Blimp Controller Benchmarking Tests

Mihir Kasmalkar

November 2023

1 Introduction

The test details were as follows:

- Two platforms were tested: a Lenovo Ideapad-3 with an Intel Core i5 processor running Ubuntu and a Raspberry Pi Zero 2 W running Raspberry Pi OS Lite (a headless OS).
- Three controllers were tested: an MPC using Gurobi, an MPC using Casadi, and a tracking controller using the feedback linearized dynamics.
- The Gurobi MPC was implemented with a time horizon of 250 time steps when run on the computer and 20 time steps when run on the Raspberry Pi, while the Casadi MPC was implemented with a time horizon of 20 time steps on both platforms. This was to get solve times less than 100-ms.
- The full nonlinear dynamics were simulated with a time step of 0.05 sec.

The controllers implemented the trajectory in Fig. 1. The blimp follows the helical path, rotating clockwise about the z-axis as it does, and stops at the end. This takes 20 seconds. At the 20-second mark, the LQR which damps oscillations in the zero dynamics is enabled in the tracking controller, while the MPCs do not change. The simulation continues for 100 seconds as the zero dynamics oscillations gradually die out.

Code: `https://github.com/MKasmalkar/blimp_mpc`

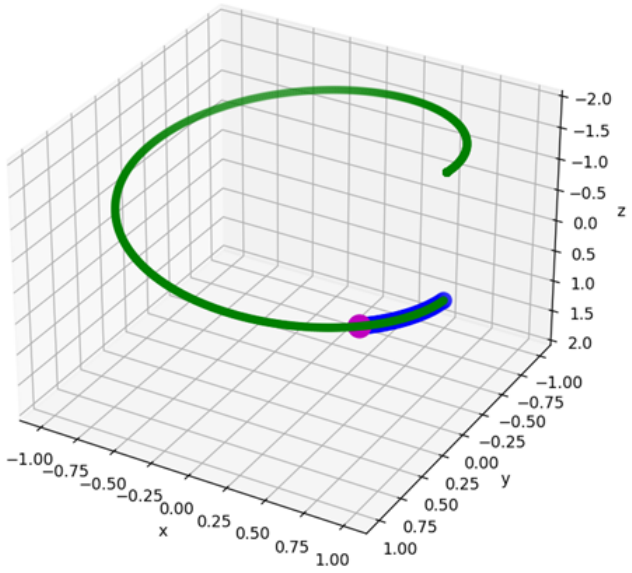

Figure 1: Trajectory

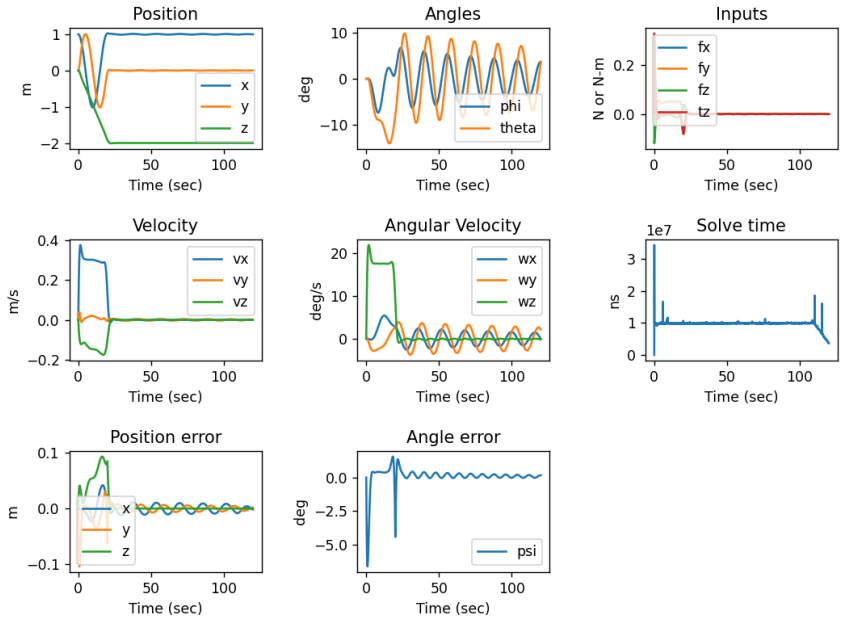

Figure 2: Plots of waveforms from Gurobi MPC simulation

2 Solve times

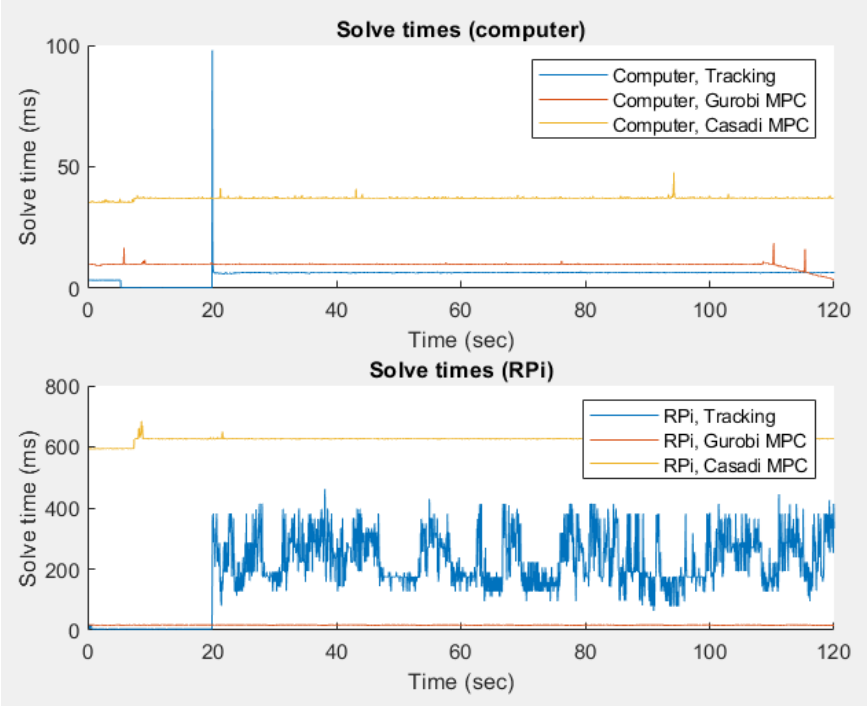

Figure 3: Solve times

2.1 Observations

- On the computer, the tracking controller is always the fastest, with a minimum solve time of 0.2-ms and a maximum solve time of 7-ms (not including the spike that occurs at 20-ms).
- On both platforms, Casadi is always the slowest, averaging 40-ms on the computer and 620-ms on the RPi.
- On the RPi, the tracking controller is the faster than the Gurobi MPC until the attitude-stabilizing LQR is enabled, at which point it becomes slower than the Gurobi MPC. The Gurobi MPC solve time is around 16-ms on the RPi and 10-ms on the computer. The tracking controller achieves solve times around 3-ms before the LQR is enabled and around 300-ms afterwards.

2.2 Position error

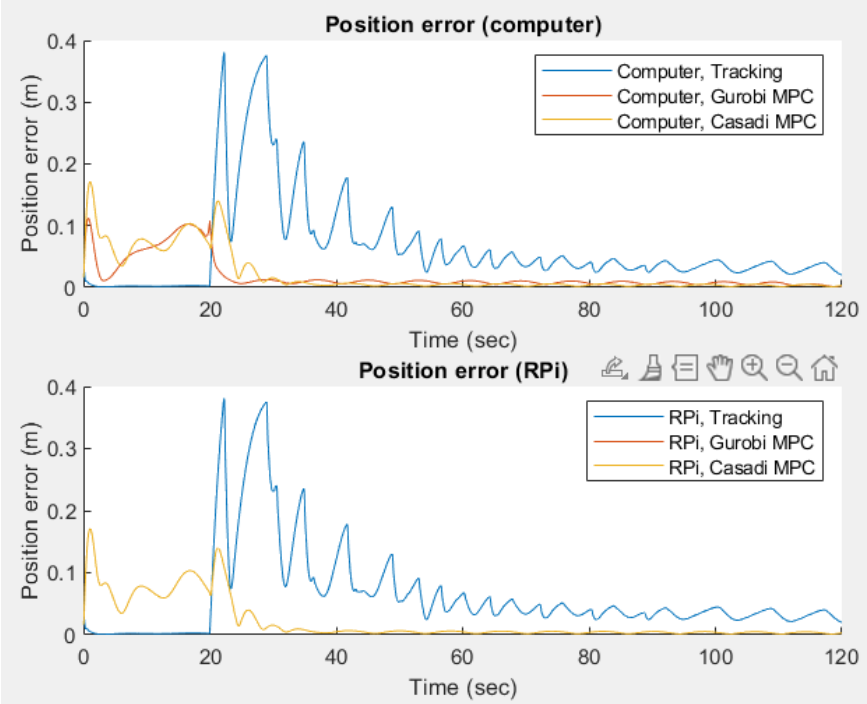

Figure 4: Position error (norm of x, y, and z error)

