Seventh Annual Title Text
Cyber-Physical Systems Principal Investigators' Meeting
Arlington, VA | October 31 – November 1, 2016

Computationally-Aware Cyber-Physical Systems (CACPS)

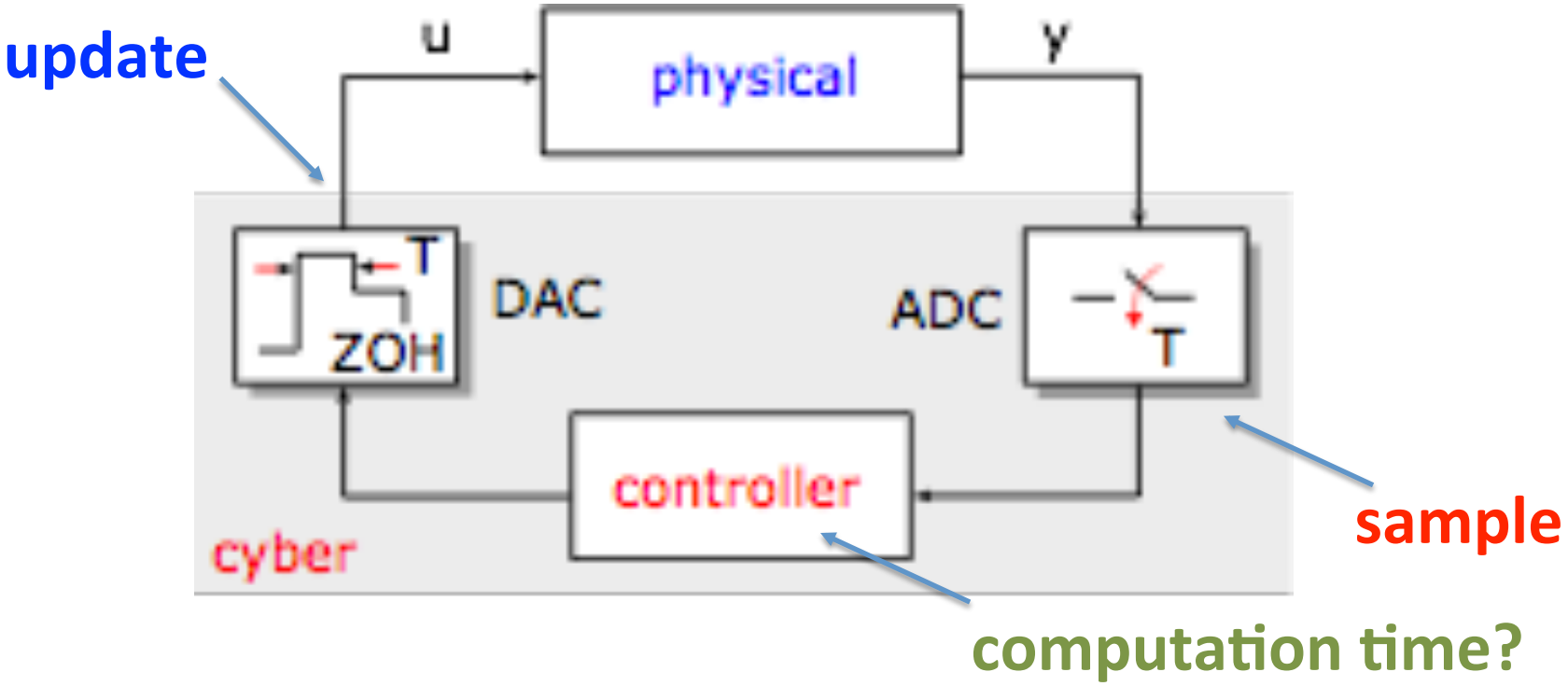
Ricardo Sanfelice (UCSC) and Jonathan Sprinkle (UofA)

Synergy: Collaborative: Computationally Aware Cyber-Physical Systems
NSF CNS-1544395 (UCSC), NSF CNS-1544395 (UofA)

UNIVERSITY OF CALIFORNIA SANTA CRUZ

THE UNIVERSITY OF ARIZONA

What to do if my control algorithm does not terminate when an input is needed?



Feedback provides robustness to small computation time, but computing time could be large, specially in optimization-based approaches

Computationally-Aware Algorithms driven by Applications

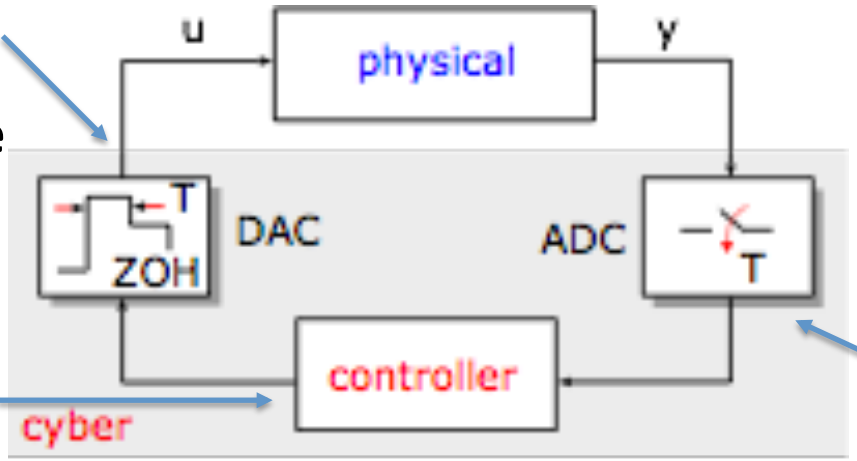
UAVs in the National Air Space
(courtesy NASA)

Self-driving Cars

MPC for Diesel Engines
(courtesy TOYOTA)

update

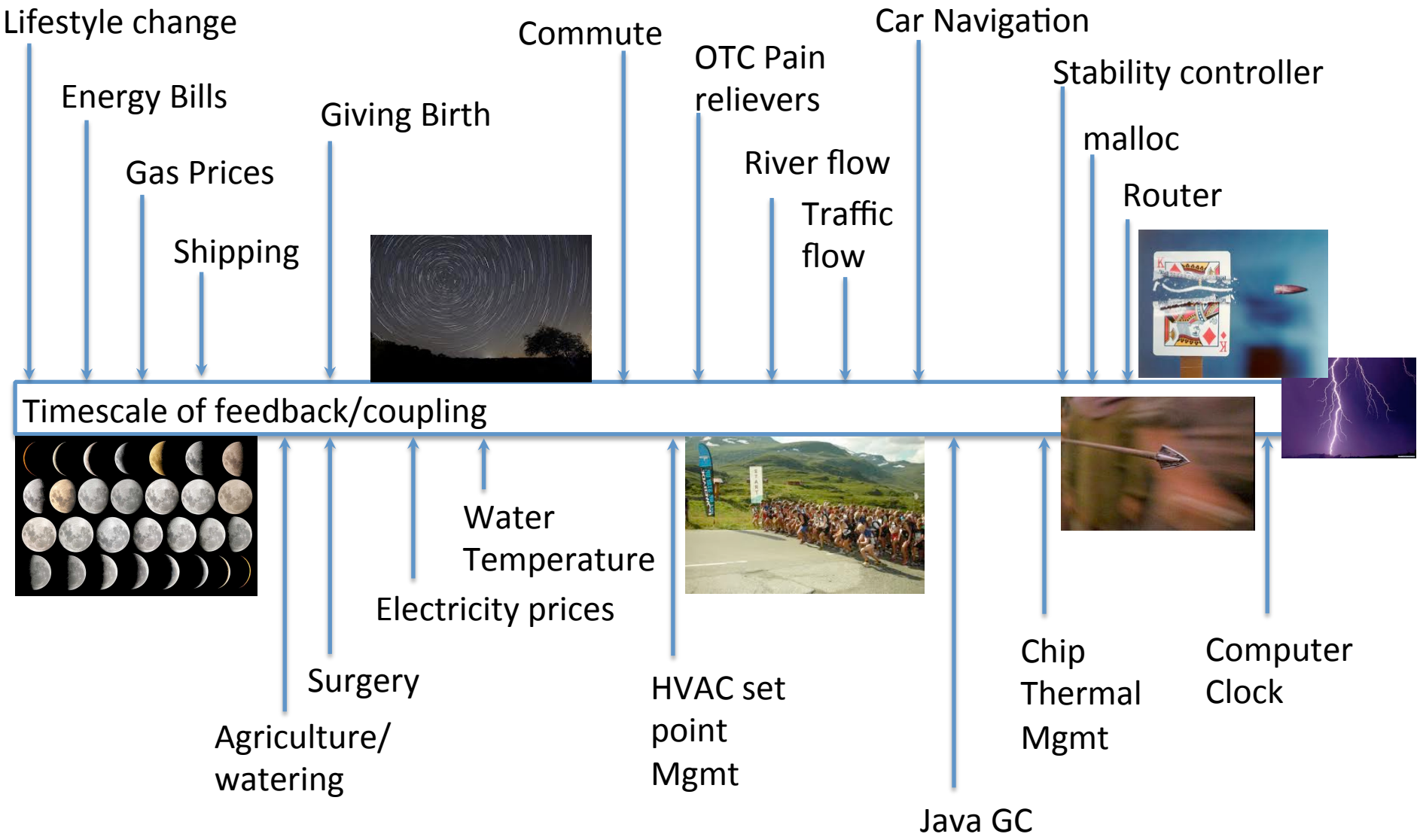
Exploit models of the physical system w/ different accuracy to reduce computation time!



sample

Trade off

Different timescales*



* Not to scale. It's not like I plotted this in MATLAB or anything...

Problem for vehicle control: comfort

http://www.youtube.com/watch?v=ChZwuj3hCvw

Or worse...

