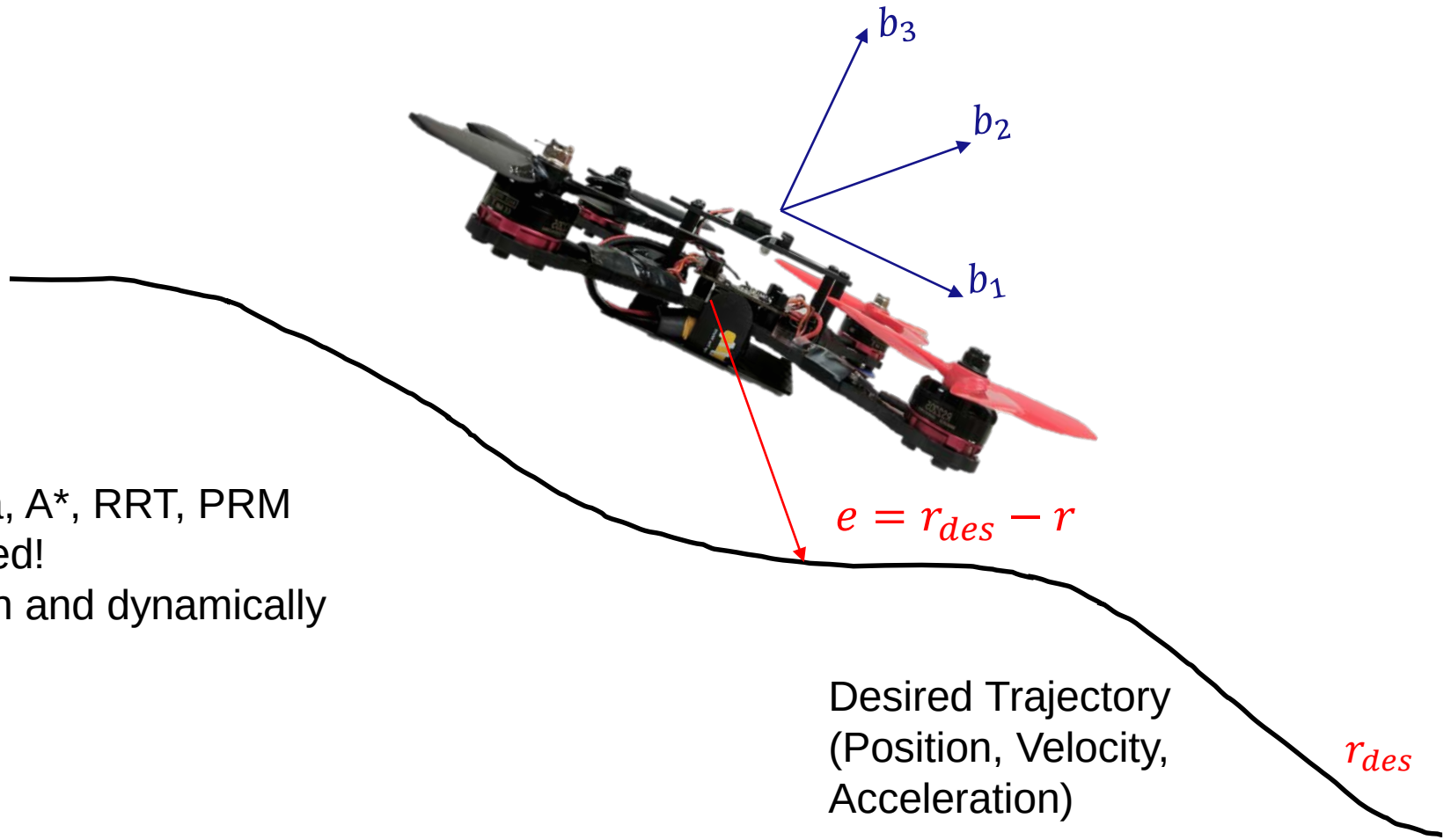
CMSC828T
Vision, Planning And Control In Aerial Robotics

TRAJECTORY PLANNING

Smooth 3D Trajectories

Trajectory generation in robotics
Interpolation in computer graphics

Why do we need this?

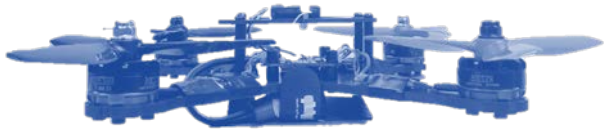


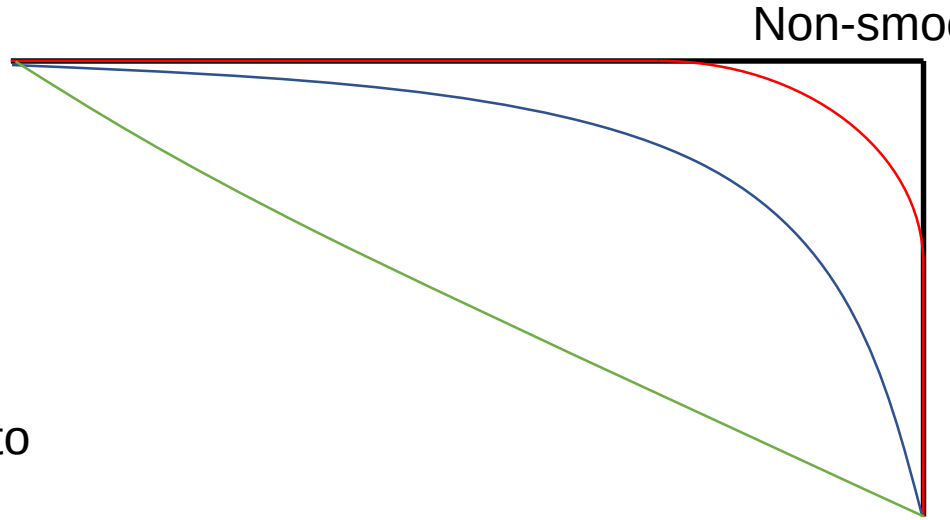
Paths generated by Dijkstra, A*, RRT, PRM need to be time parametrized!
Paths need to made smooth and dynamically feasible!

