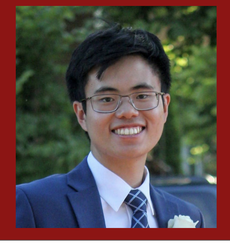
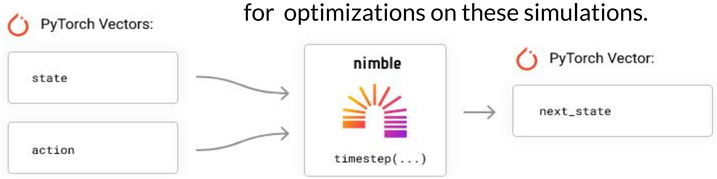
Adaptive Realtime MPC Class for the Nimblephysics Library

Raymond Suo, Keenon Werling, Prof. Karen Liu



Background

Nimble is a "analytically differentiable fork of the popular DART physics engine" that allows for the use of a "physical timestep as a non-linearity in … PyTorch neural networks" (1). Physics simulations using Nimble are represented by a skeleton with position and velocity. Forces (friction, gravity) can be applied to the skeleton during each timestep. Additionally, each Nimble timestep with the current state and control forces is a valid PyTorch function, allowing for optimizations on these simulations.

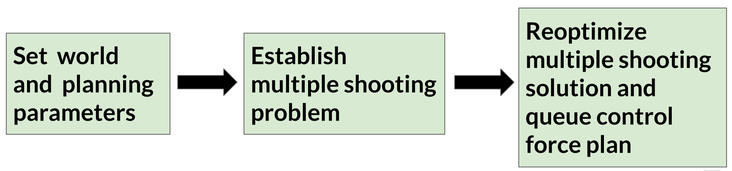


In addition to optimizing singleshot and multishot trajectories, Nimblephysics has a realtime MPC class that keeps track of the state of a realtime physics simulation and and stores future control force; it also provides the tools for realtime optimization of the simulation and planning of future controls to reach some simulation goal.

Goal

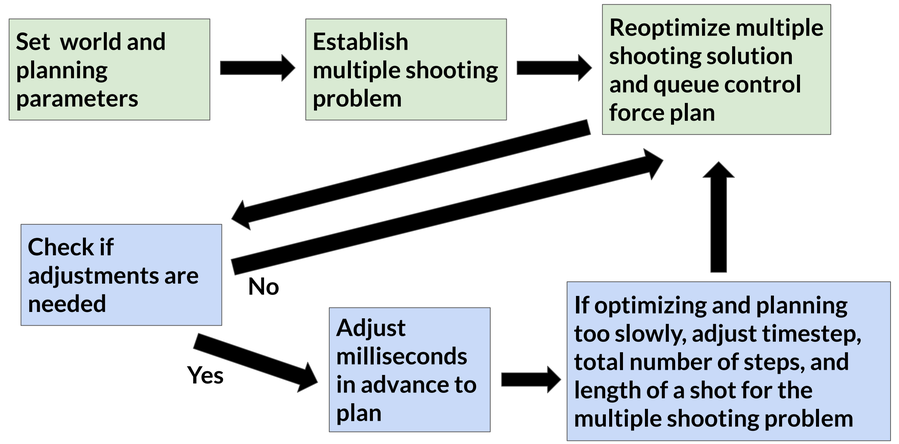
The goal of this research was to make the existing Nimble realtime MPC class more adaptive and, when needed, adjust simulation and optimization parameters to improve the performance of the optimizer. When the simulation gets out of sync with future control forces planned by the optimizer, we want the MPC to adjust and bring the simulation back on track towards reaching the simulation goal.

Existing MPC Model



The realtime MPC first initializes world and optimization parameters, including the time duration of a step, the total number of steps, and the length of a shot for the multiple shooting problem. Next, the multiple shooting problem is established using these three parameters and a loss function. Finally, the MPC continuously updates the multiple shooting problem with the current states and reoptimizes the multiple shooting solution. The resulting new control force plan is then queued.

