EN.530.678 Nonlinear Control and Planning in Robotics, Spring 2022

Planning and Control of a Quadcopter

Saicharan Balamurali (sbalamu1) and Yu-Heng Deng (ydeng33)

Johns Hopkins University

May 12, 2022

Abstract

Quadcopters have become a common sight today due to their diverse functionality and convenience when it comes to certain activities. They are generally used for surveillance and as personal cameras but are finding applications in other environments such as search and rescue, exploration and assistance and construction. Such use case scenarios require the quadcopter to maneuver around obstacles and find the optimal path to the target location while minimising power usage. Trajectories can be generated with optimal usage of fuel and travel time, but tracking the nonlinear dynamics of the robot along this trajectory is a challenge that can be solved by techniques such as feedback linearization or backstepping. In this report, we aim to discuss how we used dynamic feedback linearization to track our trajectory and also discuss why our initial attempts using block backstepping did not work.

1 Introduction

This project implements concepts in optimal trajectory generation using the Differential Dynamic Programming algorithm and tracking of the nonlinear dynamics of a quadcopter. We explored multiple different methods for trajectory tracking such as block backstepping, robust backstepping and feedback linearization. Eventually, we landed on using feedback linearization as it seemed to be the best suitable for the dynamics of the quadcopter. We used standard quadcopter dimension values of $I_x, I_y, I_z = 0.1, 0.1, 0.15 \ \frac{kg}{m^2}$ and mass $m = 0.2 \ kg$.

In this report, we discuss in more detail, the controller we used, the planning algorithm, the challenges we faced and how our work can be extended. We assumed no uncertainties were present in our model while simulating our UAV.

2 Materials & Methods

Our project consists of 2 stages - trajectory generation and trajectory tracking. We generated our trajectory using the Differential Dynamic Programming (DDP) algorithm. This is a optimal trajectory generation algorithm that takes in a cost function that is defined in terms of the controls and states of the model. The second stage is trajectory tracking using feedback linearization. This is method that separates coupled control terms to a linear way by differentiating the output multiple times .

3 Mathematical Model

The state vector is denoted by the variable s in our model shown below:

$$s = [x \ \ y \ \ z \ \ \phi \ \ \theta \ \ \psi \ \ \dot{x} \ \ \dot{y} \ \ \dot{z} \ \ p \ \ q \ \ r]^T$$

The state variables as shown in the vector above are respectively the x, y, z coordinates, the roll, pitch, yaw angles, and their corresponding variations with time. The variation of the state with respect to the existing state and the control inputs are depicted as per the equation shown below:

$$\dot{s} = f(x) + g(x)u$$

$$f(x) = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ p + q\sin(\phi)\tan(\theta) + r\cos(\phi)\tan(\theta) \\ q\cos(\phi) - r\sin(\phi) \\ q\frac{\sin(\phi)}{\cos(\theta)} + r\frac{\cos(\phi)}{\cos(\theta)} \\ 0 \\ 0 \\ g \\ \frac{(I_y - I_z)}{I_x}qr \\ \frac{(I_z - I_x)}{I_y}pr \\ \frac{(I_x - I_y)}{I_z}pq \end{bmatrix}$$

$$g(x) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ g_{71} & 0 & 0 & 0 \\ g_{81} & 0 & 0 & 0 \\ g_{91} & 0 & 0 & 0 \\ 0 & \frac{1}{I_x} & 0 & 0 \\ 0 & 0 & \frac{1}{I_y} & 0 \\ 0 & 0 & 0 & \frac{1}{I_z} \end{bmatrix}$$

Here the quadcopter parameters I_x, I_y, I_z represent its moment of inertia about the x, y and z axes respectively. The coefficients g_{71}, g_{81}, g_{91} are shown below:

$$g_{71} = -\frac{1}{m}(\sin(\phi)\sin(\psi) + \cos(\phi)\cos(\psi)\sin(\theta))$$
$$g_{81} = -\frac{1}{m}(\cos(\psi)\sin(\phi) - \cos(\phi)\sin(\psi)\sin(\theta))$$
$$g_{91} = -\frac{1}{m}(\cos(\phi)\cos(\theta))$$