Desired Trajectory (Position, Velocity, Acceleration)

Most of these slides are inspired by MEAM620 Slides at UPenn

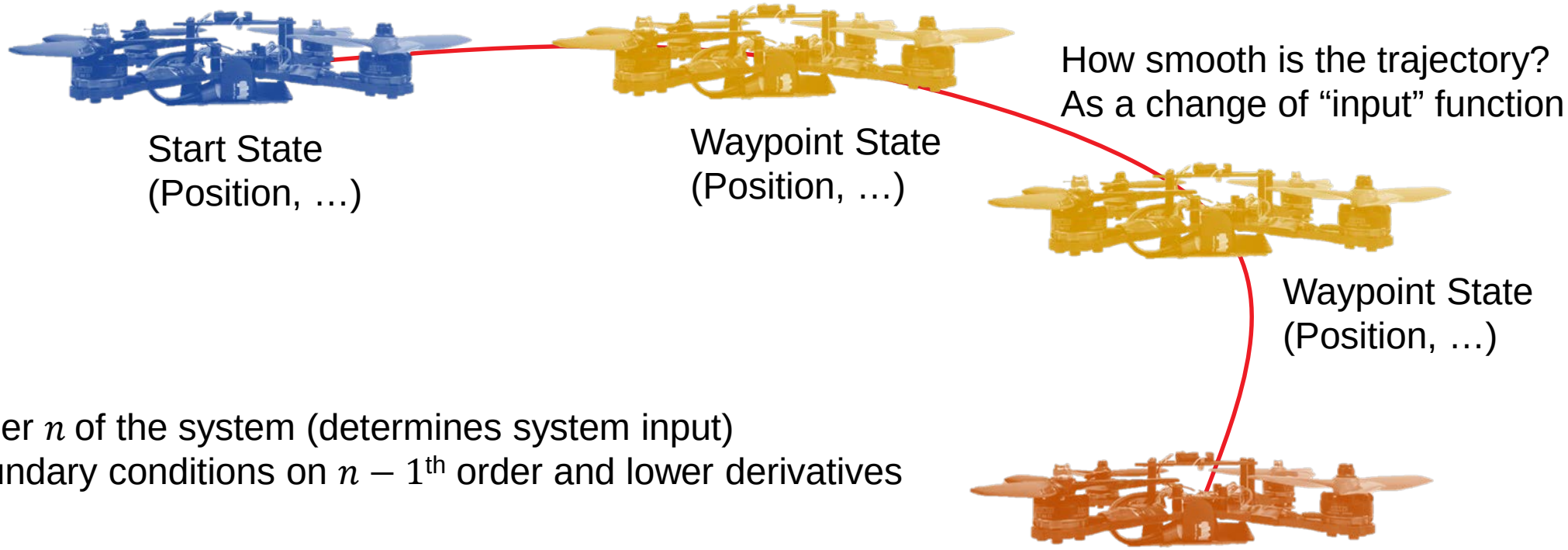
Achievable Motion





Use Calculus of Variations to obtain smooth trajectories!

Problem Setup



Order n of the system (determines system input)
Boundary conditions on $n - 1^{th}$ order and lower derivatives

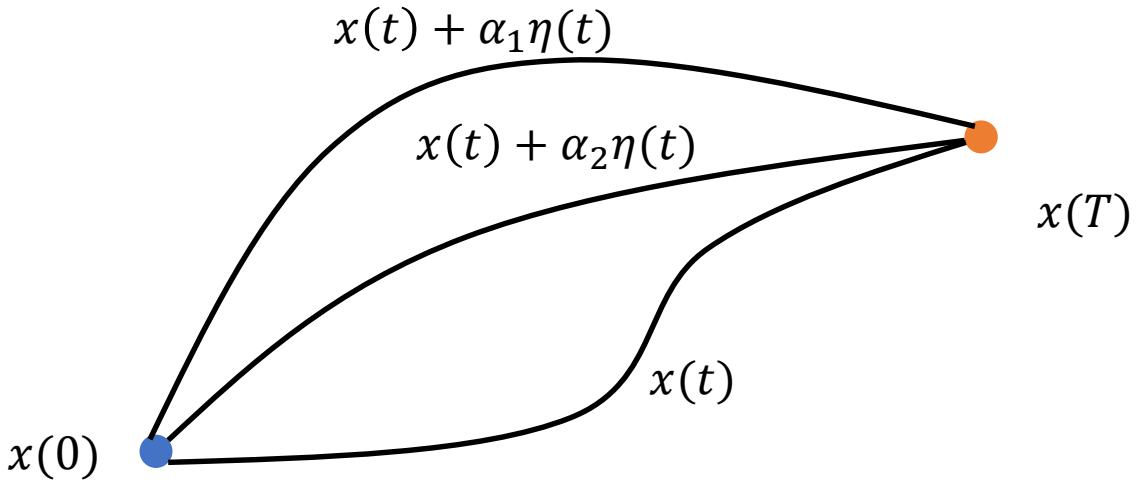
Calculus of Variations

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

Assume $x(t)$ is the optimal function

Any other path from $x(0)$ to $x(T)$ can be written as

$x(t) + \alpha\eta(t) \ni \eta(0) = \eta(T) = 0$



Calculus of Variations

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

Since $x(t)$ is the optimal function, we have the criterion that

$$\left.\frac{d}{d\alpha}\right|_{\alpha=0} \int_0^T \mathcal{L}(\dot{x} + \alpha\dot{\eta}(t), x(t) + \alpha\eta(t), t)dt = 0$$

Euler Lagrange Equation to the rescue!

Necessary condition satisfied by the optimal function

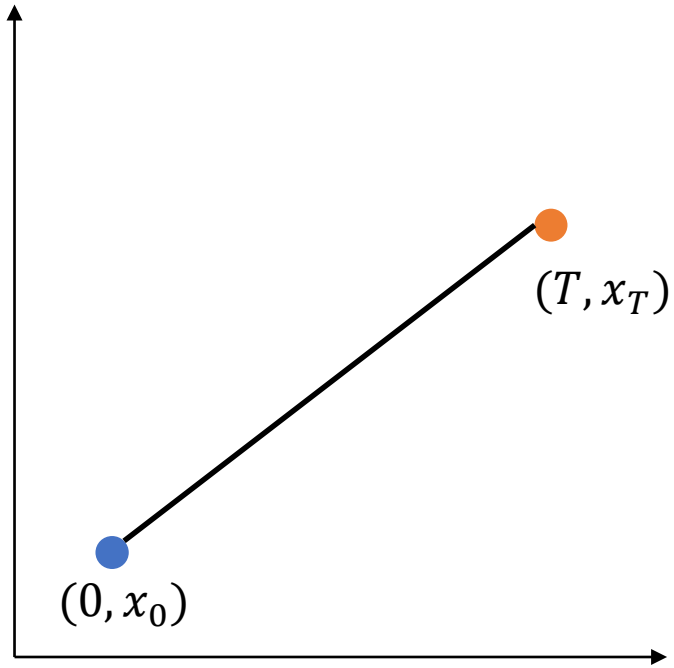
$$\boldsymbol{\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{x}}\right) - \frac{\partial \mathcal{L}}{\partial x} = 0}$$

Smooth Trajectories ($n = 1$)

$$\dot{x} = u$$

$$x^*(t) = \underset{x(t)}{\operatorname{argmin}} \int_0^T \dot{x}^2 dt \quad ; \mathcal{L} = \dot{x}^2$$

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



Smooth Trajectories $(n = 1)$

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

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

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

Shortest path is a straight line as expected!

