I. Introduction

In this coursework, a closed-loop MPC controller was implemented to move a gantry crane between two points, located in a rectangular outline, with up to ten ellipses. The crane has to move without violating the physical, work and time constraints.

II. Solution Overview

The final submission uses a linear time-varying predictive controller with decreasing horizon length. It relies on a non-linear plant model with non-linear constraints. The optimizer uses sequential quadratic programming (SQP) to find the optimal predicted path at every time step. Both the constraints, and plant model are linearized at every time step along the predicted trajectory. At the first iteration, the trajectory from a 2D path planner is used as an initial guess. Hard constraints are placed on both the load and the cart to ensure they settle in time within the tolerance ϵ of the target. Similarly to Part 1 implementation, work was minimized in the cost function by penalizing high input power and cart speed (1).

III. System Dynamics

In this second part, the trajectories of both the cart and load are more complex as the space is non-convex. For this reason, it was particularly important to minimize model errors as much as possible to avoid collisions with the obstacles. Therefore, a non-linear plant model derived from Euler-Lagrange equations was used for our controller. This model accounts for the impact of friction on the system; a factor that was not considered in Part 1. It is discretized using the implicit trapezoidal rule for the set sampling time T_s.

At every time step, the model is linearized around the entire predicted trajectory. This was implemented by making use of `fmincon` SQP option via the `nlmpc` wrapper. This approach yields much smaller model errors than the one adopted in Part 1, as the linearization happens around the predicted state rather than the equilibrium point. Using a non-linear solver was also considered. However, this led to a significant increase in computation time with no apparent difference in quality (see section VII for more details).

IV. Constraints Formulation

The constraints for the rectangular outline were included using the same method as Part 1 (1). The shape outline was included as a controller constraint by breaking it down in a pair of parallel lines. Each pair was expressed in a single inequality by converting the lines to their standard form, $Ax + By = C$, and using the coefficients in the inequality shown in equation 1. Expressing the lines in their standard form means that any orientation will be supported, even if they are vertical.

$$C_{line1} < Ax + By < C_{line2} \tag{1}$$

As the position of the load might not match the location of the cart, an additional constraint was added to force it within the shape outline.

$$C_{line1} < Ax + By + rxsin(\theta) + rysin(\phi) < C_{line2} \tag{2}$$

The elliptical constraints were added by ensuring that inequality 3 was respected. The same inequality was applied to the cart, similarly to equation 2.

$$\frac{(x - x_c)^2}{a^2} + \frac{(y - y_c)^2}{b^2} > 1 \tag{3}$$

The inputs were also forced to stay within their maximum allowed range [-1,1] with hard constraints.

Each of those constraints were applied to the entire crane trajectory. However, for the last 10 steps, a hard constraint on the position was added to force the cart to be within tolerance of the target point (see equation 4). This means that the crane will arrive $10 \cdot T_s$ seconds ahead of time, giving it some margin to settle. This approach adds robustness to the controller as the final position is not dependent on the cost function. Each of those constraints was linearized at every step along the trajectory using SQP.

$$target - \epsilon < x < target + \epsilon \tag{4}$$

All the constraints both on the outputs and inputs are hard. The disadvantage of this approach is that the optimizer may not find a solution, however, this was mitigated by using a path planner (see section VI). Using soft constraint was also considered, but this led to significantly higher computation time due to the increased number of variables.

V. Cost Function

In the final controller, the cost function is used solely to minimize the amount of work done. In order to this, we assumed that the total work consumed was proportional to the input power and velocity of the cart. Although this approach cannot guarantee that the cart will meet the set work constraint, it significantly reduces work (around 60% - see table I). Furthermore, this same technique was used in Part 1, and our controller did not break any of the work constraints in the assessed test cases. This was implemented by setting all of function weights to zero except, except those corresponding to $\mathbf{u_x}$, $\mathbf{u_y}$, $\dot{\mathbf{x}}$ and $\dot{\mathbf{y}}$. Note that as the weights on the final position are set to zero, some additional considerations need to be taken to ensure convergence to the target point. The prediction horizon must be extended to the settling time , T_f and a hard constraint on the final position is added. Finally, decreasing horizon must be used (2). The decreasing horizon

	Work *Default Course*	Work *Course 2*	Work *Course 3*
Cost Function (Part 2)	0.103	0.200	0.781
No Cost Function	0.351	0.791	1.391

TABLE I
Results after setting the cost function to reduce speed and input power for three different courses.

leads to higher computations, but the total run-time still stays significantly under the 60 seconds time-out. For example, it takes around 25 seconds to solve the default shape with $T_s = 0.05s$.