Photo By Twilight Invasion

Consider a moving ground vehicle

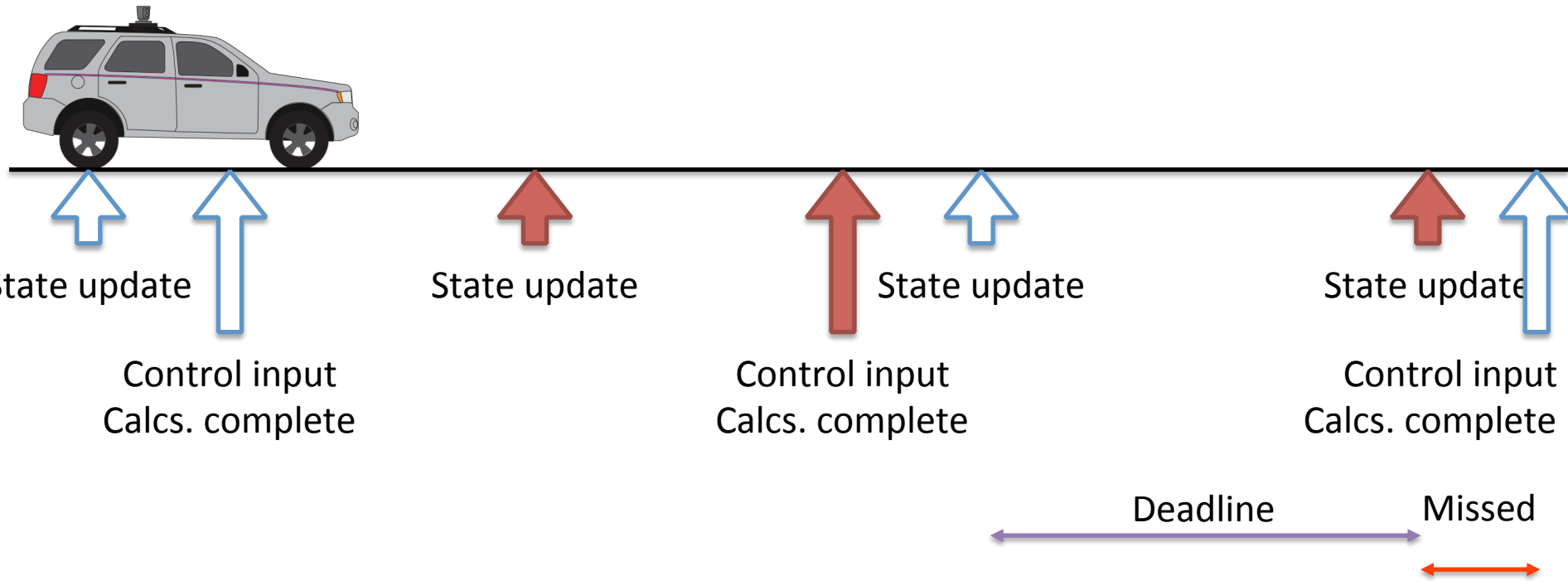
State update

State update

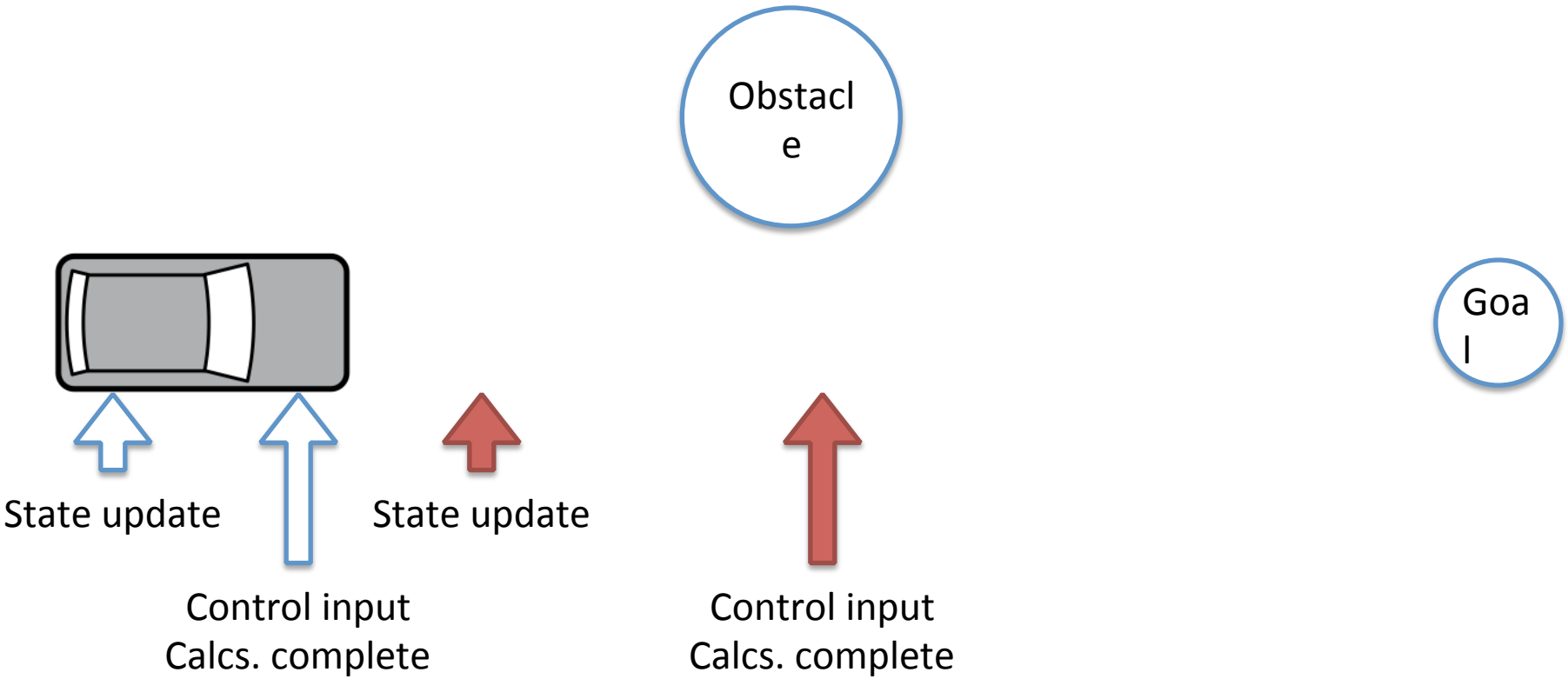
State update

State update

When computing a control decision…



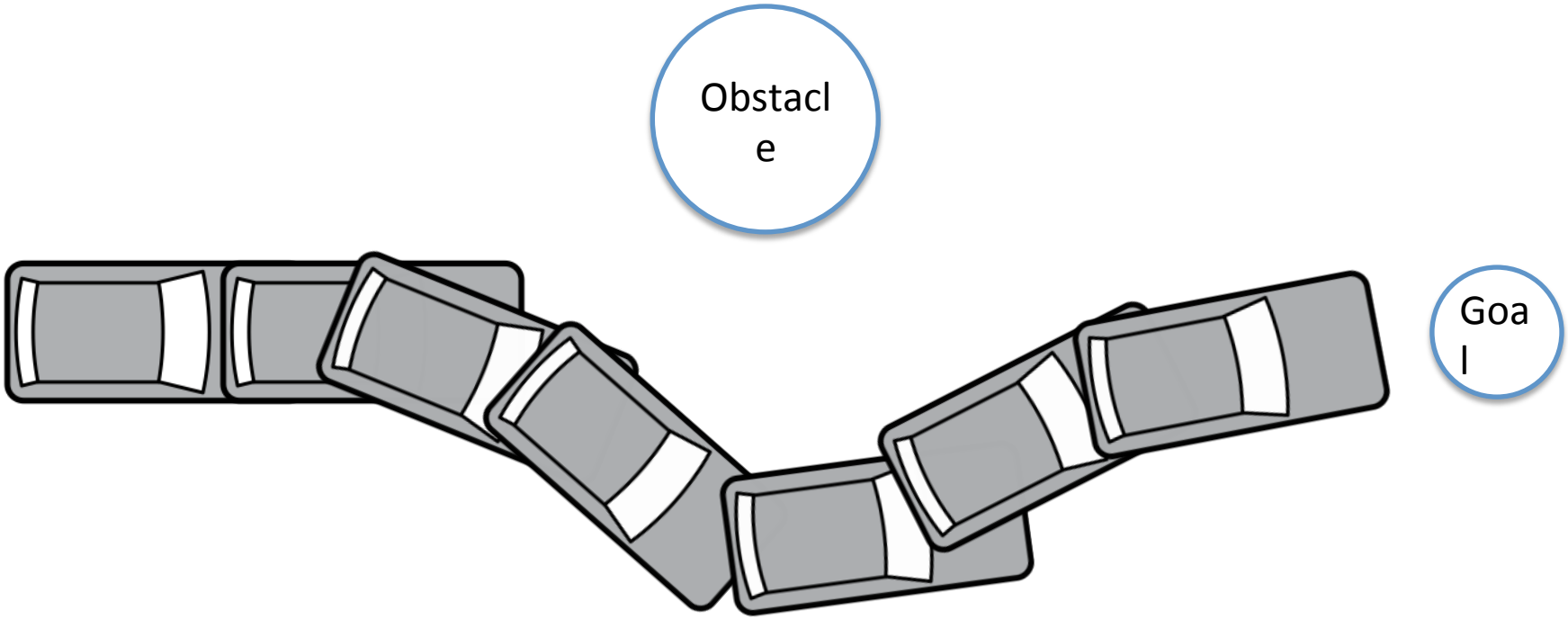
What if an obstacle moves while we compute?



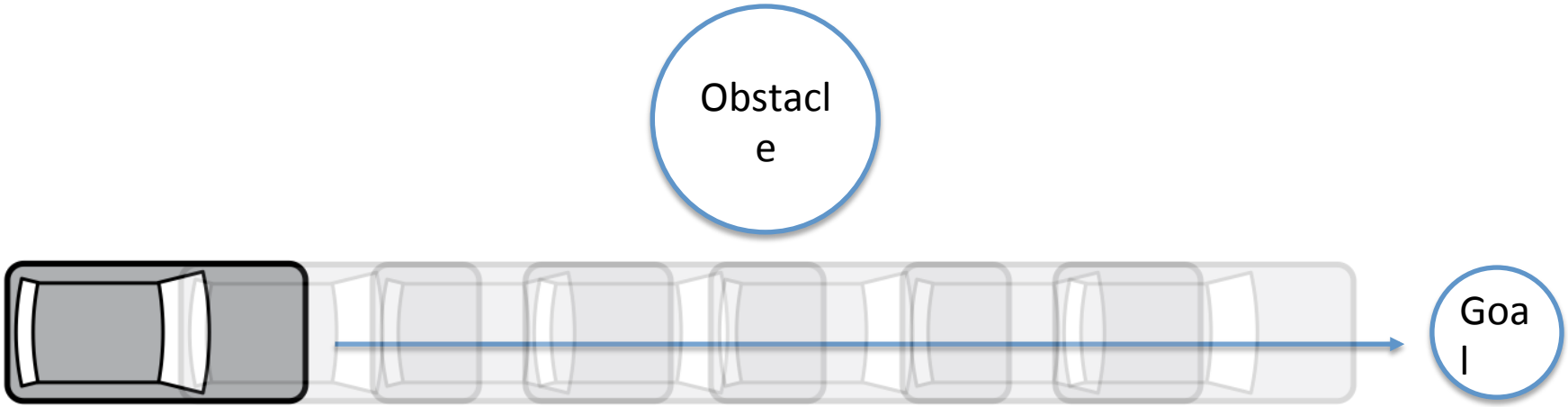
Obvious design goal: Reduce the period for state update/control input as much as possible.

This is a motivating animation, not a kinematic simulation.

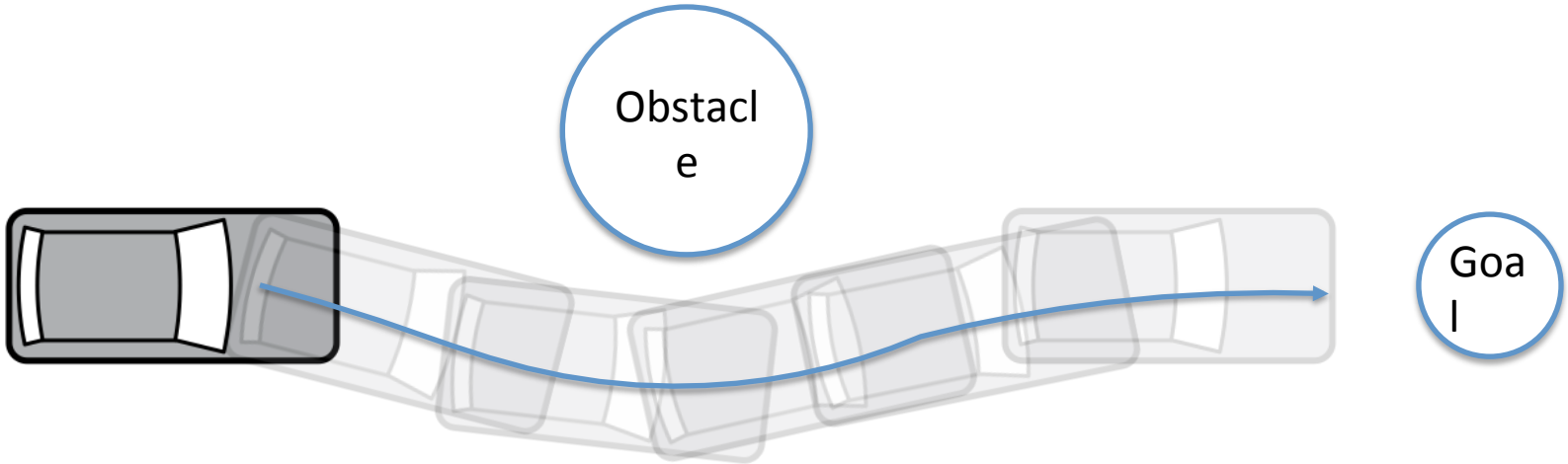
Model-Predictive Control: Plan Trajectories at Runtime



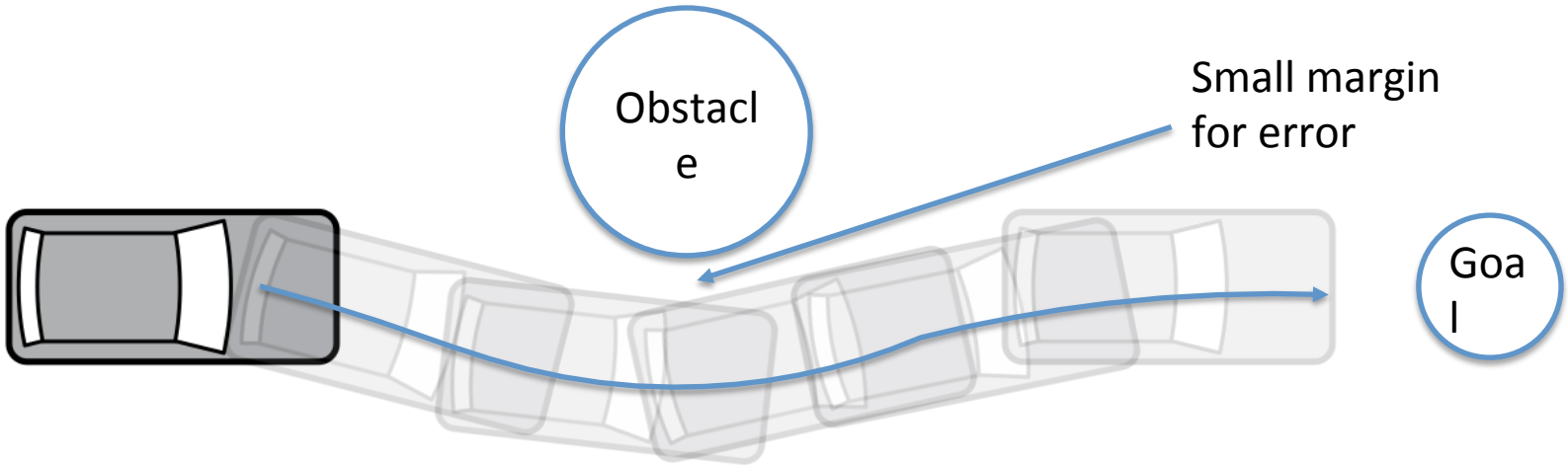
Model-Predictive Control: Plan Trajectories at Runtime



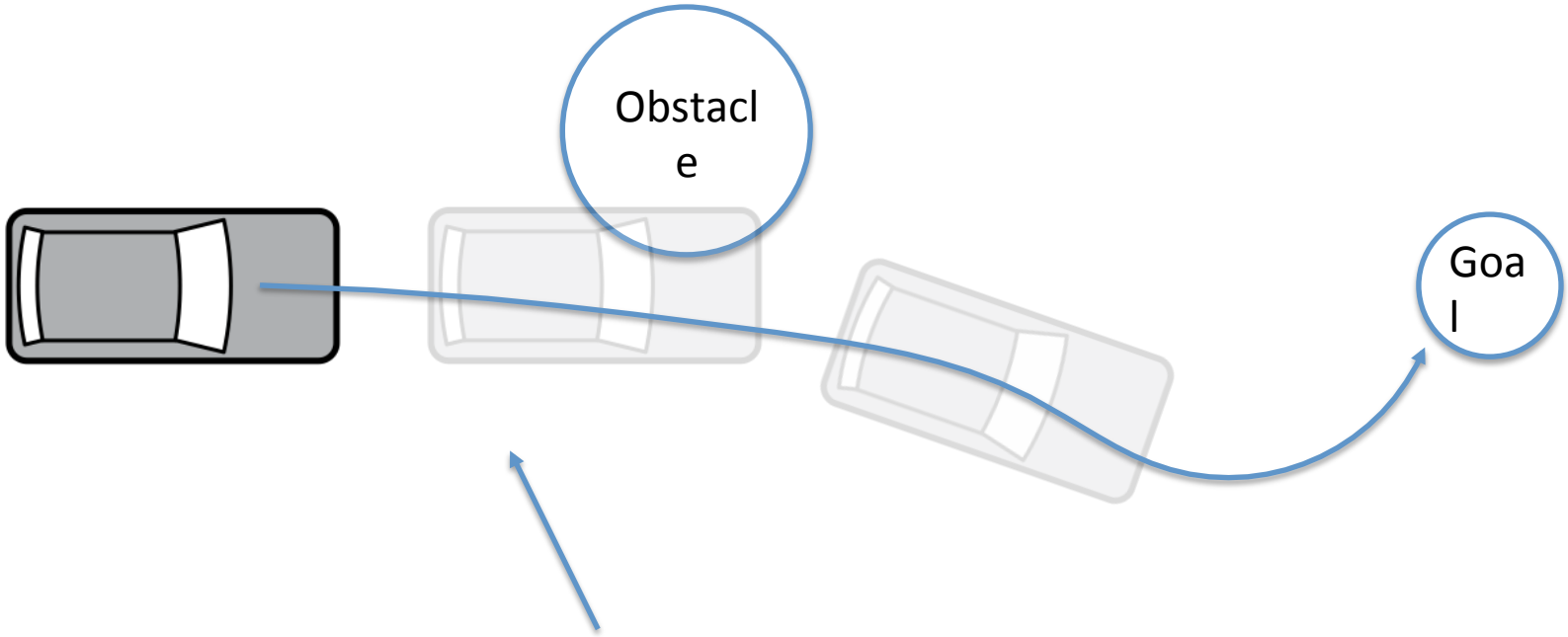
Model-Predictive Control: Plan Trajectories at Runtime



Model-Predictive Control: Plan Trajectories at Runtime

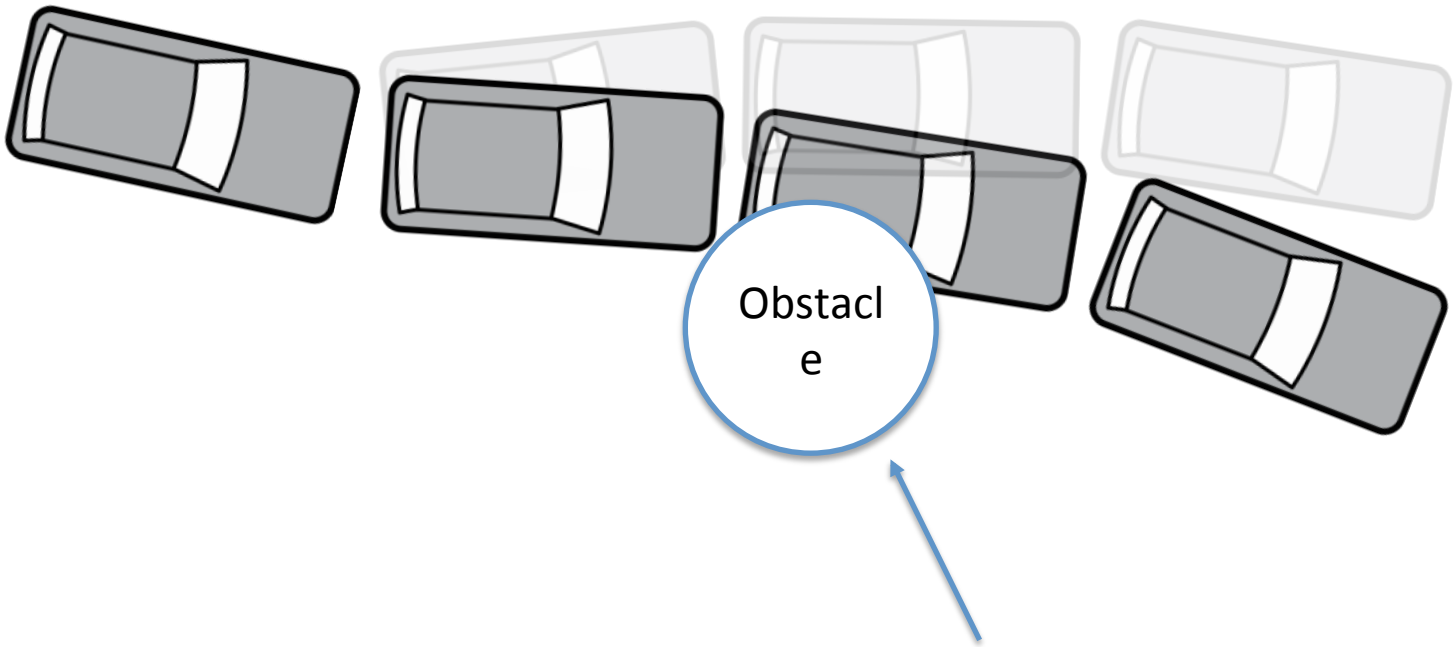


Easy…I'll use an accurate vehicle model to predict the trajectory and avoid the obstacle.



