Computing Trajectories for Vertical Landing

Computational Control Project

Naoki Sean Pross

ETH Zürich

Spring Semester 2023

Current State

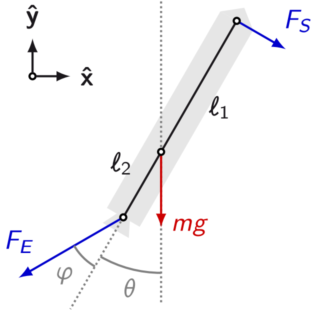
Rocket Model

Non-linear dynamics linearised around $z_s = 0$, $u_s = \begin{bmatrix} mg & 0 & 0 \end{bmatrix}^{\mathsf{T}}$:

$$z_{n+1} = Az_n + Bu_n,$$

where

$$z = \begin{bmatrix} x & y & \dot{x} & \dot{y} & \theta & \dot{\theta} \end{bmatrix}^{\mathsf{T}},$$
$$u = \begin{bmatrix} F_E & F_S & \varphi \end{bmatrix}^{\mathsf{T}}.$$



Controller

Decoupled PID controllers for F_E, F_S and φ, unaware of each other.

Behaviour

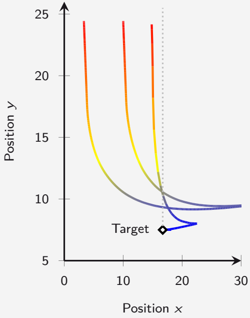
- Work well for "good" z_0
- Breaks easily $\rightsquigarrow$ need to retune
- Waits and high thrust near end

Failure Mode

Plots: Trajectories on the xy plane, color is the y velocity (red is fast).

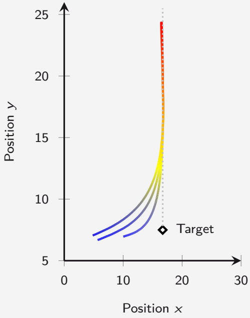
Bad x_0 Coordinate

Overshoots landing pad



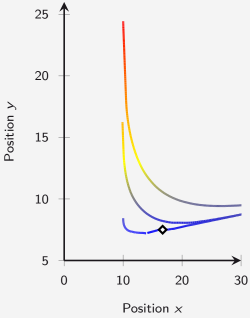
Bad θ_0 Angle

Not enough side thrust



Bad y_0 Coordinate

Too little thrust



Intuition

Decoupled controllers cannot coordinate in difficult situations (far from set point) and fail hard.

Recommendation

Proposed Controller

Relaxed linear MPC on linearised dynamics

Strengths

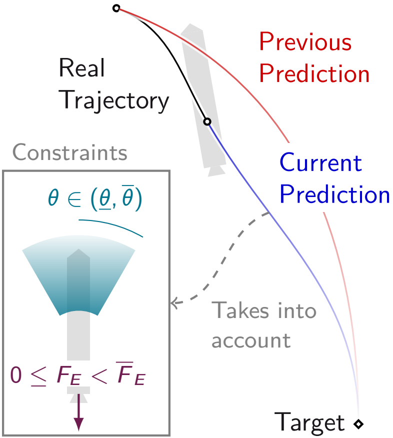
- Cutting edge, yet proven to be reliable
- Optimize fuel consumption
- "Easy" to specify constraints
- Possible to extend with more powerful theory if necessary (eg. sequential convex programming)

Weaknesses

- Computationally more expensive
- No theoretical stability guarantee (because of linearisation)

Key Idea of MPC

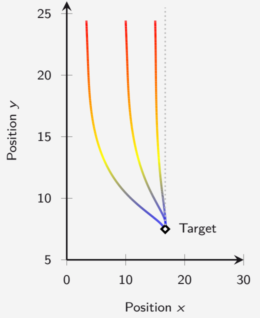
Continuously predict future to decide next action.



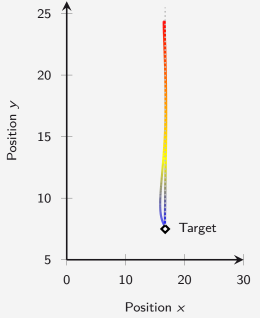
Demonstration

Plots: Trajectories on the xy plane, color is the y velocity (red is fast).

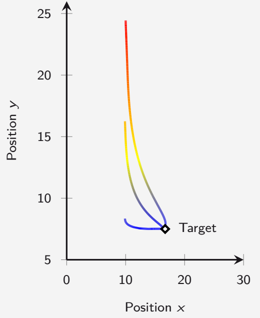
Bad x_0 Coordinate



Bad θ_0 Angle



Bad y_0 Coordinate



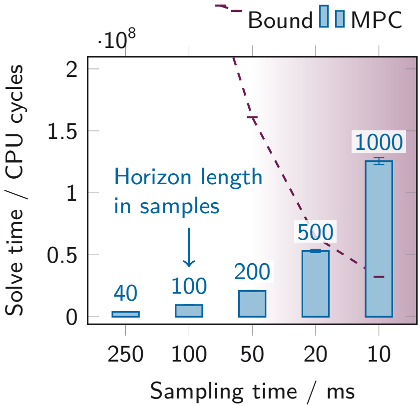
Trajectories

MPC handles all situation where PID failed, because it is "aware" of what the other actuators are doing.

Note

Performance does not come for free: it is computationally (a lot) more expensive, but worth it!

Deployment Plan



Plot: CVXPY with time horizon of 10 s.

Hardware

Modern hardware is very powerful. Decision factors are sampling time and prediction time horizon.

Computation

CPU cycles[a] needed to predict fixed amount of time into the future grows exponentially with the sampling frequency. Solve time is bounded by sampling time (need action before next sample comes).

Solver Software

There are countless options:

Commercial solutions

- Embotech AG, MOSEK ApS

Free solutions

- CVXgen, CVXPYgen, OSQP, OOQP, CVXOPT, ECOS

[a] Computation time normalized wrt CPU freq. Plot $f = 3.22$ GHz.

Backup Slides

If someone wants to know the details

(they are not officially part of the presentation)

What is Relaxed Linear MPC

Relaxed Linear MPC

Non-linear dynamics linearised at (z_s, u_s) to get LTI system (A, B), target landing pad is at z_f. In state z_n compute

$$\begin{aligned}
u^\star - u_s = \arg\min_{u_0} & \left\{ z_N^\top S z_N + \sum_{k=0}^{N-1} z_k^\top Q z_k + u_k^\top R u_k + V\|\epsilon_k\|_1 \right\} \\
\text{subject to} \quad & z_{k+1} = A z_k + B u_k && \text{(dynamics)} \\
& G_z z_k \leq g_z - G_z z_s + \epsilon_k && \text{(relaxed state constr.)} \\
& G_u u_k \leq g_u - G_u u_s && \text{(input constr.)} \\
& z_N = z_f - z_s && \text{(terminal constr.)} \\
& z_0 = z_n - z_s && \text{(parametrisation)}
\end{aligned}$$

Index n is real time, k is the prediction time. The ϵ_k are linearly penalized slack variables, and N is the "horizon length" for the prediction.

Model Uncertainty

The linearised model is very inaccurate in x and θ. To take into account make future states more expensive: $Q_k = \operatorname{diag}\left[q_0 + \varsigma_0 k/N \quad \dots \quad q_{n_x} + \varsigma_{n_x} k/N\right]$.