

# Motion Planning with Model Predictive Control



主讲人 赖叔朋

新加坡国立大学 Research Fellow  
主要研究方向为 Motion Planning 以及  
Nonlinear Model Predictive Control





# 目录

-  **1. Introduction**
-  **2. Linear Model Predictive Control (MPC)**
-  **3. Non-linear MPC**
-  **4. Homework**



# Introduction: Model Predictive Control

- Model
  - System model
  - Problem model
- Prediction
  - State space
  - Input space
  - Parameter space
- Control
  - The process of choosing the best policy

$$\min(p - p_d)$$
$$F = ma$$
$$\dot{p} = v$$
$$\dot{v} = a$$



# Introduction: Model

$$\min C_F(x(t_f)) + \int_{t=t_0}^{t_f} C_R(x, u) dt$$

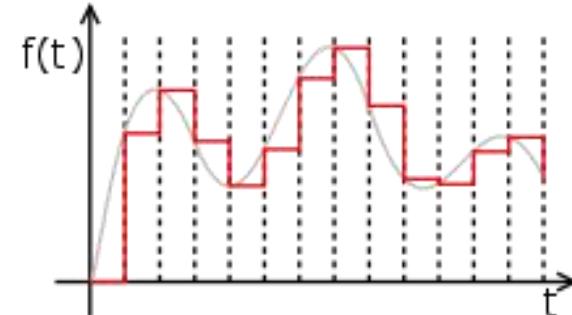
$$\begin{aligned}\dot{x} &= f(x, u) \\ g(x, u) &< 0 \\ h(x, u) &= 0 \\ x &\notin \textit{Obstacle}\end{aligned}$$



# Introduction: Parameter space

$$\min_u C_F(x(t_f)) + \int_{t=t_0}^{t_f} C_R(x, u) dt$$

- Zero order hold (direct discretization)
- Polynomial
- B-spline
- Numerical mapping
  - Jerk limited trajectory
  - Neural network based method



$$u(t) = at^3 + bt^2 + ct + d$$



# Introduction: Optimization

Searching:

Graph search

Random sampling based search

Convex optimization:

Quadratic programming

Nonconvex optimization:

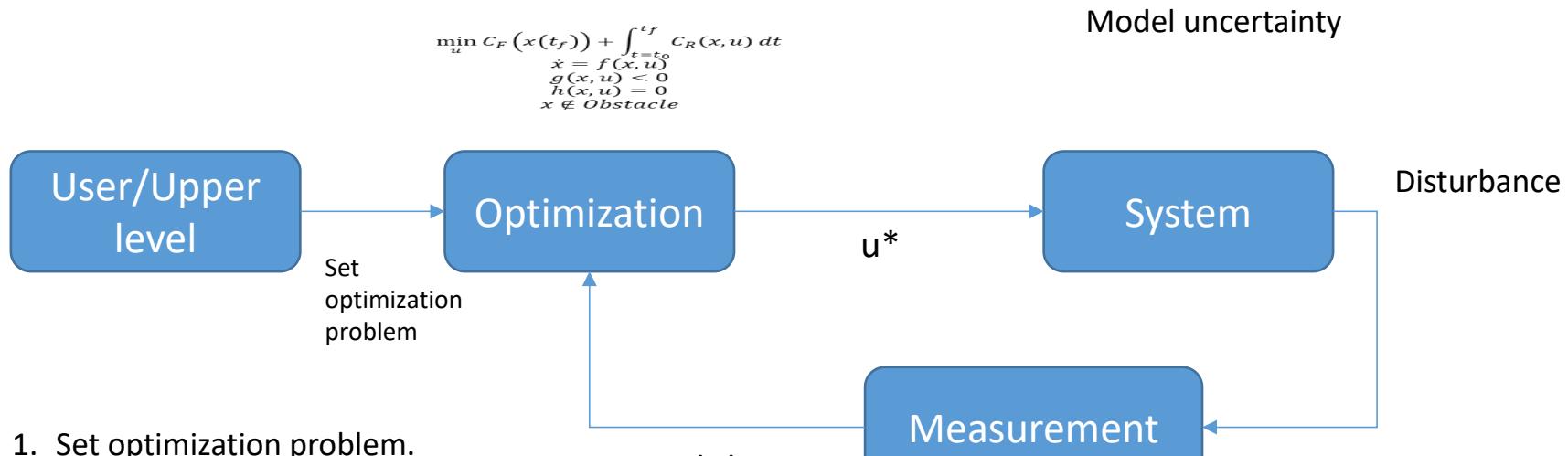
Sequential quadratic programming

**Particle swarm optimization**

*Nonconvex, nonlinear, discontinuous*



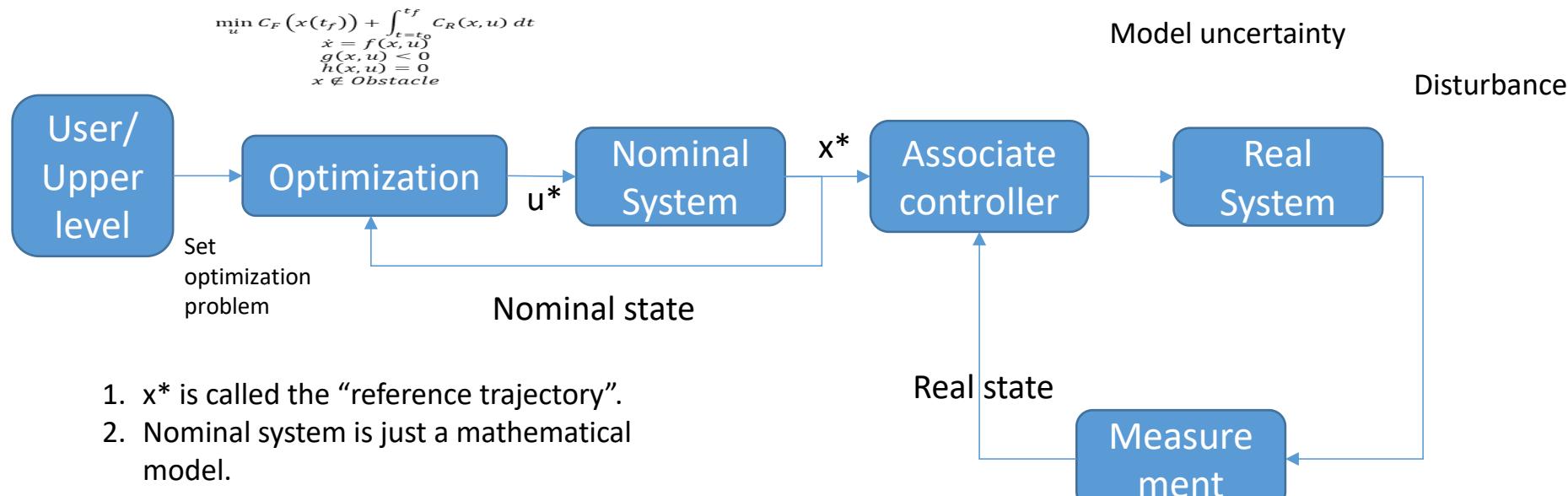
# Introduction: Control



1. Set optimization problem.
2. Take in current state measurement and finalize the optimization problem.
3. Solve the optimization and get  $u^*$ .
4. Apply  $u^*$  for a short period of time.



# Introduction: Tube based MPC



1.  $x^*$  is called the “reference trajectory”.
2. Nominal system is just a mathematical model.
3. Minimize the rate of solving the optimization problem.
4. Leave the robust tracking problem to the associate controller.

*“Tube-Based MPC: a Contraction Theory Approach”*



# Introduction: Convenient sources

- Matlab MPC toolbox: <https://www.mathworks.com/products/mpc.html>
- μAO-MPC: <http://ifatwww.et.uni-magdeburg.de/syst/muAO-MPC/>
- Acado toolkit: <https://acado.github.io/>
- YANE: <http://www.nonlinearmpc.com/>
- Multi-Parametric Toolbox 3: <https://www.mpt3.org/>



# Linear MPC: basics

Continuous model

$$\dot{p} = v$$

$$\dot{v} = a$$

$$\dot{a} = j$$

$$p_i = p(i \cdot dt)$$

$$v_i = v(i \cdot dt)$$

$$a_i = a(i \cdot dt)$$

Discrete model

$$p_{i+1} = p_i + v_i dt + \frac{1}{2} a_i dt^2 + \frac{1}{6} j_i dt^3$$

Time discretization

$$v_{i+1} = v_i + a_i dt + \frac{1}{2} j_i dt^2$$

$$a_{i+1} = a_i + j_i dt$$

$$i = [0, 2, \dots, 19]$$

$$dt = 0.2$$



# Linear MPC: basics

Discrete model

$$p_{i+1} = p_i + v_i dt + \frac{1}{2} a_i dt^2 + \frac{1}{6} j_i dt^3$$

$$\begin{aligned} v_{i+1} &= v_i + a_i dt + \frac{1}{2} j_i dt^2 \\ a_{i+1} &= a_i + j_i dt \end{aligned}$$

$$i = [0, 2, \dots, 19]$$

$$dt = 0.2$$

Linear Matrix form

Prediction model

$$P = T_p J + B_p$$

$$V = T_v J + B_v$$

$$A = T_a J + B_a$$

$$P = [p_1, p_2, p_3, \dots, p_{20}]^T$$

$$V = [v_1, v_2, v_3, \dots, v_{20}]^T$$

$$A = [a_1, a_2, a_3, \dots, a_{20}]^T$$

$$J = [j_0, j_1, j_2, \dots, j_{19}]^T$$



# Linear MPC: basics

## Prediction model

$$P = T_p J + B_p$$

$$V = T_v J + B_v$$

$$A = T_a J + B_a$$

```
1  function [Tp, Tv, Ta, Bp, Bv, Ba] = getPredictionMatrix(K,dt,p_0,v_0,a_0)
2      Ta=zeros(K);
3      Tv=zeros(K);
4      Tp=zeros(K);
5
6      for i = 1:K
7          Ta(i,1:i) = ones(1,i)*dt;
8      end
9
10     for i = 1:K
11         for j = 1:i
12             Tv(i,j) = (i-j+0.5)*dt^2;
13         end
14     end
15
16     for i = 1:K
17         for j = 1:i
18             Tp(i,j) = ((i-j+1)*(i-j)/2+1/6)*dt^3;
19         end
20     end
21
22     Ba = ones(K,1)*a_0;
23     Bv = ones(K,1)*v_0;
24     Bp = ones(K,1)*p_0;
25
26     for i=1:K
27         Bv(i) = Bv(i) + i*dt*a_0;
28         Bp(i) = Bp(i) + i*dt*v_0 + i^2/2*a_0*dt^2;
29     end
```



# Linear MPC: basics

Problem model:

Target 1: Zero position, zero velocity and zero acceleration.

Target 2: Smooth trajectory.