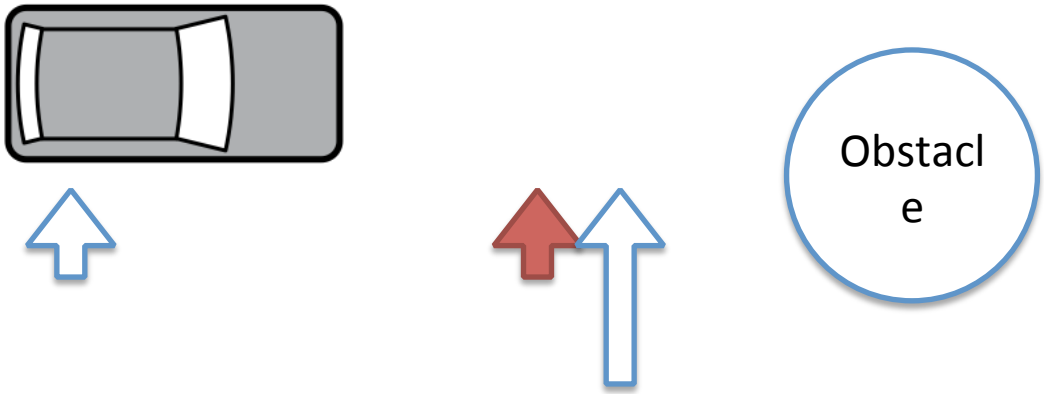
Takes longer to compute control inputs

Easy…I'll pick a simpler vehicle model to make it more likely to return control inputs in time



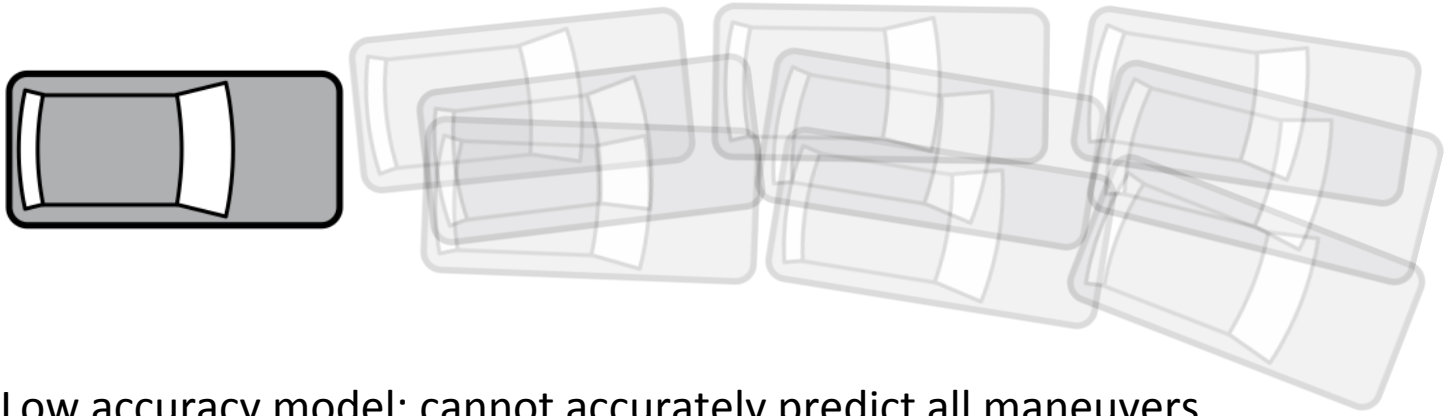
That control input does not mean what you think it means.

Competing constraints



High accuracy model: takes longer to optimize.

High vehicle speed: cannot tolerate slow return time.



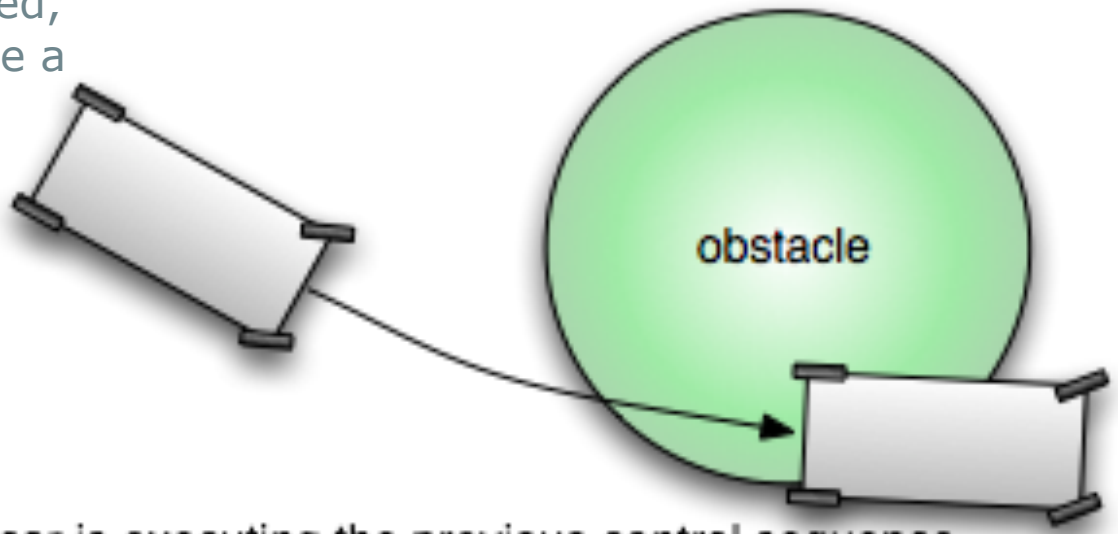
Low accuracy model: cannot accurately predict all maneuvers.

Goal: Computationally-Aware CPS

1. Consider the time required to perform the computation.
2. Switch between controllers using accuracy and time as switching criteria.
3. Explore conditions for stability and convergence.

Problems: MPC return time

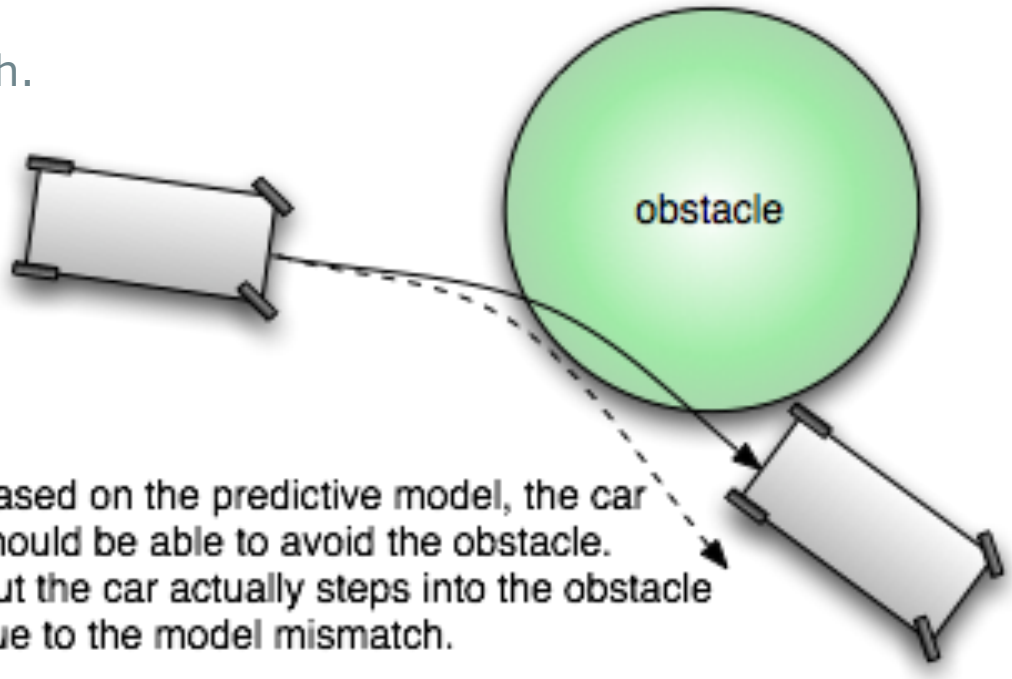
- A complex model may introduce predictive accuracy,
- however, this increases the computational burden.
- Especially under high speed, the system cannot tolerate a slow return rate.



The car is executing the previous control sequence while waiting for the new control sequence

Problems: control accuracy

- The return time problem can be addressed via model reduction,
  - potential drawback is higher model mismatch.
- Model mismatch can also introduce problems.
  - Wrong prediction
  - Infeasible trajectories



Model Predictive Control

$$\xi_{k,t+1} = \hat{f}_q(\xi_{k,t}, u_{k,t})$$
$$t \in \{k, k+1, \cdots, k+N-1\}$$

MPC solves the optimization problem $\mathbf{P}^q(\xi_k)$ at time k by using the model $\hat{f}_q$. We denote the input sequence $\{u^q_{k,k}, u^q_{k,k+1}, \cdots, u^q_{k,k+N-1}\}$ by U^q_k, and formulate the following problem:

$$\mathbf{P}^q(\xi_k) : \underset{U^q_k}{\operatorname{argmin}} \{J_N(\xi_k, U^q_k) : U^q_k \subset \mathbb{R}^m\}$$

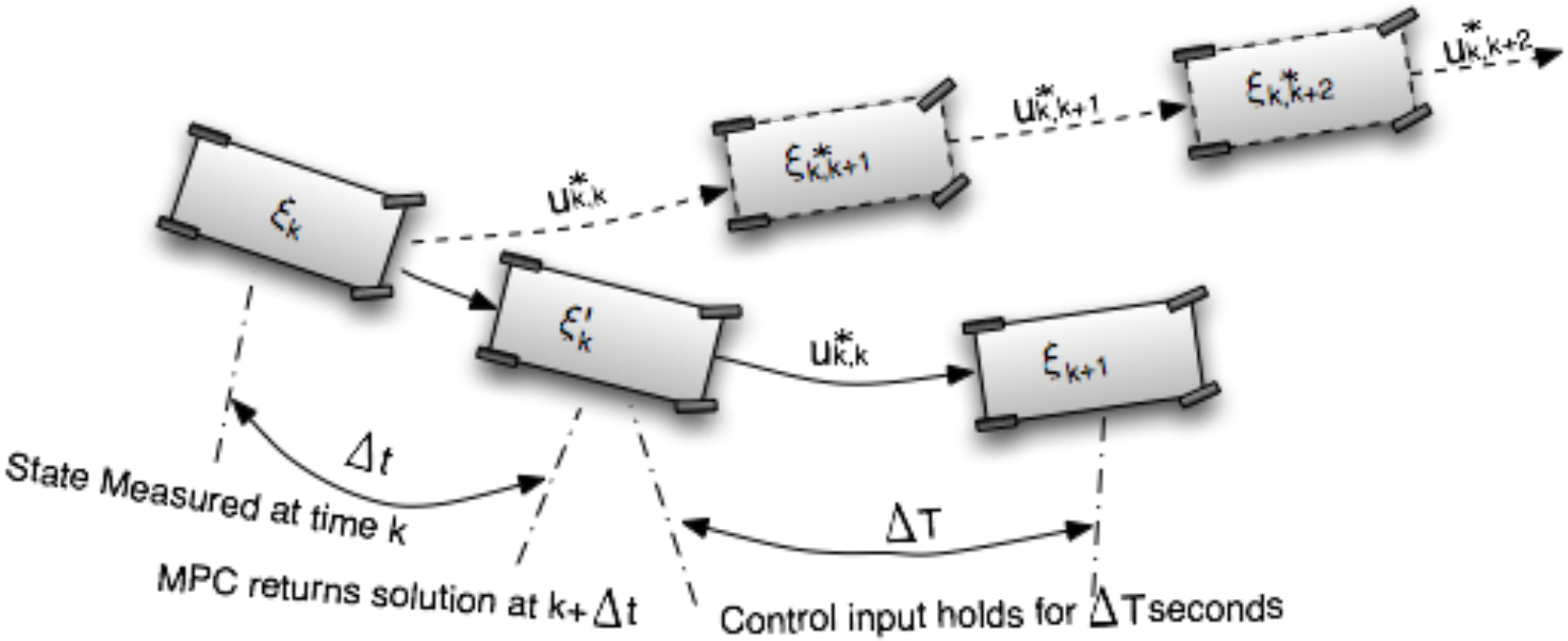
$$J_N(\xi_k, U^q_k) = \sum_{t=k}^{k+N-1} \ell(\xi^q_{k,t}, u^q_{k,t}) + F(\xi^q_{k,k+N})$$

$$U^{q*}_k = \{u^{q*}_{k,k}, u^{q*}_{k,k+1}, \cdots, u^{q*}_{k,k+N-1}\}$$

Problem Modeling

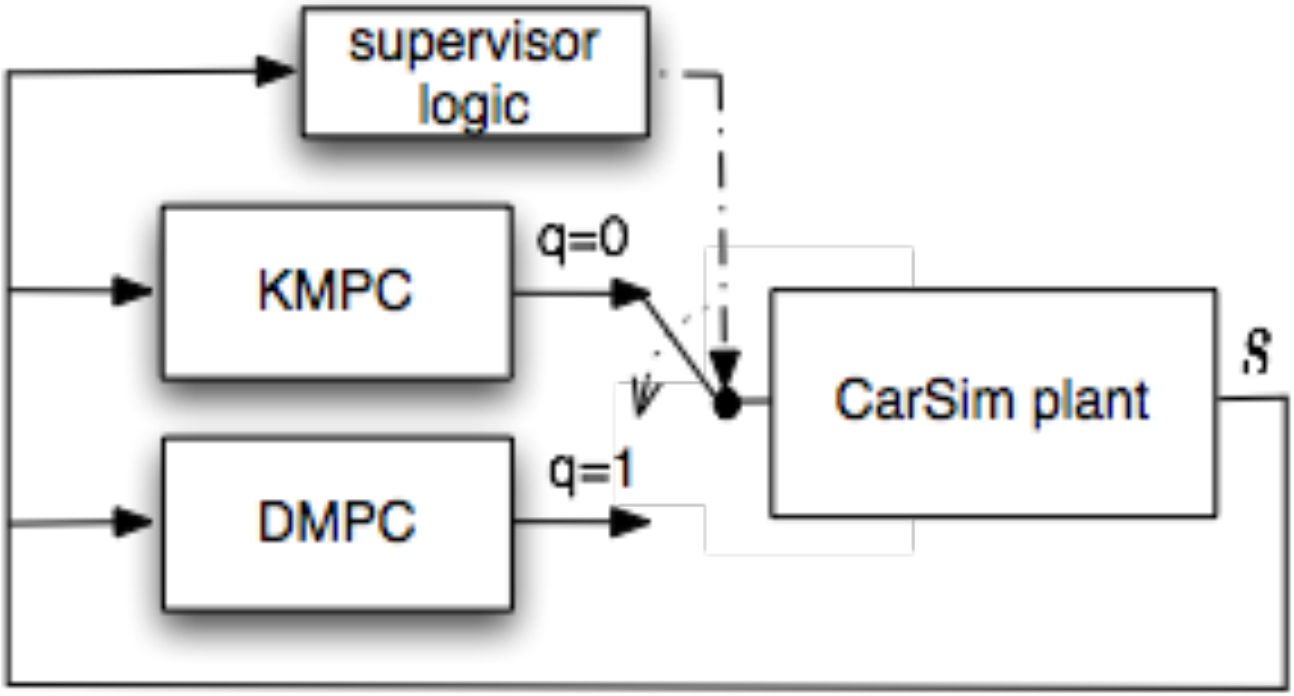
$$\xi'_k \approx \xi_k + f(\xi_k, u^*_{k-1,k})\Delta t$$

$$\begin{aligned}\xi_{k+1} &= \hat{f}\left(\xi'_k, \kappa_q(\xi_k)\right) = \hat{f}_q\left(\xi'_k, \kappa_q(\xi_k)\right) + \hat{\Gamma}_q(\xi'_k) \\ &\approx \hat{f}_q\left(\xi_k, \kappa_q(\xi_k)\right) + \hat{\Gamma}_q(\xi_k) + \left(\frac{\partial \hat{f}_q\left(\xi_k, \kappa_q(\xi_k)\right)}{\partial \xi} + \frac{\partial \hat{\Gamma}_q\left(\xi_k, \kappa_q(\xi_k)\right)}{\partial \xi}\right) f\left(\xi_k, \kappa_q(\xi_k)\right) \Delta t_q(\xi_k)\end{aligned}$$



Problem Statement

Suppose at time $k \in \{0, 1, \cdots\}$, the vehicle state ξ_k is observed for an optimization problem indexed by the predictive model in use (i.e., $\mathbf{P}^q(\xi_k)$), and that two alternative predictive models are available.