The control inputs u in our model is a column vector of the vertical thrust, and the torques about the x, y and z axes.

$$u = [u_1 \quad u_2 \quad u_3 \quad u_4]$$

In our implementation, the quadcopter was set to finish its trajectory at the origin. The obtained linearized matrices A and B are shown below:

$$A = \text{zeros}(12, 12)$$
$$A_{7:12,7:12} = I_6$$
$$A_{7,5} = -49.05$$
$$A_{8,4} = -49$$

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -5 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 6.67 \end{bmatrix}$$

Feedback Linearization:

$$\xi = \begin{pmatrix} u_1 \\ \dot{u_1} \end{pmatrix}$$

$$v = \begin{pmatrix} x_1^{(4)} \\ x_2^{(4)} \end{pmatrix}$$

$$\frac{d}{dt}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \end{pmatrix} = \begin{pmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \frac{R(x_3)}{m} * \begin{pmatrix} 0 \\ u_1 \end{pmatrix} \\ \frac{u_2}{J} \end{pmatrix}$$

Feedback Linearization 3D:

For the 3D model of the quadcopter, we are also applying feedback lineariztion to design a controller to track the generated path. The output we selected are as follow:

$$y = \begin{bmatrix} h_1 & h_2 & h_3 & h_4 \end{bmatrix}^T = \begin{bmatrix} x & y & z & \psi \end{bmatrix}^T$$

beside the position, we also want to controller the track ψ, which is the heading angle of the quadcopter. The linearized input-output system can be found at the fourth derivative of y_1, y_2, and y_3 plus the second derivative of y_4. The derived system becomes

$$\begin{bmatrix} y_1^{(4)} \\ y_2^{(4)} \\ y_3^{(4)} \\ y_4^{(2)} \end{bmatrix} = F + GU$$

$$F = \begin{bmatrix} L_f^{(4)}h_1 \\ L_f^{(4)}h_2 \\ L_f^{(4)}h_3 \\ L_f^{(2)}h_4 \end{bmatrix} \quad U = \begin{bmatrix} u_1^{(2)} \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

$$G = \begin{bmatrix} L_{g1}L_f^{(4)}h_1 & L_{g2}L_f^{(4)}h_1 & L_{g3}L_f^{(4)}h_1 & L_{g4}L_f^{(4)}h_1 \\ L_{g1}L_f^{(4)}h_2 & L_{g2}L_f^{(4)}h_2 & L_{g3}L_f^{(4)}h_2 & L_{g4}L_f^{(4)}h_3 \\ L_{g1}L_f^{(4)}h_3 & L_{g2}L_f^{(4)}h_3 & L_{g3}L_f^{(4)}h_3 & L_{g4}L_f^{(4)}h_3 \\ L_{g1}L_f^{(2)}h_4 & L_{g2}L_f^{(2)}h_4 & L_{g3}L_f^{(2)}h_4 & L_{g4}L_f^{(2)}h_4 \end{bmatrix}$$

Since u_1 term is being derived to second order, we had to treat u_1 and $\dot{u_1}$ as augmented states as part of the dynamic compensator:

$$\xi = \begin{pmatrix} u_1 \\ \dot{u_1} \end{pmatrix}$$

In the end, we then assign the virtual controls so that the error dynamic can be asymptotically stabilized to the desired trajectory. The value of virtual control v is calculated with computed torque law:

$$v = \begin{pmatrix} x_1^{(4)} \\ x_2^{(4)} \\ x_3^{(4)} \\ x_4^{(2)} \end{pmatrix} = \begin{pmatrix} y_{d1}^{(4)} - \sum_{i=0}^{3} K_i(y_1^{(i)} - y_{d1}^{(i)}) \\ y_{d2}^{(4)} - \sum_{i=0}^{3} K_i(y_2^{(i)} - y_{d2}^{(i)}) \\ y_{d3}^{(4)} - \sum_{i=0}^{3} K_i(y_3^{(i)} - y_{d3}^{(i)}) \\ y_{d4}^{(2)} - \sum_{i=0}^{1} K_i(y_4^{(i)} - y_{d4}^{(i)}) \end{pmatrix}$$