Optimization target 1:  $\min_J w_1 P^T P + w_2 V^\top V + w_3 A^\top A$

Optimization target 2:  $\min_J w_4 J^\top J$



# Linear MPC: basics

Optimization:

The overall optimization target:

$$\min_J w_1 P^\top P + w_2 V^\top V + w_3 A^\top A + w_4 J^\top J$$

Combine with

$$P = T_p J + B_p \quad V = T_v J + B_v \quad A = T_a J + B_a$$

We have

$$\begin{aligned} & \min_J J^\top (w_1 T_p^\top T_p + w_2 T_v^\top T_v + w_3 T_a^\top T_a + w_4 I) J + \\ & 2(w_1 B_p^\top T_p + w_2 B_v^\top T_v + w_3 B_a^\top T_a) J + \text{constant} \end{aligned}$$



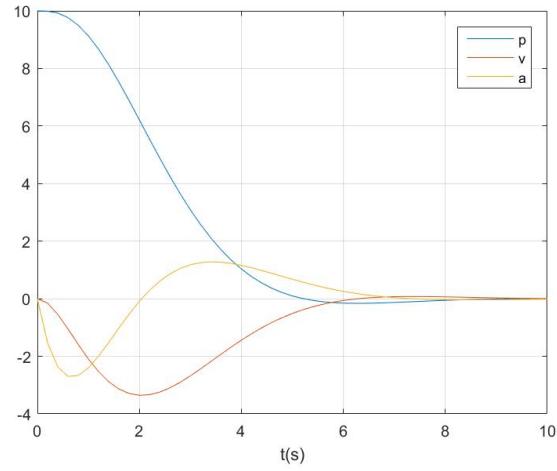
```

1 p_0 = 10;
2 v_0 = 0;
3 a_0 = 0;
4 K=20;
5 dt=0.2;
6 log=[0 p_0 v_0 a_0];
7 w1 = 1;
8 w2 = 1;
9 w3 = 1;
10 w4 = 1;
11 for t=0.2:0.2:10
12 %% Construct the prediction matrix
13 [Tp, Tv, Ta, Bp, Bv, Ba] = getPredictionMatrix(K,dt,p_0,v_0,a_0);
14
15 %% Construct the optimization problem
16 H = w4*eye(K)+w1*(Tp'*Tp)+w2*(Tv'*Tv)+w3*(Ta'*Ta);
17 F = w1*Bp'*Tp+w2*Bv'*Tv+w3*Ba'*Ta;

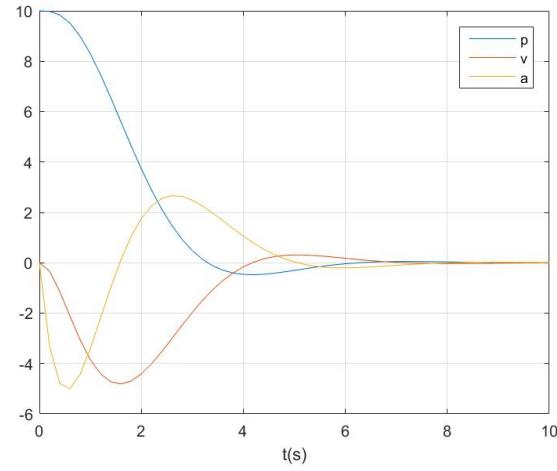
18
19 %% Solve the optimization problem
20 J = quadprog(H,F,[],[]);
21
22 %% Apply the control
23 j = J(1);
24 p_0 = p_0 + v_0*dt + 0.5*a_0*dt^2 + 1/6*j*dt^3;
25 v_0 = v_0 + a_0*dt + 0.5*j*dt^2;
26 a_0 = a_0 + j*dt;
27
28 %% Log the states
29 log = [log; t p_0 v_0 a_0];
30
31 end

```

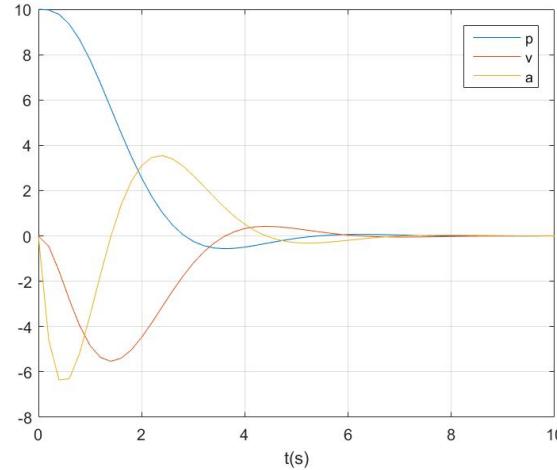




$w_1=1$   
 $w_2=1$   
 $w_3=1$   
 $w_4=1$



$w_1=5$   
 $w_2=1$   
 $w_3=1$   
 $w_4=1$



$w_1=10$   
 $w_2=1$   
 $w_3=1$   
 $w_4=1$

A plot showing Position ( $p$ ), Velocity ( $v$ ), and Acceleration ( $a$ ) versus time ( $t(s)$ ). The x-axis ranges from 0 to 16 seconds, and the y-axis ranges from -15 to 10. The initial conditions are  $p(0) = 10$ ,  $v(0) = 0$ , and  $a(0) = 0$ . The parameters are  $w_1 = 100$ ,  $w_2 = 1$ ,  $w_3 = 1$ , and  $w_4 = 1$ . The position  $p$  decreases rapidly. The velocity  $v$  starts at zero, reaches a minimum of approximately -8 at  $t \approx 1$ , and then oscillates with decreasing amplitude around zero. The acceleration  $a$  starts at zero, reaches a maximum of approximately 8 at  $t \approx 1.5$ , and then decays towards zero.

$w_1=100$   
 $w_2=1$   
 $w_3=1$   
 $w_4=1$

# Linear MPC: basics

Optimization problem

$$\begin{aligned} \min_J J^\top & (w_1 T_p^\top T_p + w_2 T_v^\top T_v + w_3 T_a^\top T_a + w_4 I) J + \\ & 2(w_1 B_p^\top T_p + w_2 B_v^\top T_v + w_3 B_a^\top T_a) J + \text{constant} \end{aligned}$$

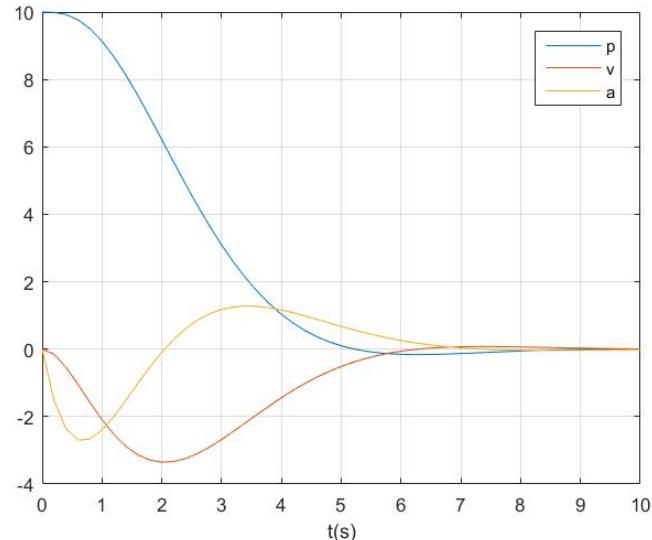
$$\text{Let } H = (w_1 T_p^\top T_p + w_2 T_v^\top T_v + w_3 T_a^\top T_a + w_4 I)$$

$$F = (w_1 B_p^\top T_p + w_2 B_v^\top T_v + w_3 B_a^\top T_a)$$

$$\min_J J^\top H J + 2FJ$$



$$J = -H^{-1}F$$



```
%% Solve the optimization problem  
J = -H\F';
```



# Linear MPC: hard constraints

Constraints

$$-1 \leq v_i \leq 1, \forall i \in \{1, 2, 3 \dots, 20\}$$

$$-1 \leq a_i \leq 1, \forall i \in \{1, 2, 3, \dots, 20\}$$

Matrix form:

$$-1_{20 \times 1} \leq V \leq 1_{20 \times 1}$$

$$-1_{20 \times 1} \leq A \leq 1_{20 \times 1}$$

$$V = T_v J + B_v$$

$$A = T_a J + B_a$$

Final form:

$$-1_{20 \times 1} \leq T_v J + B_v \leq 1_{20 \times 1}$$

$$-1_{20 \times 1} - B_v \leq T_v J \leq 1_{20 \times 1} - B_v$$

$$-1_{20 \times 1} - B_a \leq T_a J \leq 1_{20 \times 1} - B_a$$

Less equal form:

$$T_v J \leq 1_{20 \times 1} - B_v$$

$$-T_v J \leq 1_{20 \times 1} + B_v$$

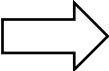
$$T_a J \leq 1_{20 \times 1} - B_a$$

$$-T_a J \leq 1_{20 \times 1} + B_a$$



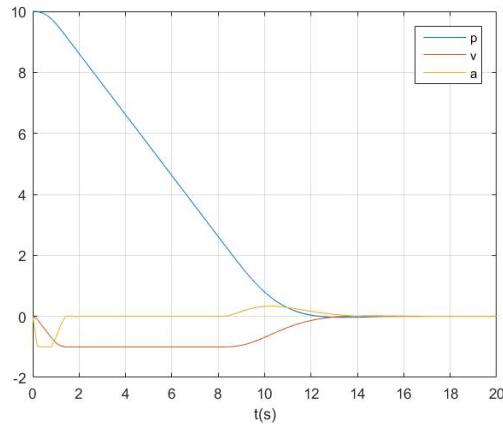
# Linear MPC: hard constraints

$$\begin{aligned} T_v J &\leq 1_{20 \times 1} - B_v \\ -T_v J &\leq 1_{20 \times 1} + B_v \\ T_a J &\leq 1_{20 \times 1} - B_a \\ -T_a J &\leq 1_{20 \times 1} + B_a \end{aligned}$$

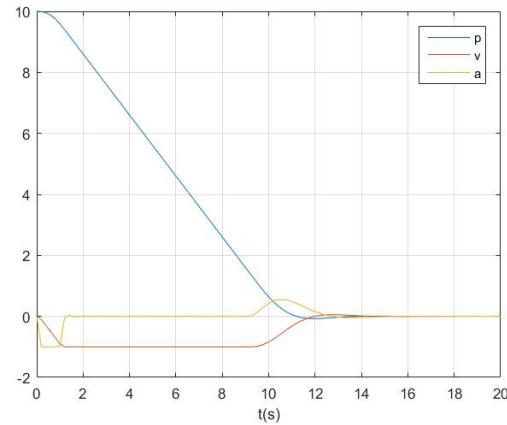


```
A = [Tv; -Tv; Ta; -Ta];
b = [ones(20,1)-Bv; ones(20,1)+Bv; ones(20,1)-Ba; ones(20,1)+Ba];
J = quadprog(H, F, A, b);
```

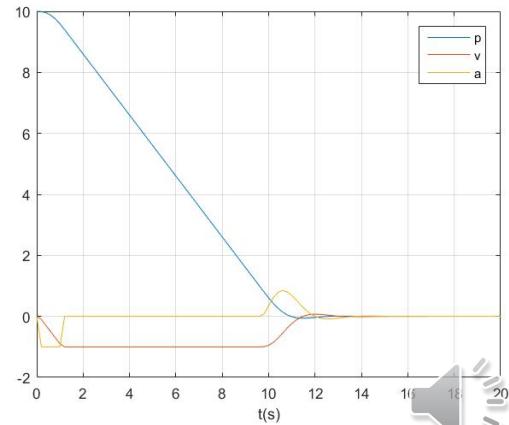
w1=1, w2=1, w3=1, w4=1



w1=10, w2=1, w3=1, w4=1



w1=100, w2=1, w3=1, w4=1



# Linear MPC: soft constraints

- What if the velocity and acceleration constraints are inevitably violated?

$$v_0 = 3m/s$$

$$-1 \leq v_i \leq 1, \forall i \in \{1, 2, 3 \dots, 20\}$$

$$-1 \leq a_i \leq 1, \forall i \in \{1, 2, 3, \dots, 20\}$$

The solver will report no solution!!

The controller won't know what to do.