2.3 Observations

- The tracking controller has almost no position error until the attitude-stabilizing LQR is enabled. This is because the tracking controller places a higher priority on damping zero dynamics oscillations than the MPCs do. There is a trade-off between position accuracy and fast damping, as the forces required to damp oscillations create translating motion.
- The MPCs have tens of centimeters of position error during the helical path traversal, but they have almost no position error once the blimp reaches its endpoint.
- The MPCs do also attempt to regulate the attitude dynamics to zero, but this is overridden by the cost associated with the larger position error (in these simulations, the running, terminal, and input cost matrices for both MPCs are identity matrices).

2.4 Control effort

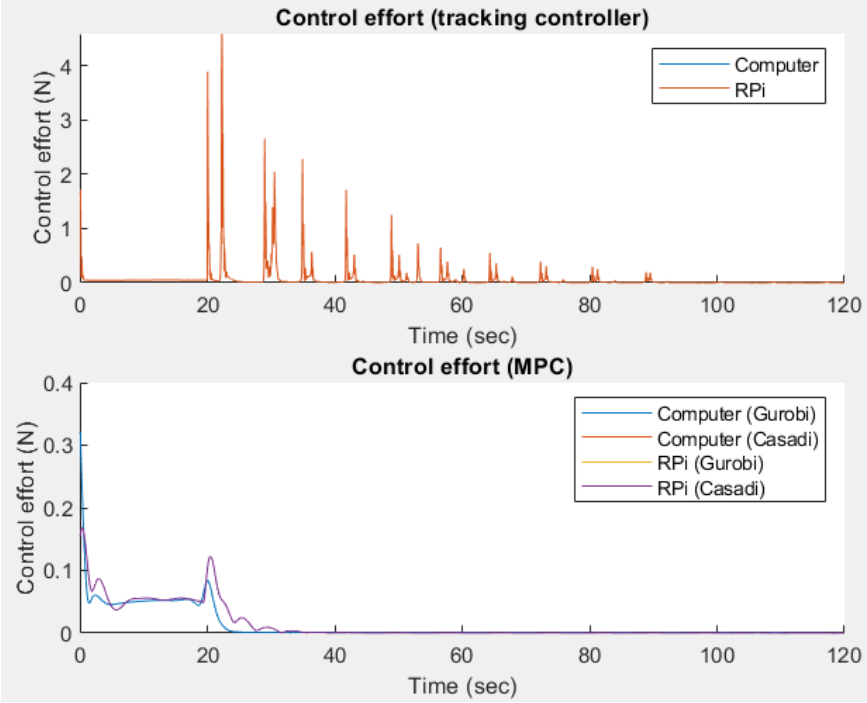

Figure 5: Control effort (norm of x, y, and z force)

2.5 Observations

- The tracking controller requires the most control effort when it is damping zero dynamics oscillations. Large damping forces occur in spikes, which are used to reverse the direction of the blimp's momentum and regulate oscillations to zero.
- The MPCs require about the same control effort as the tracking controller during the initial 20-second period when the blimp is navigating across the helical path and almost no control effort after the blimp has settled at the end.
- The control effort required by both of these controllers ignores real-world actuator saturation limits, since the blimp thrusters are limited to 10-mN.

