t=0}^{T} c(x,u)$$

$$\text{subject to:}\quad x_{t+1} = f(x_t, u_t) \quad \text{(hardware)}$$

$$x_0 = x_{\text{init}} \quad \text{(initial condition)}$$

The dynamics $f(x_t, u_t)$ is hardware, does not always have an analytical form that can be differentiated for optimization.

The numerical differentiation methods, such as score function gradient estimators [1] and policy gradient reinforcement learning [2] is data intensive.

Our objective:

Develop a data-efficient method to solve the above optimal control problem by differentiate through the unknown dynamics (robot hardware).

APPROACH

Randomized Smoothing (RS) [3]

- RS is to smooth a given (potentially non-smooth) function by convoluting the function $f(x,u)$ value with a distributions $\mu(Z)$ (e.g., Gaussian)

$$f_\varepsilon(x,u) = E_{Z\sim\mu}[f(x+\varepsilon Z, u)]$$

- Using integration by parts, the differentiation of RS smoothed function is :

$$\nabla_x f_\epsilon(x,u) = E_\mu[(f(x,u) - f(x+\varepsilon Z^i, u))\frac{\nabla \log \mu(z)^T}{\epsilon}]$$

- Using Monte Carlo to approximate

$$\nabla_x f_\varepsilon(x,u) = \frac{1}{M}\sum_{i=1}^{M}\left(f(x,u) - f(x+\varepsilon Z^i, u)\right)\frac{\nabla \log \mu(Z^i)^T}{\varepsilon}$$

- ε is used to indicate the variance of random samples. The greater the variance the smoother the function

Neural gradient field for hardware dynamics

Idea: train neural network as proxy of RS differentiation

Train neural networks with data from RS differentiation to learn gradient fields of hardware dynamics f.

$$\min_{\theta_A} \; E_{p(x,u)}||S_{\theta_A}(x,u) - \nabla_x f_\epsilon(x,u)||^2$$

$$\min_{\theta_B} \; E_{p(x,u)}||S_{\theta_B}(x,u) - \nabla_u f_\epsilon(x,u)||^2$$

$S_{\theta_A}(x,u)$ and $S_{\theta_B}(x,u)$ are the gradient fields of the hardware dynamics w.r.t. (x,u), respectively

Adaptive neural gradient field for optimal control

We integrate the neural gradient field to the iterative linear Quadratic Regulator(iLQR)[4] framework, and develop the adaptive neural gradient field for robot planning.

Input:

$Cost: J_n(x,u)$
$Hardware: f(x,u)$
$Parameters: \varepsilon, \#\ of\ samples, max\ iterations$

Initialization:

$S_{\theta_A} - neural\ gradient\ field\ \ (\nabla_x f(x,u))$
$S_{\theta_B} - neural\ gradient\ field\ \ (\nabla_u f(x,u))$
$u_0 - initial\ u\ trajectory$
$x_{init} - initial\ state$
$forward\ rollout\ on\ f\ (hardware) using\ u_0 and$
$x_{init}\ to\ get\ x_0\ trajectory$
$J_0 - initial\ cost\ value$

While loop: k

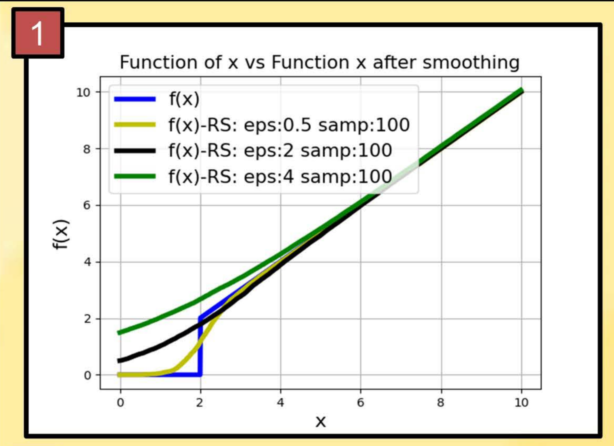
$backward\ pass\ to\ get\ \Delta u_k\ using S_{\theta_A} and S_{\theta_B}$
$forward\ rollout\ on\ f\ using\ u_k\ to\ get\ x_k$
$get\ current\ cost\ J_k$
$if\ J_k\ >\ J_{k-1}:$
$update S_{\theta_A} \& S_{\theta_B} by\ collecting\ (u,x)\ on\ f$
$around\ u_k$

until k > max iterations

Adaptive training: training was done only when the cost function dropped below the previous cost.