# Linear MPC: soft constraints

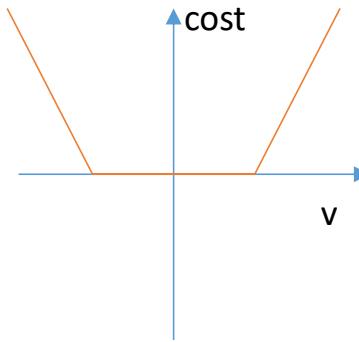
Add a penalty function to the optimization target

$$\min_J w_1 P^\top P + w_2 V^\top V + w_3 A^\top A + w_4 J^\top J + S(V)$$

$$S(V) = \sum_{i=1}^{20} s(v_i)$$

$$s(v_i) = \begin{cases} 0 & \text{if } \|v_i\| \leq 1 \\ M \cdot (\|v_i\| - 1) & \text{else} \end{cases}$$

*M is just a large positive number*



# Linear MPC: soft constraints

Original constraints

$$\begin{aligned}T_v J &\leq 1_{20 \times 1} - B_v \\-T_v J &\leq 1_{20 \times 1} + B_v \\T_a J &\leq 1_{20 \times 1} - B_a \\-T_a J &\leq 1_{20 \times 1} + B_a\end{aligned}$$

Add the slack variable L

$$\begin{aligned}-T_v J &\leq 1_{20 \times 1} + B_v + L \\-L &\leq 0 \\L &= [l, l_2, l_3, \dots, l_{20}]^\top\end{aligned}$$

New optimization target

$$\min_{J,L} w_1 P^T P + w_2 V^\top V + w_3 A^\top A + w_4 J^\top J + w_5 L^\top L$$

Set new programming variable as  $J = \begin{bmatrix} J \\ L \end{bmatrix}$

Then all the H, F, A, b matrix shall be adjusted accordingly

```
%% Construct the prediction matrix
[Tp, Tv, Ta, Bp, Bv, Ba] = getPredictionMatrix(K, dt, p_0, v_0, a_0);

%% Construct the optimization problem
H = blkdiag(w4*eye(K)+w1*(Tp'*Tp)+w2*(Tv'*Tv)+w3*(Ta'*Ta), w5*eye(K));
F = [w1*Bp'*Tp+w2*Bv'*Tv+w3*Ba'*Ta zeros(1,K)];

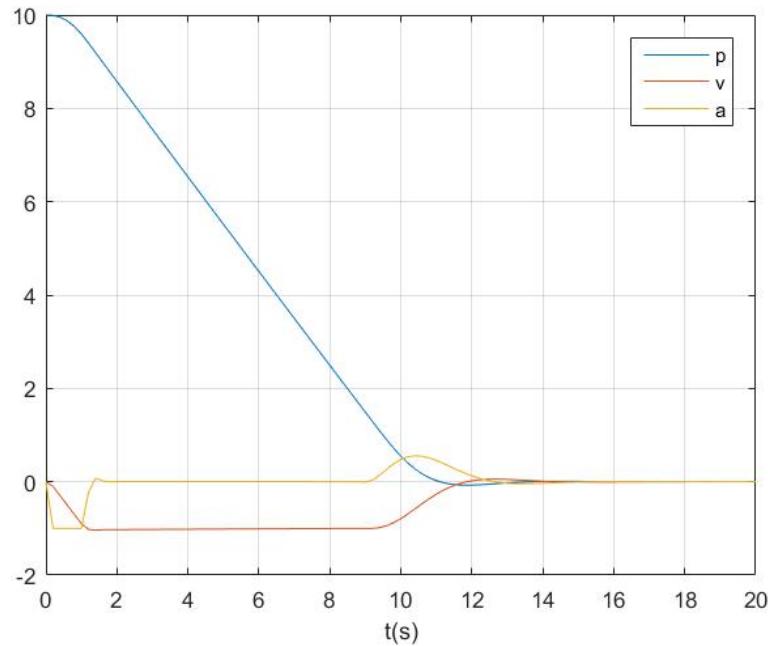
A = [Tv zeros(K); -Tv -eye(K); Ta zeros(K); -Ta zeros(K); zeros(size(Ta)) -eye(K)];
b = [ones(20,1)-Bv; ones(20,1)+Bv; ones(20,1)-Ba; ones(20,1)+Ba; zeros(K,1)];

%% Solve the optimization problem
J = quadprog(H, F, A, b);
```

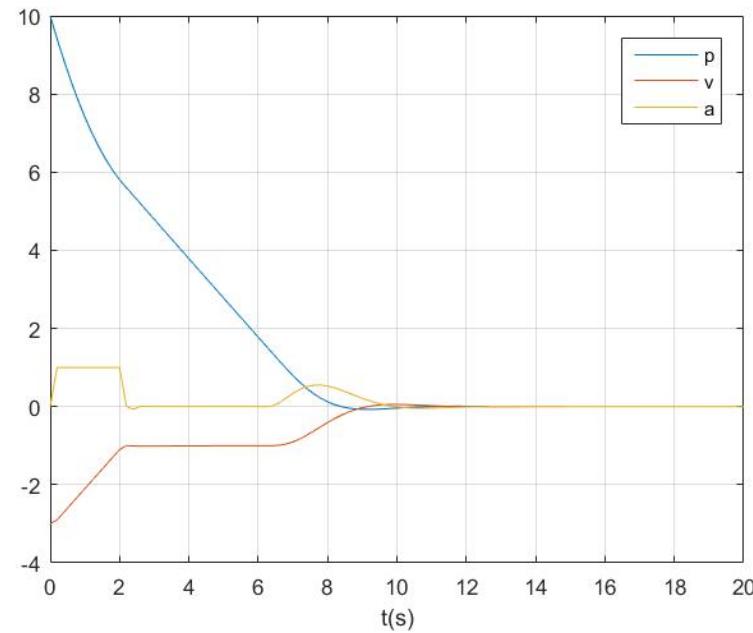


# Linear MPC: soft constraints

w1=10, w2=1, w3=1, w4=1, w5 = 1e4



w1=10, w2=1, w3=1, w4=1, w5 = 1e4



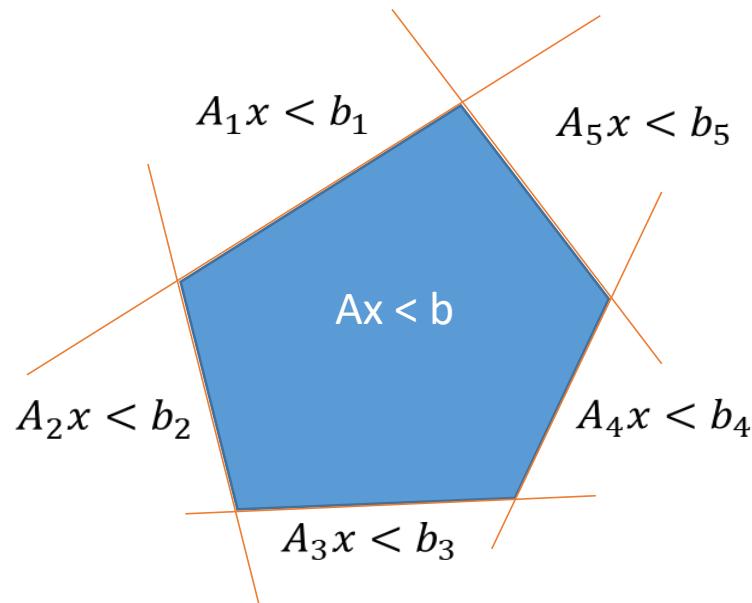
# Linear MPC: soft constraints

- State constraints -> use soft constraints
  - They are effected by measurement noise and disturbances
- Input constraints -> use hard constraints
  - They can be varied arbitrarily, and their violation might harm the physical system.



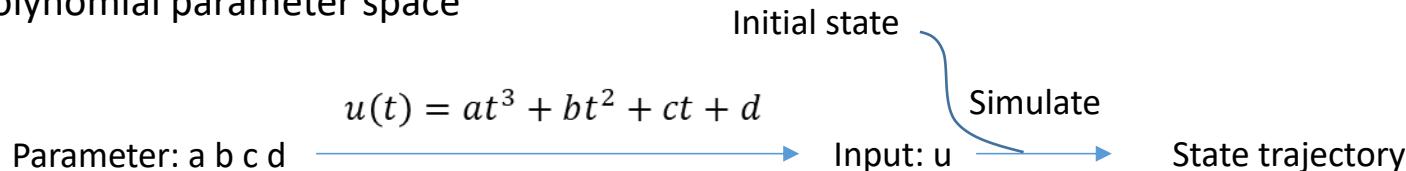
# Linear MPC: limitation

- It usually requires a linear model, or the model can be reasonably linearized (adaptive MPC).
- Obstacle constraints are usually non-convex by nature.



# Parameter space: Boundary constrained motion primitives (BSCP)

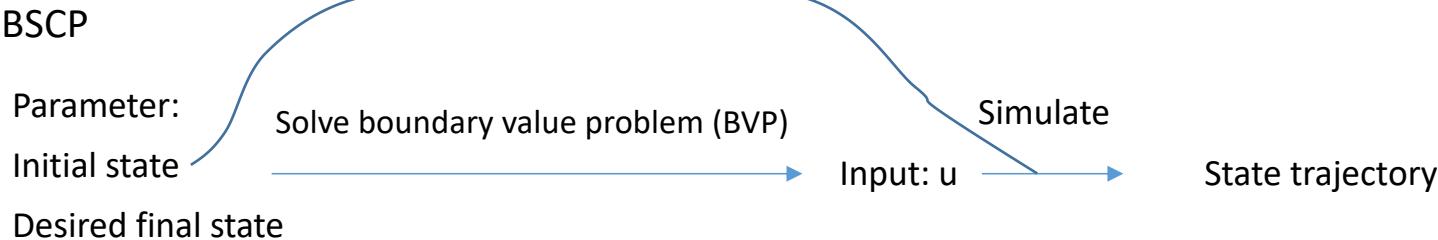
- Polynomial parameter space



- Zero order hold parameter space

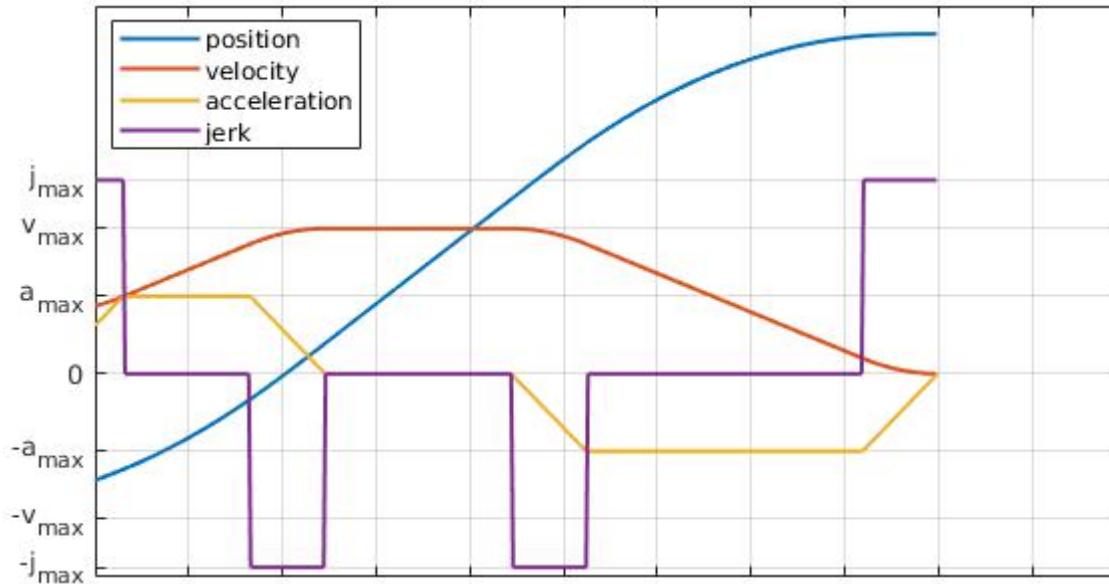


- BSCP



# Parameter space: Jerk limited trajectory

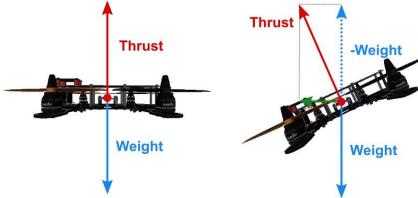
- Not only limited jerk, but also limited acceleration and velocity.
- We will refer it as JLT later.



The jerk limited trajectory for a single axis



# Parameter space: Jerk limited trajectory



## Inner loop constraints:

The limited body rate  $\omega_{\max}$  and total thrust  $f_{\max}$

## Final constraints:

The limited acceleration and jerk

$$-v_{\max} \leq v(t) \leq v_{\max},$$

$$-a_{\max} \leq a(t) \leq a_{\max},$$

$$-u_{j\max} \leq u_j(t) \leq u_{j\max},$$

Sufficient condition

Limit the velocity for safety

## Outer loop constraints:

The limited acceleration and jerk

$$\sqrt{\ddot{x}^2 + \ddot{y}^2 + \ddot{z}^2} \leq (\ddot{z}_{\min} + g)\omega_{\max}$$

$$\sqrt{\ddot{x}^2 + \ddot{y}^2 + (\ddot{z} + g)^2} \leq \frac{f_{\max}}{m_v}$$

$$\ddot{z} \geq \ddot{z}_{\min} \geq \frac{f_{\min}}{m_v} - g$$

Decoupling

Sufficient condition

## Single axis constraints:

The limited acceleration and jerk

$$-a_{\max} \leq a(t) \leq a_{\max},$$

$$-u_{j\max} \leq u_j(t) \leq u_{j\max},$$



# Parameter space: Jerk limited trajectory

The jerk limited trajectory problem considers a system characterized by

$$\begin{pmatrix} \dot{p} \\ \dot{v} \\ \dot{a} \end{pmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{pmatrix} p \\ v \\ a \end{pmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u_j$$

with  $-v_{\max} \leq v(t) \leq v_{\max}$ ,  $-a_{\max} \leq a(t) \leq a_{\max}$ ,  $-u_{j\max} \leq u_j(t) \leq u_{j\max}$

The traditional Time Optimal Control (TOC) or bang-bang control problem considers

$$\dot{x}(t) = Ax(t) + Bu(t)$$

with

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ a \end{bmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} y \\ \dot{y} \end{pmatrix} = \begin{pmatrix} y \\ \dot{y} \end{pmatrix} \quad (4.6)$$

Note that  $v = \dot{y}$  is the velocity of the system. Let the control input be constrained as follows:

$$|u(t)| \leq u_{\max}$$



# Parameter space: Jerk limited trajectory

## Problem:

From arbitrary state to a desired velocity.