VI. Path Planning

To ensure convergence of our optimizer, it is key to provide a good initial guess. Unlike Part 1, the space is non-convex making it much harder to find a solution. On top of this, as explained in section IV, they are several hard constraints on both the inputs and outputs, meaning it is more likely to find our problem infeasible, especially if the initial guess of the trajectory is inadequate. In our final controller, an A* path planner was used to provide the starting trajectory of the optimizer in the first iteration.

The constraints were first converted into a discrete occupation map. The map is a 2D array, filled with boolean values; `true` for cells that are inside a constraints, `false` otherwise. The occupation map is then fed into a standard A* path finding block which returns a trajectory (3; 4). This trajectory is resampled and divided in p equidistant way-points, w, where p is the prediction horizon. Finally, a first guess, X_0, for the optimizer is formed following equation 5.

$$X_0 = \begin{bmatrix} w_{x1} & 0 & w_{y1} & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ w_{xp} & 0 & w_{yp} & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5}$$

This approach greatly improved the robustness of the controller. Without planning, 8 out of 10 of the test courses were deemed infeasible by the optimizer. After implementing it, all of them were solved without any constraint violation.

VII. Computation Time Reduction

Several techniques were used to minimize the computation time of the controller. Linearizing the constraints and plant model greatly reduced the computation time (see table II). This was further improved by selecting a good starting point for the optimiser . As explained in the previous section, for the first iteration the trajectory from path planning block is used. For all the subsequent iterations, the previous predicted trajectory is shifted by one time step and used as a guess (5).

Due to the way work is minimized, our controller prediction must extend to, at least, the final position at T_f. To reduce the amount of computation added by this design choice, a decreasing horizon controller was implemented. This controller has a prediction horizon that extends to the settling time (T_f). This means that no time is spent calculating the trajectory of the cart after that time. Furthermore, if T_f is small, less calculations are performed compared to a receding horizon approach, as no computational power is spent on predicting further than needed.

Finally, the optimizer settings were tweaked to speed up computation by allowing sub-optimal solutions. A maximum number of iterations was set, along with `FunctionTolerance` and `StepTolerance`. Although this led to a reduction in accuracy, the controller still passed all shapes present in the test bench without any violation.

Optimizer type	**Computation Time** (sec)
Non Linear	536
Linear Time Variant	12.48
Linear Time Variant with optimizer tuning	12.01

TABLE II
Computation for different optimizer choices on the default course, with $T_s = 0.1s$.

VIII. Robustness

The robustness of the controller was improved via several techniques. As explained in section VI, providing a good guess from the path planner in the first iteration improved the robustness greatly. T_f was also decreased by $10 \cdot T_s$ to ensure it will settle in time. Finally, the geometrical constraints were enlarged by a small factor, allowing for some error in the model. However, one disadvantage of this approach is that this might make the course infeasible. For this reason, the constraints were enlarged by no more than $\frac{1}{3} \cdot \epsilon$, where ϵ is the tolerance, or the minimum allowed gap.

IX. Verification

The final controller was tested using a test-bench of 10 shapes covering a wide range of scenarios and number of constraints. It was also tested with a series of different start and target points. In all cases, the controller settled at the target point by the settling time without any violation. On top of this, random perturbations of up to 40% in each of the parameters of the plant model were added. Our controller still passed each shape in the testbench.

References

[1] N. B. Guisset, "Predictive control core assignment part 1," 2021.

[2] W. Yang, G. Feng, and T. Zhang, "Decreasing-horizon robust model predictive control with specified settling time to a terminal constraint set," *Asian Journal of Control*, vol. 18, no. 2, pp. 664–673, 2015.

[3] *Marine Autonomous Exploration Using a Lidar and SLAM*, ser. International Conference on Offshore Mechanics and Arctic Engineering, vol. Volume 6: Ocean Space Utilization, 06 2017, v006T05A029. [Online]. Available: https://doi.org/10.1115/OMAE2017-61880

[4] E. S. Ueland, "Astar-algorithm," https://github.com/EinarUeland/Astar-Algorithm, 2021.

[5] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, "From linear to nonlinear mpc: bridging the gap via the real-time iteration," *International Journal of Control*, vol. 93, pp. 1–19, 09 2016.