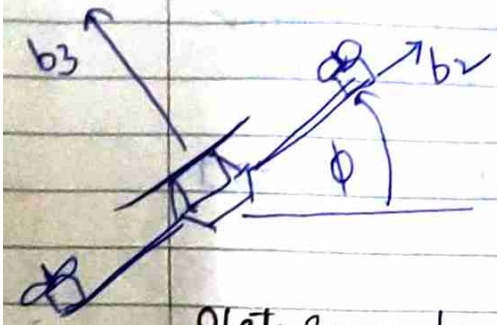
Planning & Control

2-D Quadrotor control

Equation of motion



$$\begin{bmatrix} \ddot{y} \\ \ddot{z} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 \\ -g \\ 0 \end{bmatrix} + \begin{bmatrix} -\frac{1}{m}\sin\phi & 0 \\ \frac{1}{m}\cos\phi & 0 \\ 0 & \frac{1}{I_{xx}} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

State space description of the quadrotor

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} y \\ z \\ \phi \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \end{bmatrix} \qquad \dot{x} = \begin{bmatrix} \dot{y} \\ \dot{z} \\ \dot{\phi} \\ 0 \\ -g \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -\frac{1}{m}\sin\phi & 0 \\ \frac{1}{m}\cos\phi & 0 \\ 0 & \frac{1}{I_{xx}} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

velocities

acc^{n}

input that can drive our dynamical system. By specifying appropriate u_1 & u_2, we can change the state of quadrotor & get it to go where we want it to go.

Equations of motion

$$\ddot{y} = -\frac{u_1}{m}\sin\phi$$

$$\ddot{z} = -g + \frac{u_1}{m}\cos\phi$$

$$\ddot{\phi} = \frac{u_2}{I_{xx}}$$

Dynamics are non linear. Non-linearity here comes from the fact that we have $\sin\phi$ & $\cos\phi$

Let's consider these eqns of motion at a hover configuration. We define a hover configuration at $y=0, z=0, \phi=0$

$u_1 = mg$ → the hover conf. can be at equilibrium only if the thrust vector, u_1 opposes the gravity vector.

$u_2 = 0$

So, at hover, the nominal value of u_1 must equal mg. Further, nominal value of u_2 must be equal to 0. If it were not, then the vehicle would try to accelerate.

Linearized Dynamic Model (linearization of the nonlinear model)

$\sin\phi$ & $\cos\phi$ are the two principle sources of nonlinearity. How do these nonlinear functions behave in close proximity to the hover configuration?

when ϕ is close to 0 $\rightarrow \sin\phi \approx \phi$, $\cos\phi \approx 1$

$$\ddot{y} = -g\phi$$

$$\ddot{z} = -g + \frac{u_1}{m}$$

$$\ddot{\phi} = \frac{u_2}{I_{xx}}$$

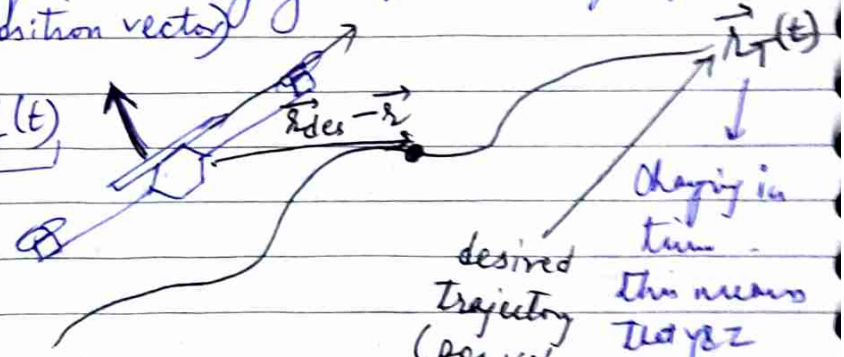
Trajectory Tracking

Wow, Let's look at the control problem.

Imagine a plane or quadrotor trying to follow a specified trajectory r_T (a position vector)

Given $\vec{r}_T(t)$, $\dot{\vec{r}}_T(t)$, $\ddot{\vec{r}}_T(t)$

Let's assume that this function is differentiable (to calculate these)



Changing in time. This means that y & z are both changing as a function of t.

$$\vec{r}_T(t) = \begin{bmatrix} y(t) \\ z(t) \end{bmatrix}$$

We are interested in $(\vec{r}_{des} - \vec{r})$ and reducing it to 0.

Here, $\vec{r}_{des}$ being the point on the traj. we want to follow and $\vec{r}$ being the current point.

error vectors :- $\vec{e}_p = \vec{r}_T(t) - \vec{r}$

$\vec{e}_v = \dot{\vec{r}}_T(t) - \dot{\vec{r}}$

commanded acc^n, calculated by the controller

We want the error to decay exponentially to 0.

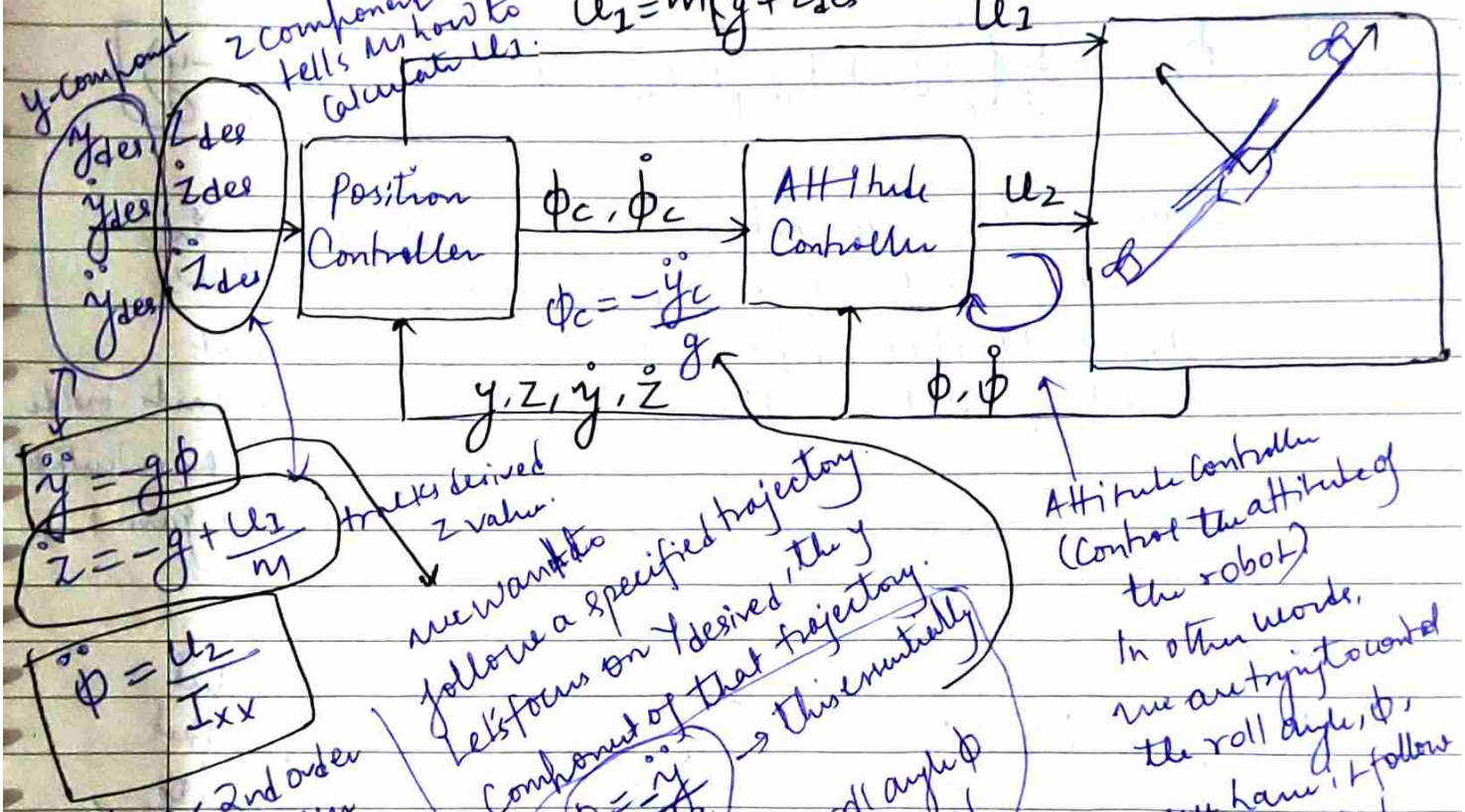
i.e., $(\ddot{\vec{r}}_T(t) - \ddot{\vec{r}}_c) + K_d \vec{e}_v + K_p \vec{e}_p = 0$

(as we have seen before, if we take the error term & make it to obey a 2nd ODE with +ve coeffs, we guarantee that error decays exponentially to 0)

So, we command an acceleration which forces the error terms to obey the 2nd order ODE.

This idea can be extended to develop a control structure that's hierarchical.

Nested Control Structure



tracks desired z value.

Linear 2nd order differential eqn in one variable → It's not hard to determine u_2 so that ϕ converges to ϕ_c

we want to follow a specified trajectory. Let's focus on $y_{desired}$, the y component of that trajectory.

If $\phi = -\frac{\ddot{y}}{g}$ → this eventually ensures that roll angle ϕ will track the desired accn in y direction

Attitude controller (Control the attitude of the robot)

In other words, we are trying to control the roll angle, ϕ, we have it follow a specified trajectory.