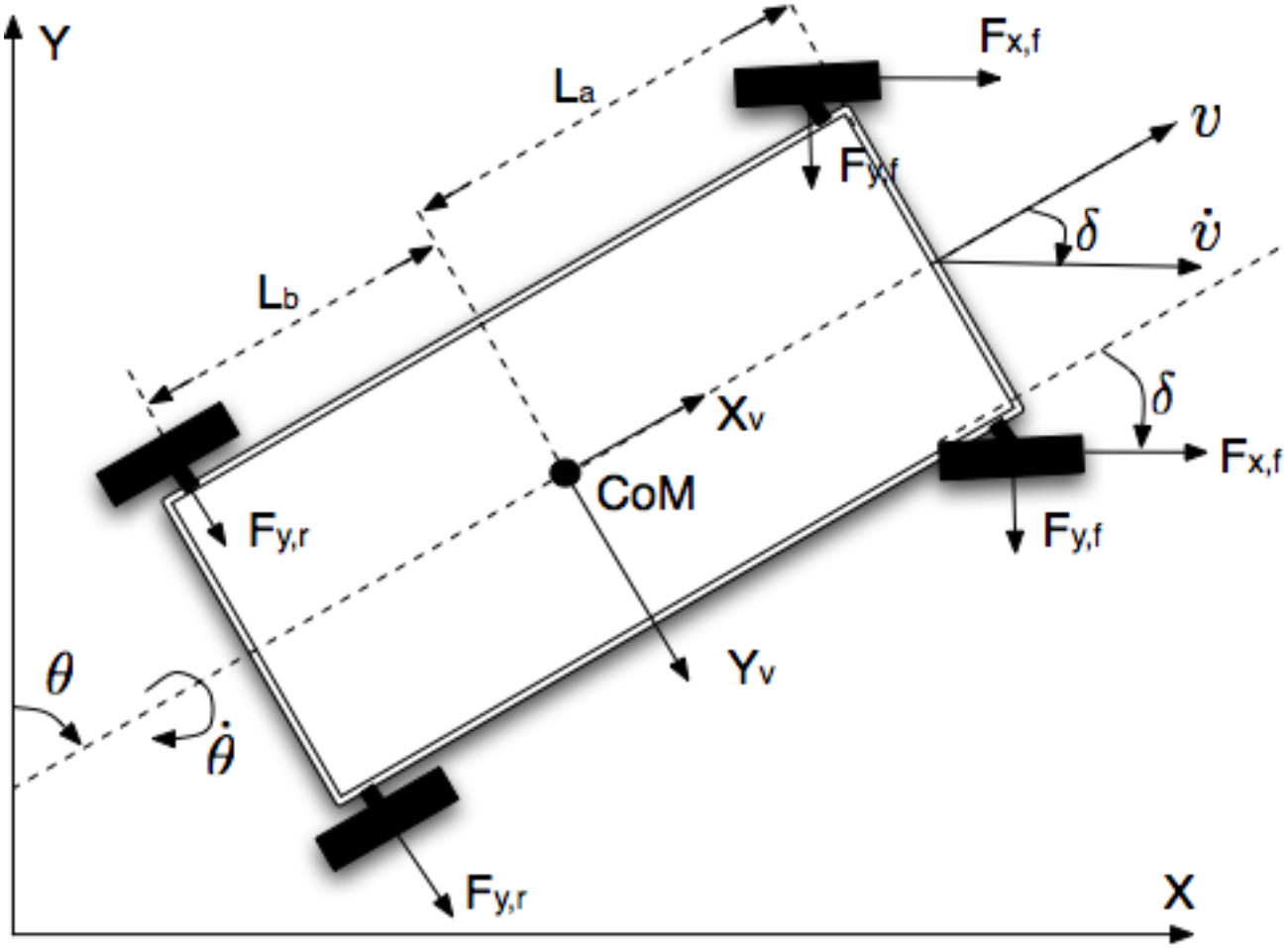
Problem: select the predictive model q such that the divergence of the state at ξ_k from the plant's state with the same inputs is minimized.

Kinematic/Dynamical Models

$$\dot{\xi} = \begin{bmatrix} v\sin\theta \\ v\cos\theta \\ \frac{v\tan\delta}{L} \end{bmatrix}$$

$$\dot{\xi} = \begin{bmatrix} v\sin(\theta) \\ v\cos(\theta) \\ \cos(\delta)a - \frac{2}{m}F_{y,f}\sin(\delta) \\ \varphi \\ \frac{1}{J}\left(L_a\left(ma\sin(\delta) + 2F_{y,f}\cos(\delta)\right) - 2L_bF_{y,r}\right) \\ \omega \end{bmatrix}$$

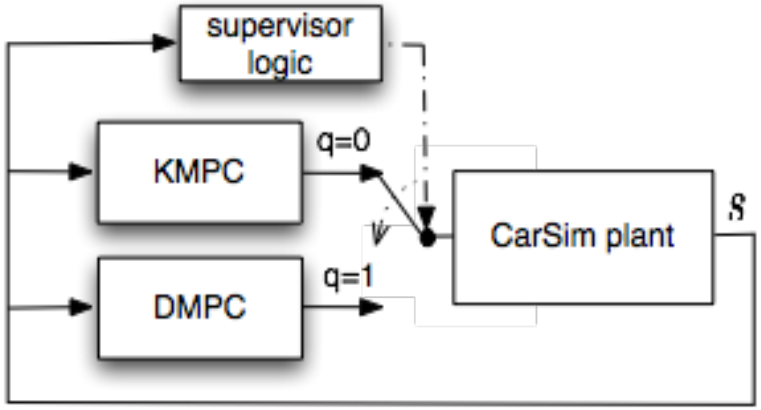
$$\dot{\xi} = \begin{bmatrix} v\sin(\theta) \\ v\cos(\theta) \\ \cos(\delta)a - \frac{2}{m}F_{y,f}\sin(\delta) \\ \varphi \\ \frac{1}{J}\left(L_a\left(ma\sin(\delta) + 2F_{y,f}\cos(\delta)\right) - 2L_bF_{y,r}\right) \\ \omega \end{bmatrix}$$



Hybrid MPC Design

$$\dot{\xi}^{q=0} = \begin{bmatrix} v\sin\theta \\ v\cos\theta \\ \frac{v\tan\delta}{L} \end{bmatrix} \qquad \dot{\xi}^{q=1} = \begin{bmatrix} v\sin(\theta) \\ v\cos(\theta) \\ \cos(\delta)a - \frac{2}{m}F_{y,f}\sin(\delta) \\ \varphi \\ \frac{1}{J}\left(L_a\left(ma\sin(\delta) + 2F_{y,f}\cos(\delta)\right) - 2L_bF_{y,r}\right) \\ \omega \end{bmatrix}$$

$$q = \underset{q}{\operatorname{argmin}} \left\| \xi_{k+1} - \xi_{k,k+1}^{q*} \right\|$$



Hybrid MPC Design

$$q = \underset{q}{\operatorname{argmin}} \left\| \xi_{k+1} - \xi_{k,k+1}^{q*} \right\|$$

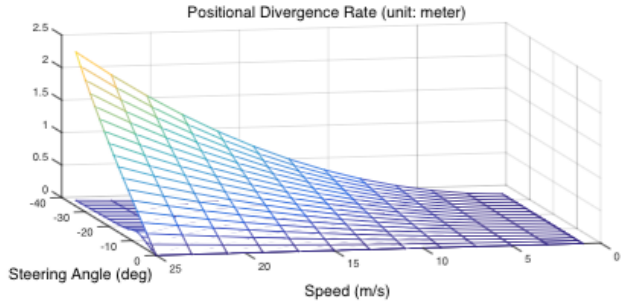
To make this decision, we need to know two things:

(1) Model mismatch **(2) Return time for MPC for this model**

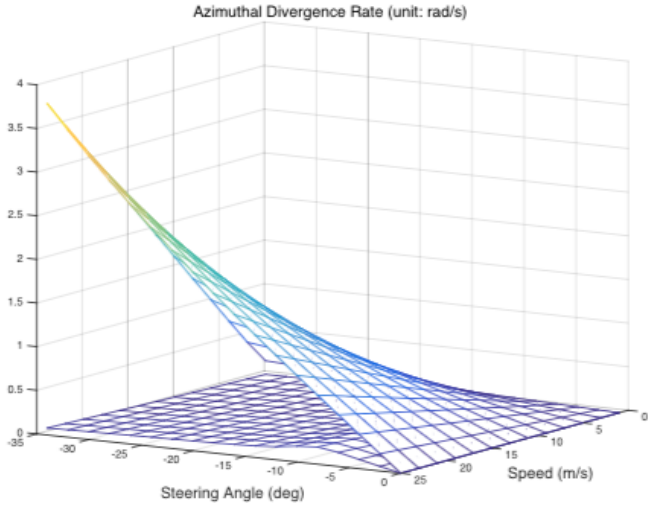
$$\left\| \xi_{k+1} - \xi_{k,k+1}^{q*} \right\| \leqslant \left\| \hat{\Gamma}_q(\xi_k) \right\| + \left\| I + \frac{\partial f\left(\xi_k, \kappa_q(\xi_k)\right)}{\partial \xi} \Delta T \right\| \left\| f\left(\xi_k, \kappa_q(\xi_k)\right) \right\| \Delta t_q(\xi_k)$$

$$\left\| \xi_{k+1} - \xi_{k,k+1}^{q*} \right\| \approx \left\| \hat{\Gamma}_q(\xi_k) \right\| + v \Delta t_q(\xi_k) \sqrt{1 + \left(\frac{\tan\delta}{L} v \Delta T \right)^2}$$

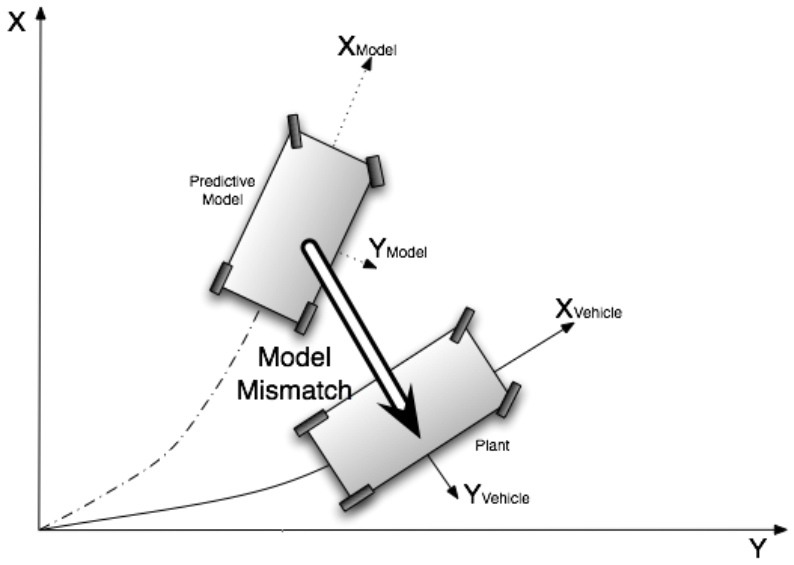
Model Mismatch & Return Time

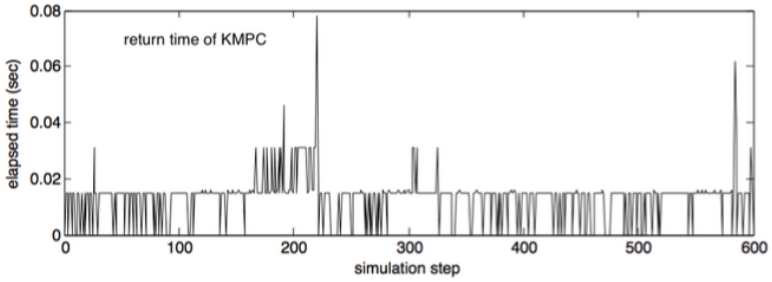


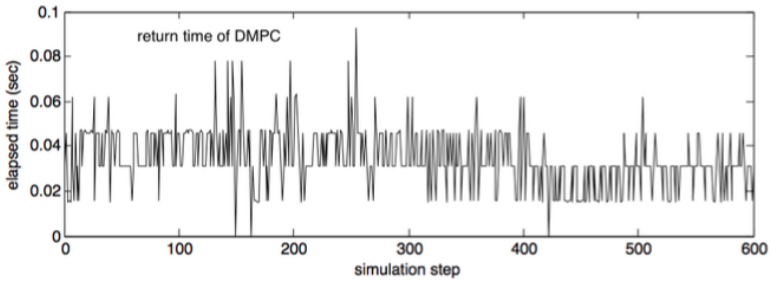
(a)



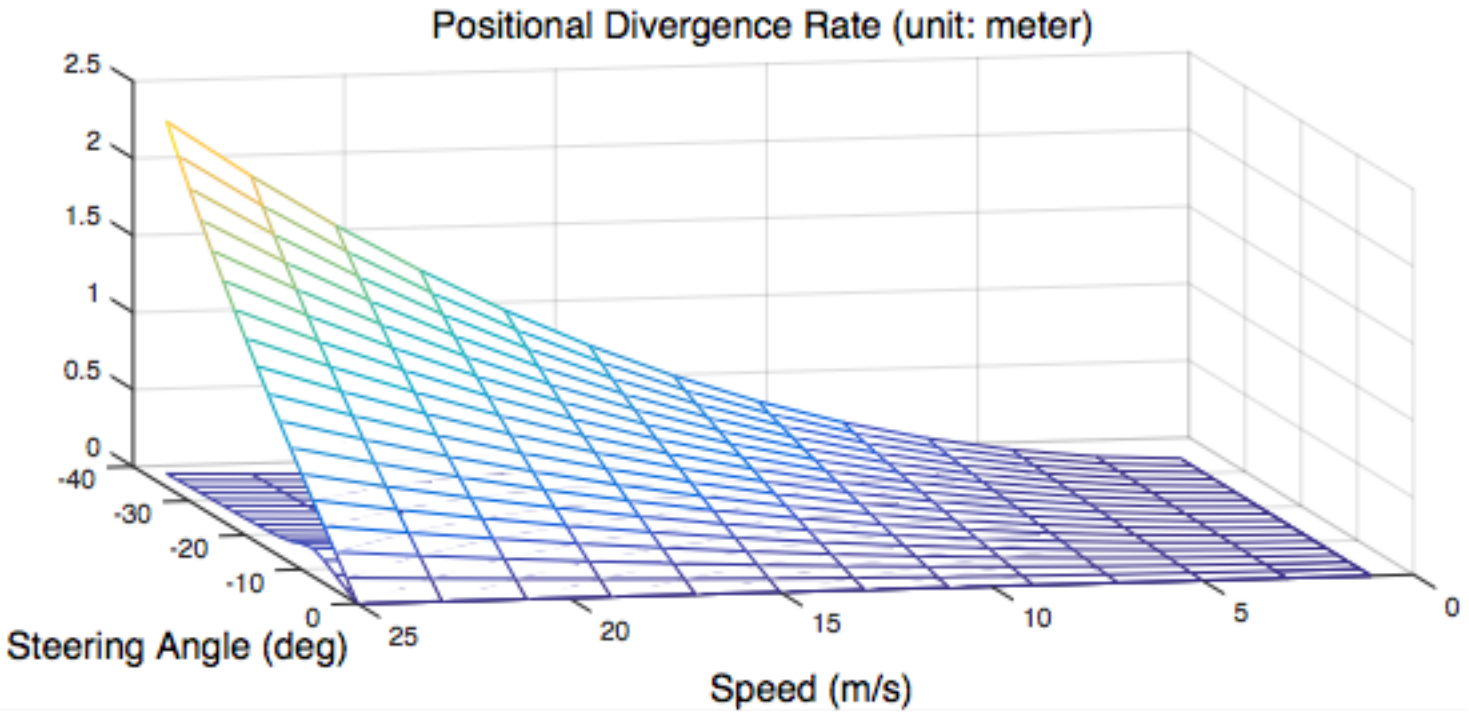
(b)