Smooth Trajectories (n)

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

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

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

$$\dot{x}(0) = \dot{x}_0, \qquad \dot{x}(T) = \dot{x}_T$$

$$\vdots$$

$$x^{(n)}(0) = x_0^{(n)}, x^{(n)}(T) = x_T^{(n)}$$

Euler-Lagrange Equation

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

The Euler-Lagrange Equation's necessary condition for a function to be optimal is given by

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

Smooth Trajectories

$n = 1$, shortest distance (minimum velocity)
$n = 2$, minimum acceleration
$n = 3$, minimum jerk
$n = 4$, minimum snap
$n = 5$, minimum crackle
$n = 6$, minimum pop

$\mathcal{L} = \left(x^{(n)}\right)^2$ leads to $x(t) = c_0 + c_1 t + \cdots + c_{2n-1} t^{2n-1}$

Extensions to Multiple Variables

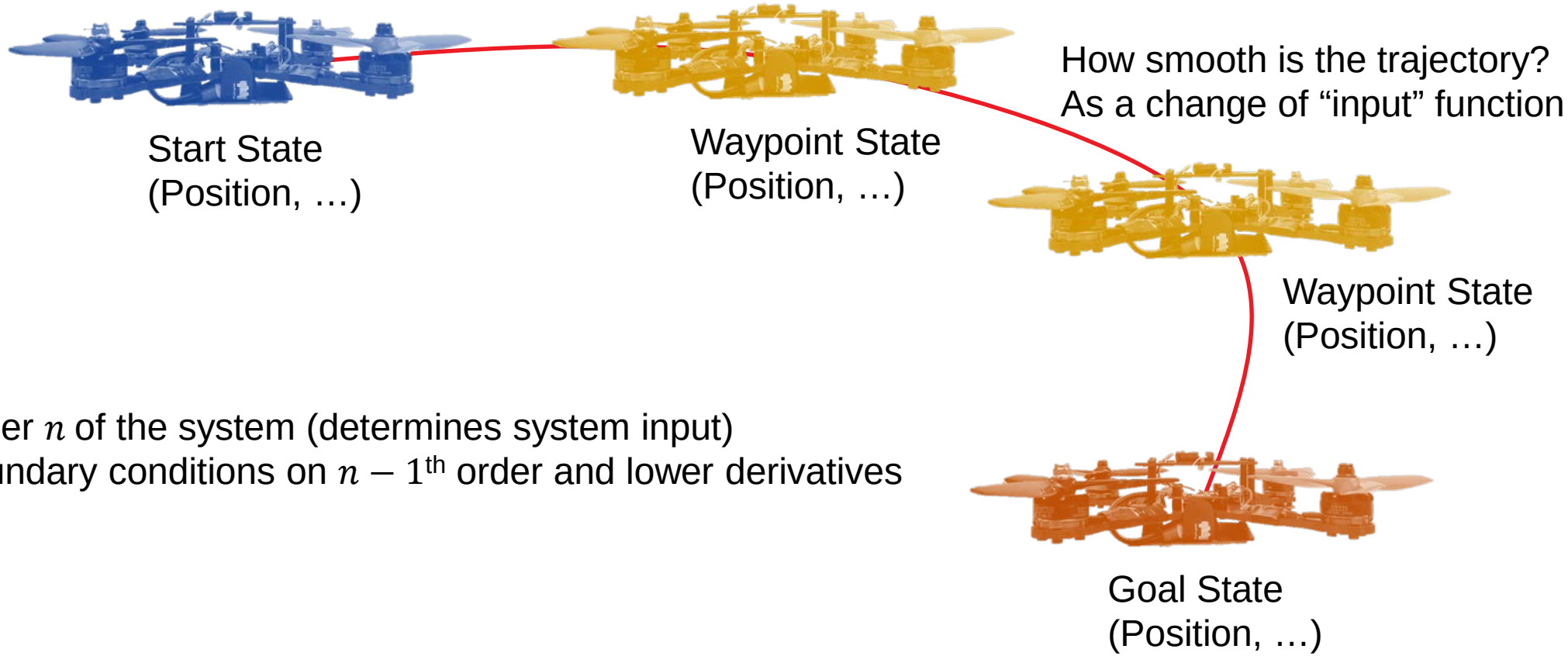
$$\left(x^*(t), y^*(t)\right) = \underset{x(t),y(t)}{\operatorname{argmin}} \int_0^T \mathcal{L}(\dot{x}, \dot{y}, x, y, t)dt$$

Euler-Lagrange Equations for each variable!

