Project-1 Report

vp2504

November 2024

1 Trajectory Optimization

The objective function $J(z)$ is defined as:

$$J(z) = \sum_{k=0}^{N} \frac{1}{2}(x_k - x_t)^T Q(x_k - x_t) + \sum_{k=0}^{N-1} \frac{1}{2} u_k^T R u_k$$

Where:

$$Q = diag(q_{px}, q_{vx}, q_{py}, q_{vy}, q_\theta, q_\omega), \quad R = diag(r_{u1}, r_{u2})$$

$$x_k = (p_{xk}, v_{xk}, p_{yk}, v_{yk}, \theta_k, \omega_k), \quad u_k = (u_{1k}, u_{2k})$$

x_k and u_k are the current states and control inputs, respectively. x_t are target states.

Parameters:

$q_{px} = 100.0, q_{vx} = 1.0, q_{py} = 100.0, q_{vy} = 1.0, q_\theta = 10.0, q_\omega = 10.0, r_{u1} = 1.0, r_{u2} = 1.0,$

System Dynamics Using the Euler Method

$\mathrm{P}_{x_{k+1}} = P_{x_k} + V_{x_k}\,\Delta t$
$V_{x_{k+1}} = V_{x_k} + \left(-\frac{(u_{1_k}+u_{2_k})\sin\theta_k}{m}\right)\Delta t$
$P_{y_{k+1}} = P_{y_k} + V_{y_k}\,\Delta t$
$V_{y_{k+1}} = V_{y_k} + \left(\frac{(u_{1_k}+u_{2_k})\cos\theta_k - mg}{m}\right)\Delta t$
$\theta_{k+1} = \theta_k + \omega_k\,\Delta t$
$\omega_{k+1} = \omega_k + \left(\frac{r(u_{1_k}-u_{2_k})}{I}\right)\Delta t$

Inequality Constraints

$0 \le u_1 \le 10$
$0 \le u_2 \le 10$

Targets

The targets are set at each time step to complete the circular trajectory with in given duration:

$$angular\_speed = \frac{2\pi}{DURATION}$$

$p_{x,target} = r \cdot \cos(angular\_speed \cdot t),$
$p_{y,target} = r \cdot \sin(angular\_speed \cdot t),$

$v_{x,target} = \frac{dp_{x,target}}{dt},$
$v_{y,target} = \frac{dp_{y,target}}{dt},$

$a_{x,target} = -r \cdot (angular\_speed^2) \cdot \cos(angular\_speed \cdot t),$
$a_{y,target} = -r \cdot (angular\_speed^2) \cdot \sin(angular\_speed \cdot t),$

$\theta_{target} = -\arctan 2\left(a_{x,target}, a_{y,target} + G\right),$

$\omega_{target} = \frac{d\theta_{target}}{dt}.$

Sequential Quadratic Programming (SQP) Algorithm Implementation

1. **Initialization:**

   - The algorithm starts with initial guesses for the states and controls, and Lagrange multipliers.
   - Key parameters such as maximum iterations, tolerance, and line search settings are defined.

2. **Quadratic Programming Subproblem:**

   - At each iteration, the algorithm computes the cost, gradient, Hessian, constraints, and their Jacobians.
   - A Quadratic Programming (QP) problem is formulated using these values to determine the search direction (p) and updates for multipliers using qpsolvers(CVXOPT).

3. **Line Search:**

   - A line search with merit function is performed to determine the step size (α) for updating the states and controls.

4. **Constraint Violation Handling:**

   - The algorithm tracks both inequality and equality constraint violations.

- Penalties for violations are incorporated into the merit function to prioritize feasible solutions.

5. **Convergence and Iteration:**

   - Convergence is checked using primal and dual optimality conditions.
   - The process repeats until the solution meets the predefined tolerance or the maximum number of iterations is reached.

