**Equations for Quadrotor dynamics**

The angular velocity is a vector pointing along the axis of rotation, while ˙θ is just the time derivative of yaw, pitch, and roll. In order to convert these angular velocities into the angular velocity vector, we can use the following relation:

where ω is the angular velocity vector in the body frame.

**Rotation matrix (body to inertial frame):**

Text, letter

Description automatically generated

**Equation for drag:**

**Total thrust on the quadcopter (in the body frame)**

Where is angular velocity of motor and k is a constant.

**Torques in the body frame:**

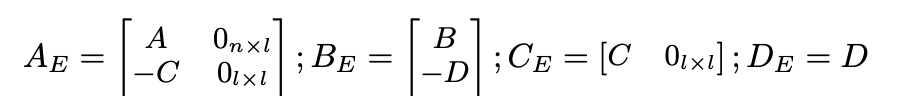
**Equation of linear motion:**

where x is the position of the quadcopter, g is the acceleration due to gravity, is the drag force, and is the thrust vector in the body frame.

**Body frame rotational equations of motion:**

Linearizing equations around hover state:

; ; ; ; .



Text

Description automatically generated with medium confidence

1. Check equations of motion. **Done**
2. Solve for PID gains
3. Include PID in LQR
4. Apply Bryson. (Prepare document stating why just x, y, z should be altered)
5. Get GPU access
6. RL DDPG to change Q matrix
   1. What should be included in state
   2. Reward should guide the agent at every step (straight line dist from target) as well as minimize the overall error (something like ISE)
   3. Apply z score norm to state like the paper
      1. Use z score norm value for rewards too?
      2. **Batch normalization is motioned in the OG paper**
   4. Intelligently choose max allowed limit for q values (gives singular matrix if all 4 Q values are allowed to be high but Bryson’s rule choses good values). Maybe choose these mathematically such that matrix is not singular
   5. Maybe initialize actions to Bryson and then train!!
7. **Limit state velocities or input U! This should be included in all controllers! This will generate a region of operation wrt Q values. Must analyse this.**
   1. **Penalty due to limits on U values (may give suboptimal results due to saturated optimal feedback)**
   2. **Penalty due to singularity of Q matrix. (small eigenvalues)**
8. HRL DDPG
9. Using random linear and angular velocities at the beginning of simulation can result in better generalization. (These states will be less visited by RL agent). Or increase the complexity of initial and final conditions as the agent learns to do well in simple initial conditions.
10. Supplementary function that will ensure all angles are between [-2\*np.pi,2\*np.pi]
11. Initialise weights in NN using glorot or some other distribution for faster convergence.

Finally compare Bryson’s rule and RL with random initial conditions. Also compare with random Q values.

Plot of successful\_tracking vs failed\_tracking.

Transform 12 state to a lower dimension like 3 and visualize the 4 Q values in separate plots. This may give insights about the kind of Q values the network is applying.

Things to think about:

Do we need HRL?

Why did we choose actor critic model? Why not other models? (DO THIS BEFORE INTERVIEWS)

As another benchmark, could include MPC control with constraints on U. Plus MPC control with constraints on U and choosing Q matrix with RL.

Does RL method help deal with error in system dynamics or in presence of noise?

**Negative reward if task not completed till end of episode**

Chart

Description automatically generated

1e-3 without z normalization