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

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

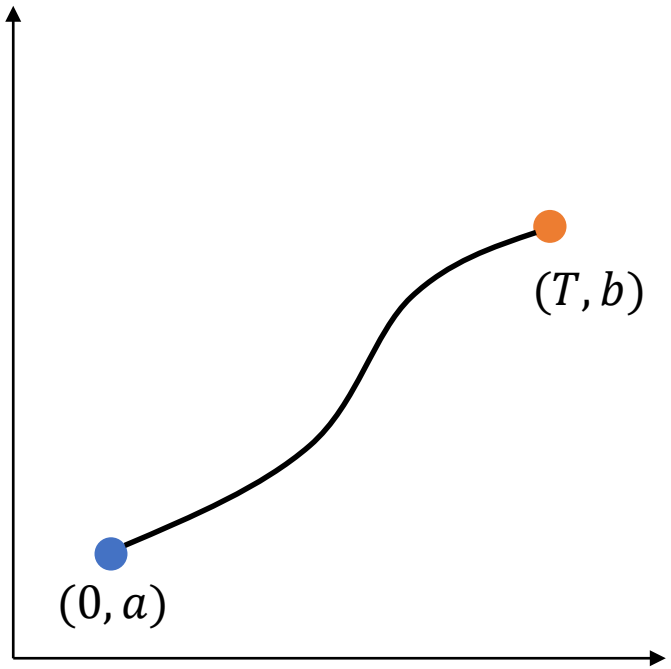
Waypoint Navigation



Order n of the system (determines system input)
Boundary conditions on $n - 1^{th}$ order and lower derivatives

Smooth 1D Trajectories

Design a trajectory $x(t) \ni x(0) = a, x(T) = b$



Minimum Acceleration Trajectory

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

Euler-Lagrange Equation gives the necessary condition satisfied by the optimal function

$$\frac{\partial \mathcal{L}}{\partial x} - \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{x}}\right) + \frac{d^2}{dt^2}\left(\frac{\partial \mathcal{L}}{\partial \ddot{x}}\right) = 0 = x^{(4)}$$

So $x(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3$

But we need to find c_i

Solving for c_i

From the problem definition we have the boundary conditions

$x(t=0)=a,\ x(t=T)=b$ (Position constraints)

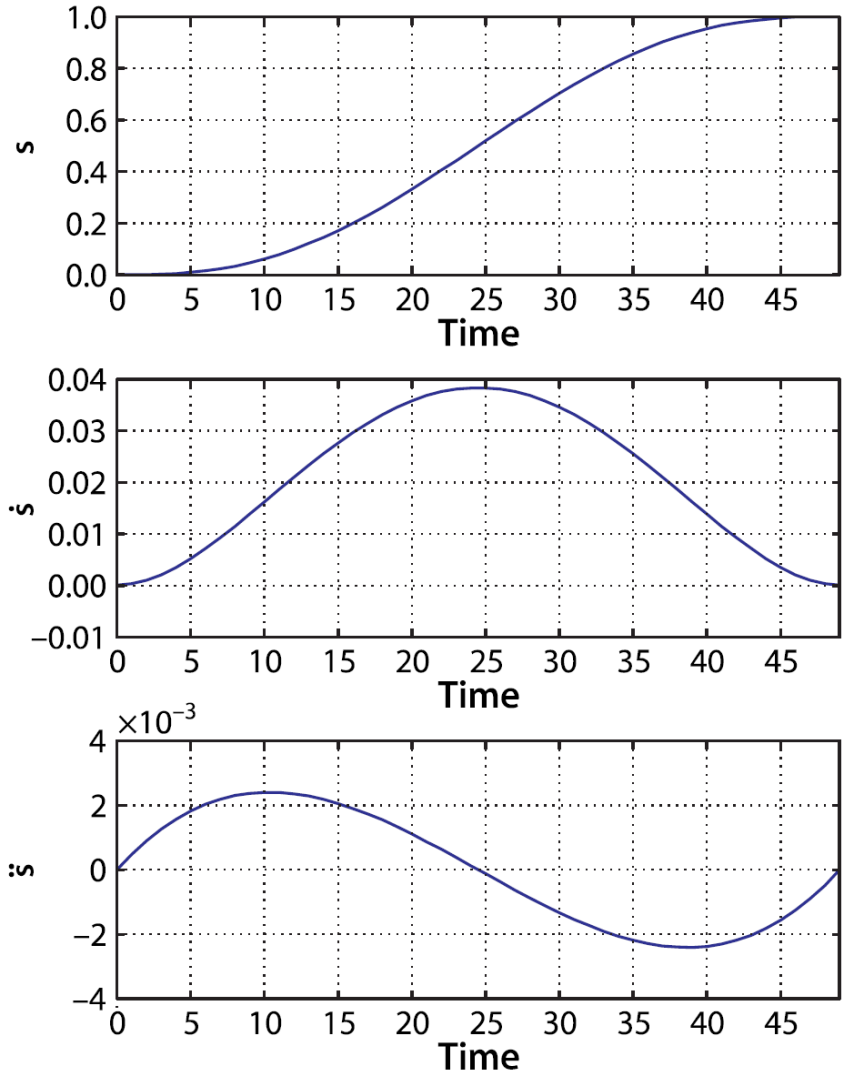
$\dot{x}(t=0)=0,\ \dot{x}(t=T)=0$ (Velocity constraints)

