Model Predictive Control Using Casadi with NeuroMANCER

Nam T. Nguyen, M.S. student at Northern Arizona University

1. Motivation

Model Predictive Control (MPC) is a powerful and widely used control strategy yet has not been fully integrated into the NeuroMANCER library [1]. Currently, the library only features Differential Predictive Control under the "Learning to Control with Differentiable Models" section. This project seeks to bridge this gap by proposing an approach to implement MPC using NeuroMANCER variables. The goal is to establish a new section in the library titled "Learning to Control with Model Predictive Control." Given the popularity and efficiency of the CasADi library [2] in solving optimization problems, this project focuses on leveraging CasADi for MPC implementation for both known and unknown dynamical systems.

2. Proposed Approaches

The MPC node is defined as: $MPC = Node(mpc\_implement, ['X', 'X\_ref'], ['U'])$.
The system node is defined as: $sys\_node = Node(model, ['X', 'U'], ['X'])$.

Here, 'X' represents the current state, 'X_ref' is the reference state, and 'U' denotes the control input. The mpc_implement function utilizes CasADi to compute the optimal control input ('U') by minimizing a defined loss function in MPC.

- *MPC for Discrete State-Space Systems*

Sate-space systems are based on PyTorch variables in NeuroMANCER. To integrate with CasADi, these models need to be converted into CasADi symbolic expressions. This process involves converting all model parameters into NumPy types using ".numpy()". The resulting NumPy-based model is the foundation for CasADi symbolic functions, which are then used in the optimization framework. After computing the optimal control inputs via the loss function of MPC, the outputs are converted back to PyTorch tensors to maintain compatibility with NeuroMANCER.

- *MPC for Unknown Dynamical Systems (using data-driven models and l4casadi)*

Data-driven models will be trained on collected data to identify unknown dynamics systems. Using "blocks.MLP" to determine neural networks, these models are always represented as Torch variables. For this reason, these neural network models must be converted to CasADi, which means all weights and biases must be transferred. This project converts them using the l4casadi [3].

The converted models are then incorporated into the mpc_implement function to compute the optimal control inputs. Afterward, the same structure as MPC for the above case can be adopted for control. "l4casadi" could be implemented as a back-end component of NeuroMANCER.

3. Simulation Results

The methodologies have been implemented and published on my GitHub repository.

4. Conclusion and Future work

This project demonstrates a framework for integrating MPC into NeuroMANCER, leveraging the strengths of CasADi for optimization. Future work ***will extend this framework to a broader range of systems with physics-informed neural networks and continuous systems*** on 'docs/psl' systems.