Results

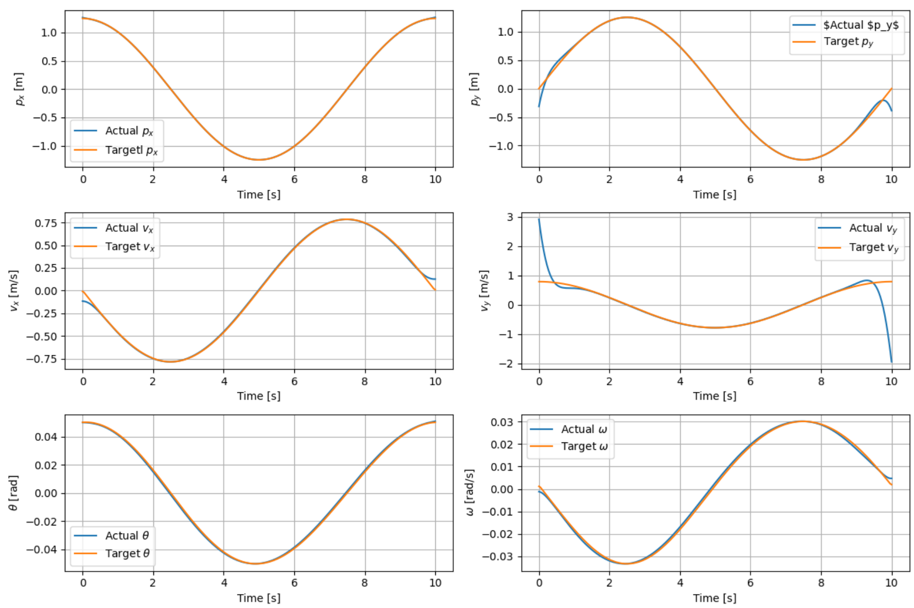

Figure 1: States as function of time

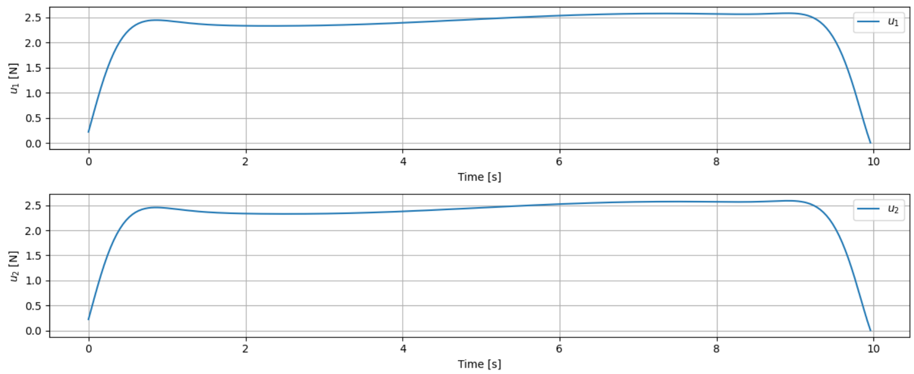

Figure 2: Controls as function of time

2 Model Predictive Control

The objective function $J(z)$ is defined as:

$$J(z) = \sum_{k=0}^{N} \frac{1}{2}(x_k - x_t)^T Q (x_k - x_t) + \sum_{k=0}^{N-1} \frac{1}{2} u_k^T R u_k$$

Where:

$$Q = diag(q_{px}, q_{vx}, q_{py}, q_{vy}, q_{\theta}, q_{\omega}), \quad R = diag(r_{u1}, r_{u2})$$

$$x_k = (p_{xk}, v_{xk}, p_{yk}, v_{yk}, \theta_k, \omega_k), \quad u_k = (u_{1k}, u_{2k})$$

x_k and u_k are the current states and control inputs, respectively. x_t are target states.

Parameters:

$$q_{px} = 300.0, q_{vx} = 1.0, q_{py} = 150.0, q_{vy} = 1.0, q_{\theta} = 10.0, q_{\omega} = 10.0, r_{u1} = 1.0, r_{u2} = 1.0,$$

The system Dynamics, Inequality constraints, and targets are same as Trajectory Optimization

MPC Controller Design

1. **Initial Check for Hover Condition:**
   - If the current time step ($t_{current}$) exceeds the total horizon (N), the controller outputs a hover thrust(M * G /2) command to maintain stability.
2. **Define Prediction Horizon:**

- Set the prediction horizon (N_p) based on the current time step ($t_{current}$) and the remaining horizon ($N - t_{current}$).

3. **Initialize Variables:**
   - Initialize the decision variable (z_0) containing states and control inputs for the entire prediction horizon.
   - The initial guess for states uses the current state ($x_{current}$), while control inputs are initialized to hover thrust.

4. **Set Final State Target:**
   - If the prediction horizon extends within the total time horizon, set the final state target (x_{N_p}) using predefined trajectory targets ($p_x, v_x, p_y, v_y, \theta, \omega$).
   - Otherwise, use the last predicted state as the final state.

