Obstacle Avoidance for A 2-D Quadrotor Using Iterative Linear Quadratic Regulator

Submitted By: Krishnesh Meratia (N12145709)

Km4943@nyu.edu

I. Abstract

In recent years quadrotors have gained popularity because of their potential to solve difficult tasks. Quadrotors are nonlinear and underactuated and developing controller and trajectory for them is complex. Iterative Linear Quadratic Regulator (ILQR) is a control approach which can handle non-linearity and simultaneously optimize for controller and trajectory. In this project obstacle avoidance for a 2D quadrotor using ILQR is presented.

II. Introduction

Quadrotor an unmanned aerial vehicle (UAV) have lately been into the limelight for solving many problems impossible and difficult for the human-being such as disaster recovery, inspection, manipulation and transportation. The availability of hardware and well-studied systems dynamics and control have accelerated research in the field.

Robotics problems often revolve around a combination of planning, tracking, and stabilization, most of the obsolete approaches dealt with solving these problems separately and manually making it time-consuming and laborious. ILQR is a control algorithm which limits these drawbacks by successfully solving for them simultaneously, optimizing the overall performance of the system and freeing the developer from extensive work.

III. Background

Quadrotors are complex systems which have non-linear dynamics, are underactuated, unstable and have to deal with noise. A general 2D quadrotor has three degrees of freedom (DOF), two degrees of actuation (DOA) and a state vector with six dimensions (position, orientation, linear velocity and angular velocity). For the motion control scenario presented, quadrotor has to avoid a square obstacle, it starts from a initial position and avoids the obstacle by crossing over it and goes land on the other side of it at a target location.

A non-linear optimal control problem can be solved using two approaches: a direct approach and an indirect. Indirect one solves for optimality in terms of the maximum principle, the adjoint equations, and the boundary conditions. Direct approach replaces the ordinary differential equation with algebraic constraints using a large set of unknowns. These methods are powerful for solving trajectory optimization problems and problems with inequality constraints but are slow [1].

IV. Process

ILQR an indirect approach is an algorithm which can solve many different control tasks by just knowing the nonlinear system dynamics and the associated cost function. It deals with the problem in discrete time and solves it using a sequence of linear quadratic subproblems with each iteration there is a decrease in the cost and we can find a sequence of controls that satisfies the necessary conditions.

Quadrotors being continuous in time requires discretization which is done using one step Euler integration. As the algorithm requires linear dynamics and quadratic cost it is done by taking one and two step Taylor series expansion respectively along an initial state and control trajectory and hence finding a direction along which we minimize the value function optimizing over both the state and control trajectories.

Derivation of iLQR adopted

$$min\ u[.] \quad l_f(x[N]) + \sum_{n=0}^{N-1} l(x[n], u[n]) \qquad (1)$$

$$subject\ to$$

$$x[n+1] = f(x[n], u[n]) \quad \forall\ n\ \varepsilon[0, N-1] \qquad (2)$$

$$x[0] = x_0 \qquad (3)$$

$x_0 = [5,0,0,0,0,0]$ for the problem

Discrete Dynamics

Equation of motion for quadrotor

$$m\ddot{x} = -(u_1 + u_2)\, sin(\theta) \quad (4)$$

$$m\ddot{y} = (u_1 + u_2)\, cos(\theta) - mg \quad (5)$$

$$I\ddot{\theta} = r\,(u_1 - u_2) \quad (6)$$

In state space form (continous dynamics)

$$State = \begin{bmatrix} x \\ y \\ \theta \\ \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} position\ in\ x\ axis \\ position\ in\ y\ axis \\ angle\ wrt\ world\ frame \\ velocity\ in\ x\ axis \\ velocity\ in\ y\ axis \\ angular\ velocity \end{bmatrix} \quad (7)$$

$$\begin{bmatrix} x \\ y \\ \theta \\ \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ 0 \\ -g \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{-sin(\theta)}{m} & 0 \\ \frac{cos(\theta)}{m} & 0 \\ 0 & \frac{1}{I} \end{bmatrix} = f(x, u) \quad (8)$$

For the problem I =1 m=1 and r =1

Discretization using one step Euler integration

$$x_{t+1} = x_t + dt * f(x_t, u_t) \quad (9)$$

Which we approximate with a first-order Taylor-series expansion about nominal trajectories X = $(x_0, \ldots\ldots x_N)$, U = $(u_0, \ldots\ldots u_{N-1})$

$$x_{k+1} + \delta x_{k+1} = f(x_k + \delta x_k, u_k + \delta u_k) \approx f(x_k, u_k) + \frac{\partial f}{\partial x}_{|\, x_t, u_t} (x - x_k) + \frac{\partial f}{\partial u}_{|\, x_t, u_t} (u - u_k) \quad (10)$$