Subject to:

$$v(0) = v_0, \quad v(T) = v_{\text{ref}}$$

$$a(0) = a_0, \quad a(T) = 0$$

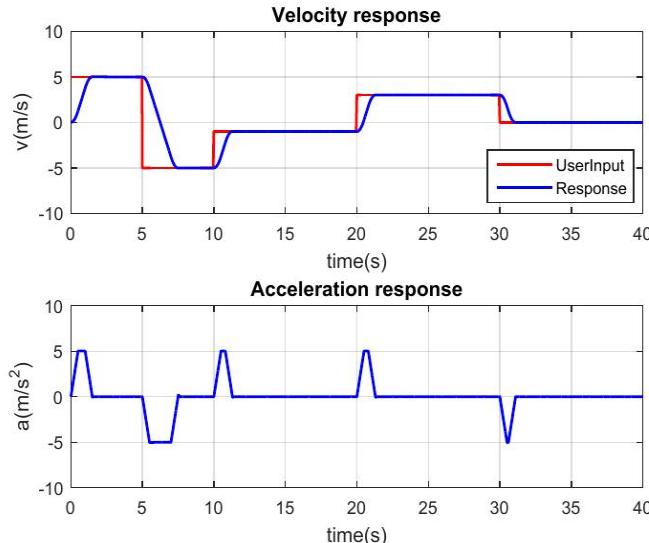
$$\dot{v}(t) = a(t)$$

$$\dot{a}(t) = u_j(t)$$

$$-a_{\max} \leq a(t) \leq a_{\max}, \quad \forall t \in [0, T]$$

$$-u_{j\max} \leq u_j(t) \leq u_{j\max}, \quad \forall t \in [0, T]$$

## An Example:



It is shown the trajectory consists of at most three segments with  $u = \pm u_{j\max}$  and 0.



# Parameter space: Jerk limited trajectory

Intuition on the solution:

Covered area (acc) =  $\Delta$  Velocity

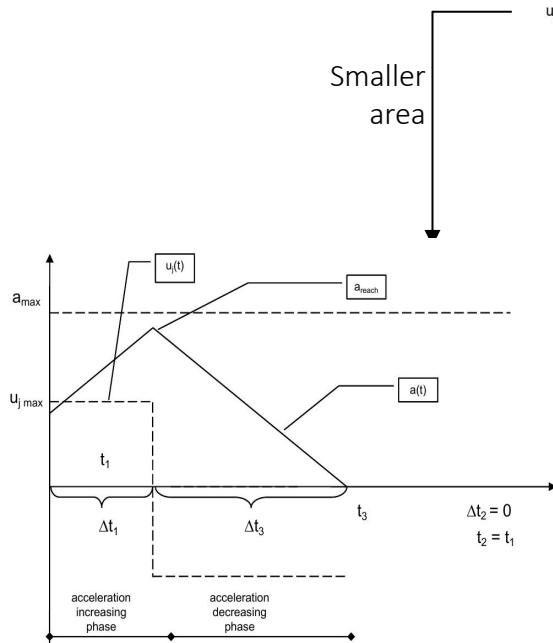


Figure 4.13: Wedge acceleration profile

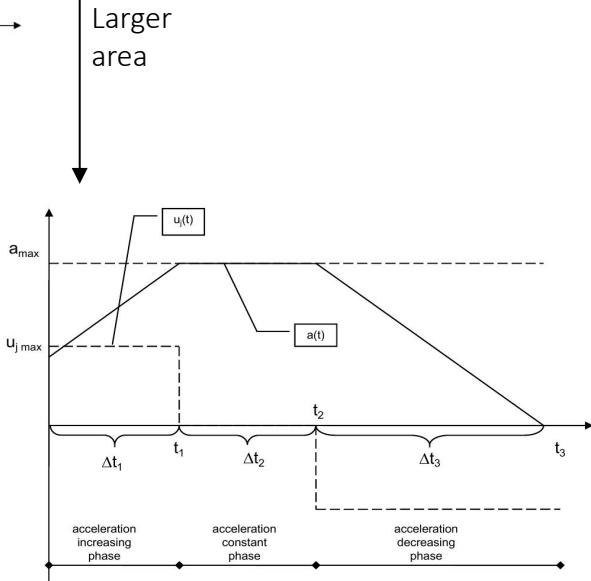
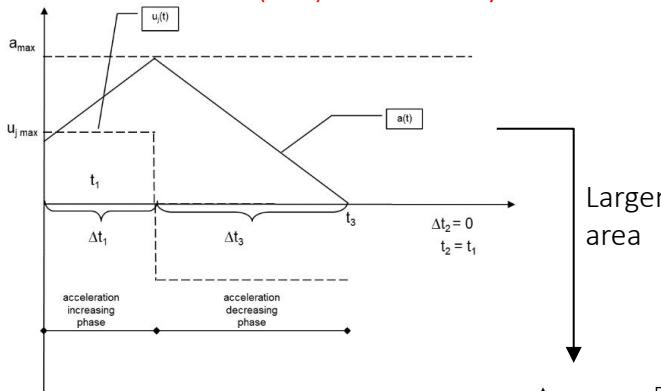


Figure 4.12: Trapezoidal acceleration profile



# Parameter space: Jerk limited trajectory

## The Problem:

From arbitrary state to a desired point subject to:

$$p(0) = p_0, \quad p(T) = p_{\text{ref}}$$

$$v(0) = v_0, \quad v(T) = 0$$

$$a(0) = a_0, \quad a(T) = 0$$

$$\dot{p}(t) = v(t)$$

$$\dot{v}(t) = a(t)$$

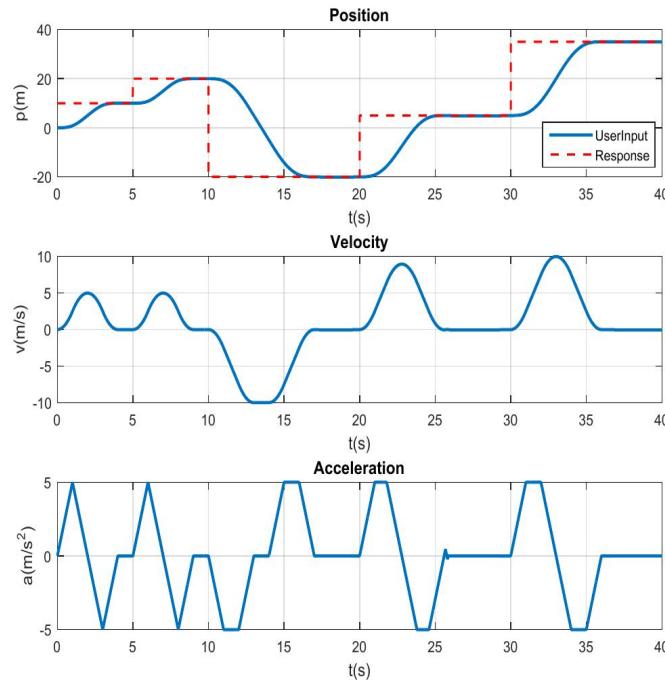
$$\dot{a}(t) = u_j(t)$$

$$-v_{\max} \leq v(t) \leq v_{\max}, \quad \forall t \in [0, T]$$

$$-a_{\max} \leq a(t) \leq a_{\max}, \quad \forall t \in [0, T]$$

$$-u_{j\max} \leq u_j(t) \leq u_{j\max}, \quad \forall t \in [0, T]$$

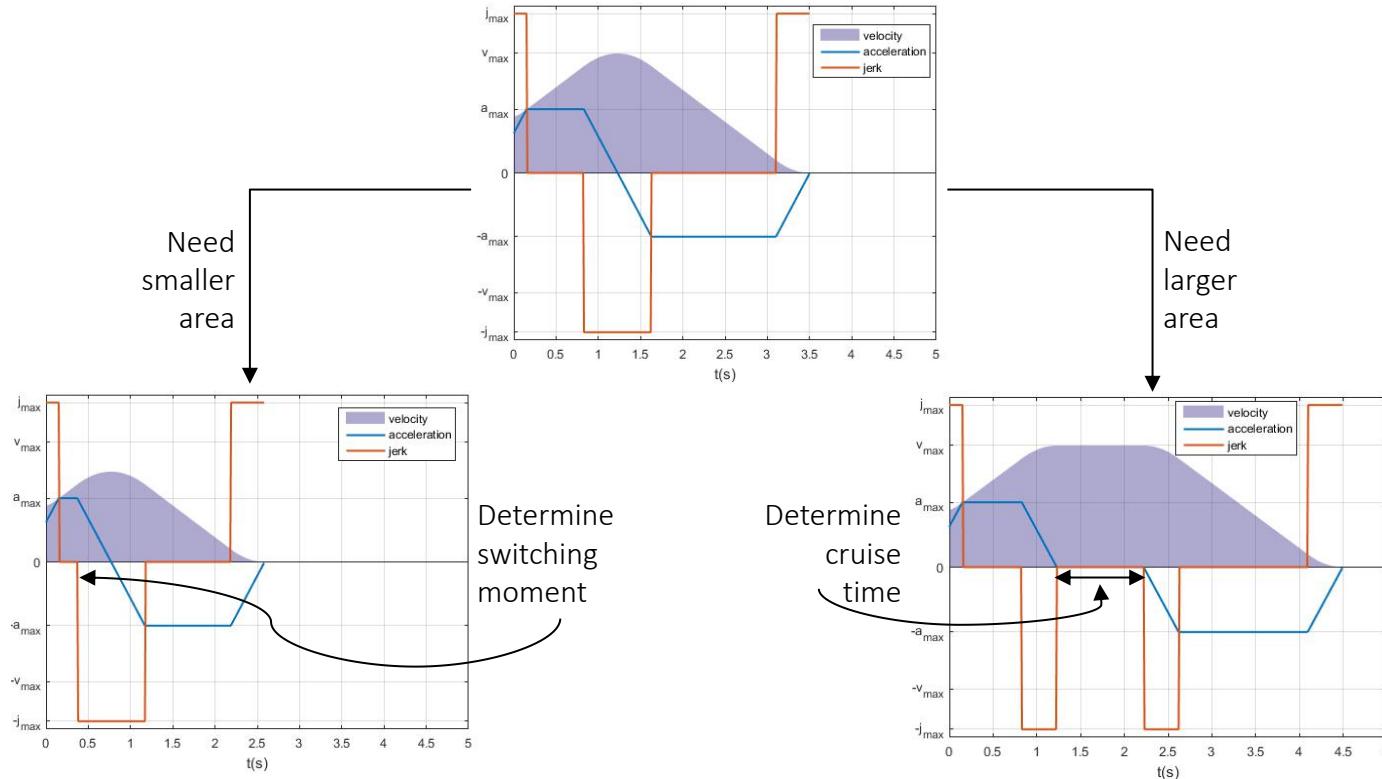
## An Example:



# Parameter space: Jerk limited trajectory

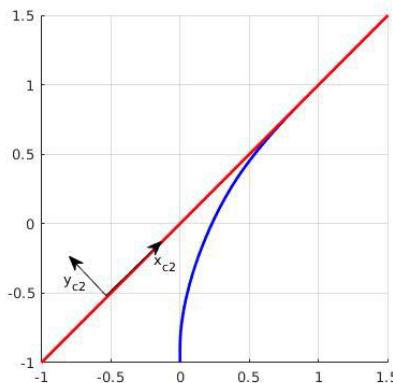
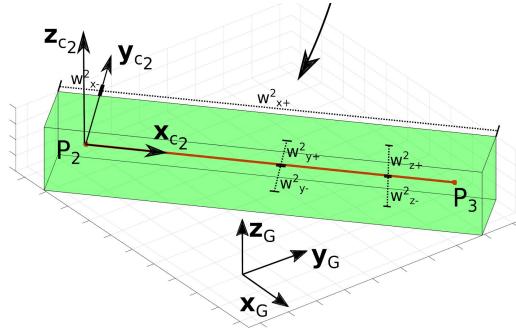
Intuition on the solution:

Covered area (vel) =  $\Delta$  Position



# Parameter space: Jerk limited trajectory

For the trajectory to "converge" to a given line segment.