5. **Solve the MPC Optimization Problem:**
   - Call the SQP solver(with CVXOPT), which solves the constrained optimization problem to minimize the cost function while satisfying system dynamics and constraints.

6. **Extract First Control Input:**
   - From the solution , extract the first control input (u_0) for immediate application.

Control Law

The optimal control law is derived iteratively using the Sequential Quadratic Programming (SQP) algorithm. At each step:

1. A Quadratic Programming (QP) subproblem is solved to determine the best control inputs and states for the prediction horizon.
2. Constraints (e.g., system dynamics, control limits) are enforced during the optimization process.
3. The first control input from the optimized sequence is extracted and applied to the system.

Results

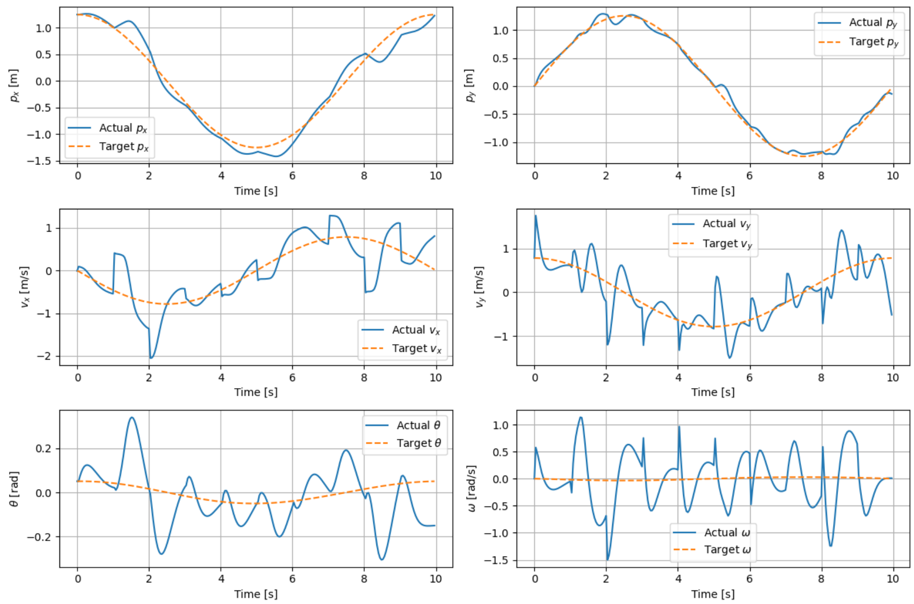

Figure 3: States with disturbance as function of time

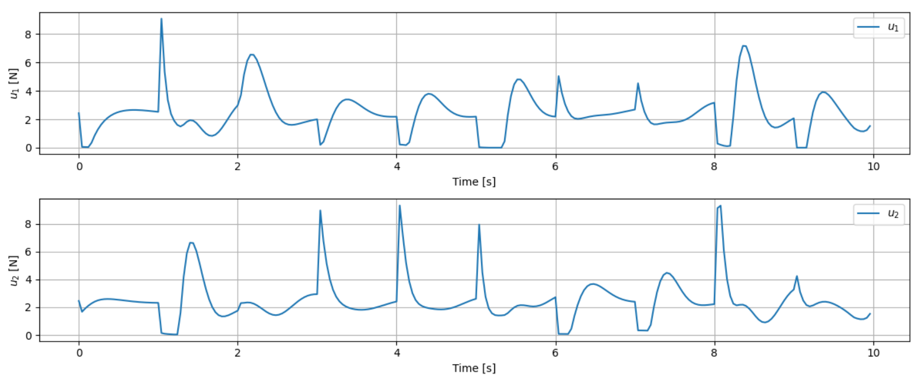

Figure 4: Controls with disturbance as function of time

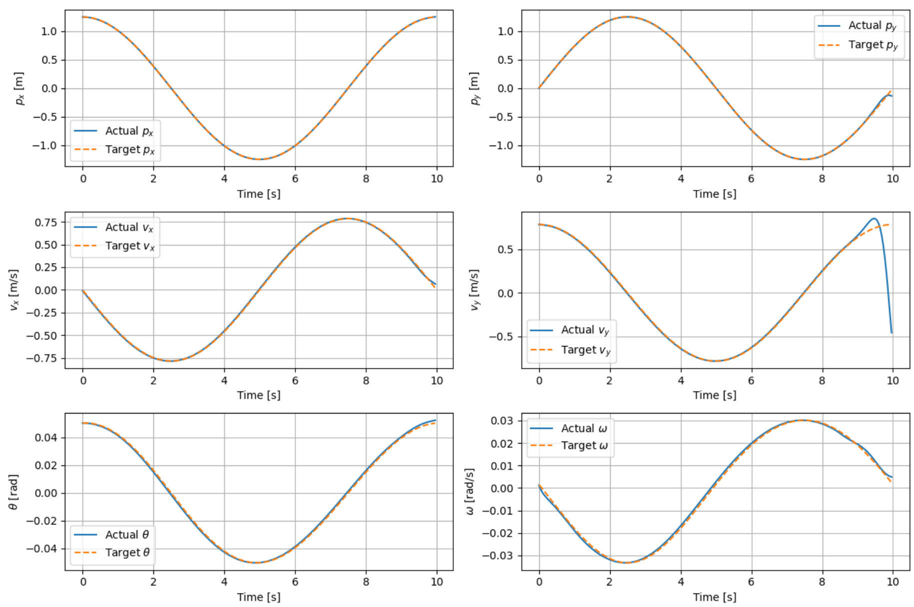

Figure 5: States without disturbance as function of time

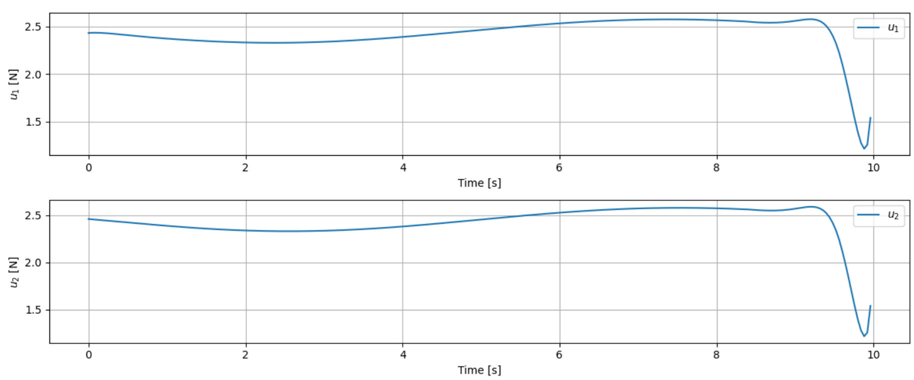

Figure 6: Controls without disturbance as function of time

3 Bonus

State constraint added for the quadrotor to maintain positive altitude.