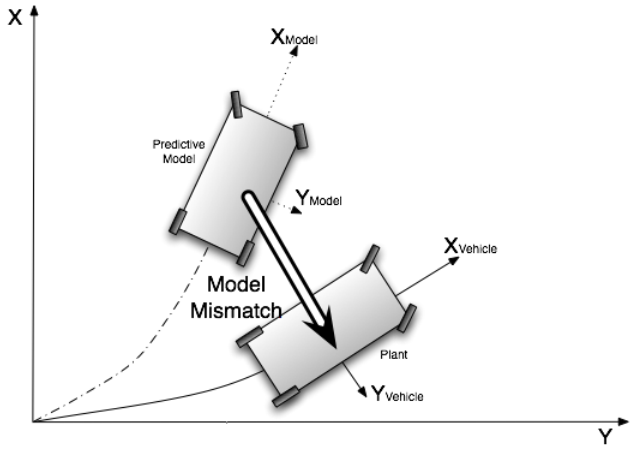




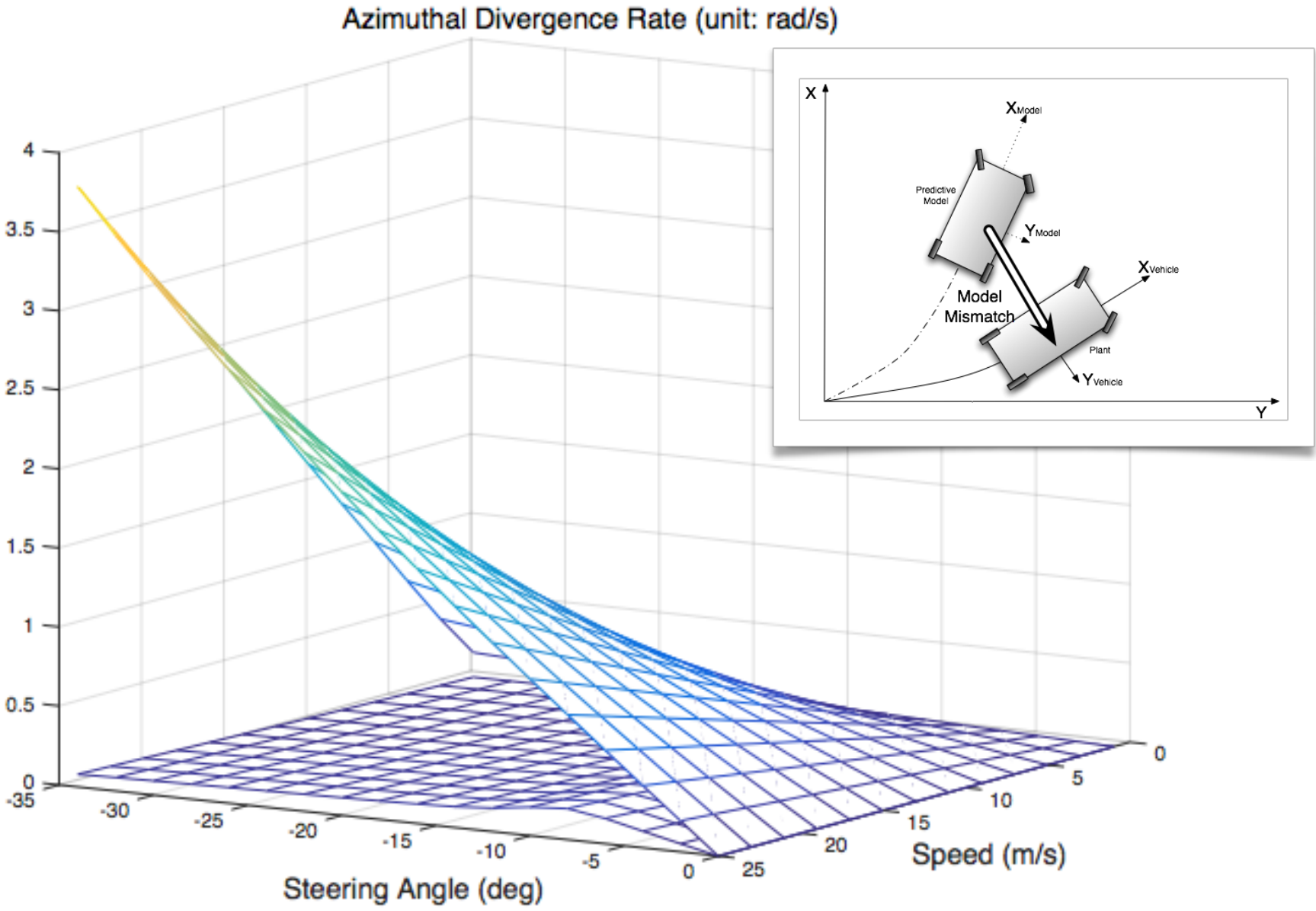


Model Mismatch

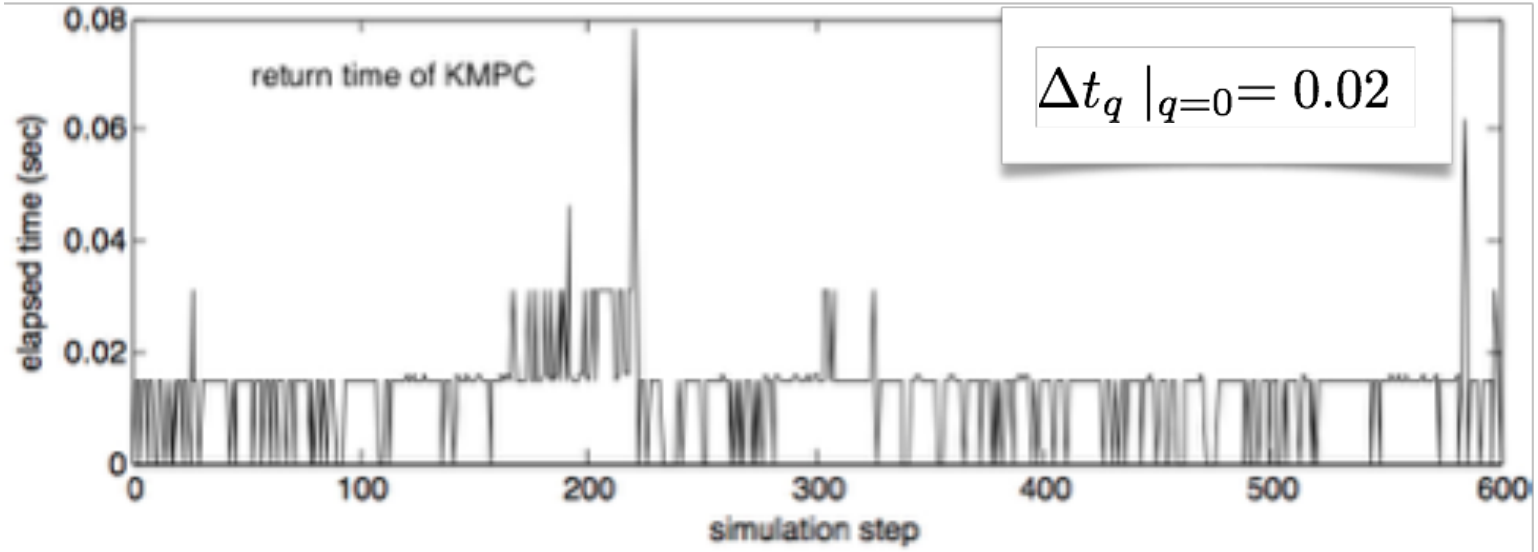


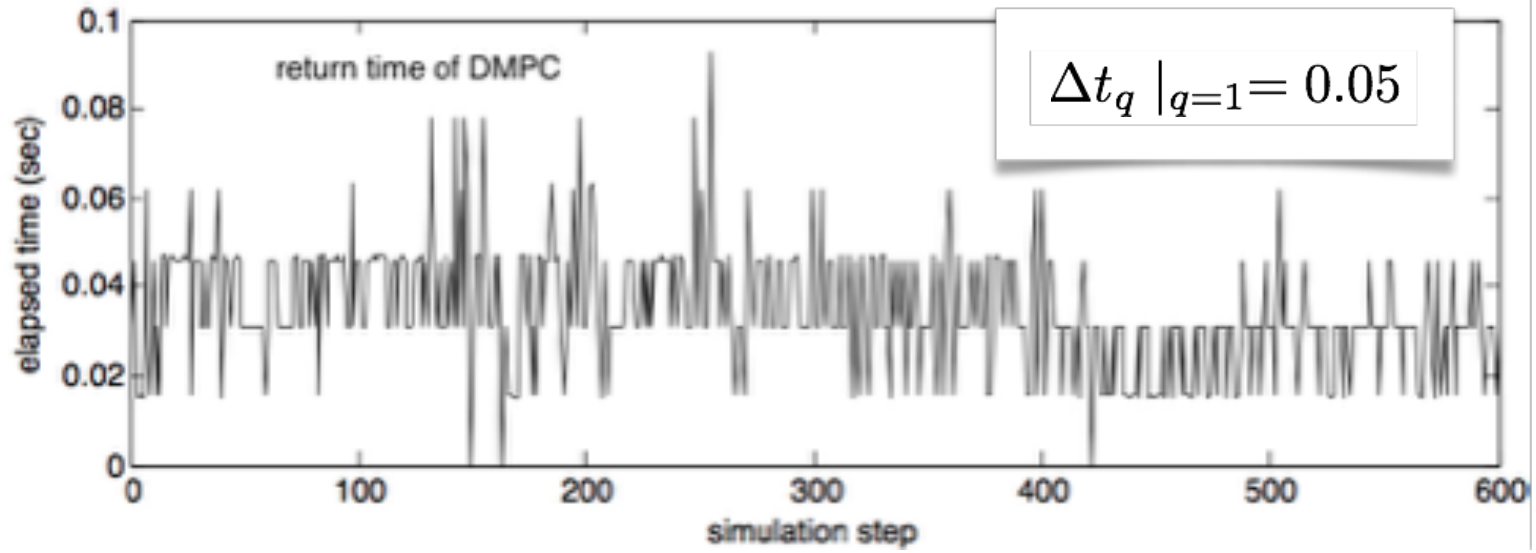


Model Mismatch

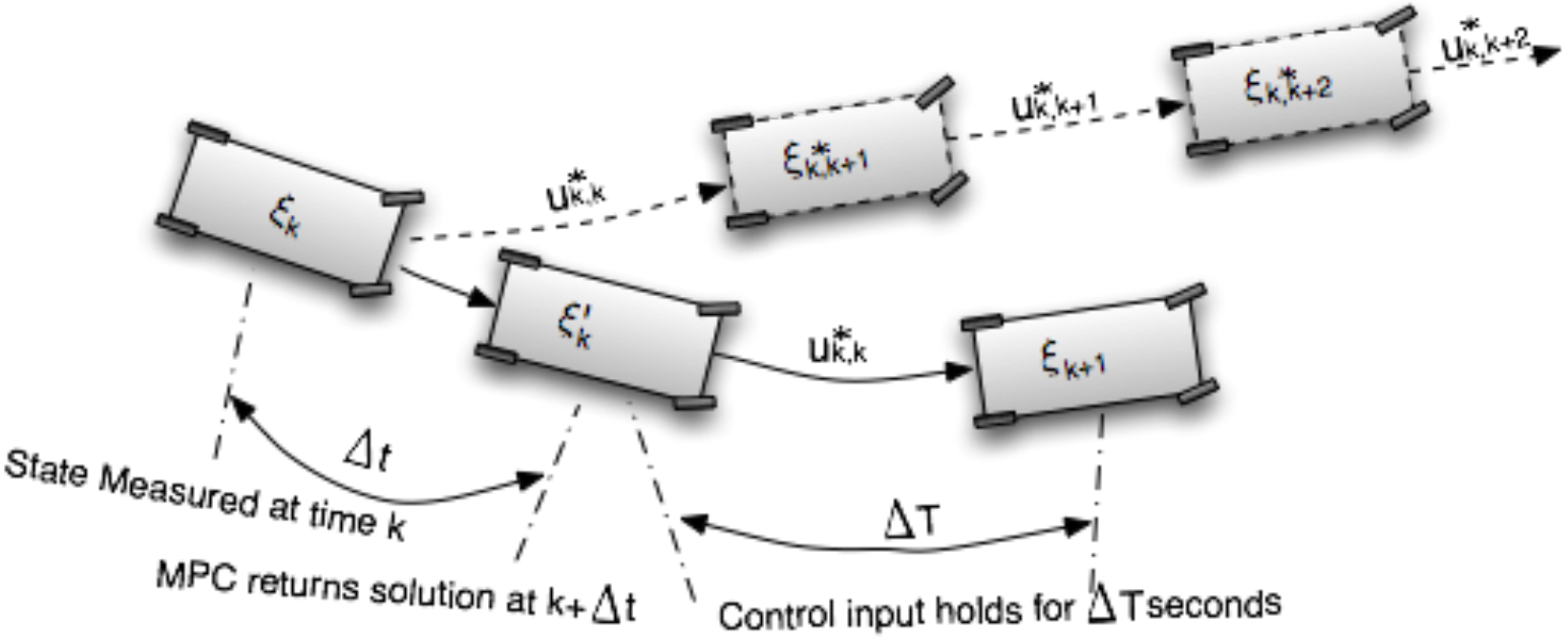


Time to return





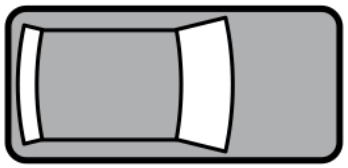
Uncontrollable divergence

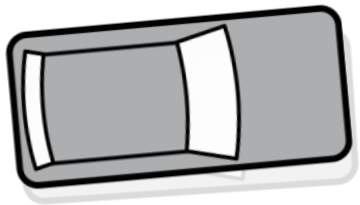


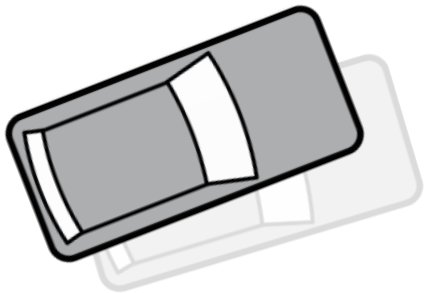
$$\left\| \xi_{k+1} - \xi^{q*}_{k,k+1} \right\|$$

Uncontrollable divergence: DMPC

$\Delta t_q = 0.05$ (DMPC value)





