ELEC5470 Convex Optimization Project:
On Trajectory Generation of a Quadcopter UAV

LO, Li-yu
20997405
e-mail: lloac@connect.ust.hk

15/05/2023

1 Introduction

In this project, we deal with the problem of trajectory generation of a quadrotor UAV. In particular, we utilize the differential flatness property of a non-linear system, where a set of carefully selected outputs σ could be harnessed to determine all state and inputs in terms of σ and its derivates [1]. The selected σ, will then be expressed in a polynomial parameterized by t, i.e., the temporal parameter; the trajectory is then optimized through retrieving a set of optimal polynomials coefficients [2]. Thereupon, below details the problem formulation (Sec. 2), problem analysis and solution (Sec. 3), and results (Sec. 4).

2 Problem Formulation

Given a series of waypoints $P = \{p1, p2, ..., p_n\}$, we want to generate a set of discrete trajectory set points for the aerial robot controller. The states of the quadrotor could first be expressed as:

$$\mathcal{X} = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \omega_x, \omega_y, \omega_z]. \tag{1}$$

Meanwhile, with differential flatness, the above states of quadrotor could be simply derived by the flat outputs and their derivatives; the flat outputs are as follows:

$$\sigma = [x, y, z, \psi], \sigma \in \mathbb{R}^3 \times SO(2) \tag{2}$$

which are the Cartesian coordinates x, y, z, and the yaw angle ψ. The derivation between flat outputs and full state are omitted here, and we refer readers to the related literatures presented in [2, 3]. The trajectory is then planned with high-order polynomial with temporal parameter $\sigma(t)$, since the formulation could readily provide smoothness as well as easy determination on derivatives. The trajectory could be expressed as:

$$\sigma(t) = \begin{cases} \sigma(t)_1 = \sum_{i=0}^{n} p_{1,i} t^i & T_0 \leq t \leq T_1 \\ \sigma(t)_2 = \sum_{i=0}^{n} p_{2,i} t^i & T_1 \leq t \leq T_2 \\ \vdots & \\ \sigma(t)_m = \sum_{i=0}^{n} p_{m,i} t^i & T_{m-1} \leq t \leq T_m \end{cases} \tag{3}$$

Based on above (equation 3), the optimization problem is then formulated.

- Objective: to generate a smooth trajectory, and minimize the maneuver cost, the square of the snap, i.e., the 4^{th} derivative of the polynomial is used.

$$J = \int_{t_0}^{t} \left\| \frac{d^4 \sigma(t)}{dt^4} \right\|^2 dt. \tag{4}$$

  With some algebraic manipulation, the cost function could be re-written into matrix form:

$$\begin{aligned} & p^T Q p \; where, \\ & p_{il} = \\ & p_i * i(i-1)(i-2)(i-3) * \\ & l(l-1)(l-2)(l-3) * p_l * \\ & (i+l-7)^{-1} T^{i+l-7}. \end{aligned} \tag{5}$$

- Variable: the optimization variable is hence $p \in \mathbb{R}^{m*(n+1)}$, which are the coefficients of the polynomial.

- Constraints: as mentioned, with a series of selected waypoints, the trajectory is generated. Therefore, the trajectory is subject to constraints induced by the waypoints. In particular, the generated trajectory should comply with the starting and ending state in terms of position, velocity, acceleration, and jerk (3^{rd} derivative).

$$\begin{aligned} f_j^{(k)}(T_j) &= x_j^{(k)} \\ \Rightarrow \sum_{i \geq k} \frac{i!}{(i-k)!} T_j^{i-k} p_{j,i} &= x_{T,j}^{(k)} \\ \Rightarrow \begin{bmatrix} \cdots & \frac{i!}{(i-k)!} T_{j-1}^{i-k} & \cdots \\ \cdots & \frac{i!}{(i-k)!} T_j^{i-k} & \cdots \end{bmatrix} \begin{bmatrix} \vdots \\ p_{j,i} \\ \vdots \end{bmatrix} &= \begin{bmatrix} x_{0,j}^{(k)} \\ x_{T,j}^{(k)} \end{bmatrix} \end{aligned} \tag{6}$$

  Additionally, between each segment of trajectories, smoothness should be guaranteed, where it could be derived as,

$$
\begin{aligned}
f_j^{(k)}(T_j) &= f_{j+1}^{(k)}(T_j) \\
\Rightarrow \sum_{i \geq k} \frac{i!}{(i-k)!} T_j^{i-k} p_{j,i} - \sum_{l \geq k} \frac{l!}{(l-k)!} T_j^{l-k} p_{j+1,l} &= 0 \\
\Rightarrow \begin{bmatrix} \dots & \frac{i!}{(i-k)!} T_j^{i-k} & \dots\dots & -\frac{l!}{(l-k)!} T_j^{l-k} & \dots \end{bmatrix} &\begin{bmatrix} \vdots \\ p_{j,i} \\ \vdots \\ p_{j+1,l} \\ \vdots \end{bmatrix} = 0 \qquad (7)
\end{aligned}
$$