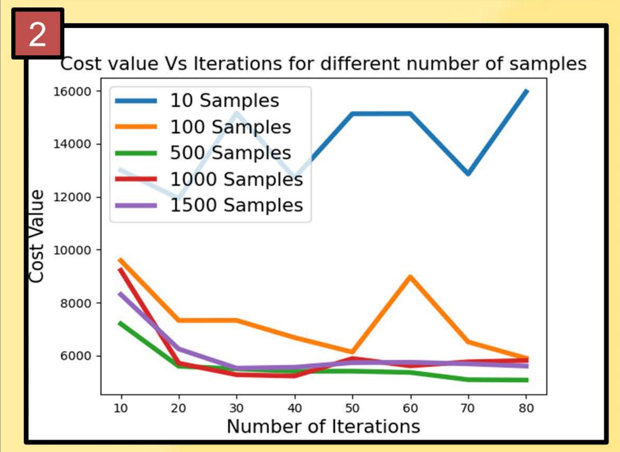
RESULTS & ANALYSIS

1



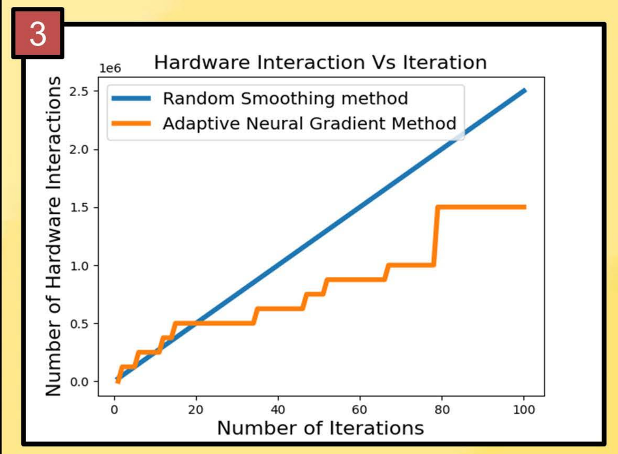
This is an example of RS, the blue function is approximated by the other curves that use RS. Increasing ε value creates a smoother curve but creates more deviation from the original function