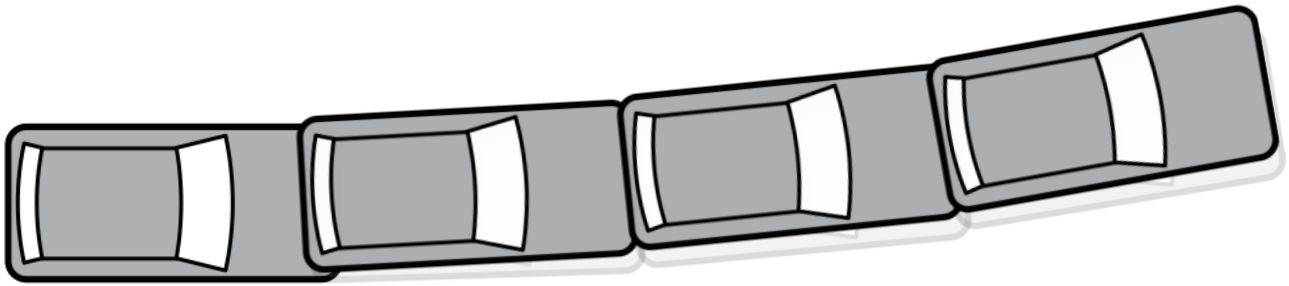


$$\dot{\xi}^{q=1} = \begin{bmatrix} v\sin(\theta) \\ v\cos(\theta) \\ \cos(\delta)a - \frac{2}{m}F_{y,f}\sin(\delta) \\ \varphi \\ \frac{1}{J}\left(L_a\left(ma\sin(\delta) + 2F_{y,f}\cos(\delta)\right) - 2L_bF_{y,r}\right) \\ \omega \end{bmatrix}$$

$$\left\| \xi_{k+1} - \xi^{q*}_{k,k+1} \right\|$$

Uncontrollable divergence: KMPC

$\Delta t_q = 0.02$ (KMPC value…)

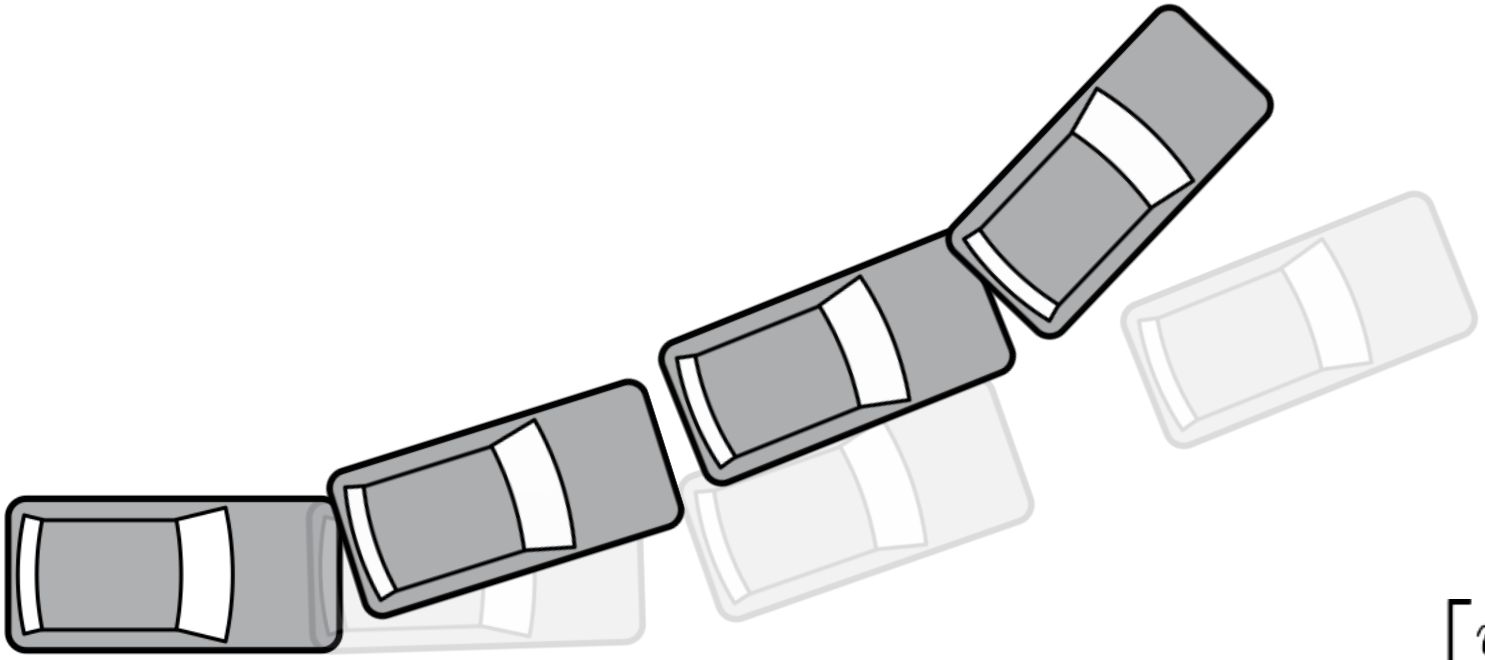


$$\dot{\xi}^{q=0} = \begin{bmatrix} v\sin\theta \\ v\cos\theta \\ \frac{v\tan\delta}{L} \end{bmatrix}$$

$$\left\| \xi_{k+1} - \xi^{q*}_{k,k+1} \right\|$$

Uncontrollable divergence: KMPC with large steering

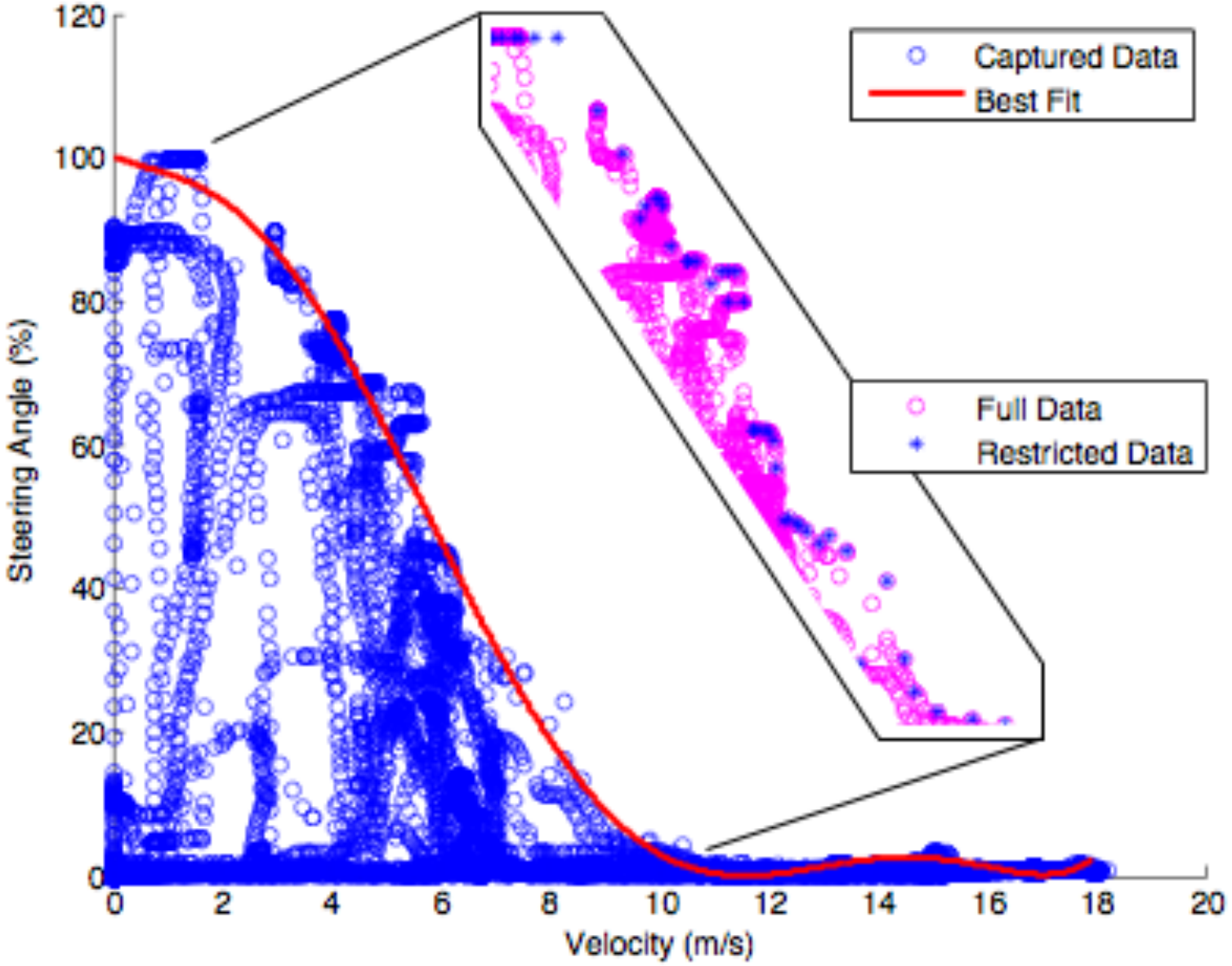
$\Delta t_q = 0.02$ (KMPC value…)



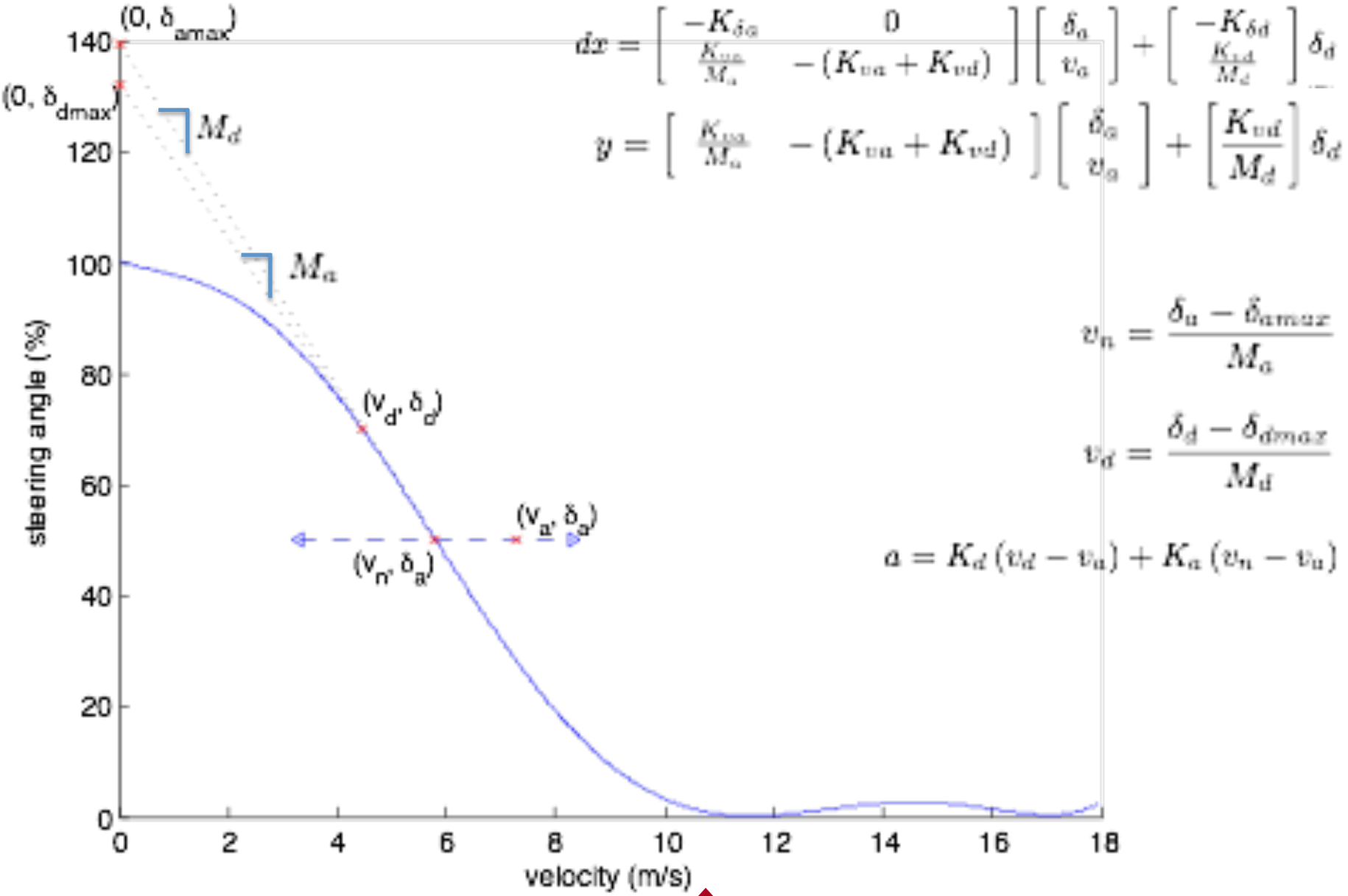
$$\dot{\xi}^{q=0} = \begin{bmatrix} v\sin\theta \\ v\cos\theta \\ \frac{v\tan\delta}{L} \end{bmatrix}$$

$$\left\| \left| \xi_{k+1} - \xi^{q*}_{k,k+1} \right| \right\|$$

Fortunate Previous Result from [1]



Take the fit data and utilize linearization techniques



$$\dot{x} = \begin{bmatrix} -K_{\delta a} & 0 \\ \frac{K_{va}}{M_a} & -(K_{va} + K_{vd}) \end{bmatrix} \begin{bmatrix} \delta_a \\ v_a \end{bmatrix} + \begin{bmatrix} -K_{\delta d} \\ \frac{K_{vd}}{M_d} \end{bmatrix} \delta_d$$

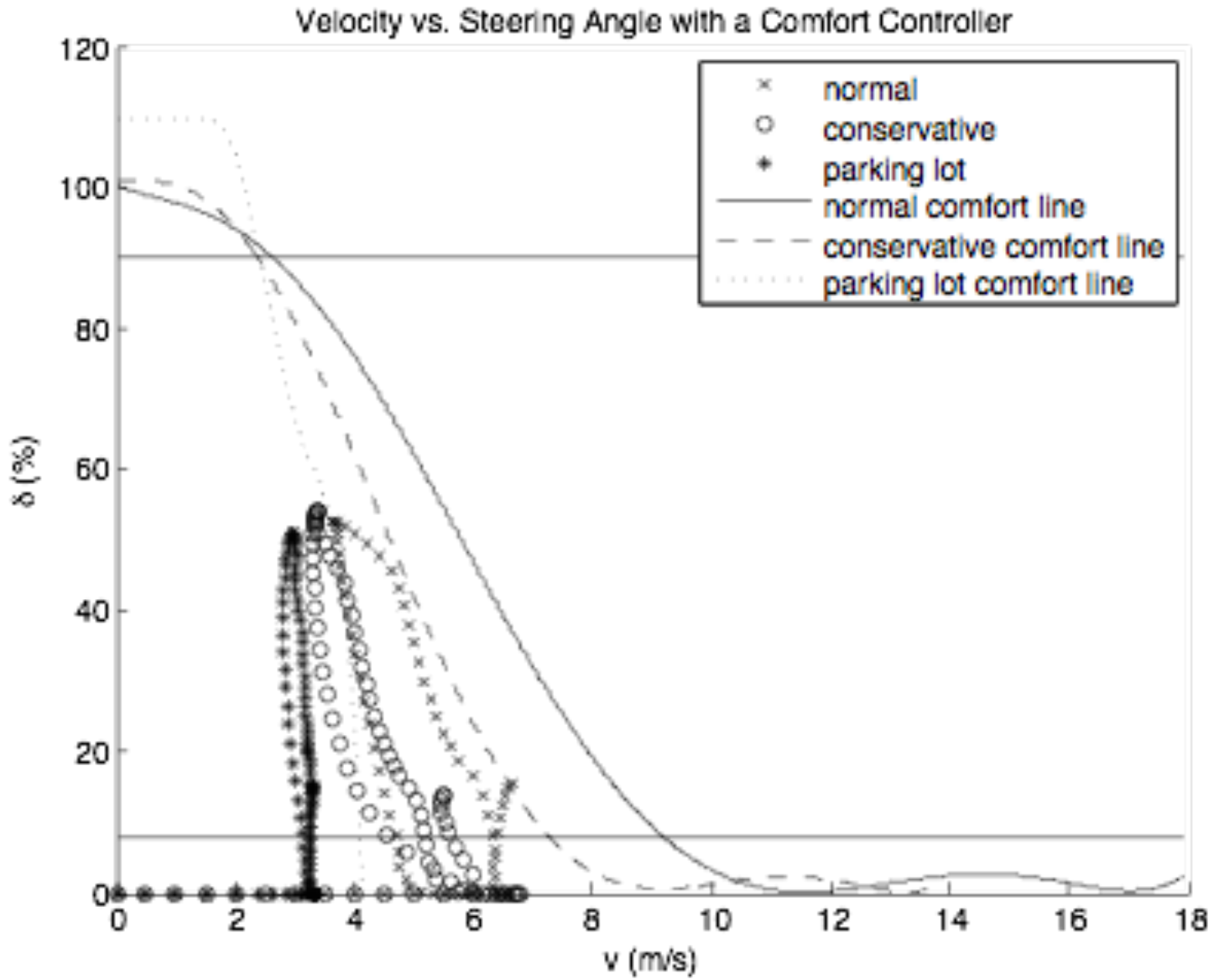
$$y = \begin{bmatrix} \frac{K_{va}}{M_a} & -(K_{va} + K_{vd}) \end{bmatrix} \begin{bmatrix} \delta_a \\ v_a \end{bmatrix} + \begin{bmatrix} \frac{K_{vd}}{M_d} \end{bmatrix} \delta_d$$

