0.1 Chameleon Mode

In the previous FOM study there was no guarantee of improvement in the obtained trajectory by considering new modes in a previously used family. It needs to be clarified that, even if in the previous study there was no global score function implemented, this effect could be so big that it could be observed with the naked eye. An ilustrative example would be the following. A family containing only the sticking mode $F_1 = S$ would greatly outperform the extended family $F_2 = S \bigcup D_2$. On the other side a further extension $F_3 = F_2 \bigcup D_1$ would outperform the two previous families TODO: ADD PHOTOS OR VIDEOS OF THIS.

The performance decrease experienced in F_2 can be explained by the fact that the FOM optimizer can choose optimal trajectories that will never be able to be applied. For example, the optimal control solution at a certain optimization iteration can be to stick for one time step to then slide up for another time step and finally stick for the rest of the finite time horizon prediction. The control law for the controller is to execute the first step of the optimal trajectory, so it the pusher will move without sliding for one time step and then the control law will be recomputed. The problem lies in the fact that no mode starts with a sliding up state, so the trajectory planned in the previous iteration is no longer considered. If the first sliding step of the trajectories obtained from S and D_2 are similar there will be no appreciable but the solution of D_2 may sometimes rely on the possibility of sliding to follow a more aggressive control trajectory that would have a smaller TODO: FINISH THIS.

The conclusion these experiences led us to was that, using [TODO: Cite] terminology, there was no sequencial improvement property when adding new modes. Solving the MPC using the FOM approach has similarities with their study: Each of the modes can be considered a heuristic and the strategy of recomputing the control policy at a certain frequency by using the same heuristics can be viewed as a rollout technique. It's important to note that the concepts of sequencial consistency or optimality cannot be applied to our case without some previous modifications, because of the existance of a receding finite time horizon, but for every FOM problem we can formulate an associated Full Horizon Family of Modes (FFOM) as follows: TODO: Define it.

If we take the family F_1 of the previous example and we consider it's associated FFOM problem, we can see that, in an ideal problem (TODO: define it in nomenclature), it's clearly sequencially consistent (and so, sequencially improving): Consider that at a certain control iteration there's a full horizon of N steps to reach the end of the trajectory. Then the S mode will get the optimal path to reach the goal while keeping the sticking contact at all times, obtaining a sequence of position states $\mathbf{x(x_0)}* = (x_1^*, x_2^*, ..., x_N^*)$ and control inputs $\mathbf{u(x_0)}^* = (u_0^*, u_1^*, ...u_{N-1}^*$ for the optimal path from the initial position to the goal. Then, the controller will execute u_0 for one control step and the pusher slider system will be in position x_1. The only heuristic it will consider in the next optimization step is to stick until reaching the goal, so the optimal trajectory obtained will be $\mathbf{x(x_1)}^* = (x_2^*, x_3^*, ..., x_N^*)$ and $\mathbf{u(x_1)}^* = (u_1^*, u_2^*, ...u_{N-1}^*$, being sequencially consistent. In a similar way, it can be shown that no sequencial consistency nor improvement can be guaranteed for family F_2: if mode D_2 is chosen at a certain time step, giving an optimal trajectory $\mathbf{x_0(x_0)}^*\mathbf{u_0(x_0)}^*$, a mode D_1 is required in the family to consider the continuation of the previous optimal trajectory $\mathbf{x_0(x_1)}^*$, $\mathbf{u_0(x_1)}^*$ again. As it is not the case, the new optimal trajectory has no relation with the previous one and consistency and improvent cannot be guaranteed. On the other hand sequencial improvent is guaranteed for family F_3 because, following the previous reasonment, including the mode D_1 in the rollout allows to continue considering optimal trajectories from a previous step of a D_2 mode and the S mode considering the ones obtained from a D_1 mode.

We considered important to guarantee sequencial improvement for the FFOM problem even taking into account that our problem is not in the ideal world (modelling errors and different frequencies for the

optimization problem and the controller can lead the state obtained when applying u_0 to x_0 to be different from x_1) and that we don't have a full horizon. It was not a sufficient condition to obtain an overall improvent when adding modes to a family, but it was a required one. Another important motivation to try to guarantee it was that the simulation results showed that families that guaranteed improvement in the FFOM problem outperformed the ones that did not.

In order to guarantee it we decided first decided to only use families that would always compare the current possible trajectories to the immediately previous optimal one (in the FFOM problem). The problem with this approach is that for any mode $M = (S_1, ..., S_N)$ you add include in the family you need to include any suffix of it extended to length N $(S_i, ...S_N, S_{N+1}, ...S_{N+i-1})$ and suffixes of the newly added modes would also be required. Given certain mode patterns and adding simple extension tails formed by i - 1 copies of the same mode may keep the number of modes low. For an S mode no extra modes are required and for a D_n one you only need to extend the family with an S mode and all the D_i modes for $i \in (1, n-1)$. Even taking this into account, we considered adopting this strategy constrained the problem of family design more than we wanted to and added more extra nodes that was strictly necessary, so we decided to change from static families to dynamical ones instead.

The dynamical adjustment we added consisted in including a dynamic mode that would force the algorithm into considering the previously chosen path to guarantee sequencial improvement. This mode, that we called chameleon mode, simply includes the maximal suffix [TODO: add definition] of the previously selected mode extended with a single sticking state. TODO: Add something.

The chameleon mode functionality was added in the simulation interface and it's performance was tested in simulation. As we expected, there was a great improvement in families that didn't have a sequencially improving structure without it. TODO: Add figures with comparison and some comment.

With the chameleon mode, an improvement when adding modes could be observed to a certain extent, as will be discussed in **??**.

Given the good properties of the chameleon mode, we decided to include it in the FOM formulation. From now on, unless spececified otherwise, we will use FOM to refer to the family of modes with chameleon mode implementation.