→ what that means is we wanna synthesize u_2 such that ϕ goes to some desired value. That desired value is known as the reference input ϕ_c (assume ϕ_c is different)

So, in summary, the inner loop is designed so that ϕ converges to ϕ_c and ϕ_c is obtained from the y component of desired trajectory.

$$u_2 = I_{xx}\left[\ddot{\phi}_c + K_{p,\phi}(\phi_c - \phi) + K_{d,\phi}(\dot{\phi}_c - \dot{\phi})\right]$$

~~finally, z component tells us how to calculate~~

#learnthesmarterway

Control Equations

Date __/__/____

$$u_1 = m\left[g + \ddot{z}_{des} + K_{d,z}(\dot{z}_{des} - \dot{z}) + K_{p,z}(z_{des} - z)\right]$$

$$u_2 = K_{p,\phi}(\phi_c - \phi) + K_{d,\phi}(\dot{\phi}_c - \dot{\phi})$$

$$\phi_c = -\frac{1}{g}\left[\ddot{y}_{des} + K_{d,y}(\dot{y}_{des} - \dot{y}) + K_{p,y}(y_{des} - y)\right]$$

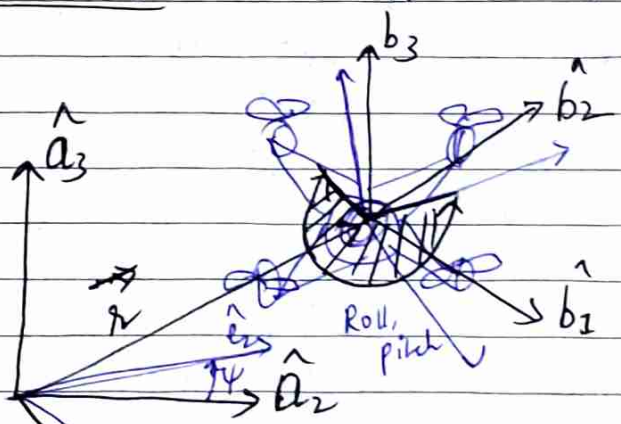
Above are the three control equation we can used to drive a quadrotor, assuming our assumption that it's close to the operating point is valid & assuming that it stays on the y-z plane.

parameters/constants inside circle needs to be calculated these are gain.

3-D Quadrotor

$$\vec{r}, \quad q = \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \end{bmatrix}, \quad \chi = \begin{bmatrix} q \\ \dot{q} \end{bmatrix}$$

ϕ pitch, θ Roll, ψ yaw



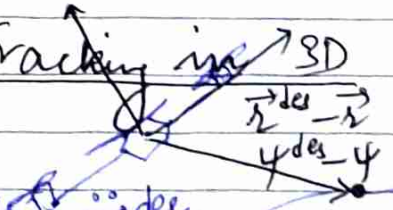
we attached a body fixed frame to the quadrotor. Also we have an inertial frame

Angular velocity component in B

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} C\theta & 0 & -C\phi S\theta \\ 0 & 1 & S\phi \\ S\theta & 0 & C\phi C\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

The State of the quadrotor consists of position & orientation. The position being the position vector of the Centre of mass.

Trajectory Tracking in 3D



given $\vec{r}_T(t)$

$\dot{\vec{r}}_T(t)$

$\ddot{\vec{r}}_T(t)$

$\vec{r}^{des}(t), \dot{\vec{r}}^{des}(t), \ddot{\vec{r}}^{des}(t)$

$\psi^{des}(t), \dot{\psi}^{des}(t), \ddot{\psi}^{des}(t)$

we assume that we are given a traj. but this traj. now consists of a pos. vector as a func of time & the yaw angle which varies as a function of time

$\vec{r}_T(t)$ desired traj. (pos, yaw)

$$\vec{r}_T(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \\ \psi(t) \end{bmatrix}$$

we are interested in the $(\vec{r}_{des} - \vec{r})$ (desired position − actual position). But, now we also interested in the $(\psi^{des} - \psi)$.

Date: __/__/____

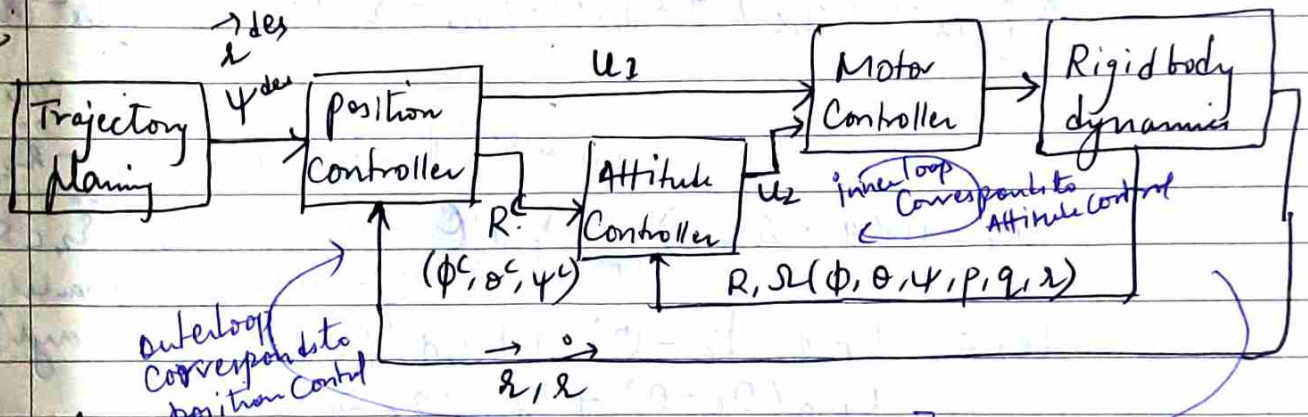
This gives us the error vector & its derivative. Again, we want the error vector to go exponentially to 0. This will allow us to calculate commanded acc^n ($\ddot{\vec{r}}$ or $\ddot{\psi}$)

$$\vec{e}_p = \vec{r}_T(t) - \vec{r}$$
$$\vec{e}_v = \dot{\vec{r}}_T(t) - \dot{\vec{r}}$$

Want
$$(\ddot{\vec{r}}_T(t) - \ddot{\vec{r}}_c) + K_d \vec{e}_v + K_p \vec{e}_p = 0$$

$\ddot{\vec{r}}_c$: Commanded acc^n, calculated by the Controller

Nested feedback Loop:



Control for hovering:

All velocities and accelerations are 0.

$$m\ddot{\vec{r}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R \begin{bmatrix} 0 \\ 0 \\ F_1 + F_2 + F_3 + F_4 \end{bmatrix}$$

Also, roll & pitch angles are 0.

$$I \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} L(F_2 - F_4) \\ L(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times I \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

u_2 (Calculate this based on desired attitude)

→ In the inner loop, we will specify the orientation either through a rotation matrix or a series of roll, pitch and yaw angles. We will feedback the actual attitude & the angular velocity, or the roll, pitch & yaw angles & the angular rates. From that we calculate u_2 (is a function of the thrust & the moments that we get from the motor speeds).

#learnthesmarterway

Outerloop (Position Feedback Loop)

Date: __/__/____

→ In the position feedback loop, we look at the specified position vector & the specified yaw angle. We look at the ~~specified~~ actual position and the actual velocity & from this we calculate U_1 (U_1 is essentially the sum of all the thrust forces).

So, in hovering, the position & orientation is fixed. In other words, all velocities are zero. Further, the roll & pitch angles are equal to 0.

$$(\ddot{r}_{i,des} - \ddot{r}_{i,c}) + K_{d,i}(\dot{r}_{i,des} - \dot{r}_i) + K_{p,i}(r_{i,des} - r_i) = 0$$

($\ddot{r}_{i,c}$: commanded; $\dot{r}_{i,des}$: commanded; $\dot{r}_i$: actual (feedback); $r_{i,des}$: specified)

$$U_1 = m(g + \ddot{r}_{3,c})$$

Commanded roll / pitch:

$$\phi_c = \frac{1}{g}(\ddot{r}_{1,c}\sin\psi_{des} - \ddot{r}_{2,c}\cos\psi_{des})$$

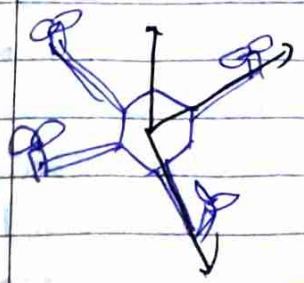
$$\theta_c = \frac{1}{g}(\ddot{r}_{1,c}\cos\psi_{des} + \ddot{r}_{2,c}\sin\psi_{des})$$

$\psi_c = \psi_{des}$ ← commanded yaw angle

$$U_2 = \begin{bmatrix} K_{p,\phi}(\phi_c - \phi) + K_{d,\phi}(p_c - p) \\ K_{p,\theta}(\theta_c - \theta) + K_{d,\theta}(q_c - q) \\ K_{p,\psi}(\psi_c - \psi) + K_{d,\psi}(r_c - r) \end{bmatrix}$$

we want to look at small perturbations around the hover position. Accordingly, we will linearize the dynamics around the current configuration.

Control for Hover



Linearize the dynamics at the hover configuration

$$m\ddot{\vec{r}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R\underbrace{\begin{bmatrix} 0 \\ 0 \\ F_1 + F_2 + F_3 + F_4 \end{bmatrix}}_{U_1}$$

$$I\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \underbrace{\begin{bmatrix} L(F_2 - F_4) \\ L(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix}}_{U_2} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times I\begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

#learnthesmarterway

In order for the home position to be one of equilibrium, input u_2 has to be 0, the rates p, q & r are equal to 0

i.e., $(u_2 \sim 0, p \sim 0, q \sim 0, r \sim 0)$

and their derivatives are equal to 0.

likewise, the sum of the thrust has to compensate the weight but the yaw angle can be non-zero as long as it's fixed.