2



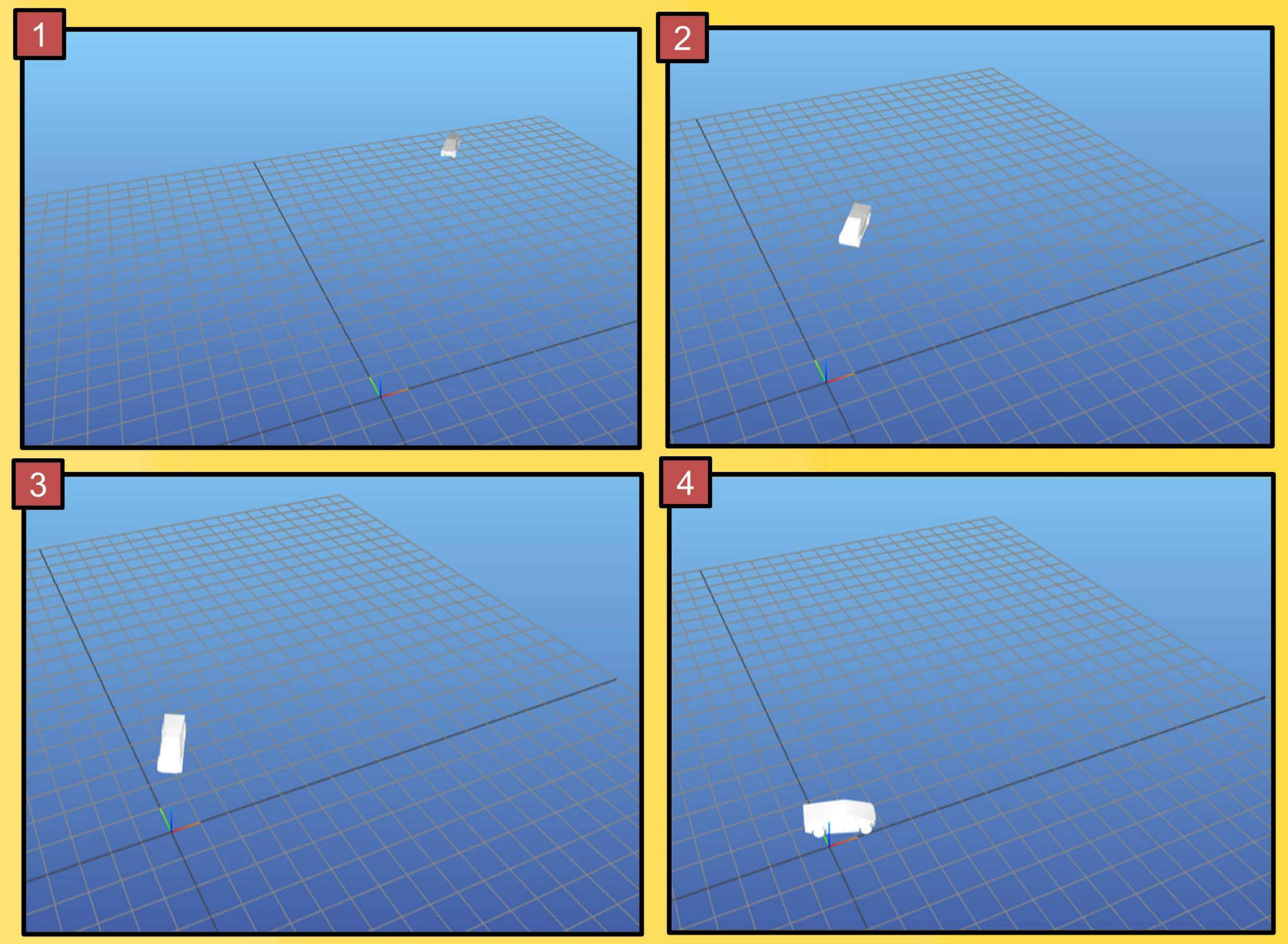
We compare the different number of samples used for the RS differentiation against the iterations. Here, we see that samples ranging from 500-1000 are best suited for hardware differentiation

3



We compare the data complexity (number of hardware interactions) of our adaptive neural gradient field method against that of the vanilla RS differentiation methods for iLQR. It can be seen there is a 60% decrease in computational requirements.

We have applied our proposed method to solve a ground vehicle navigation and planning problem. The vehicle dynamics was treated as a black box (hardware). Blow is the animation of the planned motion.



In future, we plan to test the proposed method in more complex systems, such as systems with physical contacts, e.g., robotic locomotion and manipulation, and move it onto hardware.

REFERENCES

[1] Shakir Mohamed, Mihaela Rosca, Michael Figurnov and Andriy Mnih,"Monte Carlo Gradient Estimation in Machine Learning" Journal of Machine Learning Research 21 (2020).

[2] Richard S. Sutton and Andrew G. Barto, "Reinforcement Learning: An Introduction"

[3] Quentin Le Lidec, Louis Montaut, Cordelia Schmid, Ivan Laptev and Justin Carpentier, "Leveraging Randomized Smoothing for Optimal Control of Nonsmooth Dynamical Systems ". in arXiv:2203.03986v2 [cs.RO] 11 Mar 2022

[4] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems." in ICINCO (1). Citeseer, 2004, pp. 222–229

MORE
Masters Opportunity for Research in Engineering

ASU Ira A. Fulton Schools of Engineering
Arizona State University