$$A = \frac{\partial f}{\partial x}_{|\, x_k, u_k} \qquad B = \frac{\partial f}{\partial u}_{|\, x_k, u_k} \quad (11)$$

$$\delta x_{t+1} = A(x_k, u_k)\delta x_k + B(x_k, u_k)\delta u_k \quad (12)$$

Cost Function

Stage cost

$$l = (x_k - x_k')^T Q\,(x_k - x_k') + u^T R\, u + q\, e^{-\text{distance from obstacle}} \quad (13)$$

for distance from obstacle distance from the center of the square is taken

distance from obstacle

$$= (sqrt((x - 4.5) ** 2 + (y - 4.5) ** 2) \quad (14)$$

As the center of square is at (4.5,4.5)

$$Q = diag([1,1,1,1,1,1])$$

$$R = diag\ ([0.1,0.1])$$

$$q = 1000$$

Terminal cost

$$l_f = (x_N - x_N')^T Q\,(x_N - x_N') \quad (15)$$

$$Q = diag\ ([1,1,1,1,1,1]) * 50000$$

As the cost function is nonlinear we need to take, a second-order Taylor Series Expansion to make it quadratic [2].

$$J(x_0, U) = l_f(x[N]) + \sum_{n=0}^{N-1} l(x[n], u[n]) \quad (16)$$

$$\approx \frac{1}{2} x_N{}^T Q_N x_N + q_N^T x_N + \sum_{t=1}^{N-1} x_k^T Q_k x_k + \frac{1}{2} u_k^T R_k u_k + \frac{1}{2} x_k^T H_k u_k + \frac{1}{2} u_k^T H_k^T u_k + q_k^T x_k + r_k^T u_k \quad (17)$$

$$l_x \equiv \frac{\partial l}{\partial x}_{|\, x_k, u_k} \rightarrow Q_k x_k + q_k \quad (18)$$

$$l_x \equiv \frac{\partial l}{\partial u}_{|\, x_k, u_k} \rightarrow R_k u_k + r_k \quad (19)$$

$$l_{xx} \equiv \frac{\partial^2 l}{\partial x^2}_{|\, x_k, u_k} \rightarrow Q_k \quad (20)$$

$$l_{uu} \equiv \frac{\partial^2 l}{\partial u^2}_{|\, x_k, u_k} \rightarrow R_k \quad (21)$$

$$l_{xu} \equiv \frac{\partial^2 l}{\partial x \partial u}_{|\, x_k, u_k} \rightarrow H_k \quad (22)$$

$$l_{ux} \equiv \frac{\partial^2 l}{\partial u \partial x}_{|\, x_k, u_k} \rightarrow H_k^T \quad (23)$$

Using Bellman's Principle of Optimality to define the optimal cost-to-go $V_k(x)$ by the recurrence relation:

$$V_N = l_f(X_N) \quad (24)$$

$$V_k = \min u\ \{l(x_k, u_k) + V_{k+1}(f(x_k, u_k))\} \quad (25)$$

$$V_k = \min u\ Q_k(x_k, u_k) \quad (26)$$

Approximating the cost to go function locally as locally quadratic near the nominal trajectory:

$$V_k + \delta V_k = V_k(x_k + \delta x_k) \approx V(x_k) + \frac{\partial V}{\partial x|x_k}(x\text{-}x_k) + \frac{1}{2}(x\text{-}x_k)^T \frac{\partial^2 V}{\partial x^2|_{x_k,}}(x\text{-}x_k) \quad (27)$$

$$\delta V_k = s_k^T x_k + \frac{1}{2} x_k^T S_k x_k \quad (28)$$

Similarly:

$$Q_k + \delta Q_k = Q(x_k + \delta x_{k,} u_k + \delta u_k)$$

$$\approx Q(x_k, u_k) + \frac{\partial Q}{\partial x|_{x_k,u_k}}(x\text{-}x_k) + \frac{\partial Q}{\partial u|_{x_k,u_k}}(u\text{-}u_k) + \frac{1}{2}(x\text{-}x_k)^T \frac{\partial^2 Q}{\partial x^2|_{x_k,u_k}}(x\text{-}x_k) + \frac{1}{2}(u\text{-}u_k)^T \frac{\partial^2 Q}{\partial u^2|_{x_k u_k}}(u\text{-}u_k) + \frac{1}{2}(u\text{-}u_k)^T \frac{\partial^2 Q}{\partial u \partial x|_{x_k,u_k}}(x\text{-}x_k) + \frac{1}{2}(x\text{-}x_k)^T \frac{\partial^2 Q}{\partial x \partial u|_{x_k,u_k}}(u\text{-}u_k) \quad (29)$$

$$\delta Q_k(x_k, u_k) = \frac{1}{2}\begin{bmatrix}\delta x_k \\ \delta u_k\end{bmatrix}^T \begin{bmatrix}Q_{xx} & Q_{xu} \\ Q_{ux} & Q_{xx}\end{bmatrix}\begin{bmatrix}\delta x_k \\ \delta u_k\end{bmatrix} + \begin{bmatrix}Q_x \\ Q_u\end{bmatrix}^T \begin{bmatrix}\delta x_k \\ \delta u_k\end{bmatrix} \quad (30)$$