$(u_1 \sim mg, \theta \sim 0, \phi \sim 0, \psi \sim \psi_0)$

Linearizing around this point means that we assume that u_1 is almost equal to mg.

we assume that roll & pitch angles are close to 0 & we assume that the yaw angle is fixed or close to fixed at a given value ψ_0.

i.e.,

Linearization

$(u_1 \sim mg, \theta \sim 0, \phi \sim 0, \psi \sim \psi_0)$

If we linearize the eqns, we end up with the simplified eqns —

$$\ddot{x}_1 = \ddot{x} = g(\theta \cos\psi + \phi \sin\psi)$$

$$\ddot{x}_2 = \ddot{y} = g(\theta \sin\psi - \phi \cos\psi)$$

This allows us to design inner feedback loop by simply specifying u_2 using a proportional + derivative controller.

$$u_2 = \begin{bmatrix} K_{p,\phi}(\phi_c - \phi) + K_{d,\phi}(p_c - p) \\ K_{p,\theta}(\theta_c - \theta) + K_{d,\theta}(q_c - q) \\ K_{p,\psi}(\psi_c - \psi) + K_{d,\psi}(r_c - r) \end{bmatrix}$$

(Here we assume that we have the commanded roll, pitch & yaw angles & their derivatives. And all we require for feedback are actual roll, pitch & yaw angles & their derivatives)

after feedback loop

If we look at the position eqns of motion & linearize them, we will find the expressions for the first two components of the acc^n can be written as below.

$$m\ddot{\vec{r}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R \begin{bmatrix} 0 \\ 0 \\ \sum_{i=1} F_i \end{bmatrix}$$

u_1

Date: __/__/____

Linearization

$(u_1 \sim mg,\ \theta \sim 0,\ \phi \sim 0,\ \psi \sim \psi_0)$

$$\ddot{x} = \ddot{r}_1 = g(\theta\cos\psi + \phi\sin\psi)$$
$$\ddot{y} = \ddot{r}_2 = g(\theta\sin\psi - \phi\cos\psi)$$

Using this linearized eqns of motion we now turn our attention to the error eqn that describes the error in position. (Recall we want this error to satisfy a 2nd Order differential eqn).

Commanded acc. — actual (feedback)

$$(\ddot{r}_{i,des} - \ddot{r}_{i,c}) + K_{d,i}(\dot{r}_{i,des} - \dot{r}_i) + K_{p,i}(r_{i,des} - r_i) = 0$$

specified

The commanded acceleration will tell us what thrust we want to apply with the rotors.

$$u_1 = m(g + \ddot{r}_{3,c})$$

$$u_2 = \begin{bmatrix} K_{p,\phi}(\phi_c - \phi) + K_{d,\phi}(p_c - p) \\ K_{p,\theta}(\theta_c - \theta) + K_{d,\theta}(q_c - q) \\ K_{p,\psi}(\psi_c - \psi) + K_{d,\psi}(r_c - r) \end{bmatrix}$$

Commanded roll, pitch, yaw

$$\begin{bmatrix} \phi_c = \frac{1}{g}(\ddot{r}_{1,c}\sin\psi_{des} - \ddot{r}_{2,c}\cos\psi_{des}) \\ \theta_c = \frac{1}{g}(\ddot{r}_{1,c}\cos\psi_{des} + \ddot{r}_{2,c}\sin\psi_{des}) \\ \psi_c = \psi_{des} \end{bmatrix}$$

Planning

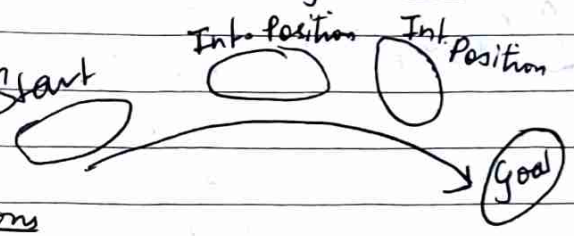
Time, Motion & Trajectories

Date __/__ ___

Generate traj.

Given a 3D environment, we want to be able to specify points in it and have the quadrotor plan obstacle free trajectories in that environment.

Smooth 3D trajectories



Suppose we have a rigid body or a quadrotor that needs to go from a start position to a goal position (also orientation) we may have intermediate position that we want the rigid body to go through.

Applications

→ traj. generation in robotics.

→ Planning trajectories for quadrotors. (Motion Planning)

General Set up

→ we are given start and goal positions (optionally orientations)

→ might want the quadrotor to visit intermediate positions or waypoints (which can also include orientation)

→ In general, we insist that the trajectories that the quadrotor follows are smooth because the quadrotor is a dynamical system, it cannot follow arbitrary trajectories.

→ This generally translates to minimizing a rate of change of "input".

→ we are particularly interested in the order of the dynamical system

→ If we consider a robot with a kinematic model. in other words, we assume that we can arbitrarily specify velocities — 1st order system

The order of the system determines the input

→ If we have a robot with second order dynamics — means we can arbitrarily specify accelerations.

→ for third order system, we should be able to control or specify or command the 3rd order derivative called jerk

→ a 4th order system involves specifying or commanding the fourth derivative — called Snap.

→ If it's an n^{th} order system — in other words, we are specifying or commanding the n^{th} derivation of position, we generally have boundary conditions on the 1st order, 2nd order, 3rd order, ... up to $(n-1)$ deriv.

Date: __/__/____

Calculus of Variations

$$x^*(t) = \underset{x(t)}{\operatorname{argmin}} \int_0^T \mathcal{L}(\dot{x}, x, t)\, dt$$

($x(t)$: function; $\int_0^T \mathcal{L}(\dot{x}, x, t)\, dt$: functional)

We are trying to minimize an integral, to find the best function $x(t)$ that minimizes a functional.

Examples → Shortest distance path (geometry)

$$x^*(t) = \underset{x(t)}{\operatorname{argmin}} \int_0^T \dot{x}^2\, dt$$

→ Fermat's principal (optics)

$$x^*(t) = \underset{x(t)}{\operatorname{argmin}} \int_0^T (1)\, dt$$ → 1 as functional

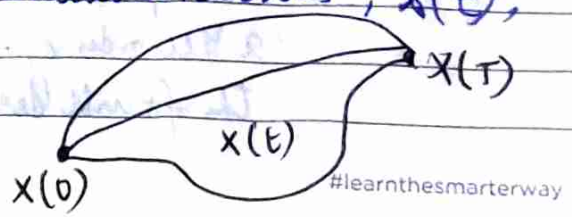
Shortest time path (problem arise in optics)

→ Principle of least action (mechanics)

$$x^*(t) = \underset{x(t)}{\operatorname{argmin}} \int_0^T (T(\dot{x}, x, t) - V(x, t))\, dt$$

(T: K.E.; V: P.E.)

⟹ In our case, we are interested in connecting two points, $x(0)$ and $x(T)$. We are looking for a trajectory, in fact, the best trajectory and we restrict ourselves to differentiable curves. In short,
Consider the set of all differentiable curves, $x(t)$, with a given $x(0)$ and $x(T)$.



we want to find something that minimizes a suitable functional. Independent of the functional, the optimal curve must satisfy what are called Euler Lagrange eqns. These are necessary conditions that must be satisfied by this optimal function.

$$x^*(t) = \arg\min_{x(t)} \int_0^T \mathcal{L}(\dot{x}, x, t)\, dt$$

Euler Lagrange Equation

Necessary condition satisfied by the "optimal" function $x(t)$.

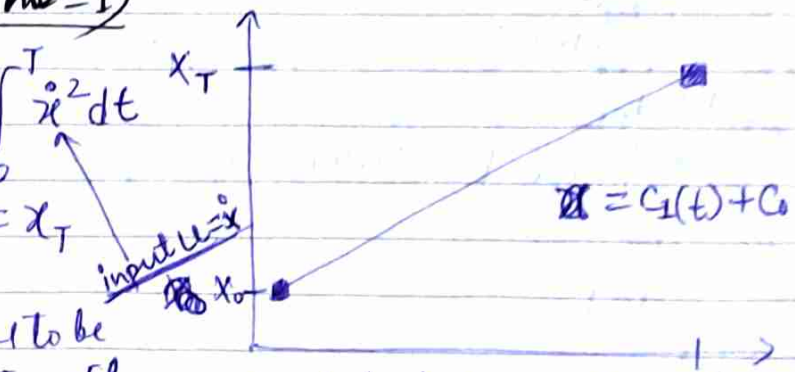
$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{x}}\right) - \frac{\partial \mathcal{L}}{\partial x} = 0$$

Total derivative w.r.t. time — partial derivatives

Smooth trajectories ($n_x = 1$)

$$x^*(t) = \arg\min_{x(t)} \int_0^T \dot{x}^2\, dt$$

$$x(0) = x_0,\ x(T) = x_T$$



Let take the functional to be the integral of $\dot{x}^2$. In other words, input is simply the velocity.

Euler Lagrange Equation $- \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{x}}\right) - \frac{\partial \mathcal{L}}{\partial x} = 0$

$$\mathcal{L}(\dot{x}, x, t) = (\dot{x})^2 \Rightarrow \ddot{x} = 0$$

$$x = C_1 t + C_2$$

Above is the case when we have a first order system with the input being the velocity and it works for robots that have a kinematic model. (Kinematic model is one where we can specify the velocity, we can command the velocity)

Minimum Velocity Trajectories from the Euler-Lagrange Equations

$$x^*(t) = \underset{x(t)}{\operatorname{argmin}} \int_0^T \mathcal{L}(\dot{x}, x, t)\, dt$$

to find the function $x^*(t)$ that minimizes the integral of a cost function $\mathcal{L}(\dot{x}, x, t)$.

when looking for the minimum velocity trajectory, this cost function is $\dot{x}(t)$

The Euler-Lagrange eqn gives the necessary condition that must be satisfied by the optimal function $x(t)$.