We need to solve the following equations to obtain the trajectory given below

$$x(t)=c_0+c_1t+c_2t^2+c_3t^3$$

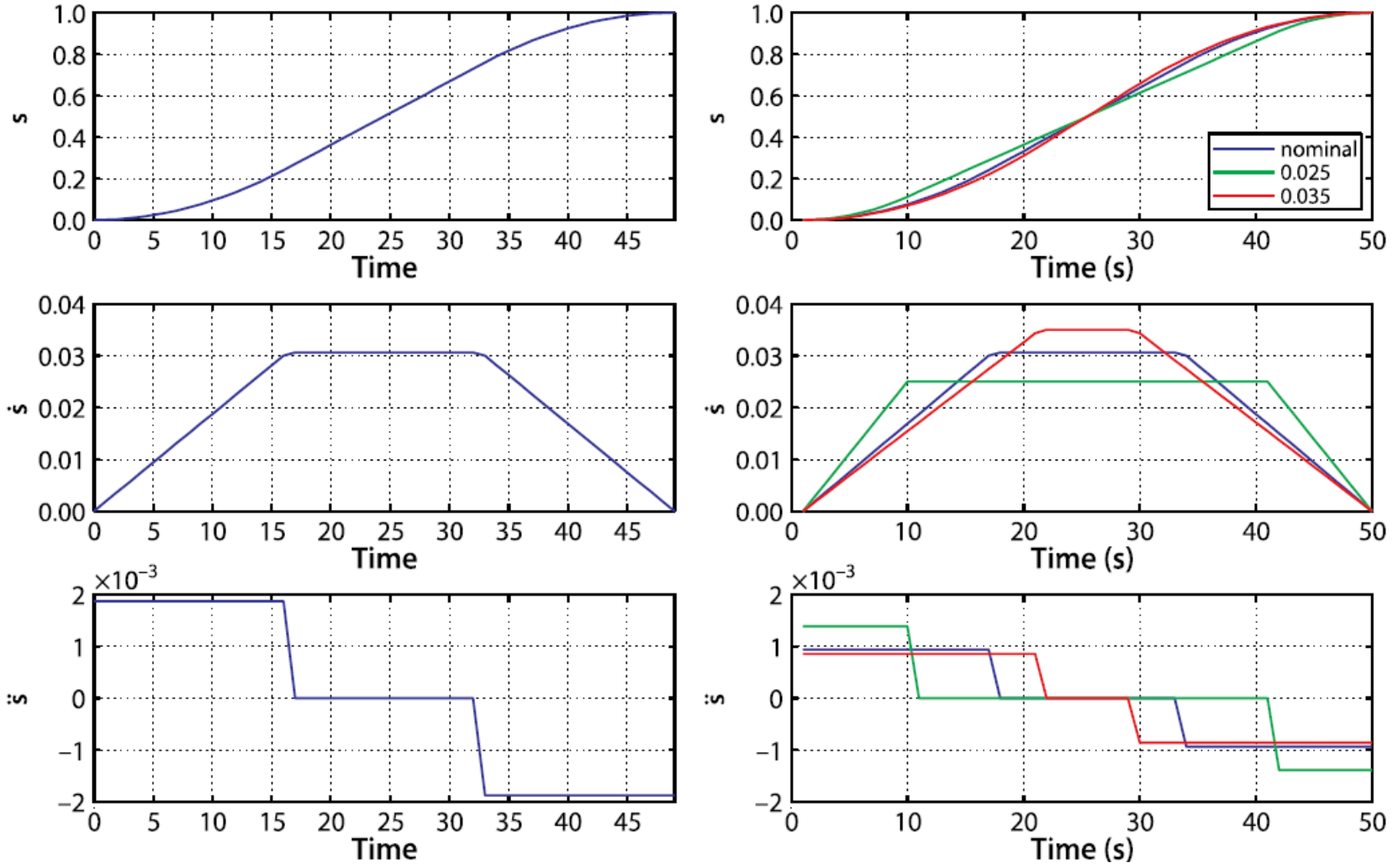
$$\begin{bmatrix} a \\ b \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & T & T^2 & T^3 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2T & 3T^2 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Motion Profiles



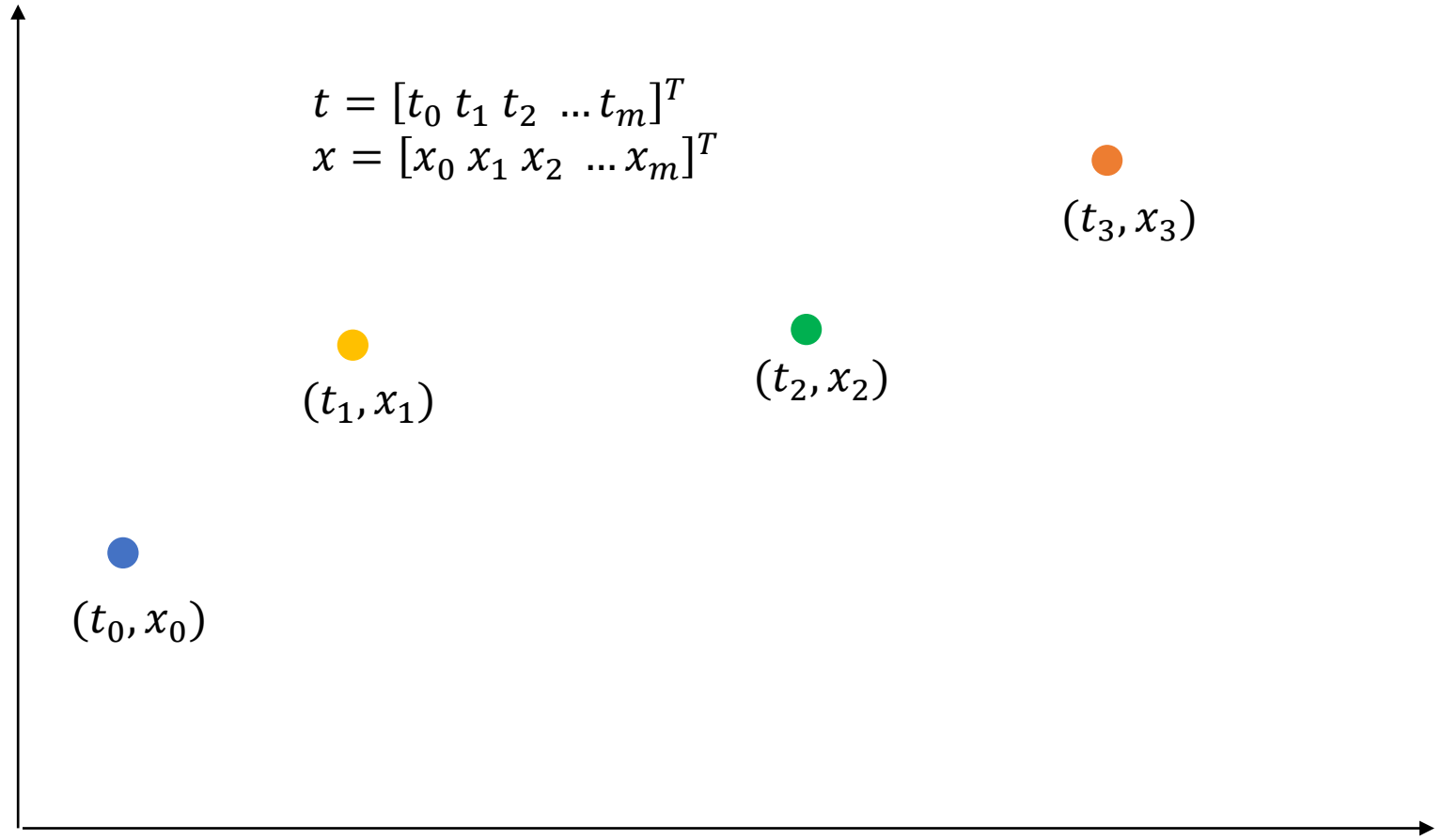
Taken from:
Peter Corke - Robotics, Vision and Control