On the y and z axes:

- Set the position set-point to zero.

On the x axis:

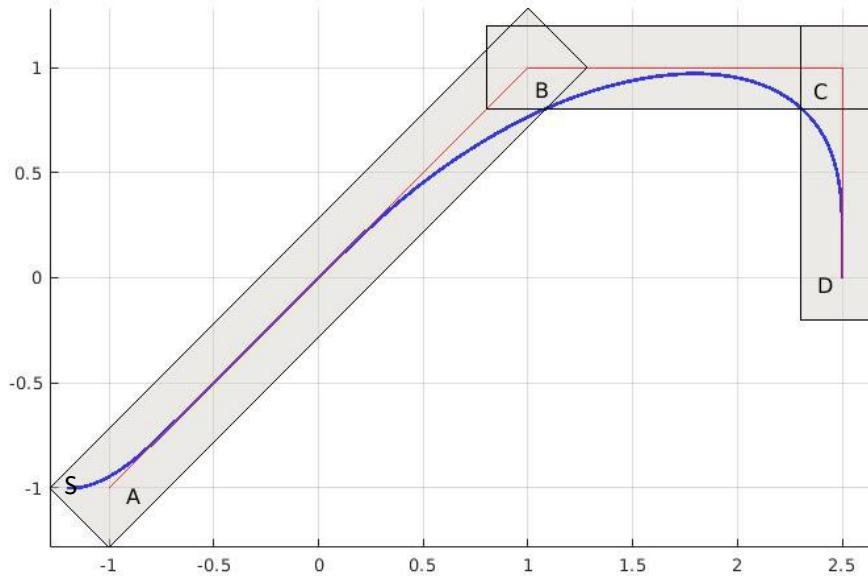
- Set the position set-point to  $|P_3 - P_2|$

Result:

- The trajectory converge to the desired line-segment path.

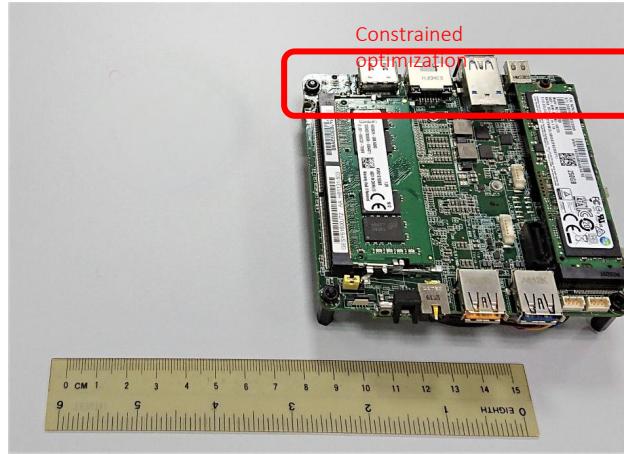
# Parameter space: Jerk limited trajectory

Online & incremental,  
sample & throw

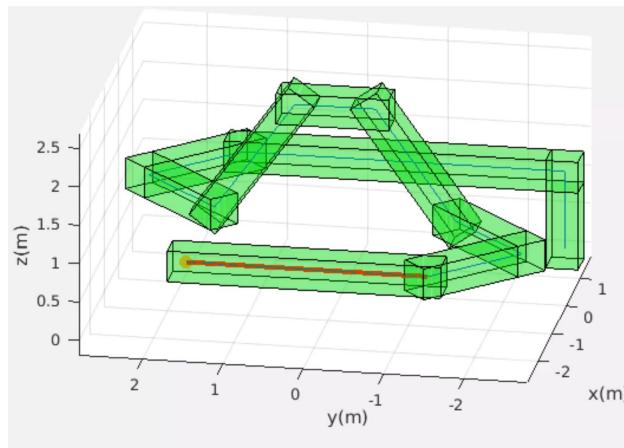


# Parameter space: Jerk limited trajectory

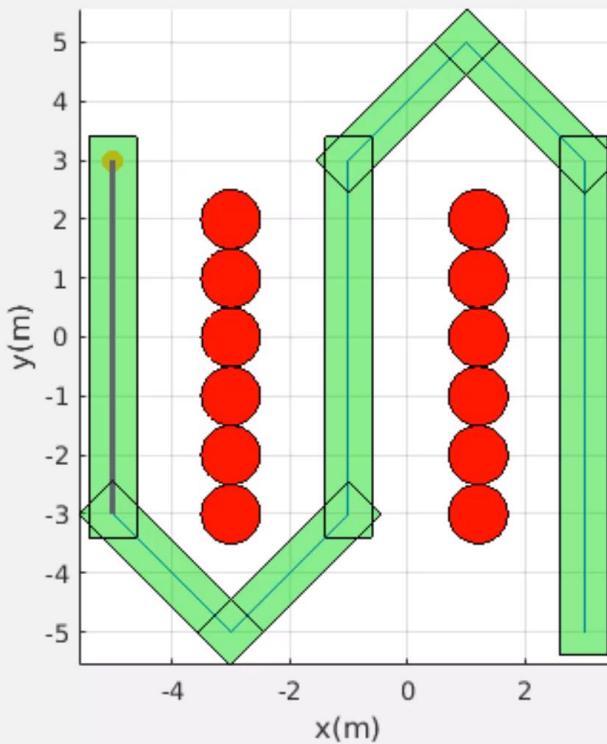
- An *efficient* algorithm to generate *3D* velocity, acceleration and jerk-limited trajectory.



- Instant reaction to changes  
(environment/mission)



# Parameter space: Jerk limited trajectory



# Parameter space: Jerk limited trajectory



Full stop at each waypoint



# JLT: Nonlinear-MPC

## Environment perception

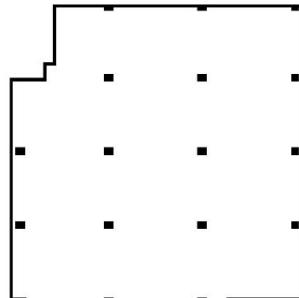
Pixelized environment

- **Occupancy map**

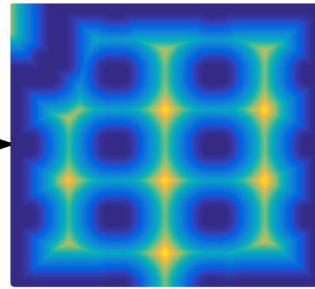
- Whether a pixel is occupied or not

- **Cost map**

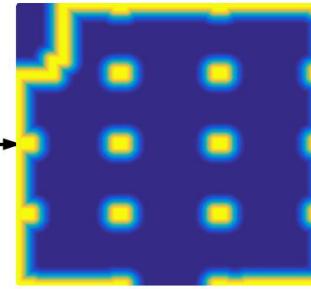
- The cost of stepping into certain pixel
  - The information on distance to nearest obstacle



Occupancy Map: Black means occupied, white means unoccupied



EDT: The darker the color, the closer to an obstacle.



Cost Map: The lighter the color, the higher the cost.



# JLT: Nonlinear-MPC

## Environment perception

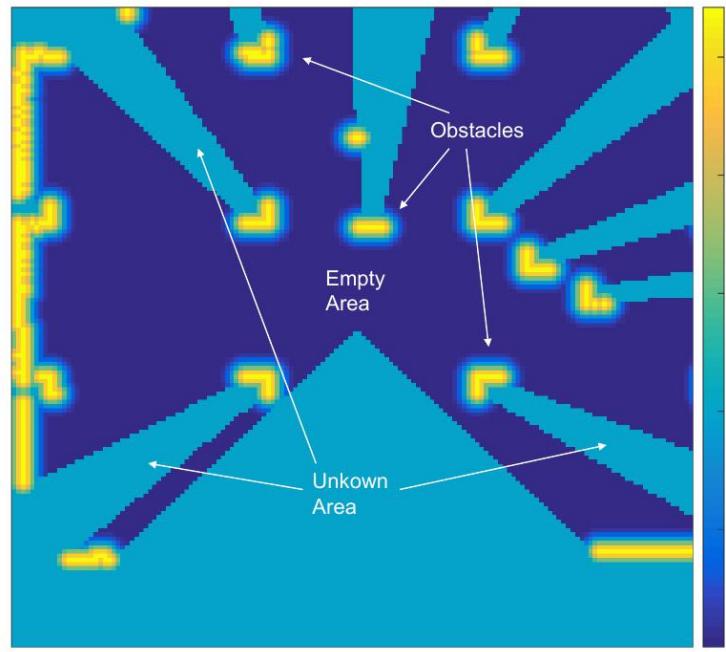
Sensor

● Laser scanner



Mapping procedure

- 1. Read sensor data**
- 2. Update the occupancy map**
  - a. Add in new obstacle
  - b. Clean LOS area
- 3. Perform EDT**
- 4. Perform Cost assigning**



# JLT: Nonlinear-MPC

## Two level guidance

Cut the problem into a series of TPBVP.

- **Global planner**

Provides a series of connected line segments

- **Trajectory planner**

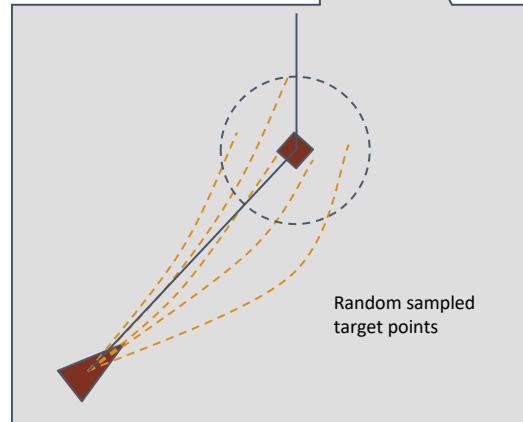
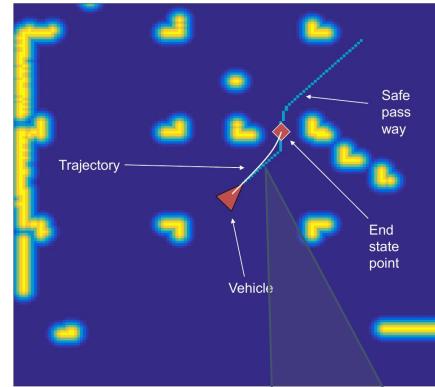
Solve TPBVP for jerk limited trajectory.

The trajectory leads the vehicle towards the first sharp turning point on line segment path.

It propose multiple trajectories for evaluator.

- **Evaluator**

Evaluate the quality of trajectories.



# JLT: Nonlinear-MPC

## Two level guidance

Cut the problem into a series of TPBVP.

- **Global planner**

Provides a series of connected line segments

- **Trajectory planner**

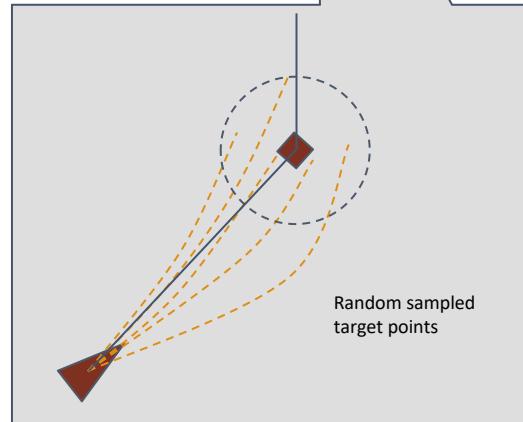
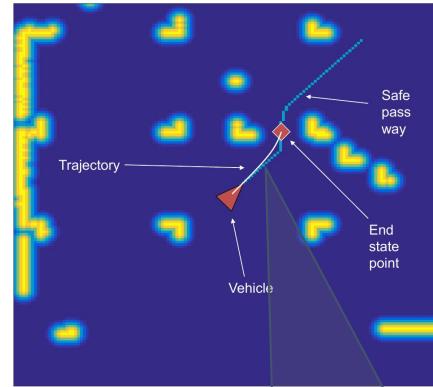
Solve TPBVP for jerk limited trajectory.