Euler-Lagrange eqn $\rightarrow \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{x}}\right) - \frac{\partial \mathcal{L}}{\partial x} = 0$

let's evaluate the E-L eqn for the problem of finding the minimum velocity trajectory

Cost fun - $\mathcal{L}(\dot{x}, x, t) = (\dot{x})^2$

Euler-Lagrange term - $\left(\frac{\partial \mathcal{L}}{\partial x}\right) = 0$ ← no x appears in $\mathcal{L}$

$$\left(\frac{\partial \mathcal{L}}{\partial \dot{x}}\right) = 2\dot{x}$$

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{x}}\right) = \frac{d}{dt}(2\dot{x}) = 2\ddot{x}$$

$\therefore 2\ddot{x} = 0 = 0$

$\Rightarrow \ddot{x} = 0$

Velocity $\rightarrow \dot{x} = C_1$

position $\rightarrow x(t) = C_1 x + C_2$.

Smooth trajectories (general n)

$$x^*(t) = \underset{x(t)}{\operatorname{argmin}} \int_0^T \left(x^{(n)}\right)^2 dt$$

In general, the system might be characterized by higher order dynamics.

input $u = x^{(n)}$

#learnthesmarterway

So, for an nth order system, we might not be able to specify the velocity or command the velocity. The system has inertia, and we can't willy-nilly specify velocities. (suddenly and without planning)

for an nth order system, the input is the n^{th} derivation of position

Euler-Lagrange Eqn

$$x^*(t) = \underset{x(t)}{\operatorname{argmin}} \int_0^T \mathcal{L}(x^{(n)}, x^{(n-1)}, \ldots, \dot{x}, x, t)\, dt$$

Necessary condition satisfied by the "optimal" function

$$\frac{\partial \mathcal{L}}{\partial x} - \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{x}}\right) + \frac{d^2}{dt^2}\left(\frac{\partial \mathcal{L}}{\partial \ddot{x}}\right) + \ldots + (-1)^n \frac{d^n}{dt^n}\left(\frac{\partial \mathcal{L}}{\partial x^{(n)}}\right) = 0$$

$n=1$, Shortest distance
$n=2$, minimum acc^n.
$n=3$, minimum jerk
$n=4$, minimum snap

$n \Rightarrow$ order of system
n^{th} derivation is input.

discrepancy between $n=1$ & the following n

Why is the minimum velocity curve also the shortest distance curve?

→ $n=1$ corresponds to the minimum velocity trajectory and yet it yields us the shortest distance trajectory. Why??

to get the minimum velocity trajectory

$$x^*(t) = \underset{x(t)}{\operatorname{argmin}} \int_0^T \dot{x}^2\, dt$$

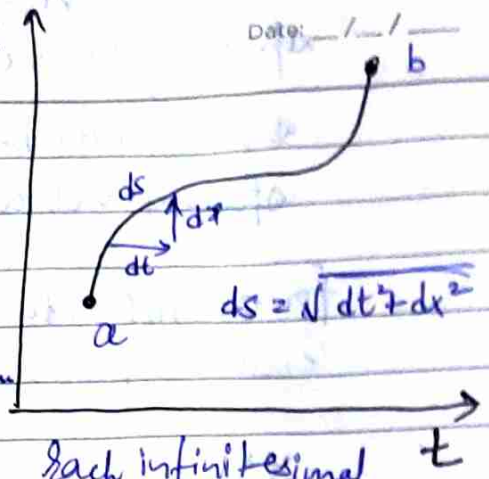
from the Euler-Lagrange equations, the solution is

$$x(t) = c_1 t + c_0$$

Minimum Distance Trajectory

Given two points, a and b, we want to find the trajectory that is the shortest in total length.

→ we can find the length of a trajectory between a and b by integrating infinitesimal segments ds along the curve



$$\text{Length of curve} = \int ds$$

$$= \int_0^T \sqrt{1+\dot{x}^2}\, dt$$

finding the function $x^*(t)$ that minimizes the integral of cost function w.r.t. 't'

$$x^*(t) = \underset{x(t)}{\operatorname{argmin}} \int_0^T \sqrt{1+\dot{x}^2}\, dt$$

Cost function,

$$L(\dot{x}, x, t) = \sqrt{1+\dot{x}^2}$$

Each infinitesimal segments 'ds' has corresponding change in 'dt' and a change in 'dx'

Length of segment ds

$$= \sqrt{dt^2+dx^2}$$

$$= \sqrt{1+\left(\frac{dx}{dt}\right)^2}\, dt$$

$$= \sqrt{1+\dot{x}^2}\, dt$$

Necessary Condition for the optimal trajectory is given by Euler-Lagrange equation —

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) - \frac{\partial L}{\partial x} = 0 \quad \text{— (1)}$$

Euler-Lagrange term: $\left(\frac{\partial L}{\partial x}\right) = 0$ ← No x appears in L

$$\frac{\partial L}{\partial \dot{x}} = \frac{\dot{x}}{\sqrt{1+\dot{x}^2}}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) = \frac{d}{dt}\left(\frac{\dot{x}}{\sqrt{1+\dot{x}^2}}\right)$$

from (1),

$$\frac{d}{dt}\left(\frac{\dot{x}}{\sqrt{1+\dot{x}^2}}\right) = 0$$

$$\Rightarrow \frac{\dot{x}}{\sqrt{1+\dot{x}^2}} = K \Rightarrow \dot{x} = \sqrt{\frac{K^2}{1-K^2}} = C_1$$

This, given the same set of boundary conditions, the minimum velocity trajectory will be the same as the minimum distance trajectory

Velocity

Position $x(t) = C_1 t + C_0$ ← same as minimum velocity solution

#learnthesmarterway

Minimum Jerk Trajectory (3rd order System)

Design a trajectory $x(t)$ such that $x(0)=a$, $x(T)=b$

$$x^*(t) = \underset{x(t)}{\operatorname{argmin}} \int_0^T \mathcal{L}(\dddot{x}, \ddot{x}, \dot{x}, x, t)\, dt$$

$$\mathcal{L} = (\dddot{x})^2$$

Euler-Lagrange :-

$$\frac{\partial \mathcal{L}}{\partial x} - \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{x}}\right) + \frac{d^2}{dt^2}\left(\frac{\partial \mathcal{L}}{\partial \ddot{x}}\right) - \frac{d^3}{dt^3}\left(\frac{\partial \mathcal{L}}{\partial x^{(3)}}\right) = 0$$

$x^{(6)} = 0$ (Condition)

Then, we want a form of the trajectory of the $x(t)$

$$x = C_5 t^5 + C_4 t^4 + C_3 t^3 + C_2 t^2 + C_1 t + C_0 \quad \text{--- (1)}$$

fifth order polynomial in time

Solving for Coefficients

Boundary Conditions –

	Position	Velocity	Acceleration
$t=0$	a	0	0
$t=T$	b	0	0

Solve :-

$$\begin{bmatrix} a \\ b \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ T^5 & T^4 & T^3 & T^2 & T & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 5T^4 & 4T^3 & 3T^2 & 2T & 1 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 20T^3 & 12T^2 & 6T & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} C_5 \\ C_4 \\ C_3 \\ C_2 \\ C_1 \\ C_0 \end{bmatrix}$$

Position Constraints – $x(t) = C_5 t^5 + C_4 t^4 + C_3 t^3 + C_2 t^2 + C_1 t + C_0$

$$x(0) = C_0 = a$$

$$x(T) = C_5 (T)^5 + C_4 (T)^4 + C_3 (T)^3 + C_2 (T)^2 + C_1 (T) + C_0 = b$$

Position Constraints in matrix form:

$$x(0) = C_0 = a$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_5 \\ C_4 \\ C_3 \\ C_2 \\ C_1 \\ C_0 \end{bmatrix} = a$$

$$x(T) = C_5(T)^5 + C_4(T)^4 + C_3(T)^3 + C_2(T)^2 + C_1(T) + C_0 = b$$

Date: __/__/____

$$\begin{bmatrix} T^5 & T^4 & T^3 & T^2 & T & 1 \end{bmatrix} \begin{bmatrix} C_5 \\ C_4 \\ C_3 \\ C_2 \\ C_1 \\ C_0 \end{bmatrix} = b$$

Velocity Constraints :-

$$\dot{x}(t) = 5C_5t^4 + 4C_4t^3 + 3C_3t^2 + 2C_2t + C_1$$

$$\dot{x}(0) = C_1 = 0$$

$$\dot{x}(T) = 5C_5(T)^4 + 4C_4(T)^3 + 3C_3(T)^2 + 2C_2(T) + C_1 = 0$$

In matrix form -

At $t=0$,

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} C_5 \\ C_4 \\ C_3 \\ C_2 \\ C_1 \\ C_0 \end{bmatrix} = 0$$

At $t=T$,

$$\begin{bmatrix} 5T^4 & 4T^3 & 3T^2 & 2T & 1 & 0 \end{bmatrix} \begin{bmatrix} C_5 \\ C_4 \\ C_3 \\ C_2 \\ C_1 \\ C_0 \end{bmatrix} = 0$$

Acceleration Constraints -

$$\ddot{x}(t) = 20C_5t^3 + 12C_4t^2 + 6C_3t^2 + 2C_2$$

In matrix form -

At $t=0$,

$$\begin{bmatrix} 0 & 0 & 0 & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} C_5 \\ C_4 \\ C_3 \\ C_2 \\ C_1 \\ C_0 \end{bmatrix} = 0$$

At $t=T$

$$\begin{bmatrix} 20T^3 & 12T^2 & 6T & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} C_5 \\ C_4 \\ C_3 \\ C_2 \\ C_1 \\ C_0 \end{bmatrix} = 0$$

Combine Constraints into one matrix expression –

Date: __/__/____

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ T^5 & T^4 & T^3 & T^2 & T & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 5T^4 & 4T^3 & 3T^2 & 2T & 1 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 20T^3 & 12T^2 & 6T & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} C_5 \\ C_4 \\ C_3 \\ C_2 \\ C_1 \\ C_0 \end{bmatrix} = \begin{bmatrix} a \\ b \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