From equation 6 and 7, the constraints could be further summarized into matrix form as

$$
\begin{aligned}
A_{state} p &= b_{state}, \\
A_{continuity} p &= b_{continuity}. \qquad (8)
\end{aligned}
$$

The final optimization problem is

$$
\begin{aligned}
&\underset{p \in \mathbb{R}^{m*(n+1)}}{\text{minimize}} \; p^T Q p \\
&\textit{subject to} \\
&A_{state} p = b_{state} \\
&A_{continuity} p = b_{continuity}, \qquad (9)
\end{aligned}
$$

3 Problem Analysis and Solution

The formulated problem is clearly a convex problem, as

- The objective is a convex function. It is a function in quadratic form with its hessian $H(f_0) \succeq 0$, i.e., $Q \succeq 0$ in this case.
- The constraints are affine.

Above hence show the convexity of the formulated problem.

The above problem is a convex optimization with equality constraints, hence, we utilize Newton method to acquire the solution. Yet, meanwhile, as a feasible starting point could not be guaranteed, we use an infeasible start Newton method [4, p. 531], where the algorithm is first shown in algorithm 1. In particular, different from equality constrained Newton method, the solved linear system has a slight difference when constructing the linear matrices, which are as follows,

$$
\begin{bmatrix} H(f(x)) & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x_{pd} \\ \Delta \nu_{pd} \end{bmatrix} = - \begin{bmatrix} g(f(x)) + A^T \nu \\ Ax - b \end{bmatrix} \qquad (10)
$$

The Matlab script of the implementation is attached at the end of this report.

Algorithm 1 Infeasible Start Newton Method

Given starting point $x \in domf$, ν, tolerance $\epsilon > 0$, $\alpha \in (0, 1/2)$, $\beta \in (0, 1)$.
Repeat
  1. Compute primal and dual Newton steps $\Delta x_{\text{nt}}, \Delta \nu_{\text{nt}}$.
  2. Backtracking line search on $\|r\|_2$
    $t \leftarrow 1$
    while $\|r(x + t\Delta x_{\text{nt}}, \nu + t\Delta \nu_{\text{nt}})\|_2 > (1 - \alpha t)\|r(x, \nu)\|_2$
      $t := \beta t$
  3. Update, $x \leftarrow x + t\Delta x_{\text{nt}}, \nu \leftarrow \nu + t\Delta \nu_{\text{nt}}$

Until
  $Ax = b$ and $\|r(x, \nu)\|_2 \leq \epsilon$

4 Results

We first show one instance of the optimized trajectories in $2D$:

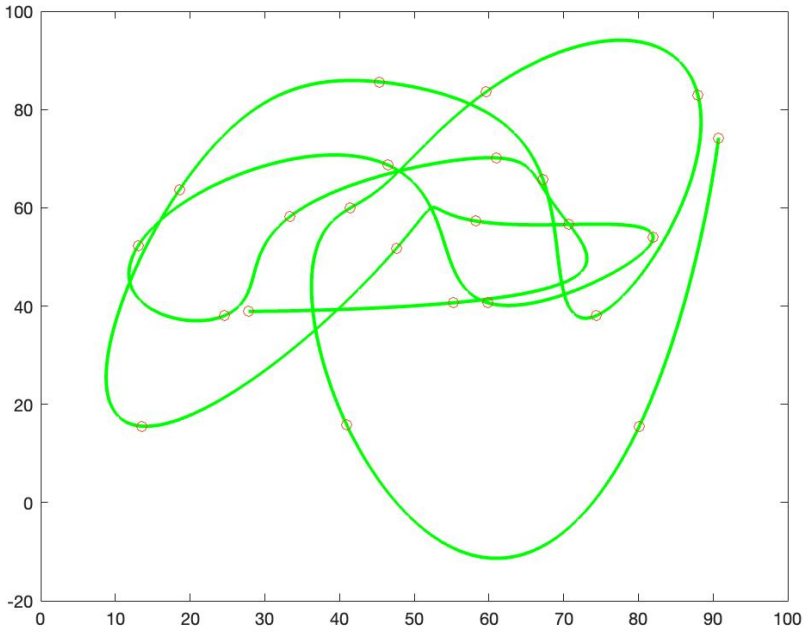

Figure 1: Case 1.

