Aditya Pulipaka

COE379L – Joe Stubbs

Project 3 Initial Proposal

Developing Simulation Environment and RL Pipeline for Hybrid EM Levitation

Introduction and Problem Statement

The transportation of tomorrow will be defined by efficiency and speed. To maximize these two factors, the reduction of friction is paramount, resulting in the conception of the magnetic levitation train, and, soon after, the hyperloop. This idea gave birth to Texas Guadaloop, UT Austin's very own hyperloop engineering student organization. As the lead of the levitation sub-team, I have been pondering for months on how to develop not only a robust control algorithm, but a system that would allow for repeated success in HEMS (Hybrid Electromagnetic Suspension) control, which utilizes a u-shaped 'yoke' with two permanent magnets covered with copper coils as shown in Fig. A. The overarching goal is to be able to control systems of various shapes and sizes with relatively minimal recalculation, something that I believe is made significantly more possible by machine learning and ANNs/DNNs.

To lay the goals out clearly, this project aims to do the following. These steps will be further clarified in Data Sources / Methods Employed section. Firstly, a finite-element simulation must be conducted to generate a 4-dimensional lookup table for the magnetic yoke. Secondly, a basic physics simulation environment must be set up in python which takes in four PWM (Pulse-width-modulation) values and applies forces to a simulated test rig (shown in Fig. B) based on lookup table results. Thirdly, a cost function must be developed to optimize for successful levitation. Finally, a neural network architecture must be designed and implemented to take physical state as input and output PWM controls, then trained to optimize on the cost function.

Data Sources Used + High-Level Methods

The data used in this project will be a mix of real-world observations and Ansys Maxwell sparse simulation data, interpolated using python. Reference observations made in the lab will be compared with Ansys calculations to perform a basic sanity check of the viability of this project. If this succeeds, the following steps are dependent only on my programming and logical abilities, rather than the robustness of existing software.

The 4D lookup table will have (1) an average gap height axis, (2) a roll degree axis, (3) a left-coil current axis, and (4) a right-coil current axis. These variables correspond to the variable parameters of the Ansys model, allowing the population of the lookup table to be automated via pyAedt, a library that links python to Ansys Electronics Desktop.

The actual training data for the neural network will be provided in real-time by the python simulation script and the cost function. To simplify the scope of *this project*, time-dependence will be captured by passing the following input state vector: [Front_mm, Back_mm, Left_mm,

Right_mm, *Z_velocity, Roll_velocity, Pitch_velocity*]^T (velocities capture the necessary scope of time-dependence).

Deliverables

This project's deliverables include (1) a python notebook and csv file that show initial lookup table data collected through Ansys simulation, (2) a python notebook that develops the simulation environment and cost function, then walks through the formulation of a machine-learning model that optimizes on the cost function by interacting with the simulation environment, (3) a completed model and visuals that demonstrate effective learning of the environment and performance on multiple test runs (that slightly tweak environment parameters success rate vs. parameter variation %) (This should display robustness (or room for improvement) in diverse environments, which may display a strength (or weakness) of the reinforcement learning method detailed above), and (4) a video detailing the process and documenting the project as a whole, as well as possible future directions. Other visuals will be included as well, such as model architecture and input vector comparisons for overall learned algorithm success rate (% success vs # iterations, iterations to 90% success vs # parameters, etc.)

Fig. A

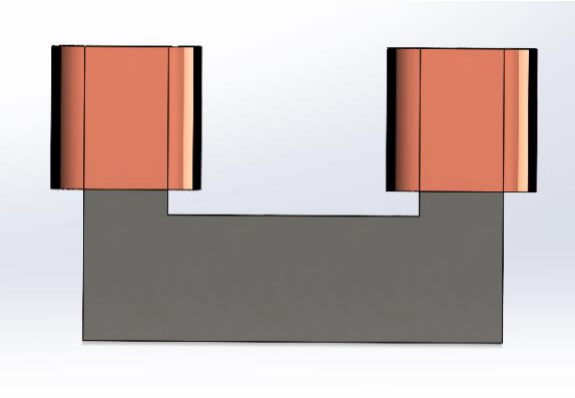
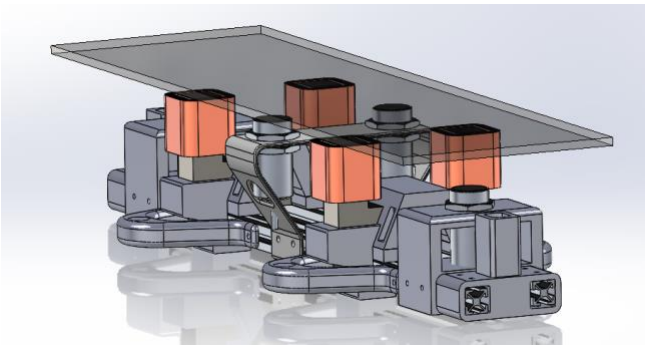


Fig. B



Reinforcement Learning Pipeline Visualization