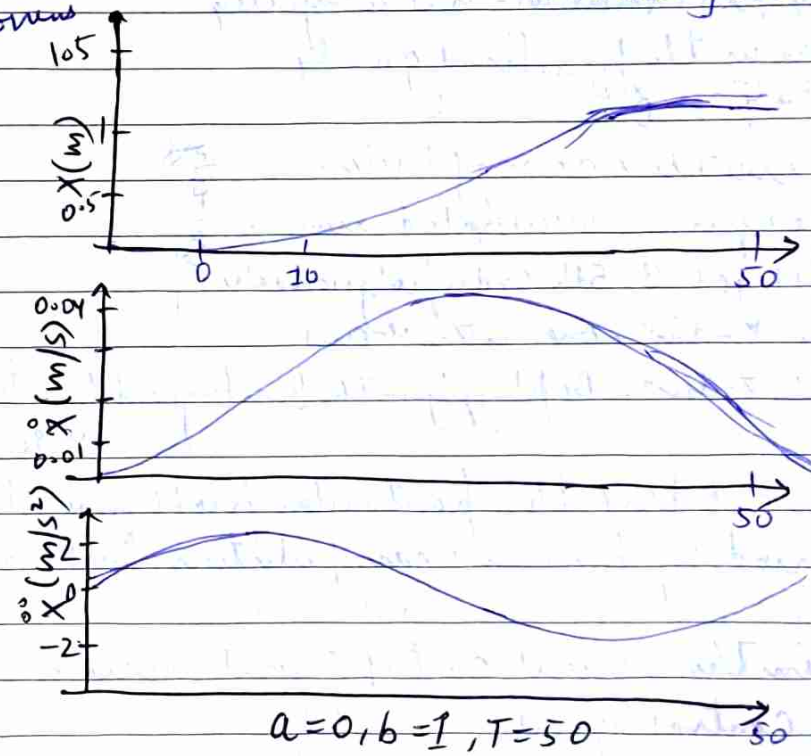
Here, T, a & b are known. we can solve using known terms

$AX = b$

$X = A^{-1}b$ → put value of $C_0, C_1 \dots C_5$ into eqn (1)

we will get the desired Minimum Jerk Trajectory

Note the order of Coeffs. in the matrix of unknowns matters



Extensions to multiple dimensions

trying to generate trajectories in xy plane

$$(x^*(t), y^*(t)) = \arg\min_{x(t), y(t)} \int_0^T \mathcal{L}(\dot{x}, \dot{y}, x, y, t)\, dt$$

Euler Lagrange Eqn – Necessary condition satisfied by the "optimal" function

x-dirn $\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{x}}\right) - \frac{\partial \mathcal{L}}{\partial x} = 0$

ydirn $\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{y}}\right) - \frac{\partial \mathcal{L}}{\partial y} = 0$

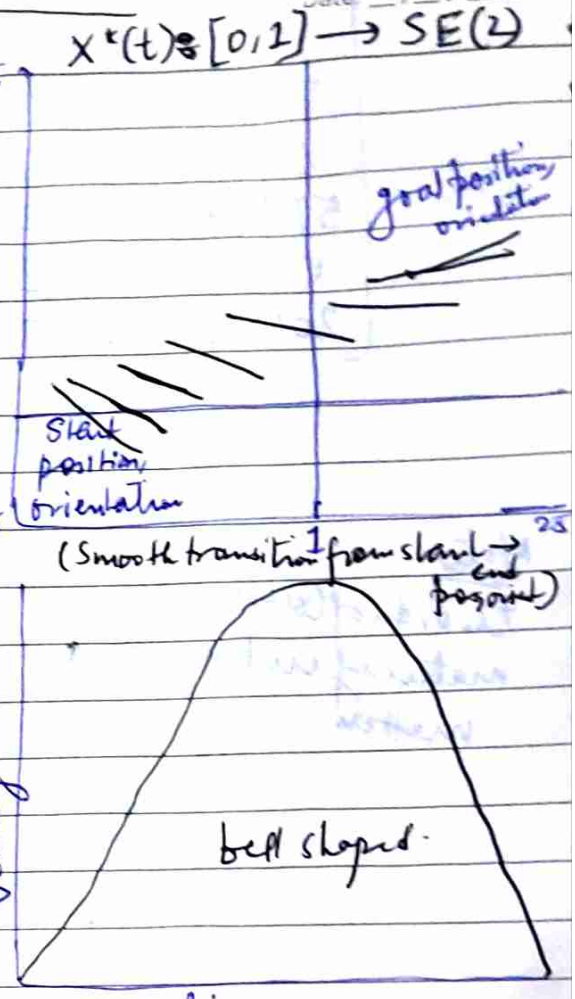
Minimum Jerk for Planar Motion

Minimum jerk trajectory in (x, y, θ)

$$\min_{x(t), y(t), \theta(t)} \int_0^1 \left(\dddot{x}^2 + \dddot{y}^2 + \dddot{\theta}^2 \right) dt$$

Here we are looking at a trajectory in the x, y and θ space we are assuming we are given a starting x & y position and starting orientation θ. Also, we are given a final x, y & θ. We are trying to find a trajectory that minimizes the functional given by $\left(\dddot{x}^2 + \dddot{y}^2 + \dddot{\theta}^2 \right)$.

→ If we use the version of Euler-Lagrange eqns for multiple dimensions, we will get 3 5th order polynomials, one in the x-dirn, one in the y and one in the z-dirn. By plugging in the boundary condition we get a bell shaped plot.



→ It turns out that this particular problem is also relevant to modeling human manipulation tasks & Human two-handed manipulation tasks –

→ Noise in the neural control signal increases with size of the control signal.

→ Rate of change of muscle fiber length is critical in relaxed, voluntary motions.

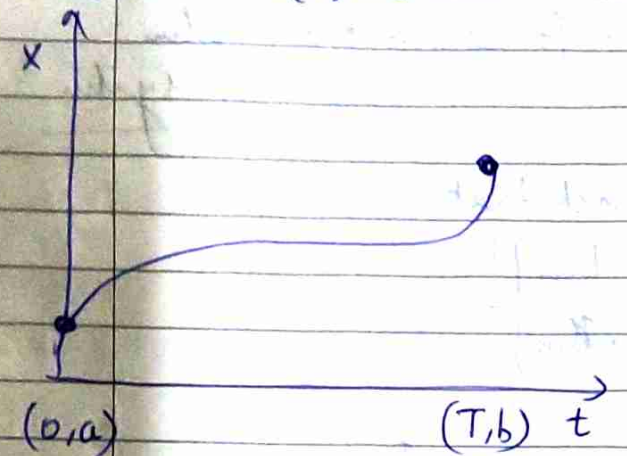
→ In robotics, it's not sufficient to specify just the initial and final position & orientation. In order to ensure that the trajectory is safe we might want to specify the intermediate waypoints.

Waypoint Navigation

(Instead of thinking simply about trajectories with known starting & end configurations we want to think about trajectories that pass through intermediate waypoints)

Smooth 1D trajectories

Design a trajectory $x(t)$ such that $x(0) = a$, $x(T) = b$

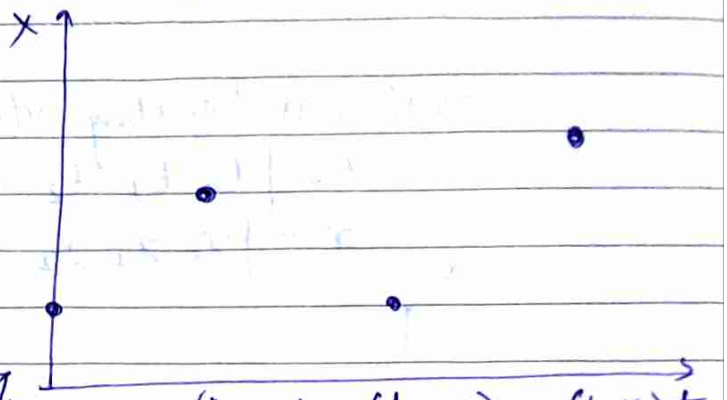


Multi-Segment 1D trajectories

Design a trajectory $x(t)$ such that

$t = [t_0 \; t_1 \; t_2 \; \dots \; t_m]^T$

$x = [x_0 \; x_1 \; x_2 \; \dots \; x_m]^T$

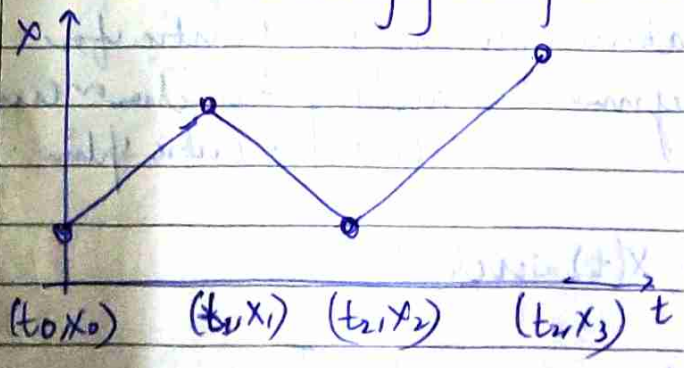


Define piecewise continuous trajectory (one for each segment)

$$x(t) = \begin{cases} x_1(t), & t_0 \le t \le t \\ x_2(t), & t_1 \le t < t_2 \\ \dots \\ x_m(t), & t_{m-1} \le t < t_m \end{cases}$$

intermediate time points & positions.

The obvious way of doing it to simply connect the waypoint



But this is not practical

What if the system is a 2nd order system with inertia

once we get started on the first segment, it's impossible for a 2nd order system to make the sharp turn at the kink.