Adaptive Changes To MPC Model



Problems with the existing MPC class arise when the system and control force plans get out of sync or when the optimization and queuing of plans for future controls happens too slowly. To address these problems, an adjustment step was added (shown in blue) in between the planning. The first adjustment was to set the time in advance to plan equal to 1.2 multiplied by the time it took to optimize the previous plan. This ensures that the plans start in the future and that the optimizer does not change control forces that already happened. The next adjustment addresses the case when the planning of future steps happens too slowly, or when the following inequality is not satisfied:

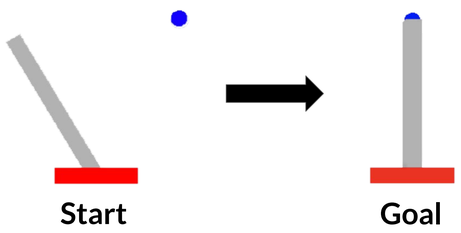
lastOptimizeTime / numSteps <= timeStep / 9

In other words, we want the time it takes to plan a step at least 9 times faster than the time of the actual step in the simulation. If this constraint is not satisfied, then in the newly added adjustment step, the timestep is increased until equality holds.

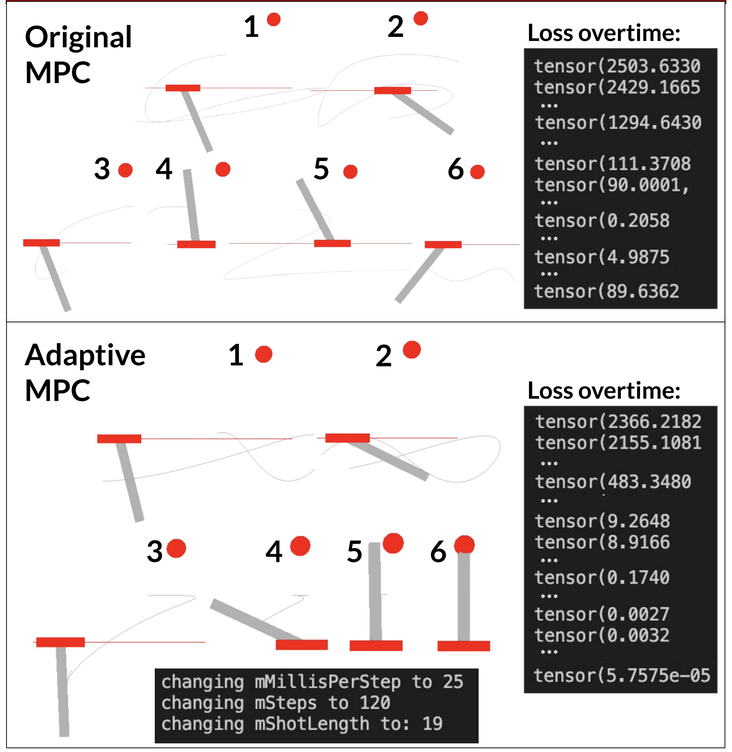
Additionally, the number of steps and the length of a shot are decreased by the same factor so that the total time we are planning in the future does not increase.

Testing

To test the realtime MPC class, a cartpole simulation was set up using Nimble. At each step, a control force was applied to the cart, and the goal of the simulation was for the cart to stabilize at a specific position with the pole stabilized vertically. The loss function was calculated using the cart's position.



Results



The results after testing on the cartpole simulation demonstrate an improvement with the adaptive MPC. Using the original MPC, the cartpole attempts to reach the stable vertical position, but fails to plan fast enough, and so the cartpole falls behind. Using the adaptive MPC, the program adjusts to speed up the optimizer, and the cartpole eventually stabilizes at the vertical position. This is also shown in the printed loss overtime.

Future Work

In the future, more testing needs to be done on the effectiveness of the adaptive changes to the Nimble MPC class. Additionally, there is only one criteria that is currently being used to determine when the simulation and optimization parameters need to adjusted; there is a need to add more criteria and adjustments to keep the realtime planning in sync with the simulation.

References

(1) https://nimblephysics.org/, (2) https://nimblephysics.org/docs/,
(3) https://github.com/keenon/nimblephysics