Bang-(Coast)-Bang Trajectories



Taken from:
Peter Corke - Robotics, Vision and Control

Multi-Segment 1D Trajectories



Multi-Segment 1D Trajectories

Design a trajectory such that

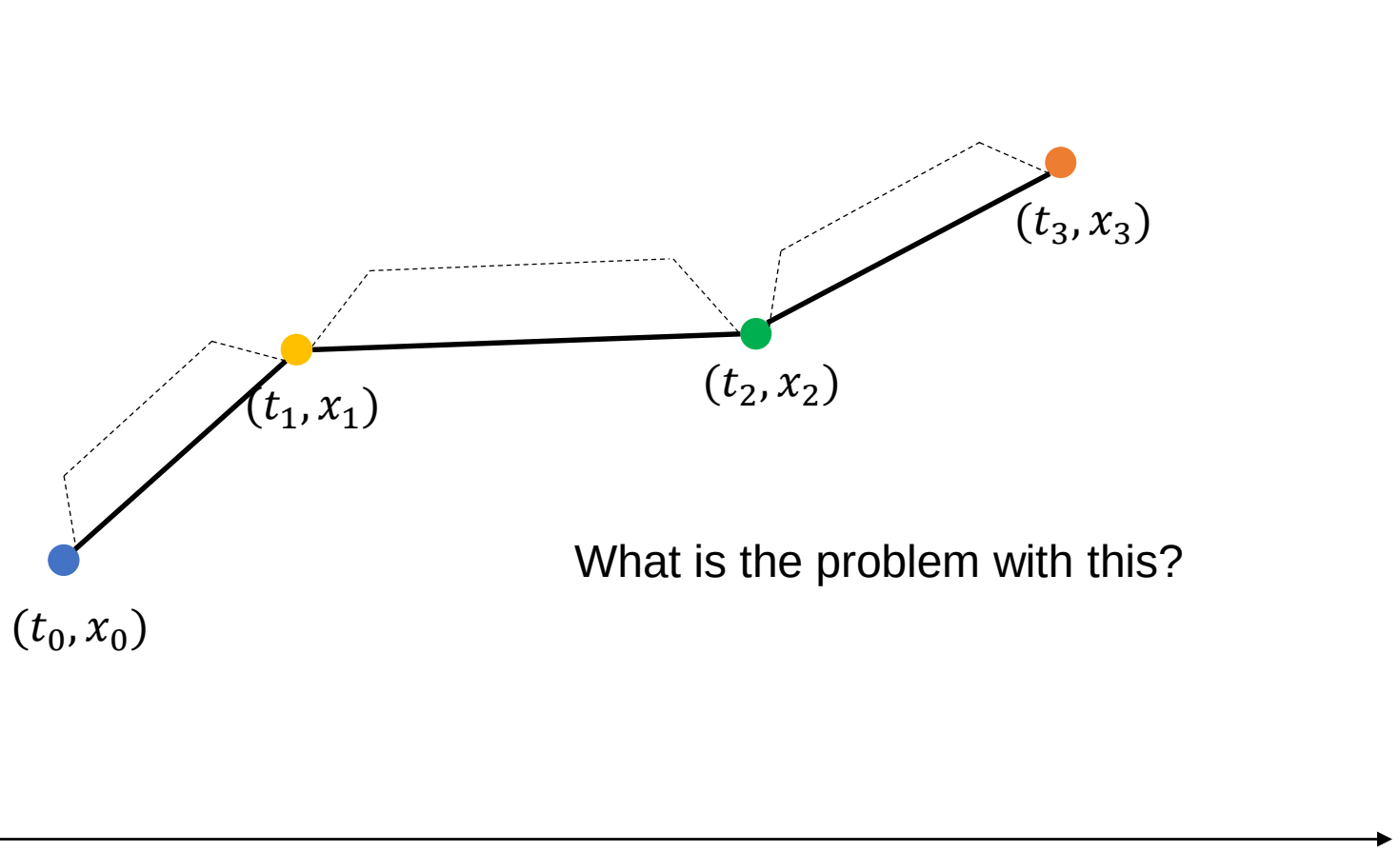
$t = [t_0\ t_1\ t_2\ \ldots t_m]^T$

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

Define a piecewise trajectory

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

Bang-(Coast)-Bang Segments



What is the problem with this?