Our error matrix can be defined in terms of the following equation:

$$\dot{z} = Az$$

$$z = \begin{pmatrix} y - y_d \\ \dot{y} - \dot{y_d} \\ \ddot{y} - \ddot{y_d} \\ y^{(3)} - y_d^{(3)} \end{pmatrix}$$

$$A = \begin{pmatrix} 0 & I_2 & 0 & 0 \\ 0 & 0 & I_2 & 0 \\ 0 & 0 & 0 & I_2 \\ -k_0 & -k_1 & -k_2 & -k_3 \end{pmatrix}$$

where A is Hurwitz if we choose the gains k_i, that are 2x2 matrices, correctly. In our implementation, it is very difficult to correctly choose the gains. Hence, to get the best possible guess, we developed a script that generates random values of k_i and printed out the eigenvalues of A to see if all of them had negative real parts. We picked one such value we found and decided to tune the gains further using that starting value.

4 Results

Our process first generated a trajectory by optimizing the cost of controls and a terminal cost term but did not have any running cost for the states. We did not want to penalize the system for travelling through intermediary states. This makes sense because we want our optimized trajectory to minimize fuel, behave according to obstacle constraints and reach the final position (i.e. the origin) using the best possible path. The next step was to perform feedback linearization. However, we had a discrete vector of our states which we converted into a continuous function of time to be fed into ode45. Further, we converted our 3D trajectory into a 2D trajectory by flattening along our y coordinate.

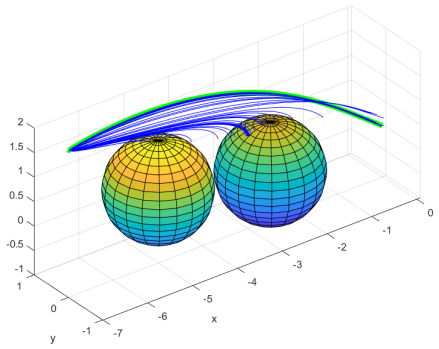

Figure 1: 3D Trajectory using DDP

For trajectory tracking, we used gain values k0, k1, k2 and k3 which are 2x2 diagonal matrices in our definition. The first trial was to check our feedback linearization controller's performance without any perturbation from the initial state. It was found that the trajectory was tracked well without any major problems. The flattened optimal trajectory and the trajectory tracked by feedback linearization are shown in the image below. Further, the x, y and yaw angle converging to 0 is shown in another image below.

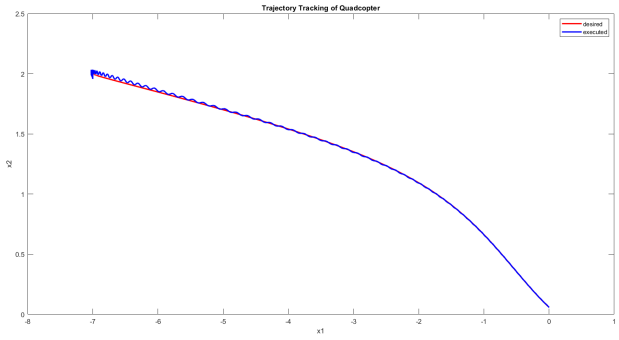

Figure 2: Trajectory tracking with no perturbation

The next trial was to try perturbing the initial state by a small amount of 0.5 m to see if the trajectory would be still be tracked as expected. The image below shows the desired trajectory and the trajectory tracked by our feedback linearization controller.