Defining the following variables using matrix calculus:

$$Q_x = l_x + s_{k+1}A_k \quad (31)$$

$$Q_u = l_u + s_{k+1}B_k \quad (32)$$

$$Q_{xx} = l_{xx} + A_k^T S_{k+1} A_k \quad (33)$$

$$Q_{uu} = l_{uu} + B_k^T S_{k+1} B_k \quad (34)$$

$$Q_{ux} = l_{ux} + B_k^T S_{k+1} A_k \quad (35)$$

Cost to go at terminal state

$$s_N \equiv \frac{\partial V}{\partial x|_{x_N}} = \frac{\partial}{\partial x}(x\text{-}x_f)^T Q_f(x\text{-}x_f)|\, x_N = Q_f(x_N\text{-}x_f) \quad (36)$$

$$S_N \equiv \frac{\partial^2 V}{\partial x^2|_{x_N}} = \frac{\partial}{\partial x} Q_f(x\text{-}x_f)|\, x_N = Q_f \quad (37)$$

Backward Pass

After solving the terminal problem we apply the principle of optimality and define the process for solving the k time step given the values at the k+1 time step.

$$V_k = \min u \{l(x_k, u_k) + V_{k+1}(f(x_k, u_k))\} \quad (38)$$

$$V_k = \min u\, Q_k(x_k, u_k) \quad (39)$$

$$\delta V_k = \min \delta u \quad \{\delta Q(x, u)\} \quad (40)$$

$$= \min \delta u \{Q_x \delta x + Q_u \delta u + \frac{1}{2}\delta x^T Q_{xx}\delta x + \frac{1}{2}\delta u^T Q_{uu}\delta u + \frac{1}{2}\delta x^T Q_{xu}\delta u + \frac{1}{2}\delta u^T Q_{ux}\delta x$$

$$\frac{\partial \delta Q}{\partial \delta u} = Q_u + \frac{1}{2}Q_{ux}\delta x + \frac{1}{2}Q_{xu}^T \delta x + Q_{uu}\delta u = 0 \quad (41)$$

$$\delta u^* = -Q_{uu}^{-1}(Q_{ux}\delta x_k + Q_u) \quad (42)$$

$$= K\delta x_k + d$$

$$d_k = -Q_{uu}^{-1} Q_u \quad (43)$$

$$K_k = -Q_{uu}^{-1} Q_{ux} \quad (44)$$

After calculating the optimal control as a function of the next time step we can plug it back into Equation

$$\delta Q_k(x_k, u_k) = \frac{1}{2}\begin{bmatrix}\delta x_k \\ K\delta x_k + d\end{bmatrix}^T \begin{bmatrix}Q_{xx} & Q_{xu} \\ Q_{ux} & Q_{xx}\end{bmatrix}\begin{bmatrix}\delta x_k \\ K\delta x_k + d\end{bmatrix} + \begin{bmatrix}Q_x \\ Q_u\end{bmatrix}^T \begin{bmatrix}\delta x_k \\ K\delta x_k + d\end{bmatrix} \quad (45)$$

$$\Delta V = \frac{1}{2}d^T Q_{uu} d + d^T Q_u \quad (46)$$

$$s = Q_x + K^T Q_{uu} d + K^T Q_u + Q_{ux}^T d \quad (47)$$

$$S = Q_{xx} + K^T Q_{uu} K + K^T Q_{ux} + Q_{ux}^T K \quad (48)$$

Forward Pass

$$\delta x = x_k\text{-}\bar{x}_k \quad (49)$$

$$u_k = \bar{u}_k + \delta u_k^* = \bar{u}_k + K_k \delta x_k + \alpha d_k \quad (50)$$

$$x_{k+1} = f(x_k, u_k) \quad (51)$$

Where α is the step size used to perform a simple line search.

If the cost decrease we keep α constant otherwise we decrease it.