Cubic Spline

Design a trajectory such that

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

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

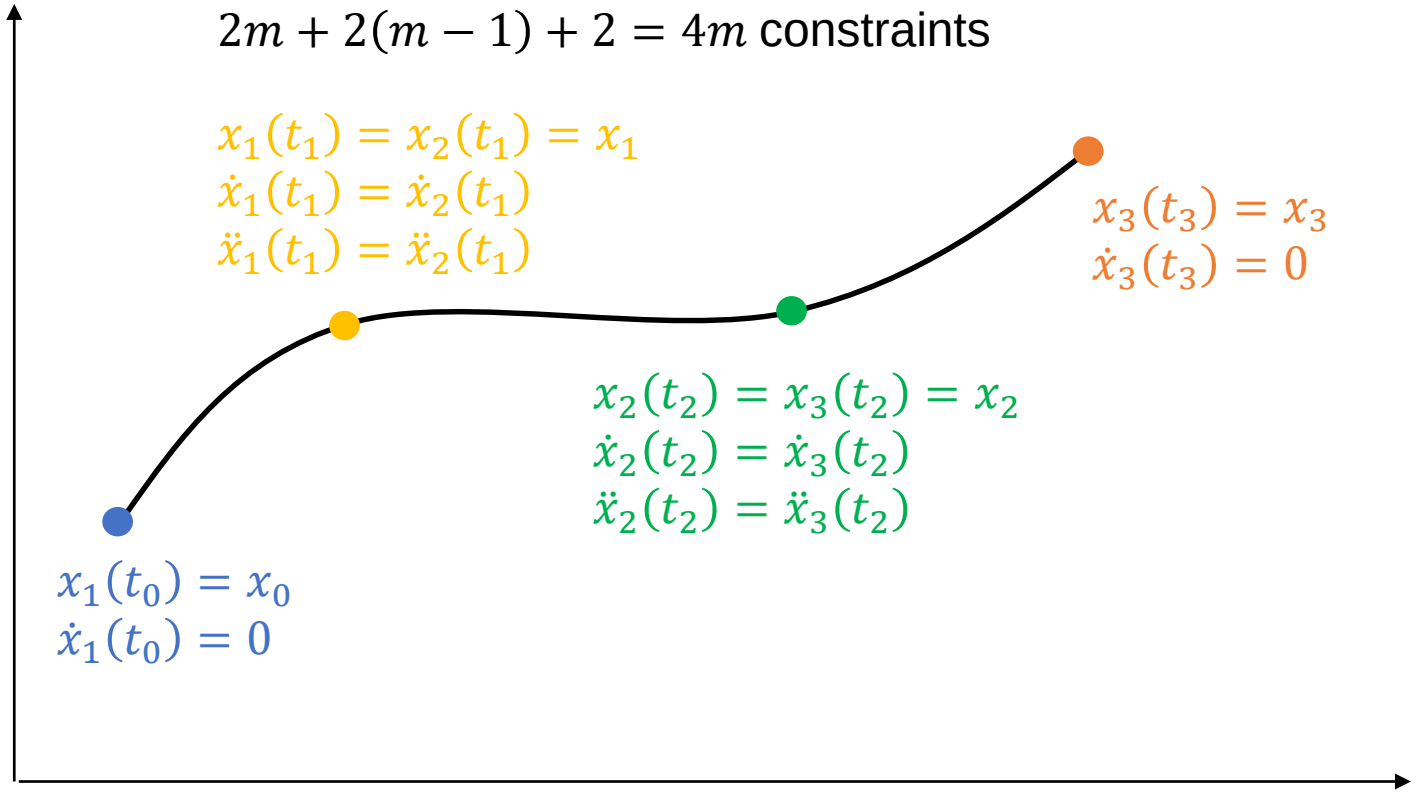
Define a piecewise trajectory

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

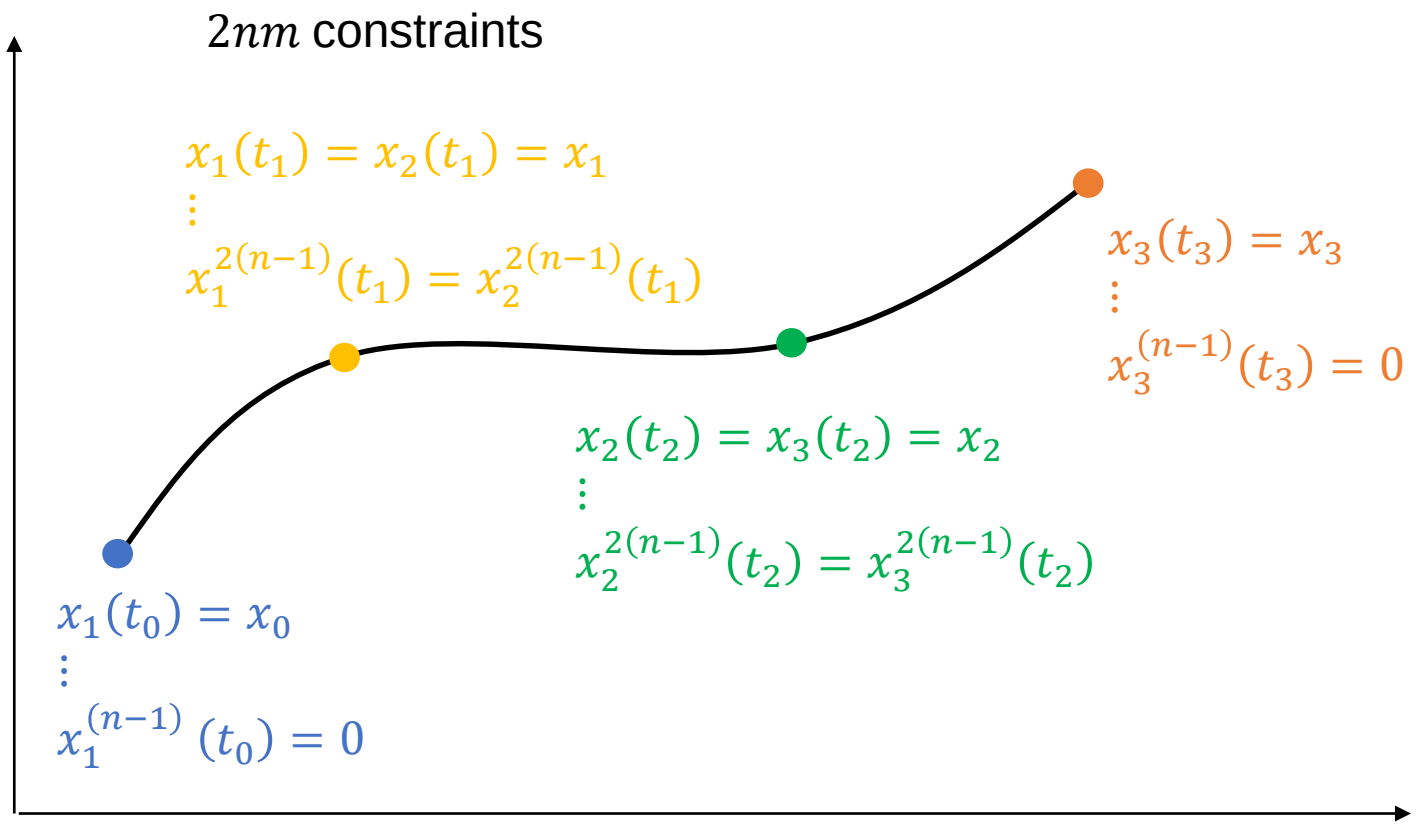
$4m$ degrees of freedom!