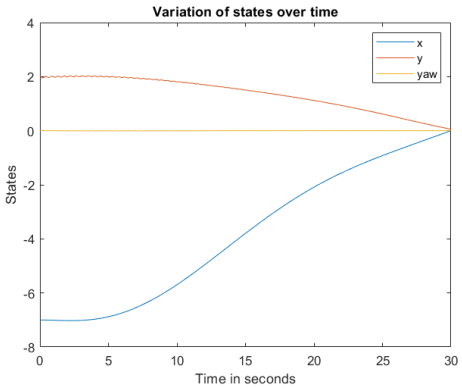

Figure 3: Variation of states over time

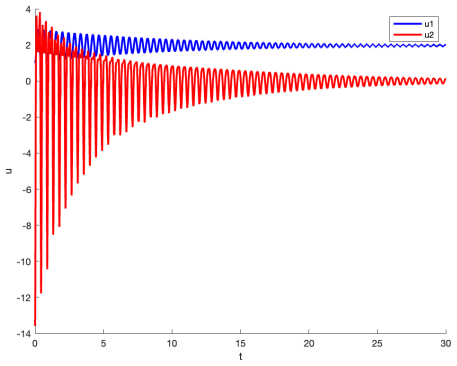

Figure 4: Variation of controls over time

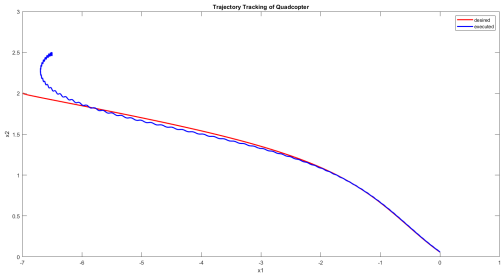

Figure 5: Perturbation of 0.5 m

As a final demonstration of the robustness of our controller, we increased the perturbation up to 2 meters to see how well it tracks the desired trajectory. The figure shown below shows the tracking of the desired trajectory.

5 Discussion

Our path planning algorithm is designed for optimal control and terminal cost. The obstacle constraints are handled in the DDP algorithm by augmenting the cost function with a scalar value to model the softness of the constraint and by modifying the accompanying gradients of the

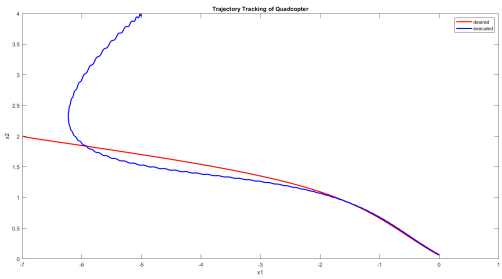

Figure 6: Perturbation of 2 m

cost function. We found that this works really well for trajectory generation and for obstacle avoidance. We experimented with multiple trajectories and eventually documented results for a trajectory with 2 obstacles. It is evident that the obstacles are avoided competently and an optimal trajectory is generated for our UAV.

Our feedback linearization seems to initially overshoot slightly above the trajectory and then eventually stabilizes to the trajectory. This slight issue can be resolved by tuning the gains further. The most significant part of our controller is the fact that it is highly robust to any initial state. Even when starting 2 meters away from the actual initial value, the trajectory is tracked by the controller.

For further details about our algorithm and to reproduce our results, please check out our github repository at this link.

In the 3D model controller, we found it really challenging to find a set of suitable control gains in limited time to make the error dynamic robust enough the track a trajectory with any perturbation. With more experiments in tuning the parameters, we should be able to find one that is able to track the trajectory generated with DDP.

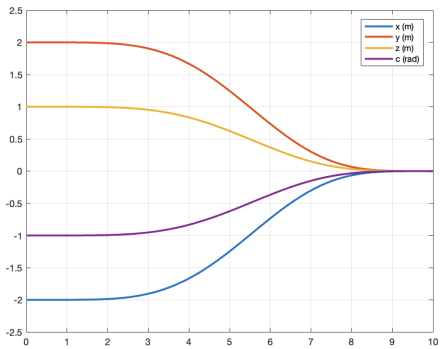

Figure 7: Trajectory tracking without perturbation

Beside feedback linearization controller, we also developed block backstepping controller for

the 3D model trajectory tracking. Although, we did not successfully find a working controller, we'd like to explain why it failed here. At first glance, our system seems to perfectly match the block backstepping format written in the lecture.