$$-p_y \leq 0$$

Results

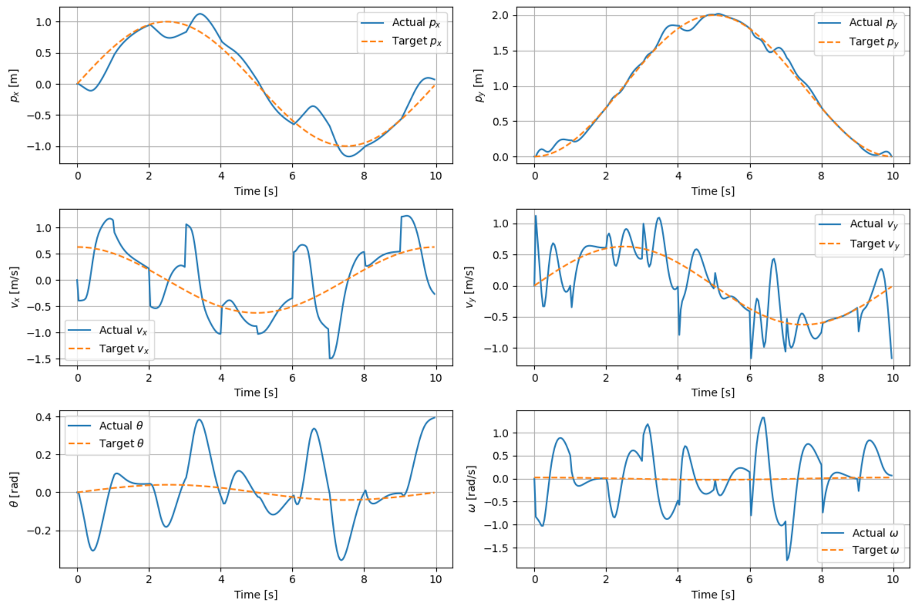

Figure 7: States with disturbance as function of time

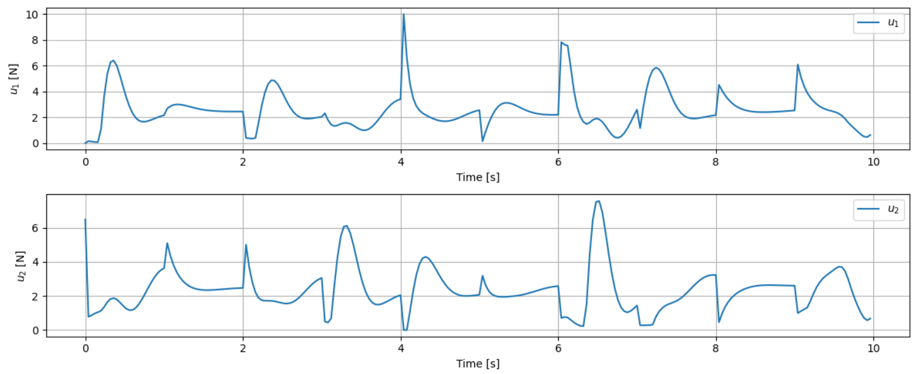

Figure 8: Controls with disturbance as function of time

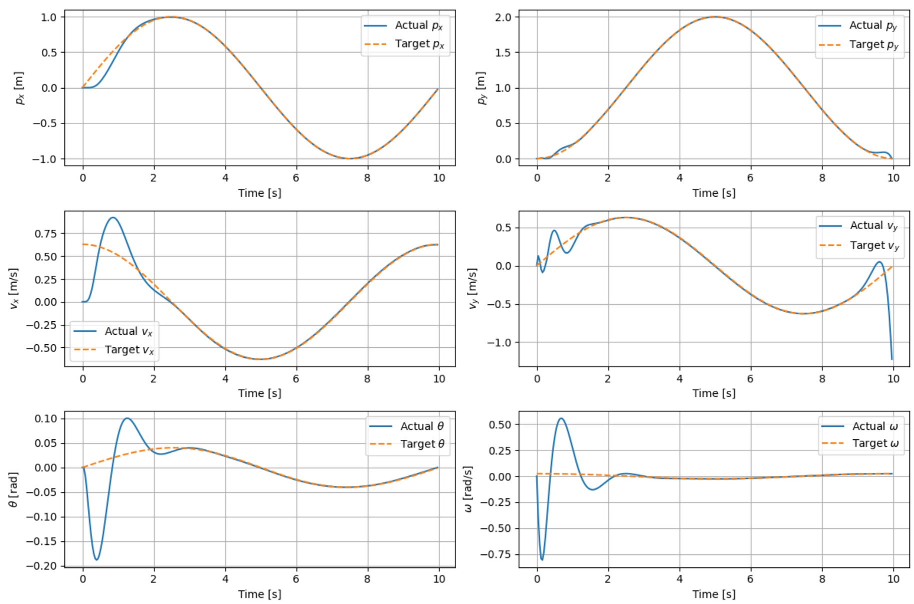

Figure 9: States without disturbance as function of time

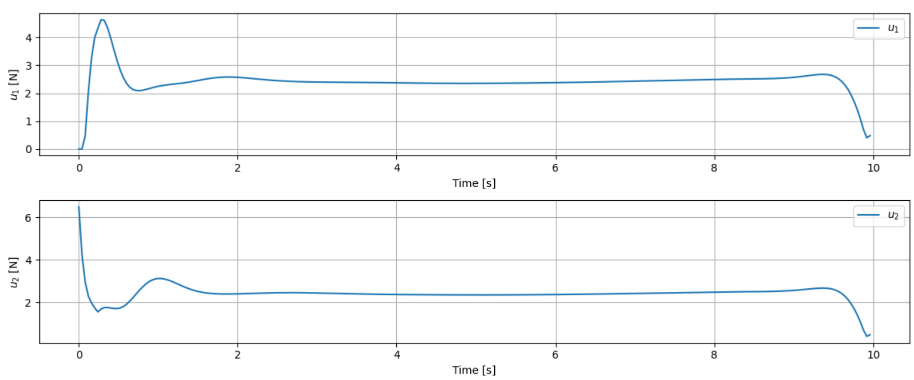

Figure 10: Controls without disturbance as function of time

Note : All animations are available in the runnable Jupyter Notebook or as videos included in the submitted zip folder.