Steps:

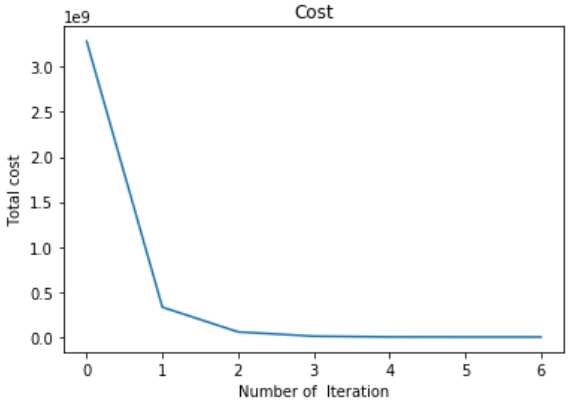
1) We start with initial condition and nominal control trajectory.
2) Compute state trajectory using discrete dynamics.
3) Get s and S for terminal state
4) Calculate gains using all the partial derivatives and applying backward pass.
5) Compute the new control trajectory using forward pass.
6) Check for decrease in cost if decreased keep alpha constant otherwise decrease alpha,

7) If cost < threshold jump out of the loop otherwise go to 1 for a max number of iterations.

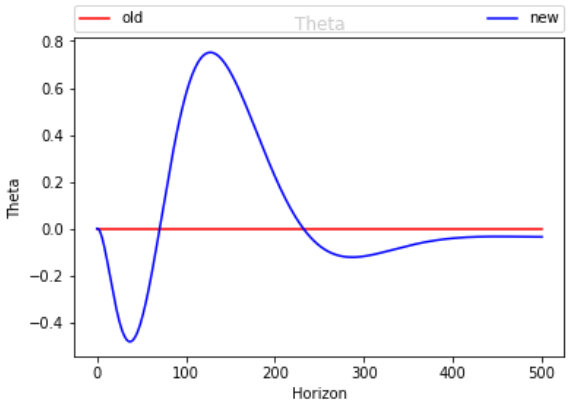
Cost function being the gist of the problem played the most important role and is a function of all the states and controls that we want to optimize over. There is a terminal cost and a cost to go as the name suggests the terminal cost is associated with the terminal position and the cost to go is the cost incurred as we transition from one state to another. The task at hand required constraining state to avoid the obstacle which was done by penalizing the distance from the obstacle.

V. Results

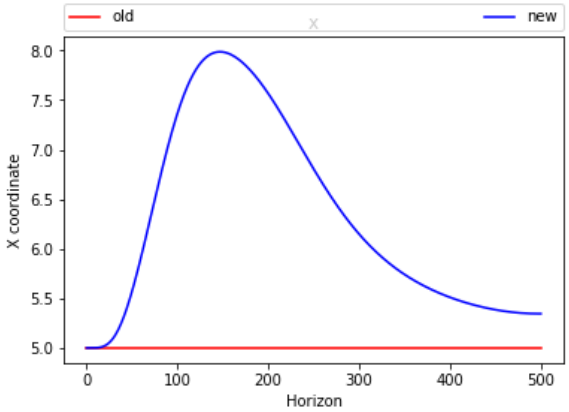
The quadrotor was able to avoid the obstacle and reach the target location assigned to it. The algorithm was able to find the optimal solution in 6 iterations

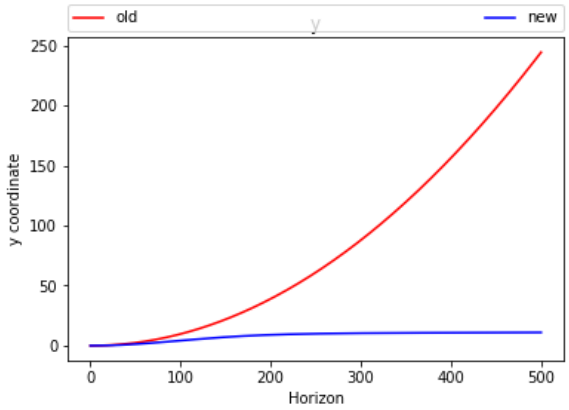


We can see the reduction in cost with each iteration.



The angle was constrained to stop the quadrotor from spinning.





VI. Conclusion

Iterative control algorithms can be applied to systems where we want to optimize control and state trajectories. As both feedback and feedforward controllers are found it reduces the burden to solve for these separately. It can handle both the input constraint and process noise which makes it feasible for most of the real-world tasks with just the requirement of knowing underlying system dynamics and the cost function associated. It's computationally efficient and can find solutions within a few iterations.

The algorithm sacrifices guarantee of global optimality; they only guarantee local optimality and can be subject to local minimum. They cannot handle hard constraints and one might have to switch to direct methods which are time consuming. The need to know the dynamics and cost function makes the problem little complex where one can switch to reinforcement learning methods. There have been many variations of the algorithm to solve the problem at hand.

RFERENCES

[1] A. Sideris and J. Bobrow, *"An efficient sequential linear quadratic algorithm for solving nonlinear optimal control problems," Automatic Control, IEEE Transactions on, vol. 50, no. 12, pp. 2043–2047, 2005.*

[2] Brian Jackson, Taylor Howell *"iLQR Tutorial Robotic Exploration Lab, Stanford University"*