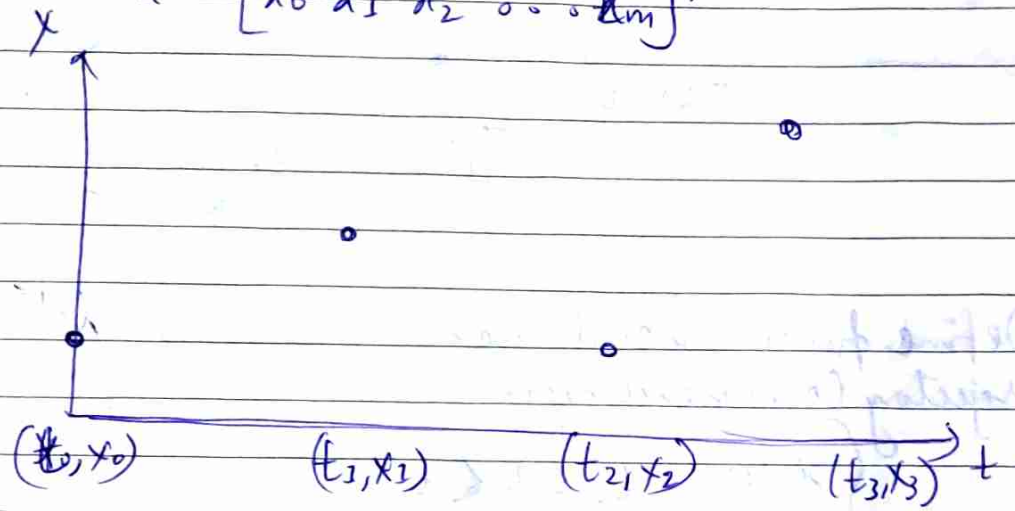
In order to make the trajectory more amenable to a second order system, we want to insist Date: __/__/____ that the trajectory is smooth and not have the kink at the intermediate points. In order to solve this problem, let's look at a min^m acc^h curve for second order system.

Minimum Acceleration Curve for 2nd order system

Design a trajectory $x(t)$ such that:

$$t = [t_0\ t_1\ t_2 \cdots t_m]^T$$

$$x = [x_0\ x_1\ x_2 \cdots x_m]^T$$



$$\min_{x(t)} \left[\int_{t_0}^{t_1} (\dddot{x})^2 dt + \cdots + \int_{t_{m-1}}^{t_m} (\dddot{x})^2 dt \right]$$

the functional consisting of many terms, each term corresponds to a particular segment

from the Euler-Lagrange eqns. if we solve above eqn, we get a cubic for each segment. The resulting function or curve is called a cubic spline.

Cubic Splines

Design a trajectory $x(t)$ such that

$$t = [t_0\ t_1\ t_2 \cdots t_m]^T$$

$$x = [x_0\ x_1\ x_2 \cdots x_m]^T$$

Date: __/__/____

$$x(t) = \begin{cases} x_1(t) = c_{1,3}t^3 + c_{1,2}t^2 + c_{1,1}t + c_{1,0}, & t_0 \le t < t_1 \\ x_2(t) = c_{2,3}t^3 + c_{2,2}t^2 + c_{2,1}t + c_{2,0}, & t_1 \le t < t_2 \\ \cdots \\ x_m(t) = c_{m,3}t^3 + c_{m,2}t^2 + c_{m,1}t + c_{m,0}, & t_{m-1} \le t < t_m \end{cases}$$

4m degrees of freedom
(4m Constants for a cubic spline)
each of these constants corresponds to a degree of freedom

In order to determine these constants we have to specify 4m different boundary conditions or intermediate conditions

We know where the intermediate points are
This gives us (2m) boundary points.
We further insist that the curve is smooth.
More specifically, differentiable up to two times at each of these intermediate points.
This gives us an additional [2(m-1)] boundary condition
Finally, we insist that the curve starts with a specified velocity & ends with a specified velocity. In this case, velocity is taken to be 0.
This gives us additional 2 boundary conditions.

$x_1(t_1) = x_2(t_1) = x_1$
$\dot{x}_1(t_1) = \dot{x}_2(t_1)$
$\ddot{x}_1(t_1) = \ddot{x}_2(t_1)$

$x_3(t_3) = x_3$

$x_1(t_0) = x_0$

$x_2(t_2) = x_3(t_2) = x_2$
$\dot{x}_2(t_2) = \dot{x}_3(t_2)$
$\ddot{x}_2(t_2) = \ddot{x}_3(t_2)$

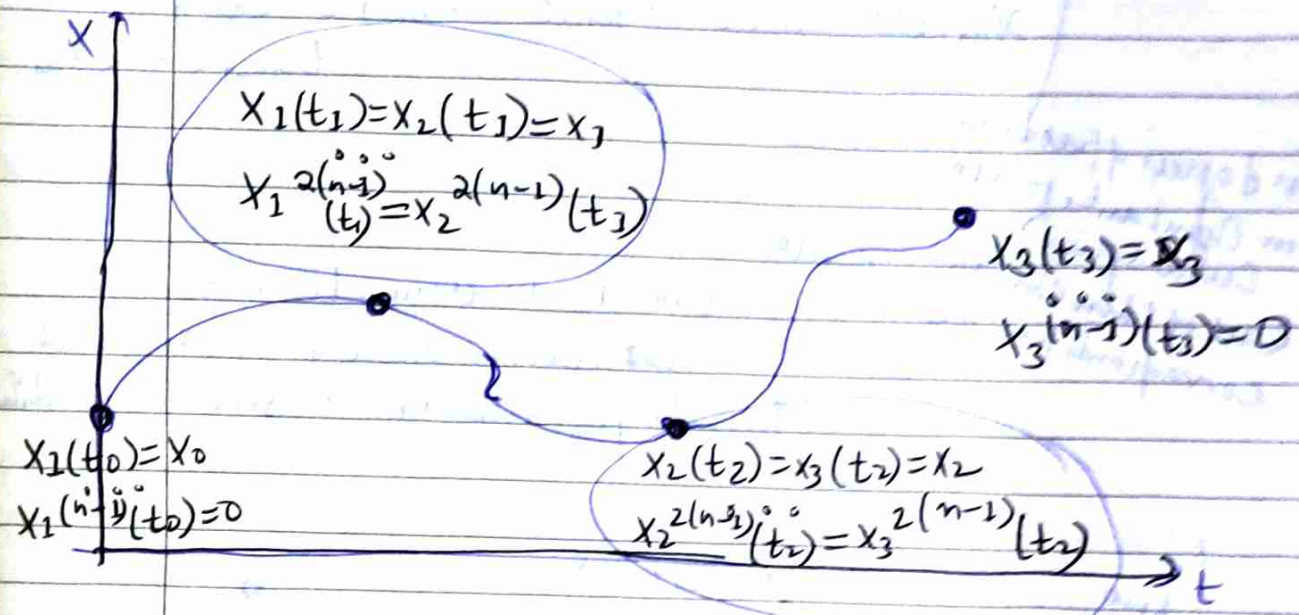
$2m + 2(m-1) + 2$
$= 4m$ constraints
Allowing us to specify 4m degrees of freedom associated with the undetermined constants

Spline for nth order System

Design a trajectory $x(t)$ such that

$$t = [t_0 \; t_1 \; t_2 \; \ldots \; t_m]^T$$

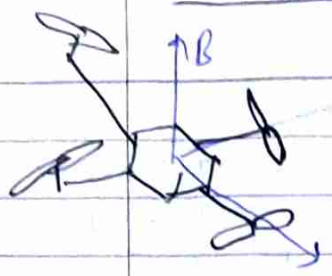
$$x = [x_0 \; x_1 \; x_2 \; \ldots \; x_m]^T$$



Above intermediate boundary conditions specify continuity upto $2(n-1)$ order derivative. This leads upto minimum snap trajectories where $n = 4$.

Minimum Snap Trajectories

Motion Planning for Quadrotors



Dynamic Eqns of motion

Newton-Euler Equations

$${}^A\vec{\omega}^B = p\hat{b}_1 + q\hat{b}_2 + r\hat{b}_3$$

Rotation of thrust vector from B to A

$$m\ddot{\vec{r}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R\begin{bmatrix} 0 \\ 0 \\ F_1+F_2+F_3+F_4 \end{bmatrix} u_1$$

two set of inputs, u_1 & u_2

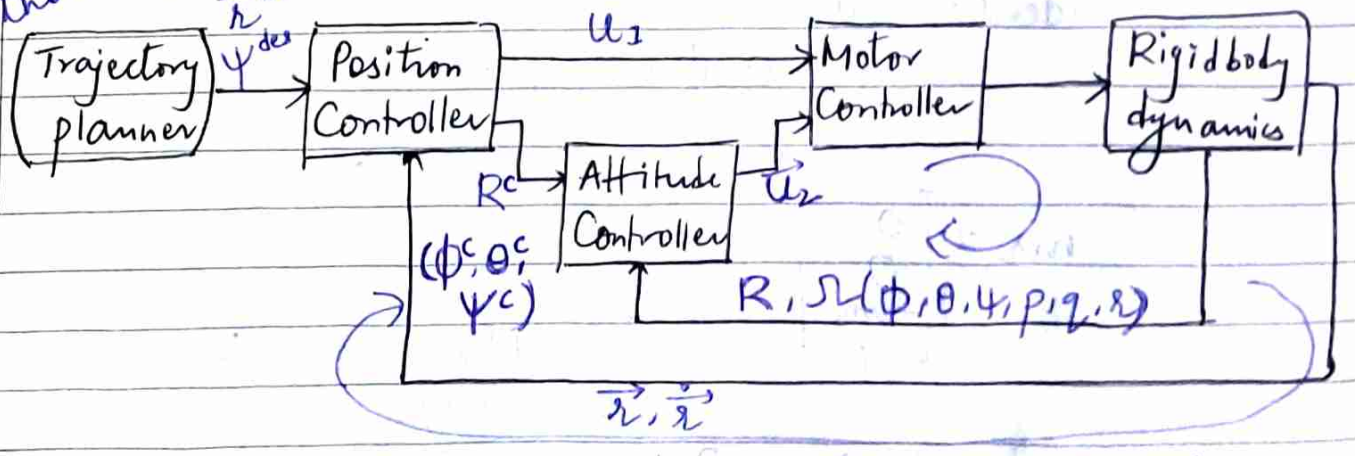
Components in the inertial frame as, $\hat{a}_1, \hat{a}_2, \hat{a}_3$ along

$$I\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} L(F_2-F_4) \\ L(F_3-F_1) \\ M_1-M_2+M_3-M_4 \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times I\begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

u_2 (Components in the body frame along $\hat{b}_1, \hat{b}_2$ & $\hat{b}_3$, the principal axes)

we are trying to generate trajectories in terms of the quadrotor position and trying to figure out how we want to control the motors to drive the vehicle to those trajectories

Position Controller



The quadrotor controller consists of two loops, first the outer position control loop that specifies u_1 and then second, an inner attitude control loop that determines u_2. In order to control the position, the inner loop, the attitude control loop has to work perfectly

$$m\ddot{\vec{r}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R \begin{bmatrix} 0 \\ 0 \\ \sum_{i=1}^{4} F_i \end{bmatrix}$$

u_1 & u_2 are two inputs

u_1 is a scalar quantity; there is only one input in u_1 but we are trying to control x, y & z components of acc^n.