As shown, with randomly selected waypoints, the algorithm is able to generate a smooth trajectory, while obeying the constraints.

In particular, the above shown case randomly selected 23 waypoints were fed to the solver, and a total of 176 coefficients were optimized. In particular, Newton method converges extremely fast, during the majority of the numerical experiments, the stopping iterations are usually less than 4. Furthermore, for most of the time, the final quadratically convergent phase could not really be obviously observed due to the swift convergence. Below shows the converging results of the above presented case:

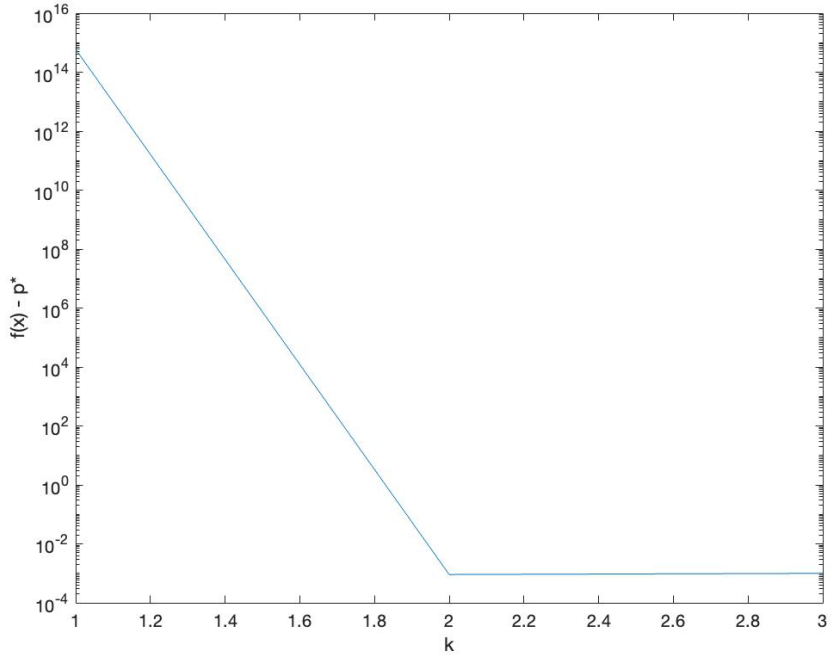

Figure 2: Converging Results, where X-axis is the iteration number and y being the difference between $x^{(k)}$ and $f(x)$.

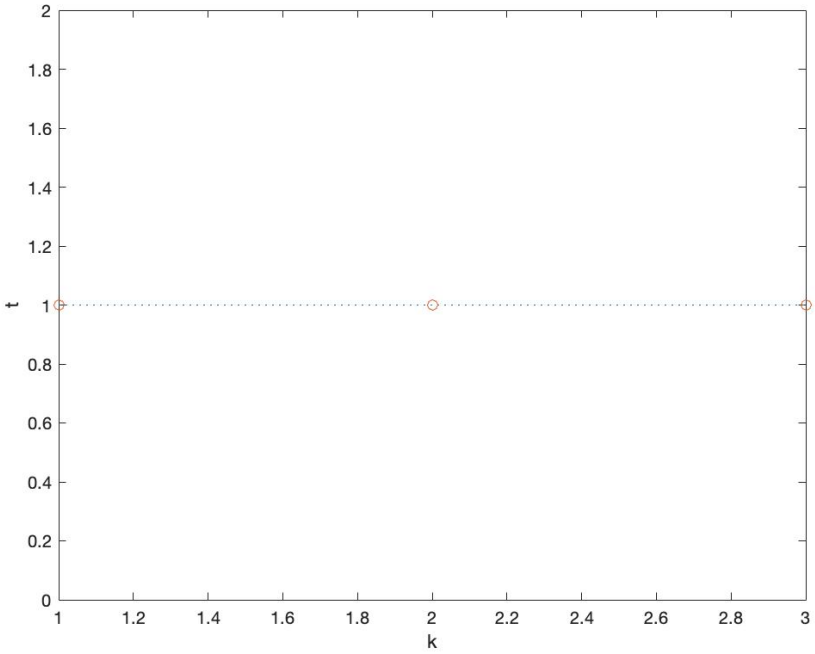

Figure 3: The step length at each iteration.

References

[1] M. J. Van Nieuwstadt and R. M. Murray, "Real-time trajectory generation for differentially flat systems," *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, vol. 8, no. 11, pp. 995–1020, 1998.

[2] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 2520–2525.

[3] G. Loianno, C. Brunner, G. McGrath, and V. Kumar, "Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and imu," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 404–411, 2016.

[4] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.