Cubic Spline

Second order system $\mathcal{L}(\ddot{x}, \dot{x}, x, t)$



Spline for n^{th} Order System



Multi-Segment Multi-Dimensional Trajectories

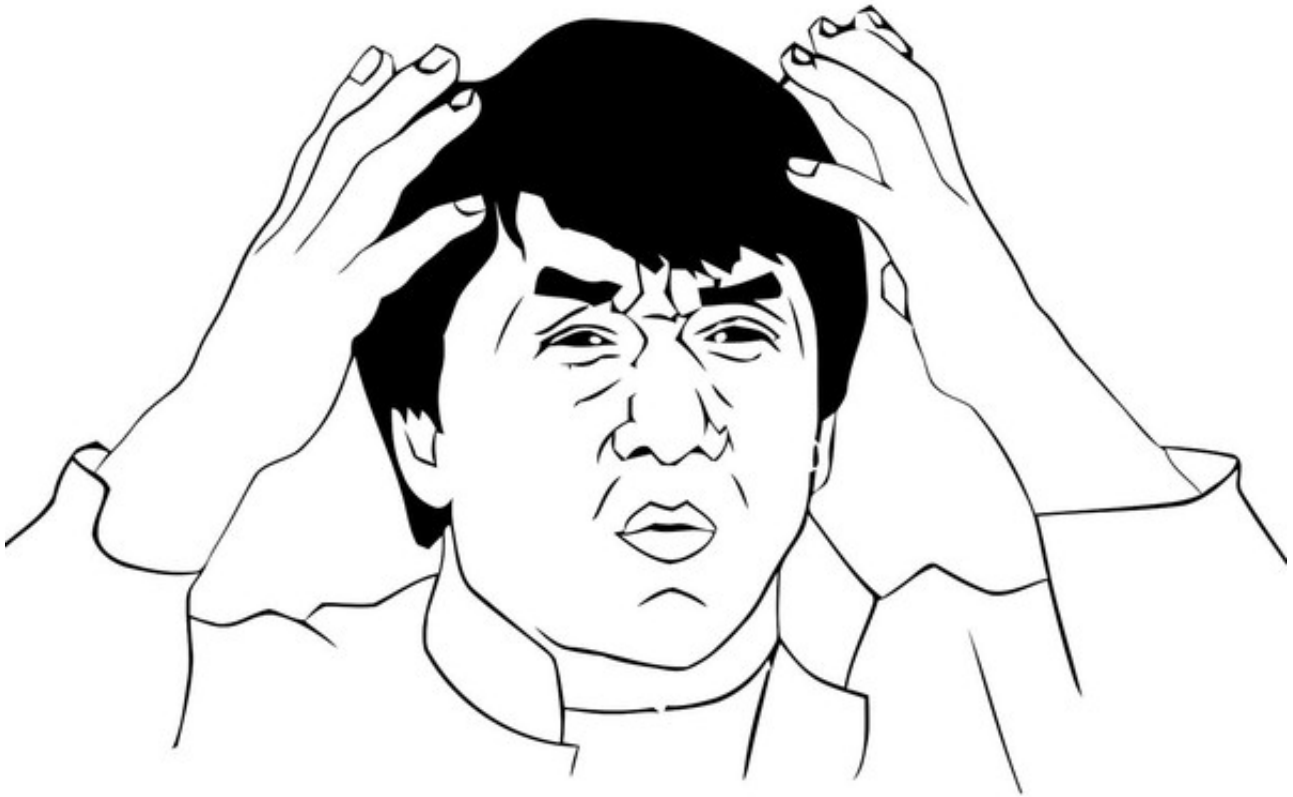
Bang-(Coast)-Bang trajectories

- Project the desired acceleration/velocity profile along the straight line connecting two waypoints

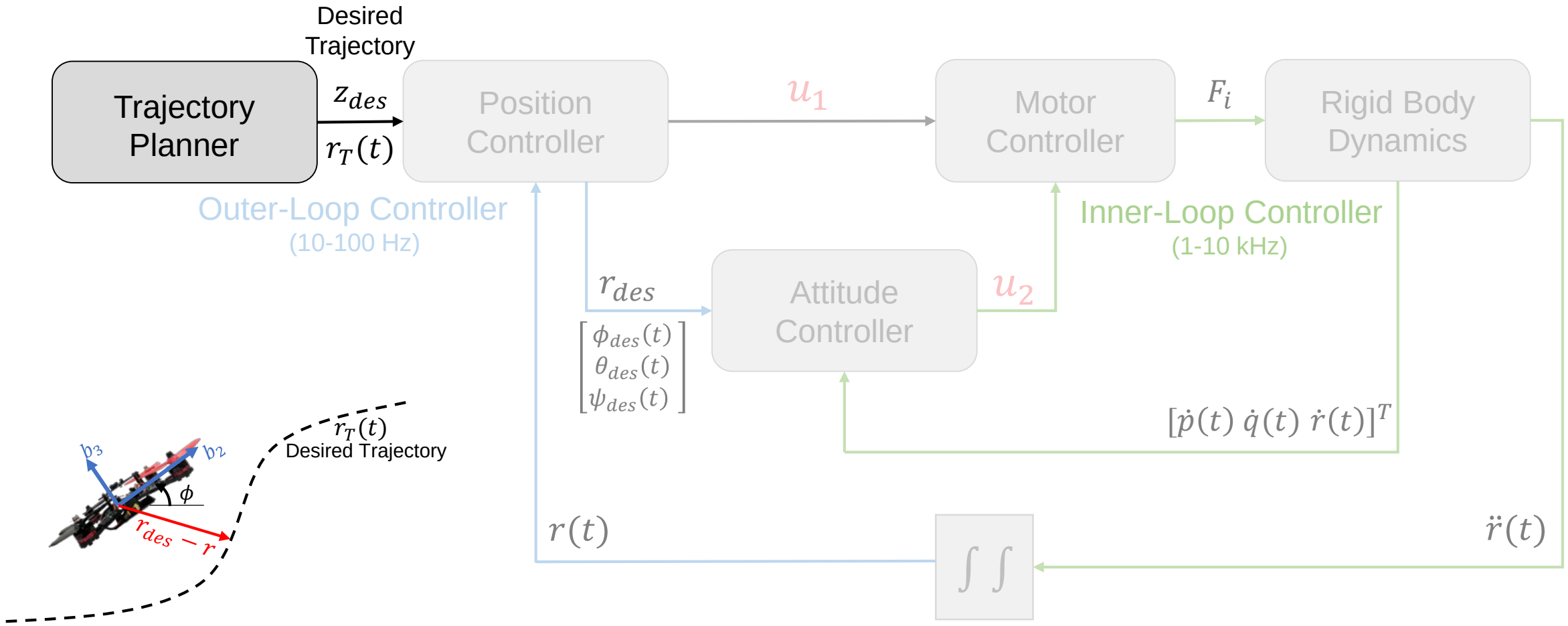
Polynomial Trajectories

- Solve each dimension independently
- Euler-Lagrange equations decouple
- Make sure time constraints are the same

How does all this go on a Quad?



Quadrotor Control



Minimum Snap Trajectory Generation

Refer to:
Mellinger, Daniel, and Vijay Kumar. "Minimum snap trajectory generation and control for quadrotors." *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011.

Go fly some autonomous quads!

References

- Robotics, Vision and Control: Fundamental Algorithms in MATLAB by Peter Corke.
- Mellinger, Daniel, and Vijay Kumar. "Minimum snap trajectory generation and control for quadrotors." *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011.