$R(\theta, \phi, \psi)$

relate the angular velocities p, q and r to the derivatives of the roll, pitch & yaw angles

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} C\theta & 0 & -C\phi S\theta \\ 0 & 1 & S\phi \\ S\theta & 0 & C\phi C\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

$$I \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \underbrace{\begin{bmatrix} L(F_2 - F_4) \\ L(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix}}_{\vec{u}_2} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times I \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

The third set of eqns relates the input u_2 to the rate of change of angular velocity.

from the last two eqns ← (second derivative of the rotation matrix depends on u_2)

Linearized Model of the System

In the linearized model, the rotation matrix is the identity. The angular velocities are related to the rates of change of roll, pitch and yaw through an identity matrix.

$(\theta \sim 0, \phi \sim 0, \psi \sim 0)$

$(p \sim 0, q \sim 0, r \sim 0)$

$$m\ddot{\vec{r}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + I R \begin{bmatrix} 0 \\ 0 \\ F_1 + F_2 + F_3 + F_4 \end{bmatrix} u_1$$

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = I \begin{bmatrix} C\theta & 0 & -C\phi S\theta \\ 0 & 1 & S\phi \\ S\theta & 0 & C\phi C\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

$R(\theta, \phi, \psi)$

$$I \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \underbrace{\begin{bmatrix} L(F_2 - F_4) \\ L(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix}}_{u_2} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times I \begin{bmatrix} p \\ q \\ r \end{bmatrix} \to 0$$

#learnthesmarterway

the second derivative of position is proportional to u_1 and the **fourth** derivative of position is **proportional** to u_2.

In summary, the position control loop involves attitude control & because of that, the position control system leads to a 4th order system. To specify trajectories, we want to consider only those trajectories that are differentiable at least 4 times.

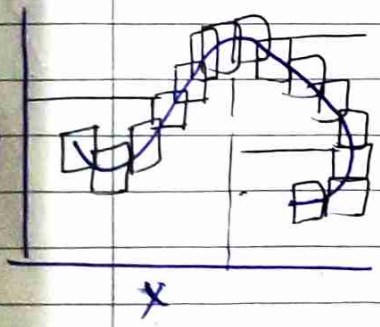
This motivates the use of a minimum snap trajectory which tries to minimize the fourth derivative of position integrated over the time history. We use minimum snap trajectories for motion planning for quadrotors.

Minimum Snap Trajectory

$$x^*(t) = \arg\min_{x(t)} \int_0^T \left(x^{(iv)}\right)^2 dt$$

Works only if we have a robust inner attitude control loop.

Minimum Snap trajectory with Constraints



this traj. is obtained by slicing together many minimum snap trajectories (reminiscent of what we looked earlier, when we synthesized cubic splines or nth order splines to go through many different waypoints).

In this case, there are no waypoints, instead the robot knows where the obstacles are.

How do we synthesize minimum snap trajectories in a known environment?

- Circular Trajectory
- Navigating Obstacles
- Aerial grasping & manipulation
- Perching (a quadrotor equipped with a special grip at the end, is able to perch on near vertical surfaces) & planning minimum snap trajectories to reach the desired end configuration at a close to 0 velocity but with high attitude

*****

Linearization of Quadrotor Equations of Motion

Date: __/__/__

In the linearized eqns of motion for the quadrotor, the 2nd derivative of position is proportional to u_3 & the 4th derivative of position is proportional to u_2.

Quadrotor Eqns of Motion

Linear momentum balance:

$$m\ddot{\vec{r}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R \begin{bmatrix} 0 \\ 0 \\ \sum_{1}^{4} F_i \end{bmatrix}$$

Angular momentum balance:

$$I\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} L(F_2 - F_4) \\ L(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times I \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

Equations of motion in terms of u_1 & Components of u_2.

Linear momentum balance

$$m\ddot{\vec{r}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R \begin{bmatrix} 0 \\ 0 \\ u_1 \end{bmatrix}$$

Angular momentum balance

$$I\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} u_{2x} \\ u_{2y} \\ u_{2z} \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times I \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

Equilibrium Hover Configuration

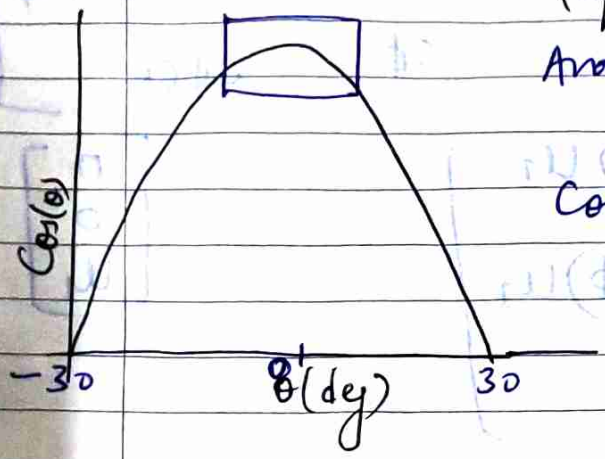
$$\vec{r} = \vec{r}_0, \quad \theta = \phi = 0, \quad \psi = \psi_0$$

$$\dot{\vec{r}} = 0, \quad \dot{\theta} = \dot{\phi} = \dot{\psi} = 0$$

Linearization of trigonometric functions

What is the value of $\cos\theta$ near $\theta = 0$?

(equilibrium configuration)



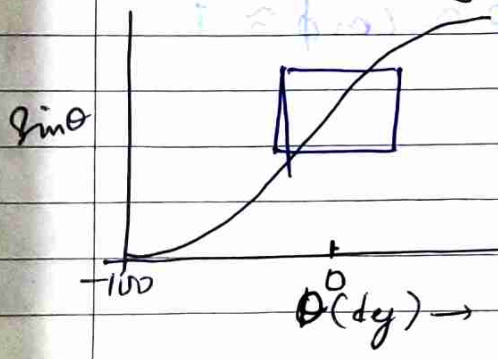
Around $\theta = 0$, $\cos\theta$ can be approximated using the Taylor series.

$$\cos\theta \approx \cos\theta\big|_{\theta=0} + \frac{d(\cos\theta)}{d\theta}\Big|_{\theta=0}\theta + \text{higher order terms}$$

$$\approx 1 - \sin\theta\big|_{\theta=0}\theta$$

$$\approx 1.$$

What is the value of $\sin\theta$ near $\theta = 0$?



Can be approximated using the Taylor series —

$$\sin\theta \approx \sin\theta\big|_{\theta=0} + \frac{d}{d\theta}\sin\theta\big|_{\theta=0}\theta + \text{higher order term.}$$

$$\approx 0 + \cos\theta\big|_{\theta=0}\theta$$

$$\approx \theta$$ (Around $\theta = 0$, we expect sin function to look linear.

Linearized Equations of Motions

What are the eqns of motion of the quadrotor when it is near the equilibrium hover configuration?

$$\vec{x} \approx \vec{x}_0 ; \theta \approx \phi \approx 0, \psi \approx \psi_0$$

$$\dot{\vec{x}} \approx 0, \dot{\theta} \approx \dot{\phi} \approx \dot{\psi} \approx 0$$

(a) Linear momentum Equation near Hover

$$m\ddot{\vec{x}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R \begin{bmatrix} 0 \\ 0 \\ u_1 \end{bmatrix}$$

we can explicitely write the rotation matrix in terms of the Euler angles.

$$m\ddot{\vec{r}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + \begin{bmatrix} C\psi C\theta - S\phi S\psi S\theta & -C\phi S\psi & C\psi S\theta + C\theta S\phi S\psi \\ C\theta S\psi + C\psi S\theta S\phi & C\phi C\psi & S\psi S\theta - C\psi C\theta S\phi \\ -C\phi S\theta & S\phi & C\phi C\theta \end{bmatrix}$$

Date: __/__/____

$$\begin{bmatrix} m\ddot{x} = (C\psi S\theta + C\theta S\phi S\psi)u_1 \\ m\ddot{y} = (S\psi S\theta - C\psi C\theta S\phi)u_1 \\ m\ddot{z} = -mg + (C\phi C\theta)u_1 \end{bmatrix} \qquad \begin{bmatrix} 0 \\ 0 \\ u_1 \end{bmatrix}$$

Substituting in the approximation,

$$\sin\theta \approx \theta, \sin\phi \approx \phi, \cos\theta \approx \cos\phi \approx 1$$

we get

$$\begin{bmatrix} m\ddot{x} = (\theta C\psi + \phi S\psi)u_1 \\ m\ddot{y} = (\theta S\psi - \phi C\psi)u_1 \\ m\ddot{z} = -mg + u_1 \end{bmatrix}$$

clearly, the second derivative of position is proportional to u_1.