2.6 Attitude-keeping

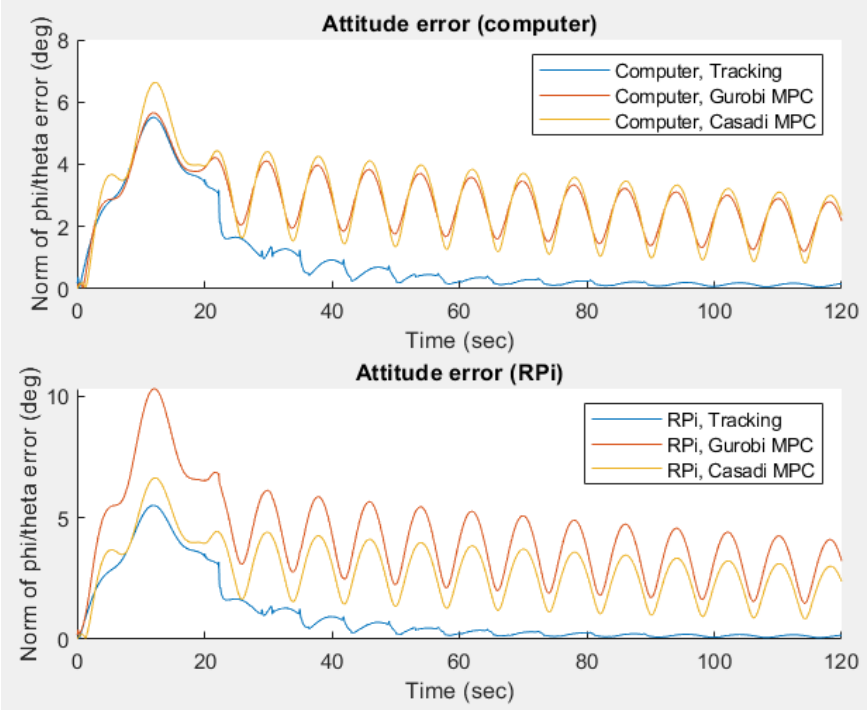

Figure 6: Attitude error (norm of pitch and roll angle error)

2.7 Observations

- The tracking controller most effectively damps the zero dynamics.
- The MPCs allow the zero dynamics to persist, essentially undamped. There is a cost term associated with nonzero pitch and roll in the MPC optimization problem, but this is negligible compared with the cost associated with position error.

3 Shortened time horizon

I ran additional tests of the MPCs on the Raspberry Pi with the following modifications:

- The time horizon was shortened to 5 time steps.
- The period of the simulation loop (which is also the period of the control loop) was increased to 0.2 seconds.

These changes were to achieve a time period sufficiently large and a solve time sufficiently small that the solution to the MPC optimization problem could be computed within a control loop period, making it possible to implement the controller in practice.

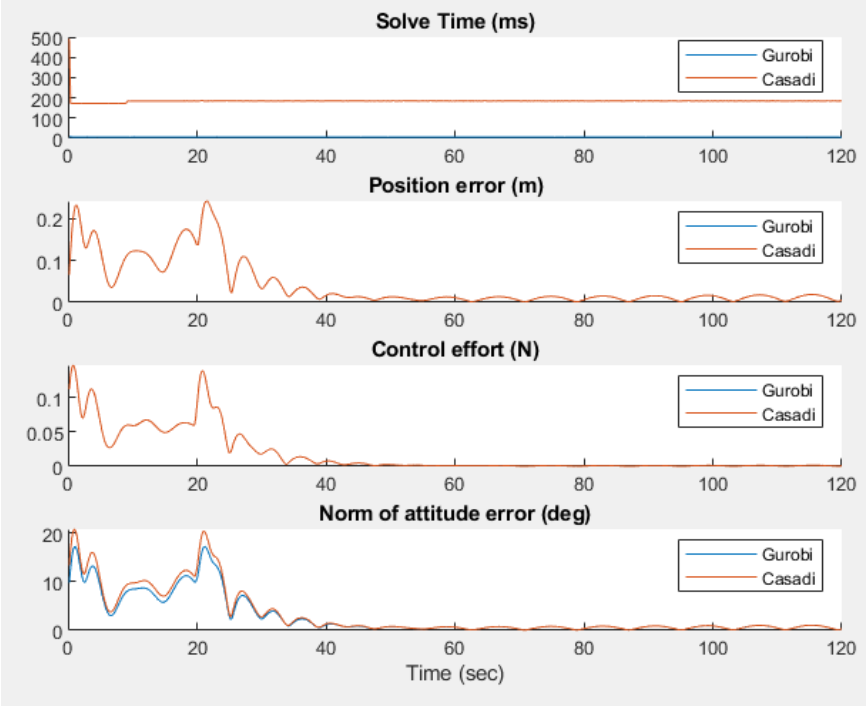

Figure 7: Shortened time horizon

3.1 Observations

- The solve times on both platforms are shorter than in the previous simulations, with Gurobi taking 6-ms and Casadi taking 200-ms.
- The position and attitude error are slightly larger, a consequence of the shorter time horizon.