$$v_n = \frac{\delta_a - \delta_{amax}}{M_a}$$

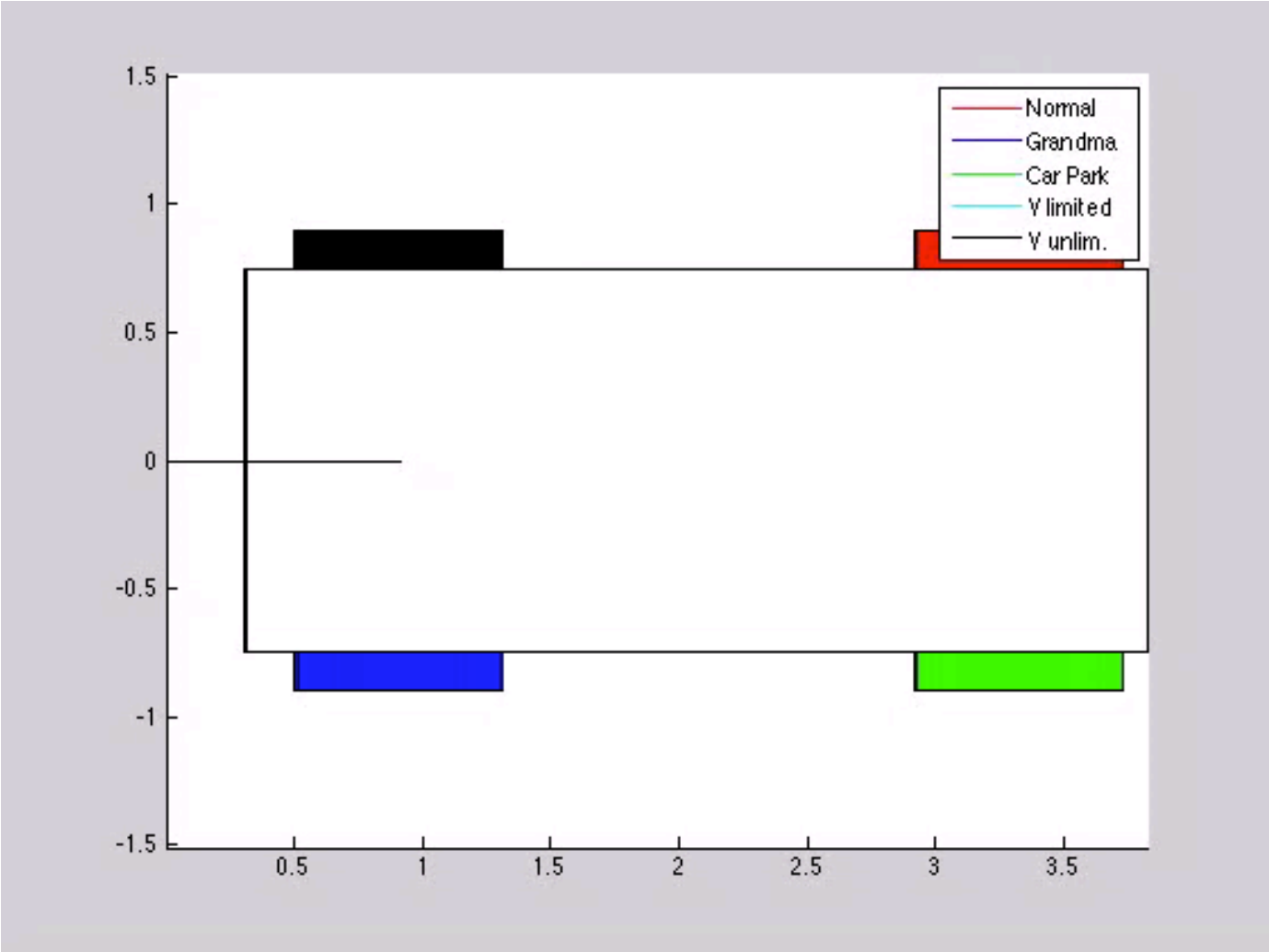
$$v_d = \frac{\delta_d - \delta_{dmax}}{M_d}$$

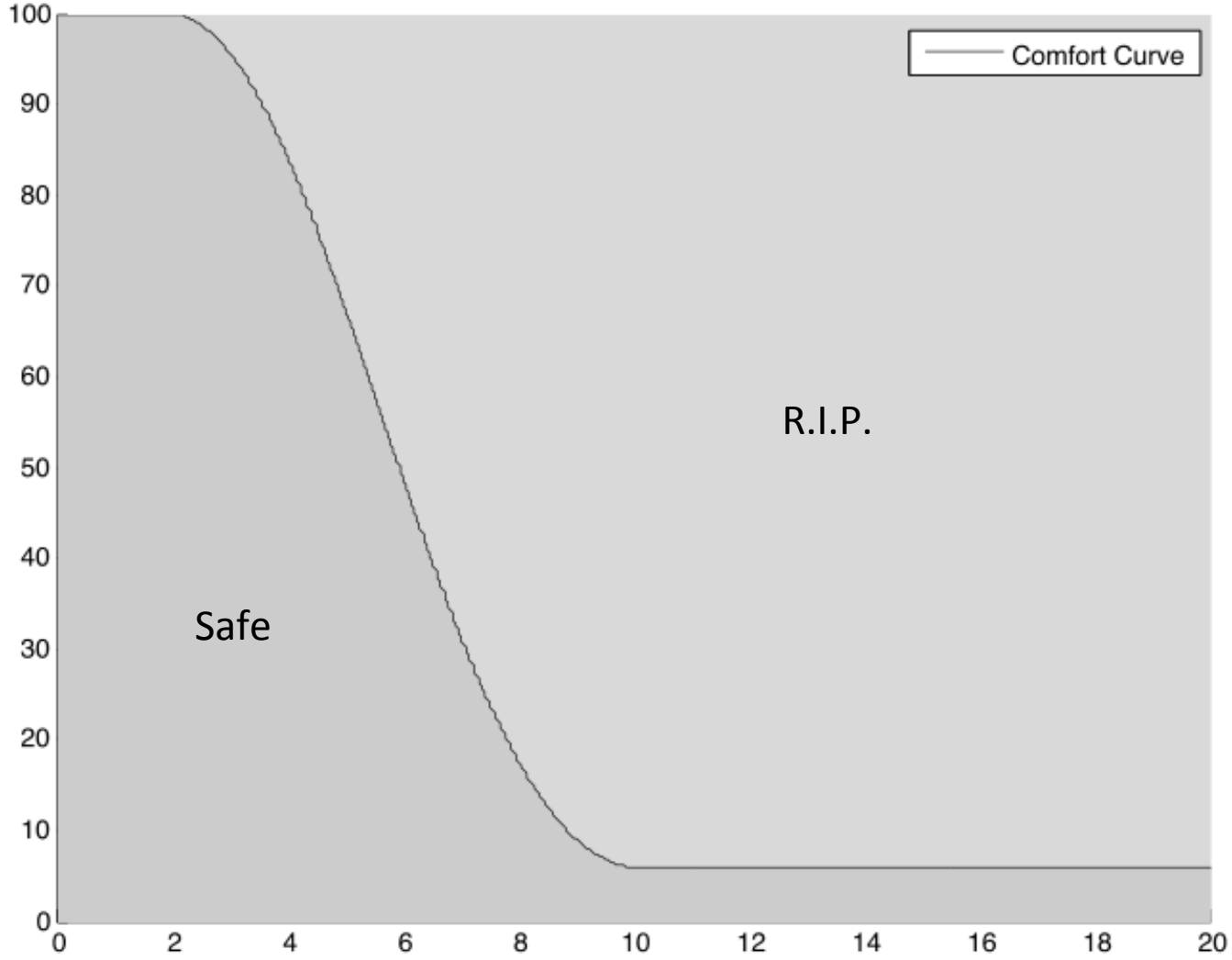
$$a = K_d (v_d - v_a) + K_a (v_n - v_a)$$

Scatter plot with comfort controller

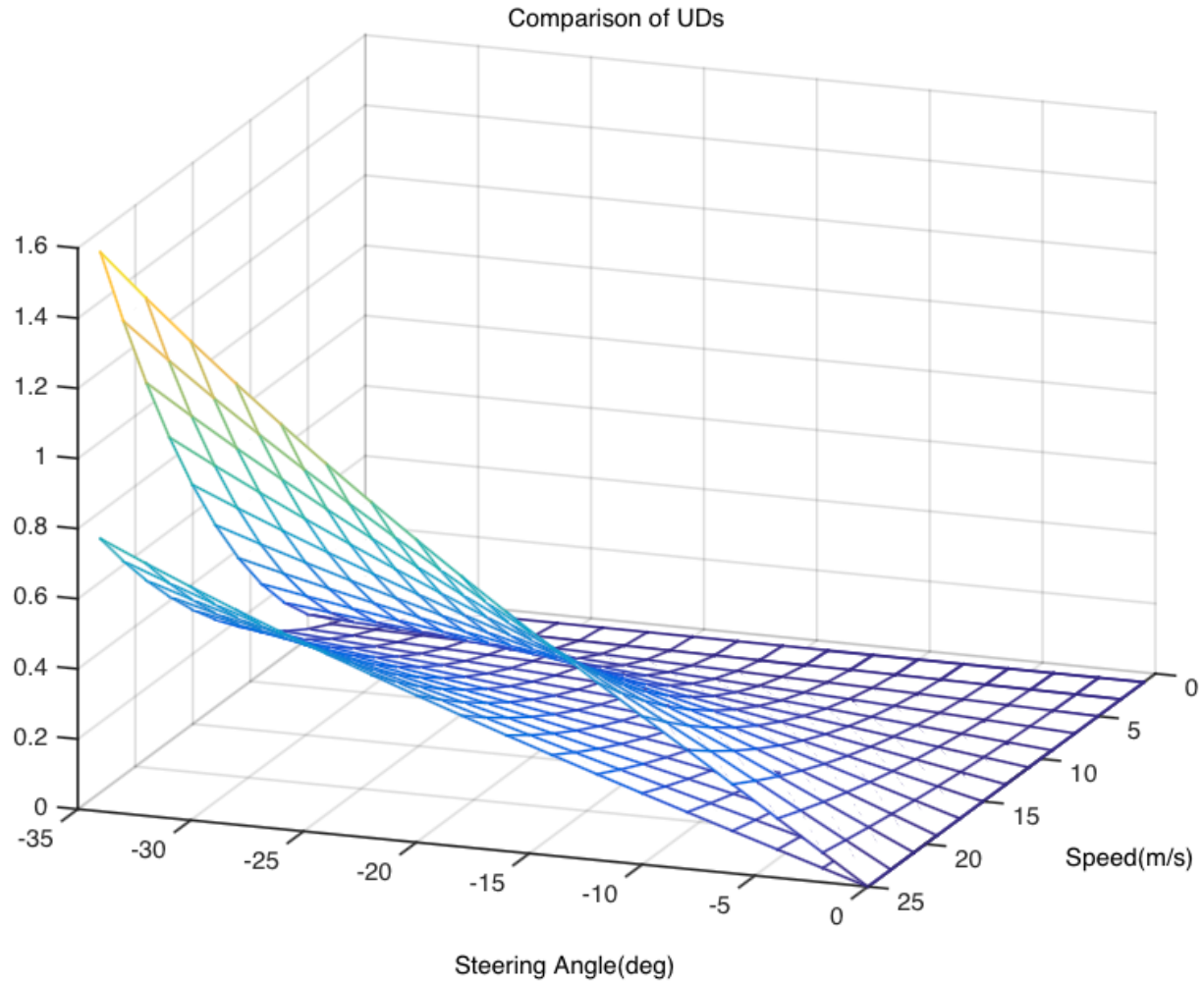


Simulation

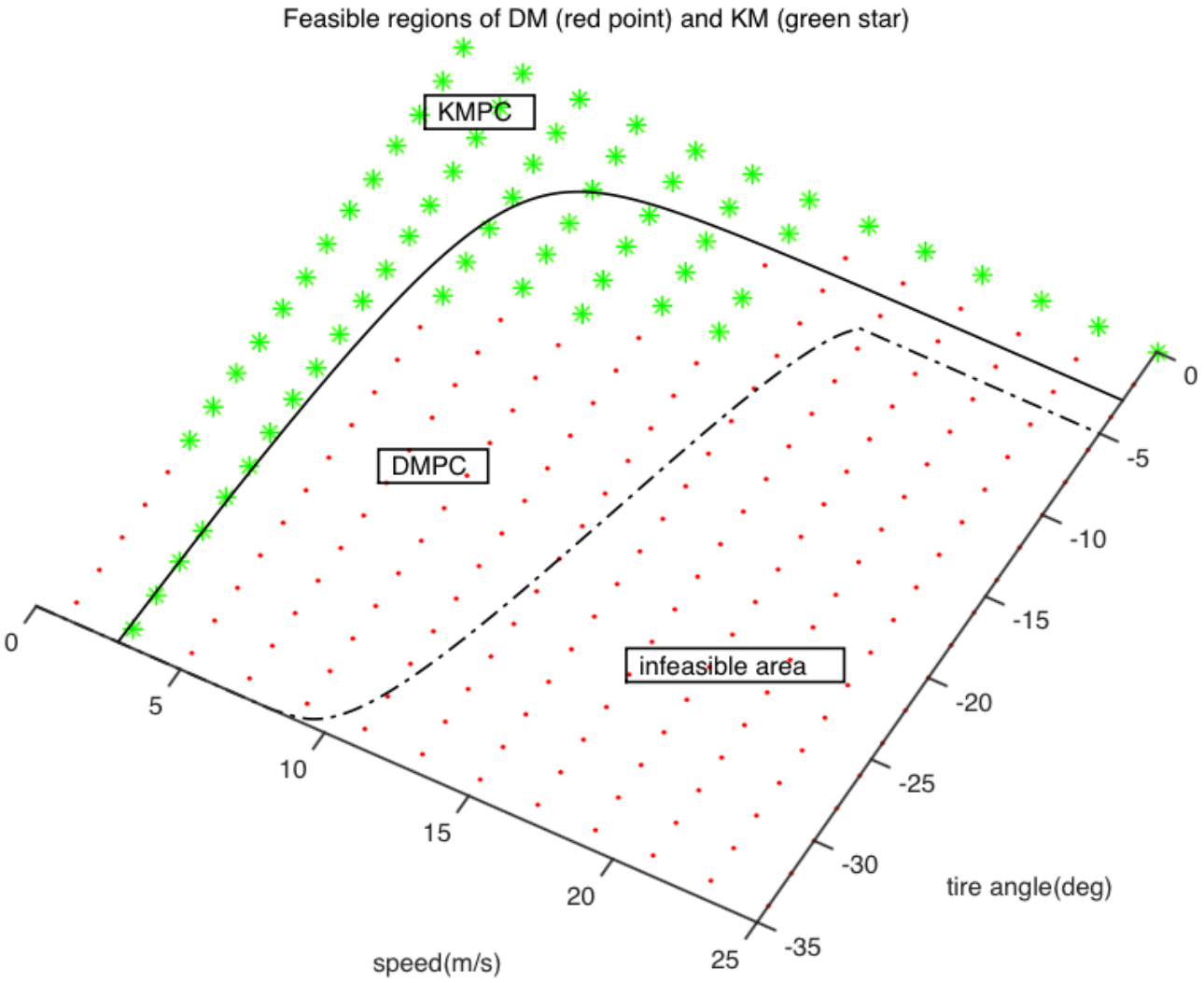

Normal
Grandma
Car Park
V limited
V unlim.
1.5
1
0.5
0
-0.5
-1
-1.5
0.5
1
1.5
2
2.5
3
3.5