The trajectory leads the vehicle towards the first sharp turning point on line segment path.

It propose multiple trajectories for evaluator.

- **Evaluator**

Evaluate the quality of trajectories.

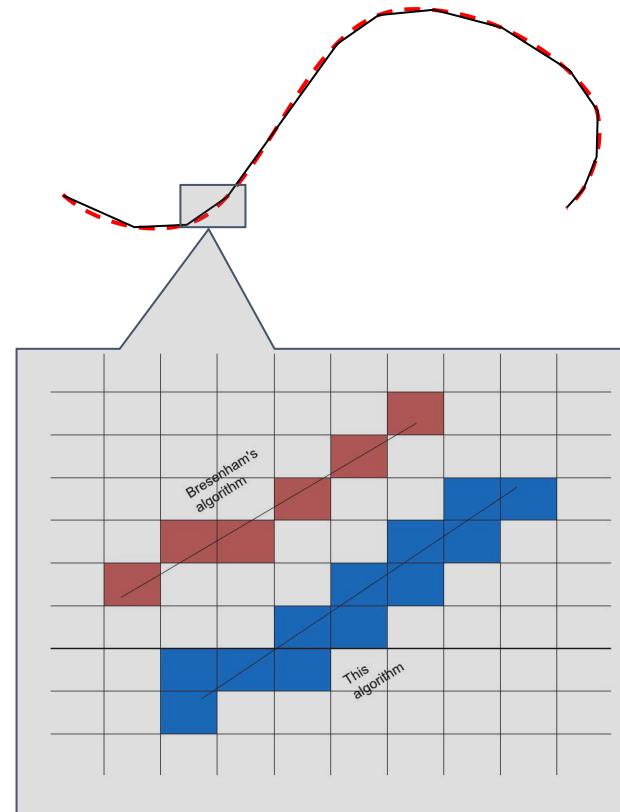


# JLT: Nonlinear-MPC

## Evaluator

Check the quality of each trajectory

- Represent original trajectory using connected line segments.
- Perform DDA marching to exam each pixel covered by trajectory.
- Find the pixel with highest cost value.
- Evaluate the trajectory based on its time duration and clearness.



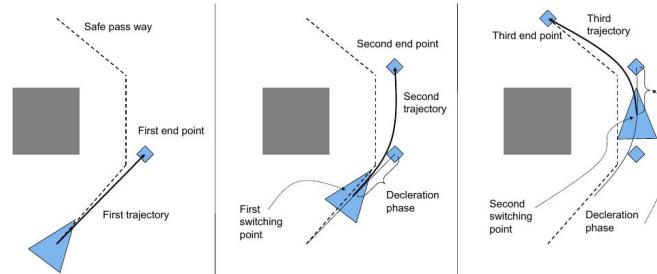
# JLT: Nonlinear-MPC

## Event Manager

Determine when to use new trajectory

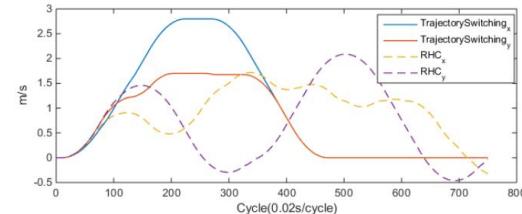
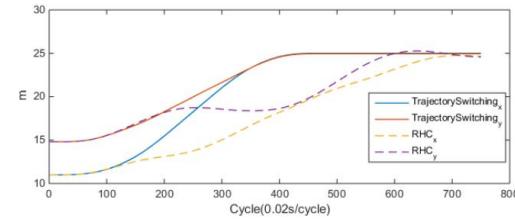
### ● Trajectory governor

- Constantly monitoring whether the current trajectory is safe.



### ● Event handler

- Fire a replanning event if the current trajectory is about to end
- Fire emergency replanning event if the current trajectory is unsafe



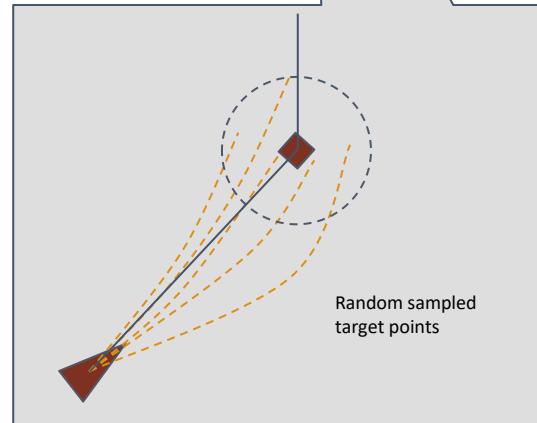
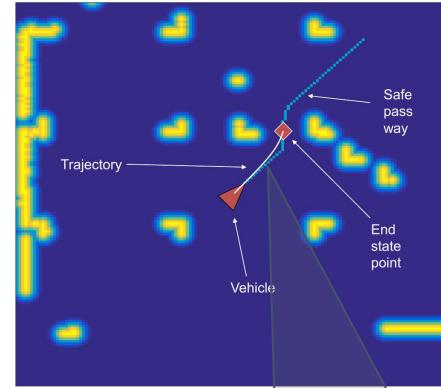
### ● Saving computational power and smoother flight



# JLT: Nonlinear-MPC

$$\min C_F(x(t_f)) + \int_{t=t_0}^{t_f} C_R(x, u) dt$$

$$\begin{aligned}\dot{x} &= f(x, u) \\ g(x, u) &< 0 \\ h(x, u) &= 0 \\ x &\notin \text{Obstacle}\end{aligned}$$

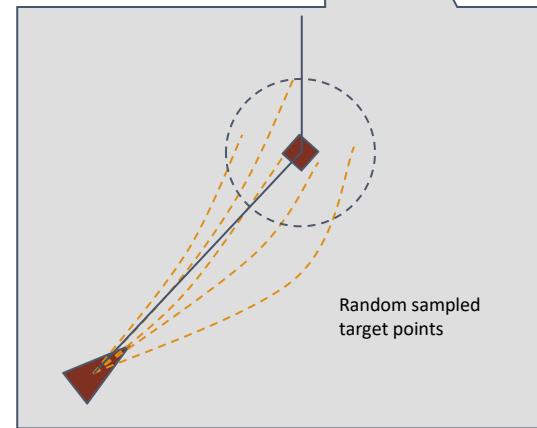
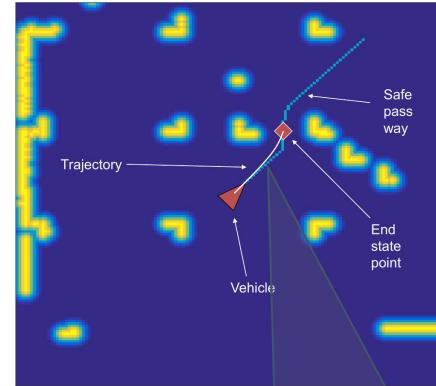


# JLT: Nonlinear-MPC

$$\min C_F(x(t_f)) + \int_{t=t_0}^{t_f} C_R(x, u) dt$$

Hard constraint:  $x \notin \text{Obstacle}$

Soft constraint:  $s(x) = \begin{cases} 0, & \text{if } x \notin \text{Obstacle} \\ M, & \text{otherwise} \end{cases}$



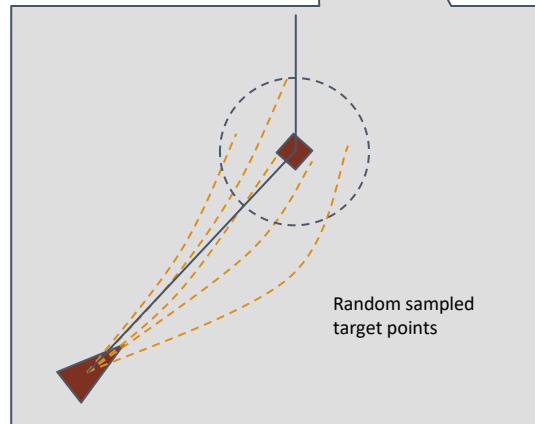
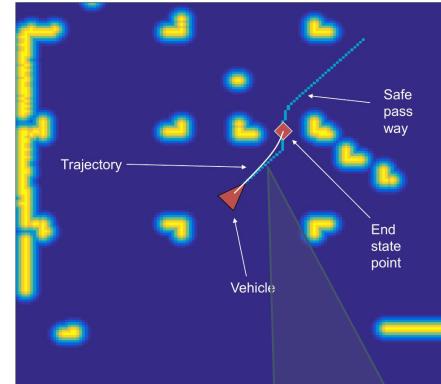
# JLT: Nonlinear-MPC

$$\min C_F(x(t_f)) + \int_{t=t_0}^{t_f} C_R(x, u) dt$$

$$C_R(x, u) = (x - x_d)^T Q(x - x_d) + u^T R u + s(x)$$

$$C_F(x) = (x - x_d)^T W(x - x_d) + s(x)$$

$$s(x) = \begin{cases} 0, & \text{if } x \notin \text{Obstacle} \\ M, & \text{otherwise} \end{cases}$$



# JLT: Nonlinear-MPC

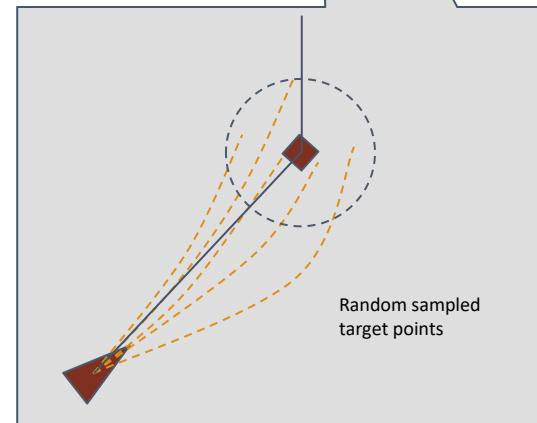
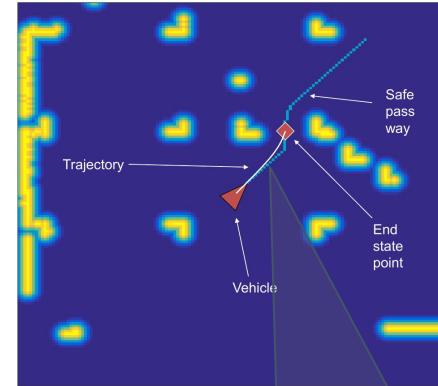
$$x_0, x_f \xrightarrow{\text{JLT}} x(t)$$

$$\min C_F(x(t_f)) + \int_{t=t_0}^{t_f} C_R(x, u) dt$$

$$C_R(x, u) = (x - x_d)^T Q(x - x_d) + u^T R u + s(x)$$

$$C_F(x) = (x - x_d)^T W(x - x_d) + s(x)$$

$$s(x) = \begin{cases} 0, & \text{if } x \notin \text{Obstacle} \\ M, & \text{otherwise} \end{cases}$$



# JLT: Nonlinear-MPC

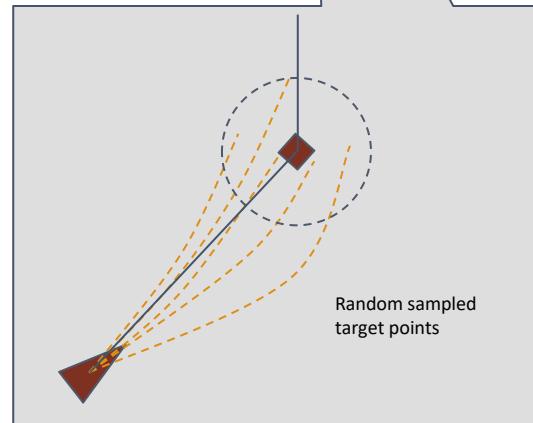
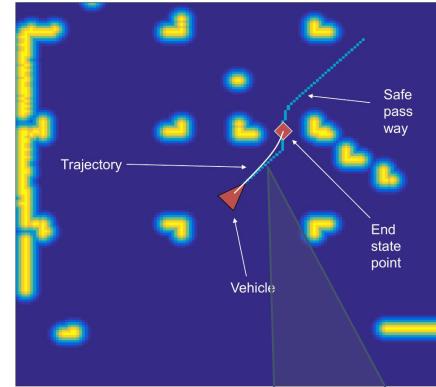
**Algorithm 1:** Particle Swarm Optimization.