Angular Rates Near Hover

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} C\theta & 0 & -C\phi S\theta \\ 0 & 1 & S\phi \\ S\theta & 0 & C\phi C\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

$$\begin{cases} p = \dot{\phi} C\theta - \dot{\psi} C\phi S\theta \\ q = \dot{\theta} + \dot{\psi} S\phi \\ r = \dot{\phi} S\theta + \dot{\psi} C\phi C\theta \end{cases}$$

#learnthesmarterway

Substituting in the approximation,

$\sin\theta \approx \theta$, $\sin\phi \approx \phi$, $\cos\theta \approx \cos\phi \approx 1$

we get

$$p = \dot{\phi} - \dot{\psi}\theta$$

$$q = \dot{\theta} + \dot{\psi}\phi$$

$$r = \dot{\phi}\theta + \dot{\psi}$$

Substituting in the approximation $(\dot{\psi}\theta \approx \dot{\psi}\phi \approx \dot{\phi}\theta \approx 0)$

Higher order terms: Product of two terms around 0 is approx. 0.

So, around hover,

$$\begin{bmatrix} p = \dot{\phi} \\ q = \dot{\theta} \\ r = \dot{\psi} \end{bmatrix}$$

the angular velocity components are approximately the time derivatives of the Euler angles.

Angular Momentum Equation Used

$$I\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} U_{2x} \\ U_{2y} \\ U_{2z} \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times I\begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

We approximate the off diagonal inertial Terms close to 0.

This allows us to simplify the inertia matrix.

Substituting in the approximation:

$$I_{xy} \approx I_{yx} \approx I_{xz} \approx I_{zx} \approx I_{yz} \approx I_{zy} \approx 0$$

This allows us to simplify the inertia matrix.

$$\begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} U_{2x} \\ U_{2y} \\ U_{2z} \end{bmatrix} - \begin{bmatrix} 0 & r & -q \\ -r & 0 & p \\ q & -p & 0 \end{bmatrix}\begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}\begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

$$I_{xx}\dot{p} = U_{2x} - I_{yy}qr + I_{zz}qr$$

$$I_{yy}\dot{q} = U_{2y} + I_{xx}pr - I_{zz}pr$$

$$I_{zz}\dot{r} = U_{2z} - I_{xx}pq + I_{yy}pq$$

Substituting in the approximation:

$$qr \approx pr \approx pq \approx \dot{\theta}\dot{\psi} \approx \dot{\phi}\dot{\psi} \approx \dot{\phi}\dot{\theta} \approx 0$$

← Higher order terms: Product of two terms around 0 is approximately 0.

This gives us the following set of eqns:

$$I_{xx}\dot{p} = u_{2x}$$

$$I_{yy}\dot{q} = u_{2y}$$

$$I_{zz}\dot{r} = u_{2z}$$

Substituting in the approximation

$$p \approx \dot{\phi},\ q \approx \dot{\theta},\ r \approx \dot{\psi}$$

approximation of the angular velocity components as the Euler angle derivatives

$$\ddot{\phi} = \frac{u_{2x}}{I_{xx}},\quad \ddot{\theta} = \frac{u_{2y}}{I_{yy}},\quad \ddot{\psi} = \frac{u_{2z}}{I_{zz}}$$

Equation of motion

Linearized linear momentum eqn:

$$m\ddot{x} = (\theta c\psi + \phi s\psi)u_1$$

differentiating the equation,

$$m\dddot{x} = (\theta c\psi + \phi s\psi)\dot{u}_1 + (\dot{\theta}c\psi - \theta s\psi\dot{\psi} + \dot{\phi}s\psi + \phi c\psi\dot{\psi})u_1$$

diff. again,

$$m\ddot{x}^{iv} = (\theta c\psi + \phi s\psi)\ddot{u}_1 + 2(\dot{\theta}c\psi - \theta s\psi\dot{\psi} + \dot{\phi}s\psi + \phi c\psi\dot{\psi})\dot{u}_1 + (\ddot{\theta}c\psi - \dot{\theta}s\psi\dot{\psi} - \theta s\psi\ddot{\psi} - \theta c\psi\dot{\psi}^2 + \ddot{\phi}s\psi + \dot{\phi}c\psi\dot{\psi} + \phi c\psi\ddot{\psi} - \phi c\psi\dot{\psi}^2)u_1$$

Substituting in the approximations:

$$\ddot{\phi} = \frac{u_{2x}}{I_{xx}}, \quad \ddot{\theta} = \frac{u_{2y}}{I_{yy}}, \quad \ddot{\psi} = \frac{u_{2z}}{I_{zz}}$$

The linear momentum equation becomes:

$$m\ddot{x} = \ldots + \left(\frac{u_{2y}}{I_{yy}} c\psi + \frac{u_{2z}}{I_{zz}} \theta(c\psi - s\psi) + \frac{u_{2x}}{I_{xx}} s\psi \right) u_1$$

Carrying out this procedure in y & z dirⁿ yields similar eqns.

So, the fourth derivative of position is proportional to u_2

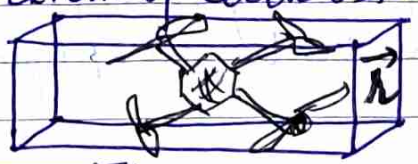
Obstacles

* * * * *

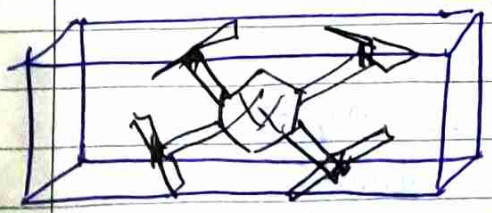
Imagine we have obstacles (here we are assuming there are more than one quadrotor flying through the obstacle filled environment at the same time).

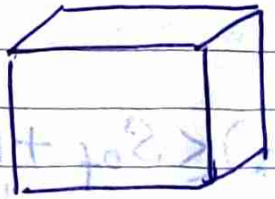
We are going to box off these quadrotors inside virtual rectangular parallelopipeds. We assume that these rectangular parallelopipeds are known and we want to make sure that they don't intersect each other.

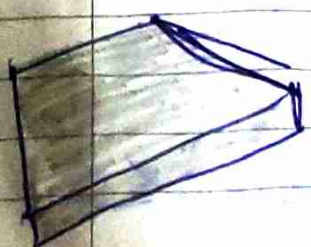
In addition there are many obstacles. We assume that every obstacle is convex & they are polyhedral models that can characterize the extent of each obstacle.

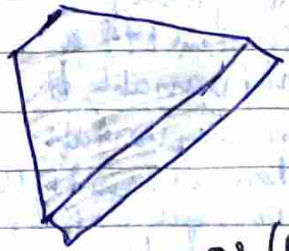


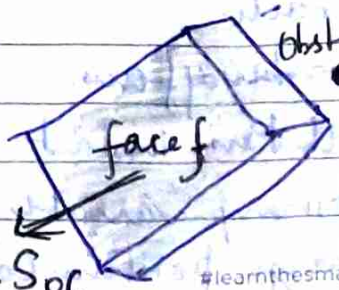
✓ Convex
✓ Polyhedral model











$n_{of} \cdot r(t_k) \leq s_{of}$

#learnthesmarterway

If the obstacles are non-convex, we can always describe them as a union of convex obstacles.

Let's consider a generic obstacle, obstacle O. (see prev page)

Because each obstacle is a convex polyhedron, we want to look at collisions in terms of the rectangular parallelopiped that characterizes the quadrotor and each of the faces characterizing the obstacle.

for a generic face f on an obstacle O, for a generic time instant t_k, we want to make sure that the convex polyhedron characterizing the robot and the face f don't intersect. That fact is written as a linear inequality.

$$\vec{n}_{of} \cdot \vec{r}(t_k) \leq s_{of}$$

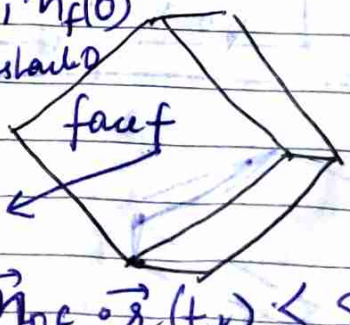
The inequality restricts the position $\vec{r}$ that can be occupied by the robot (it of course assumes that we know exactly where the obstacle is and we know the normal N for each face f for all obstacles, in this case considering obstacle O).

We can similar eqns for every face, for every obstacle & every robot. This gives us the set of eqns.

Integer Constraints for Obstacle Avoidance

$$\vec{n}_{of} \cdot \vec{r}(t_k) \leq s_{of} + \underbrace{M b_{ofk}}_{\text{consists of a large, +ve constant M \& a binary variable b. This binary variable is characterized by the obstacle O, face f \& time instant } t_k.} \quad \forall f = 1, \ldots, n_f(O)$$

O obstacle
n_f number of faces.
t_k Kth time instant
b_{ofk} binary variable
M large positive constant



#learnthesmarterway

$$\sum_{f=1}^{n_f(o)} b_{ofk} \leq n_f(o) - 1$$

Date __/__/____

The reason we want the extra term Mb_{ofk} is simple. In order for the robot to avoid a particular obstacle o, it is enough that this inequality be satisfied for any one of the faces. It is not necessary that this inequality to be satisfied for every face on the obstacle. That fact is captured by the inequality given below

$$\sum_{f=1}^{n_f(o)} b_{ofk} \leq n_f(o) - 1$$

If there are n_f faces, we want to make sure that atleast one of these binary variables is 0. And that is captured in this inequality.

Transporting Suspended Payloads

This approach has now been used to synthesize trajectories for suspended payloads when the vehicle is carrying a payload and the length of the suspended payload is larger than the height of the window it needs to go through.

The same technique can be extended to ~~navigation~~ multiple quadrotors.

Multiple Quadrotors

Imagine we have obstacles. We want quadrotor 1 to go to position 1, quadrotor 2 to go to position 2, quadrotor 3 to go to position 3 and we don't want them to intersect.

Every trajectory is a minimum snap trajectory.