$$\dot{\eta} = f(\eta) + G(\eta)\xi$$
$$\dot{\xi} = f_a(\eta, \xi) + G_a(\eta, \xi)u$$

by modifying our dynamics we got

$$\eta = \begin{pmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \end{pmatrix} \quad f = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \xi = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ p \\ q \\ r \end{pmatrix}$$

$$f_a \begin{pmatrix} 0 \\ 0 \\ g \\ (I_x - I_z)qr/I_x \\ (I_z - I_x)pr/I_y \\ (I_x - I_y)pq/I_z \end{pmatrix} \quad u = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix}$$

$$G = \begin{pmatrix} I_{3*3} & & O_{3*3} & \\ & 1 & sin(\phi)tan(\theta) & cos(\phi)tan(\theta) \\ O_{3*3} & 0 & cos(\phi) & -sin(\phi) \\ & 0 & sin(\phi)/cos(\theta) & cos(\phi)/cos(\theta) \end{pmatrix}$$

$$Ga = \begin{pmatrix} g_{71} & & & \\ g_{81} & & O_{3*3} & \\ g_{91} & & & \\ 0 & 1/I_x & 0 & 0 \\ 0 & 0 & 1/I_y & 0 \\ 0 & 0 & 0 & 1/I_z \end{pmatrix}$$

It appears to aligns very well with the format. But there are two reasons why it can not produce a working controller. First, G_a needs to be a full rank square matrix, so that when we calculate the control inputs, we could obtain the correct G_a^{-1}. In our case, G_a is not a square matrix. If we use pseudo-inverse instead, at most only one of x, y, or z position will converge. Another problem with this formulation is that f and f_a must vanish when the system approaches to the origin. But in our case, the gravity term g in f_a is permanent. Therefore, block backstepping in our current application is not suitable for this dynamic model.

6 Future Developments

For the future developments of this project, we feel that there are a lot to do. First, we would like to find the correct gain for the 3D model tracking controller. We can use grid search approach to scan though all combinations of the gains and find one that has the best performance.

The next thing we like to do is to develop the controller that can handle uncertainty in the environment or the model error. It is especially usefully in our case, because the quadcopter definitely will experience wind in outdoor environment, and it is almost a requirement for a quadcopter to have the capability to deal with it.

The last one we like to keep exploring is to develop the RRT for our trajectory generation. Since DDP is coupled with the dynamic of the quadcoptor, is is quite hard to tune how it generates the trajectory. Another reason for switching to RRT is because it can handle more complicated obstacles. Even if the trajectory isn't that smooth, with a robust controller, it should be able to track it.

Acknowledgements

We would like to thank Dr. Marin Kobilarov for the opportunity and the resources to conduct this project. In addition, the teaching assistants for Nonlinear Control and Planning in Robotics, Peiyao Zhang and Brian Kim, have been really helpful throughout the duration of this course.

7 References

(1) Lysukho, G.V. & Maslennikov, Andrey. (2020). Quadcopter: dynamics and control. Politechnical student journal. 10.18698/2541-8009-2020-5-604.;

(2) H. Voos, "Nonlinear control of a quadrotor micro-UAV using feedback-linearization," 2009 IEEE International Conference on Mechatronics, 2009, pp. 1-6, doi: 10.1109/ICMECH.2009.4957154.

(3) Francesco Sabatino, Quadcopter Control: Modelling, Nonlinear Control and Simulation, Master's Thesis Project, June 2015, KTH, Stockholm, Sweden

(4) Z. Xie, C. K. Liu and K. Hauser, "Differential dynamic programming with nonlinear constraints," 2017 IEEE International Conference on Robotics and Au-

tomation (ICRA), 2017, pp. 695-702, doi: 10.1109/ICRA.2017.7989086.;

(5) M. Kobilarov, Nonlinear Control and Planning in Robotics (Course Id: EN.530.678), Autonomous Systems, Control and Optimization Lab, Johns Hopkins University.;