---

**Input:**  $x_{\text{ini}}$ ,  $\mathcal{M}(\text{map})$   
**Output:**  $\theta^*$  (best end state constraint)

- 1:  $\Theta \leftarrow \text{Particle\_Initialization}();$
- 2:  $c_i^* \leftarrow \infty$ ,  $\theta_i^* \leftarrow \theta_i$ ,  $\delta_i \leftarrow \text{rand}$ ,  $\forall i \in [1, \text{size}(\Theta)]$
- 3: **for**  $m = 1$  to MAX\_ITERS **do**
- 4:     **for each**  $\theta_i \in \Theta$  **do**
- 5:          $[x(t), u(t)] = \mathcal{S}_{\text{NN}}(x_{\text{ini}}, \theta_i)$
- 6:          $c_i = J(x(t), u(t), \mathcal{M})$
- 7:         **if**  $c_i < c_i^*$  **then**
- 8:              $c_i^* = c_i$
- 9:              $\theta_i^* = \theta_i$
- 10:         $i^* = \underset{i}{\operatorname{argmin}}(c_i^*)$
- 11:         $\theta^* = \theta_{i^*}$
- 12:        **for each**  $\theta_i \in \Theta$  **do**
- 13:              $\delta_i = \delta_i + k_1 \cdot \text{rand} \cdot (\theta_i^* - \theta_i) + k_2 \cdot \text{rand} \cdot (\theta^* - \theta_i)$
- 14:              $\theta_i = \theta_i + \delta_i$

---



# JLT: Nonlinear-MPC

---

**Algorithm 1:** Particle Swarm Optimization.

---

**Input:**  $\mathbf{x}_{\text{ini}}, \mathcal{M}(\text{map})$   
**Output:**  $\theta^*$  (best end state constraint)

- 1:  $\Theta \leftarrow \text{Particle\_Initialization};$
- 2:  $c_i^* \leftarrow \infty, \theta_i^* \leftarrow \theta_i, \delta_i \leftarrow \text{rand}, \forall i \in [1, \text{size}(\Theta)]$
- 3: **for**  $m = 1$  to MAX\_ITERS **do**
- 4:   **for each**  $\theta_i \in \Theta$  **do**
- 5:      $[\mathbf{x}(t), \mathbf{u}(t)] = \mathcal{S}_{\text{NN}}(\mathbf{x}_{\text{ini}}, \theta_i)$
- 6:      $c_i = J(\mathbf{x}(t), \mathbf{u}(t), \mathcal{M})$
- 7:     **if**  $c_i < c_i^*$  **then**
- 8:        $c_i^* = c_i$
- 9:        $\theta_i^* = \theta_i$
- 10:       $i^* = \underset{i}{\operatorname{argmin}}(c_i^*)$
- 11:      $\theta^* = \theta_{i^*}$
- 12:     **for each**  $\theta_i \in \Theta$  **do**
- 13:        $\delta_i = \delta_i + k_1 \cdot \text{rand} \cdot (\theta_i^* - \theta_i) + k_2 \cdot \text{rand} \cdot (\theta^* - \theta_i)$
- 14:        $\theta_i = \theta_i + \delta_i$

---



# JLT: Nonlinear-MPC

---

**Algorithm 1:** Particle Swarm Optimization.

---

**Input:**  $\mathbf{x}_{\text{ini}}, \mathcal{M}(\text{map})$   
**Output:**  $\theta^*$  (best end state constraint)

- 1:  $\Theta \leftarrow \text{Particle\_Initialization};$
- 2:  $c_i^* \leftarrow \infty, \theta_i^* \leftarrow \theta_i, \delta_i \leftarrow \text{rand}, \forall i \in [1, \text{size}(\Theta)]$
- 3: **for**  $m = 1$  to MAX\_ITERS **do**
- 4:     **for each**  $\theta_i \in \Theta$  **do**
- 5:          $[\mathbf{x}(t), \mathbf{u}(t)] = \mathcal{S}_{\text{NN}}(\mathbf{x}_{\text{ini}}, \theta_i)$
- 6:          $c_i = J(\mathbf{x}(t), \mathbf{u}(t), \mathcal{M})$
- 7:         **if**  $c_i < c_i^*$  **then**
- 8:              $c_i^* = c_i$
- 9:              $\theta_i^* = \theta_i$
- 10:          $i^* = \underset{i}{\operatorname{argmin}}(c_i^*)$
- 11:          $\theta^* = \theta_{i^*}$
- 12:         **for each**  $\theta_i \in \Theta$  **do**
- 13:              $\delta_i = \delta_i + k_1 \cdot \text{rand} \cdot (\theta_i^* - \theta_i) + k_2 \cdot \text{rand} \cdot (\theta^* - \theta_i)$
- 14:          $\theta_i = \theta_i + \delta_i$

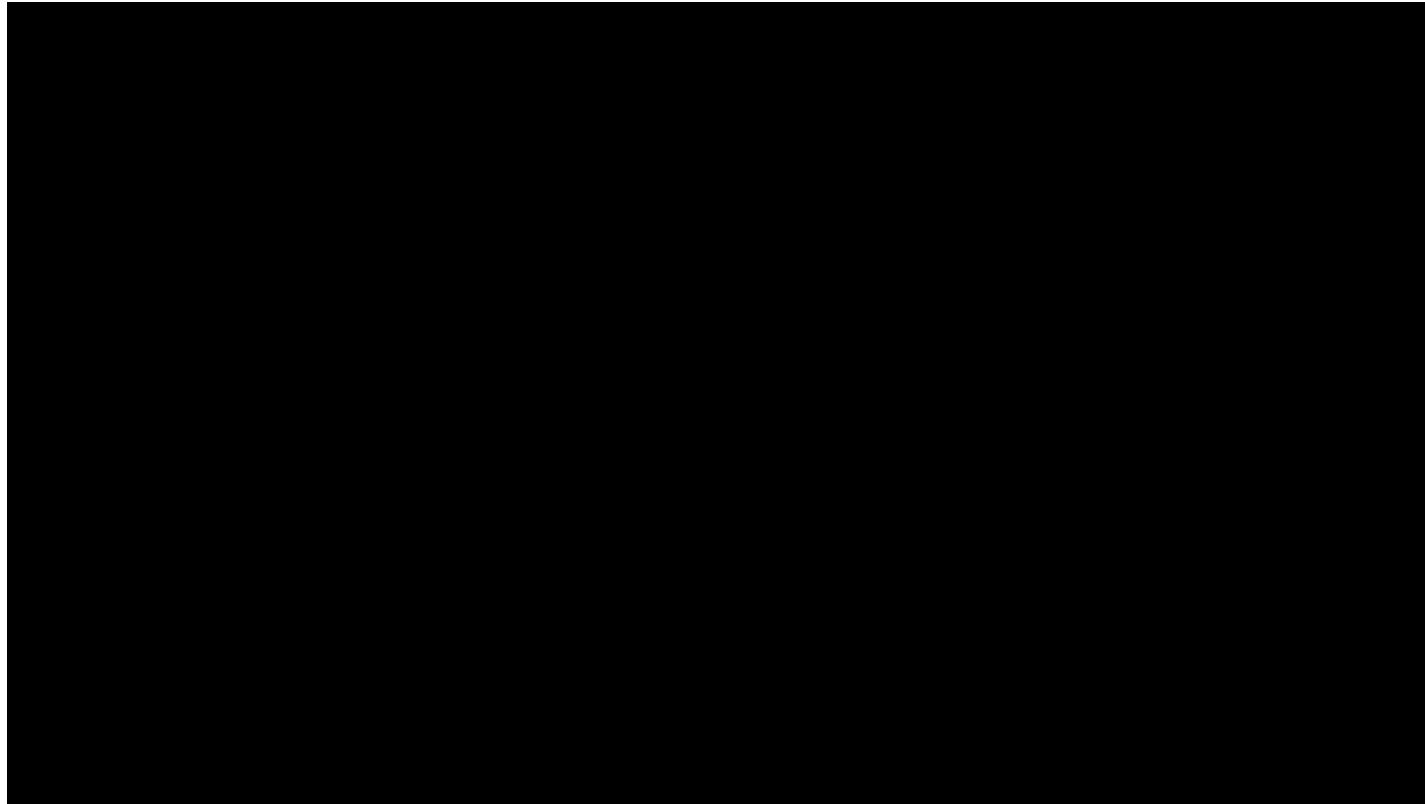
---



# JLT: Nonlinear-MPC



# JLT: Nonlinear-MPC



# Parameter space: General BSCP

A general BVP problem:

$$\min_{\mathbf{u}(t), \mathbf{x}(t), t_f} G(\mathbf{x}(t), \mathbf{u}(t), t_f)$$

$$g(\mathbf{x}(t_f), \boldsymbol{\theta}) = 0$$

$$h(\mathbf{x}, \mathbf{u}) = 0, \tilde{h}(\mathbf{x}, \mathbf{u}) \leq 0$$

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}),$$

Often requires a numerical optimization.

Some times can be solved through a controller:

$$\|g(\mathbf{x}(t_f), \boldsymbol{\theta})\| < \epsilon, \forall t \geq t_f,$$

$$\mathcal{S} : \langle \mathbf{x}_0, \boldsymbol{\theta} \rangle \rightarrow \langle \hat{\mathbf{u}}(t), \hat{\mathbf{x}}(t), \hat{t}_f \rangle .$$



# Parameter space: General BSCP

A general BVP problem:

$$\min_{\mathbf{u}(t), \mathbf{x}(t), t_f} G(\mathbf{x}(t), \mathbf{u}(t), t_f)$$

$$g(\mathbf{x}(t_f), \boldsymbol{\theta}) = 0$$

$$h(\mathbf{x}, \mathbf{u}) = 0, \tilde{h}(\mathbf{x}, \mathbf{u}) \leq 0$$

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}),$$

Often requires a numerical optimization.

Some times can be solved through a controller:

$$\|g(\mathbf{x}(t_f), \boldsymbol{\theta})\| < \epsilon, \forall t \geq t_f,$$

$$\mathcal{S} : \langle \mathbf{x}_0, \boldsymbol{\theta} \rangle \rightarrow \langle \hat{\mathbf{u}}(t), \hat{\mathbf{x}}(t), \hat{t}_f \rangle .$$

# Parameter space: General BSCP

A general BVP problem:

$$\min_{\mathbf{u}(t), \mathbf{x}(t), t_f} G(\mathbf{x}(t), \mathbf{u}(t), t_f)$$

$$g(\mathbf{x}(t_f), \theta) = 0$$

$$h(\mathbf{x}, \mathbf{u}) = 0, \tilde{h}(\mathbf{x}, \mathbf{u}) \leq 0$$

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}),$$

$$\min C_F(x(t_f)) + \int_{t=t_0}^{t_f} C_R(x, u) dt$$

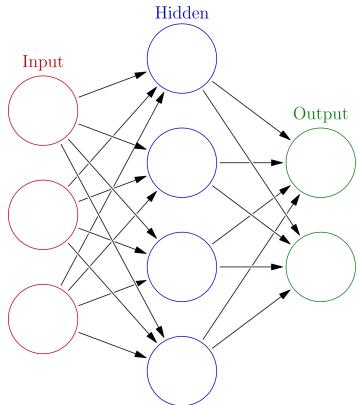
$$\mathcal{S} : \langle \mathbf{x}_0, \theta \rangle \rightarrow \langle \hat{\mathbf{u}}(t), \hat{\mathbf{x}}(t), \hat{t}_f \rangle.$$

$$\min_{\theta} L_F(\theta) + \int_{t=t_0}^{t_f} L_R(\theta) dt$$



# Parameter space: General BSCP

$$\mathcal{S} : \langle \mathbf{x}_0, \theta \rangle \rightarrow \langle \hat{\mathbf{u}}(t), \hat{\mathbf{x}}(t), \hat{t}_f \rangle .$$



Update  $x_0$ .

Optimize  $\min_{\theta} L_F(\theta) + \int_{t=t_0}^{t_f} L_R(\theta) dt$  and get  $\theta^*$ .

- 1) Use the NN approximate input  $\mathbf{u}$ .
- 2) Solve the original BVP

$$\min_{\mathbf{u}(t), \mathbf{x}(t), t_f} G(\mathbf{x}(t), \mathbf{u}(t), t_f)$$

$$g(\mathbf{x}(t_f), \theta) = 0 \quad \dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}),$$

$$h(\mathbf{x}, \mathbf{u}) = 0, \tilde{h}(\mathbf{x}, \mathbf{u}) \leq 0$$

$$\min C_F(x(t_f)) + \int_{t=t_0}^{t_f} C_R(x, u) dt$$

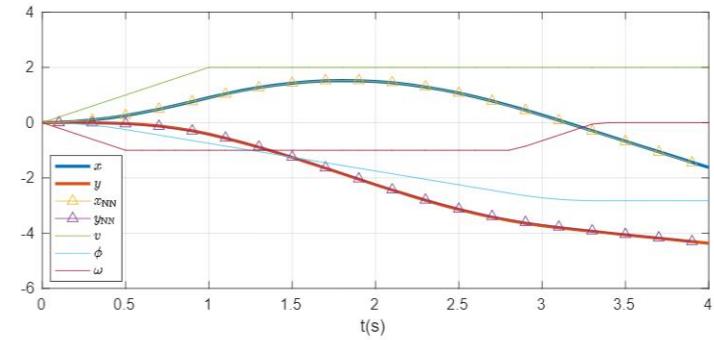
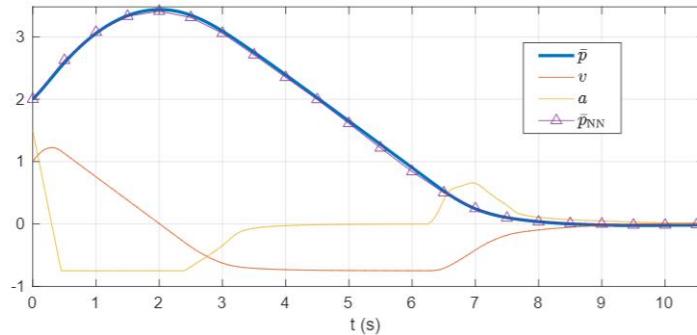


# Parameter space: General BSCP

Network size and training details.

	NN structure	Training set size	Testing set size	Batch size	Epoch	Average MSE	Maximum MSE
Quadrotor horizontal	64-128-128-41 MLP	800,000	200,000	20	12	$4.72 \times 10^{-4}$	0.071
Quadrotor vertical	64-128-128-41 MLP	800,000	200,000	10	15	$1.2 \times 10^{-3}$	0.088
2nd order unicycle	80-256-128-81 MLP	860,000	130,000	10	20	$3.4 \times 10^{-3}$	0.094

\* MSE: Mean Squared Error. MLP: Multi-Layer Perceptron.

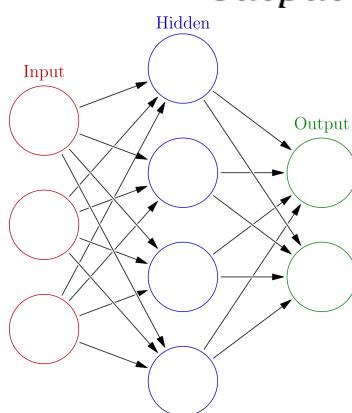


# Parameter space: General BSCP

Approximated gradient

$$\text{input} = \theta, x_0$$

$$\frac{dx}{d\theta}, \frac{du}{d\theta}$$

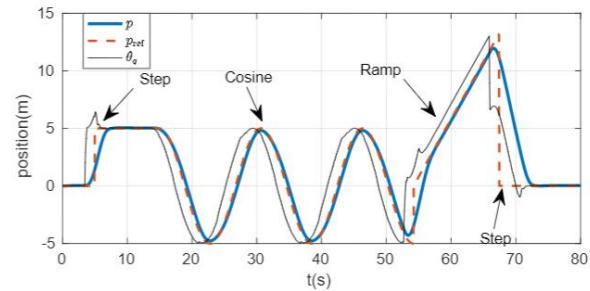


$$\text{output} = x(t), u(t)$$

$$\frac{dJ}{dx}, \frac{dJ}{du}$$

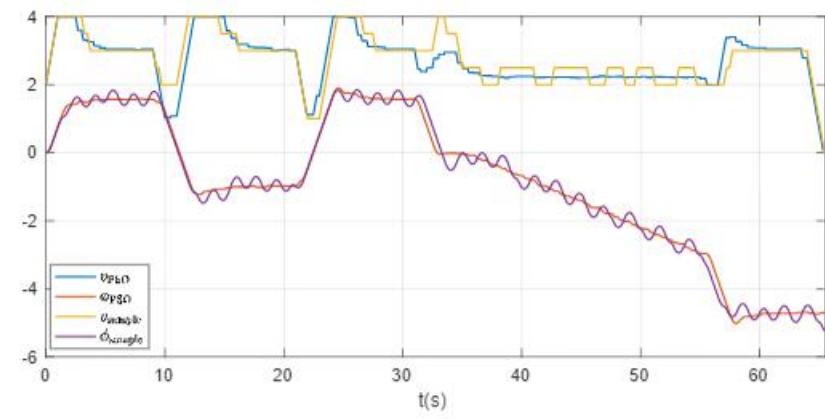
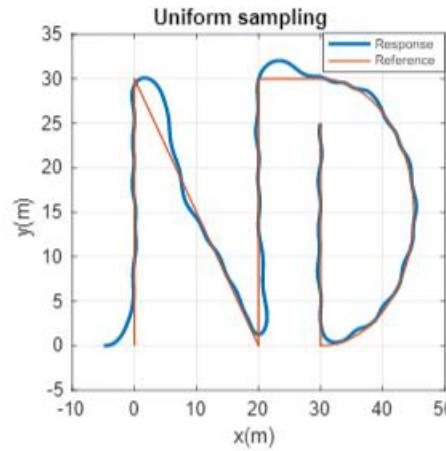
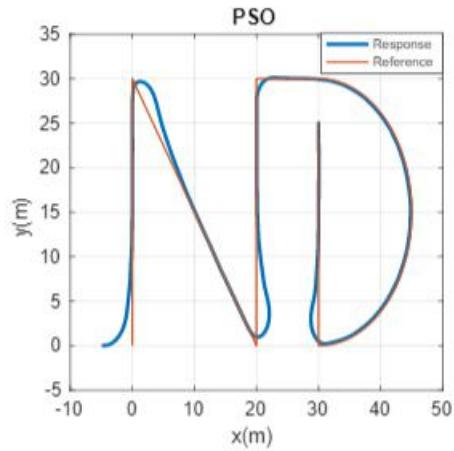
$$J = C_F(x(t_f)) + \int_{t=t_0}^{t_f} C_R(x, u) dt$$

$$\frac{dJ}{d\theta}$$

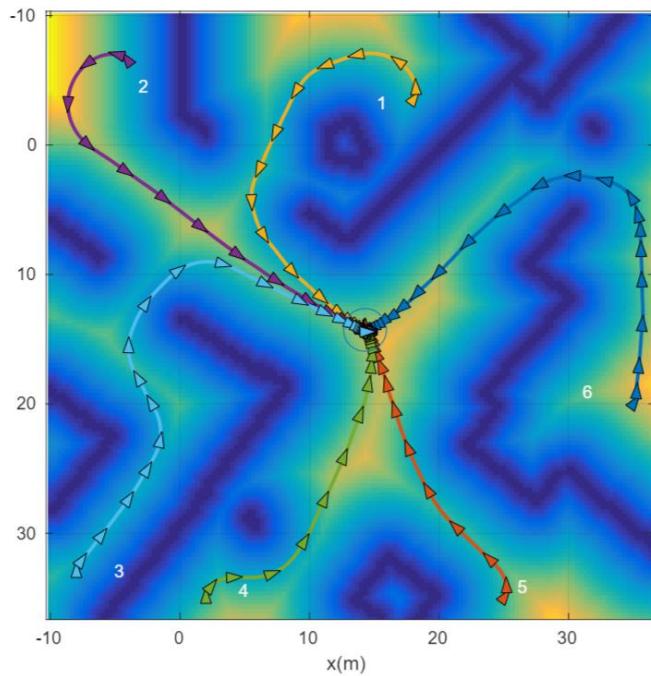


# Parameter space: General BSCP

PSO: 10 particle, 10 iterations. VS Uniform sampling: 100 particles 1 iteration.



# Parameter space: General BSCP



$$\min C_F(x(t_f)) + \int_{t=t_0}^{t_f} C_R(x, u) dt$$

$$C_R(x, u) = (x - x_d)^T Q(x - x_d) + u^T R u + s(x)$$

$$C_F(x) = (x - x_d)^T W(x - x_d) + s(x)$$

$$s(x) = \begin{cases} 0, & \text{if } x \notin \text{Obstacle} \\ M, & \text{otherwise} \end{cases}$$

cost\_to\_goal(x), through method like Dijkstra.



# Parameter space: General BSCP

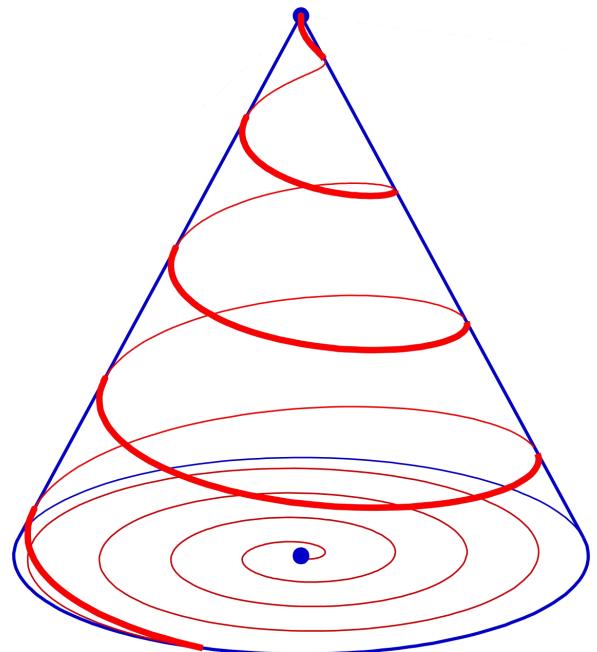
Model Predictive Motion Planning With  
Boundary State Constrained Primitives  
Experiment videos



# Homework

Previously, we have discussed how to design a quadratic programming based MPC to allow a single-axis triple integrator to travel from an arbitrary state to the centre of the state space, a.k.a with zero position, velocity and acceleration.

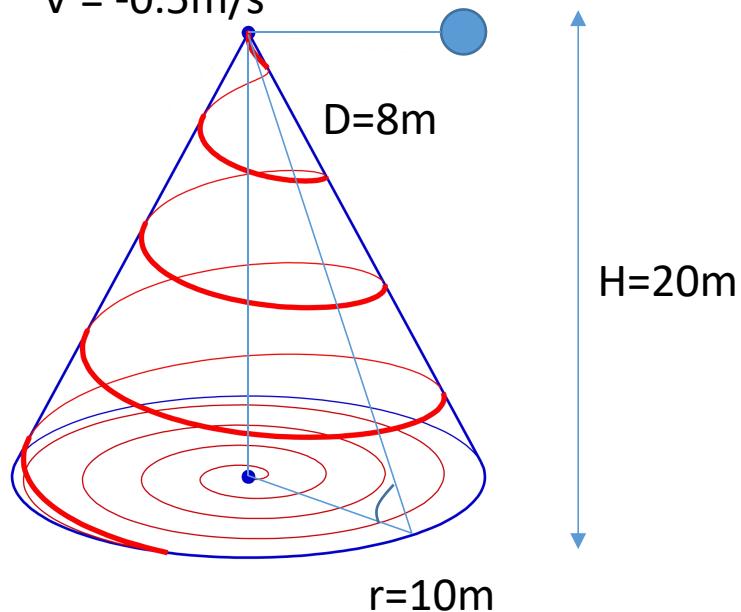
Now please design a quadratic programming based MPC to track conical spiral for a 3-axis triple integrator (basically a quadrotor model).



# Homework

$$\omega = 0.08 \text{ rad/s}$$

$$v = -0.5 \text{ m/s}$$



$$-6 \leq v_{x,y} \leq 6 \quad -1 \leq v_z \leq 6$$

$$-3 \leq a_{x,y} \leq 3 \quad -1 \leq a_z \leq 3$$

$$-3 \leq j_{x,y} \leq 3 \quad -2 \leq j_z \leq 2$$





感谢各位聆听！  
Thanks for Listening!