Comfort Curve
R.I.P.
Safe
100
90
80
70
60
50
40
30
20
10
0
0
2
4
6
8
10
12
14
16
18
20

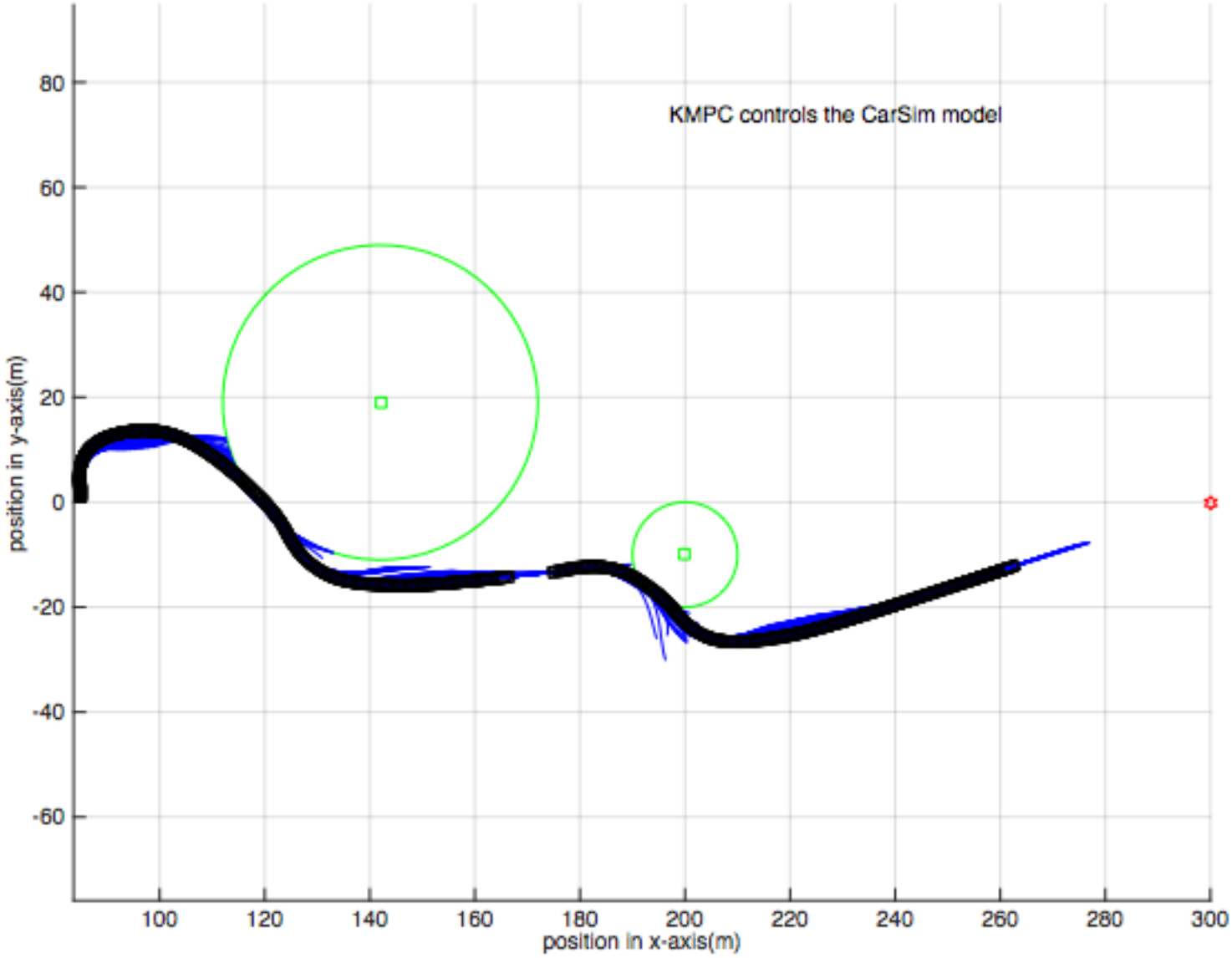
Hybrid MPC Design



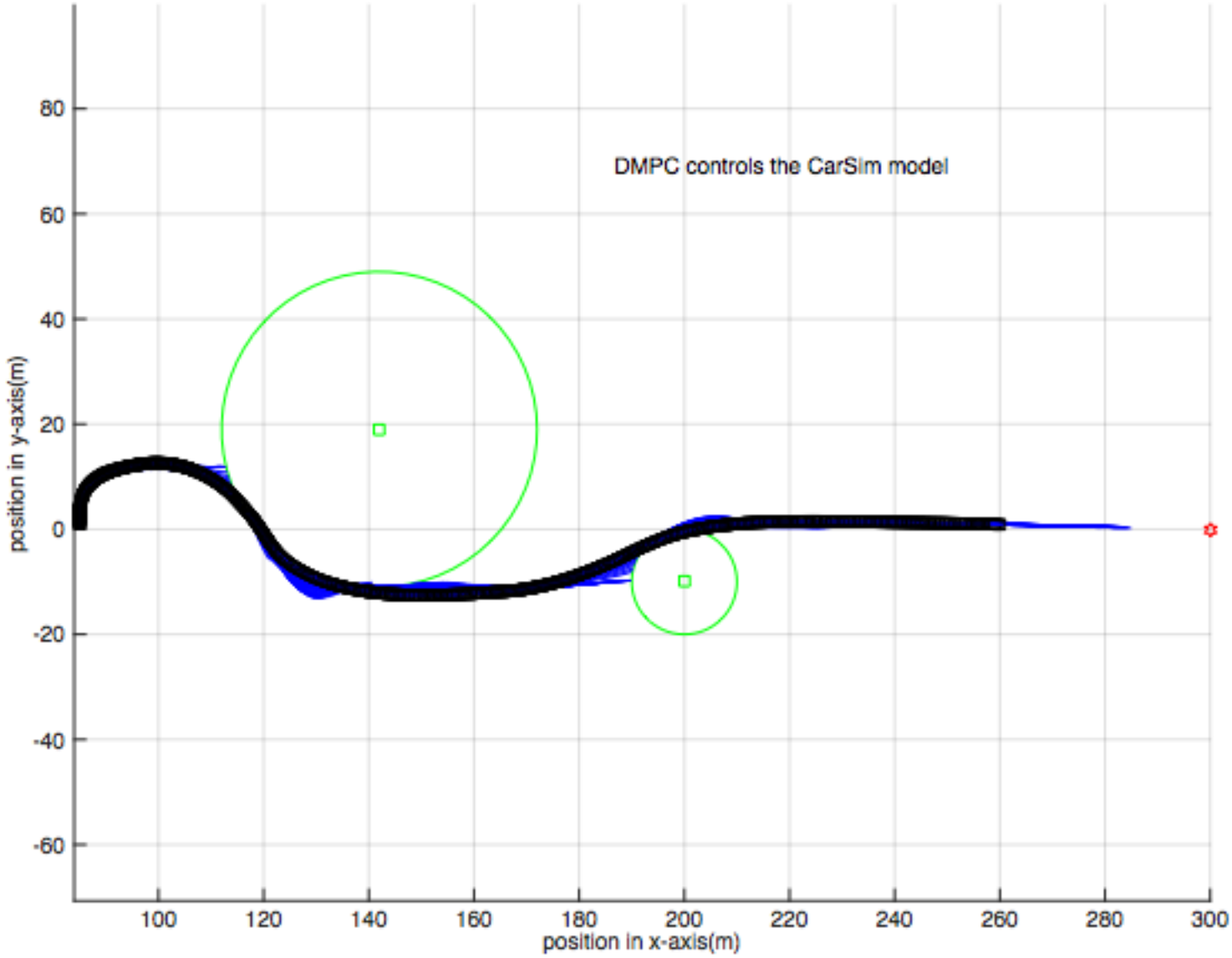
Hybrid MPC Design



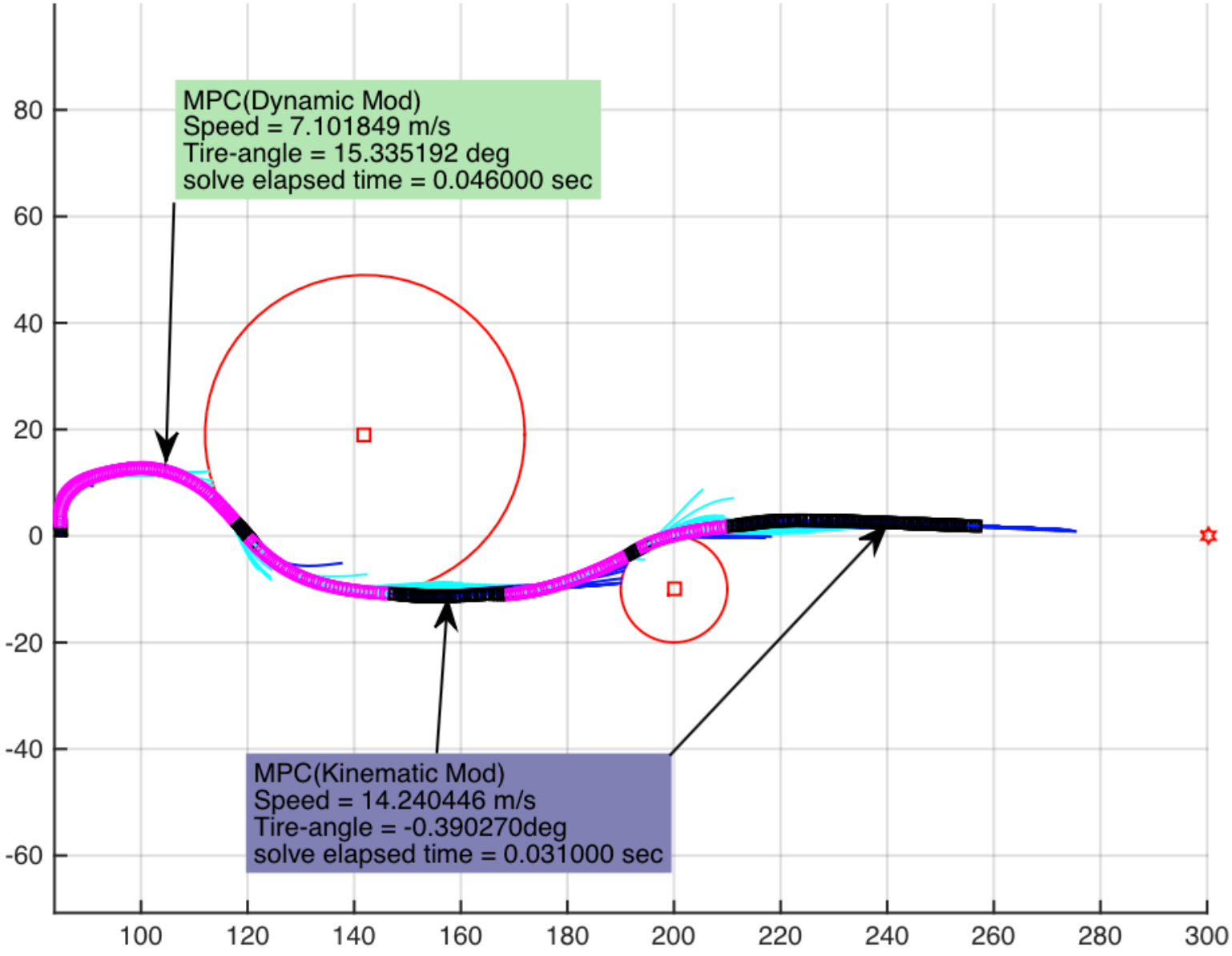
Simulation Results



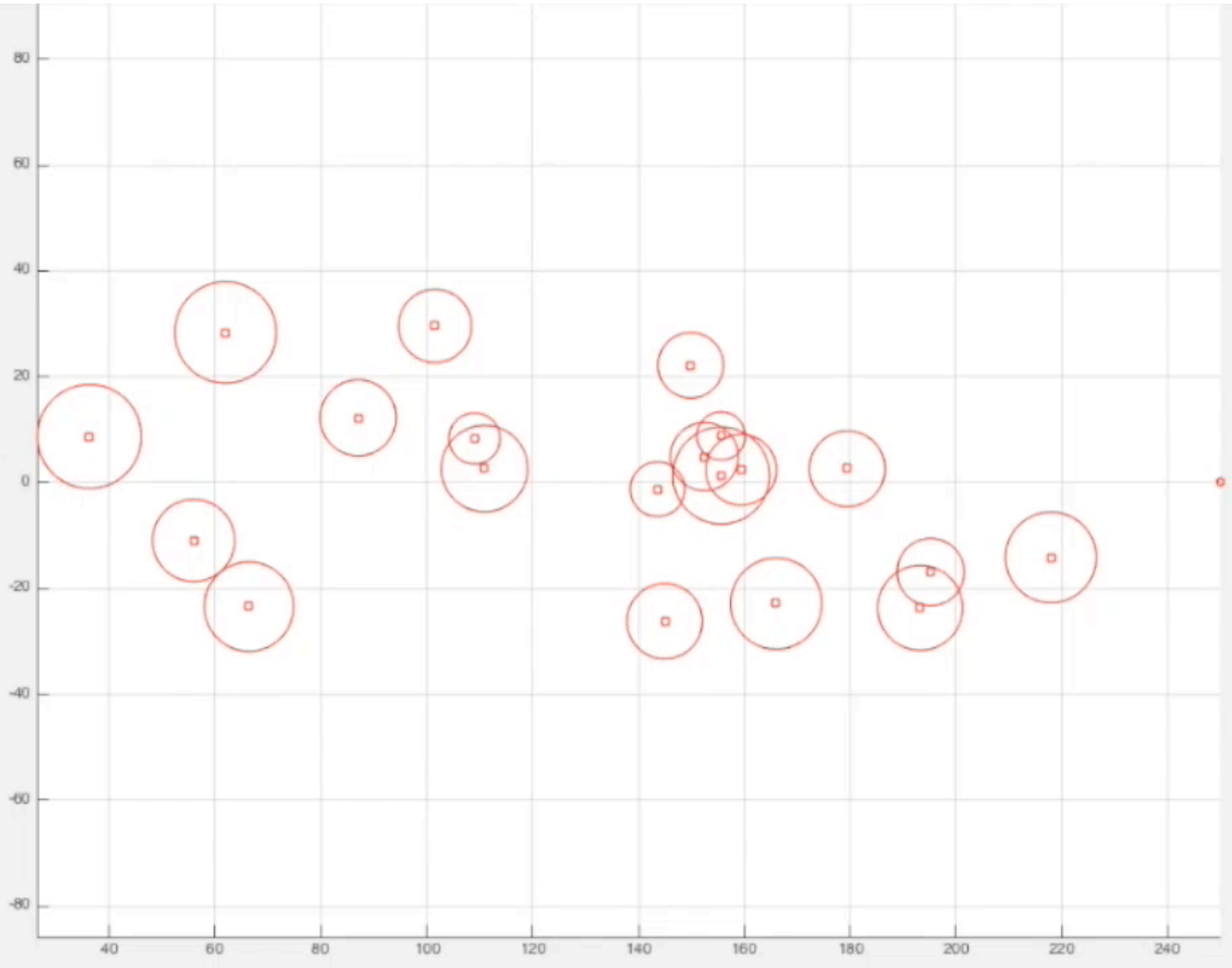
Simulation Result



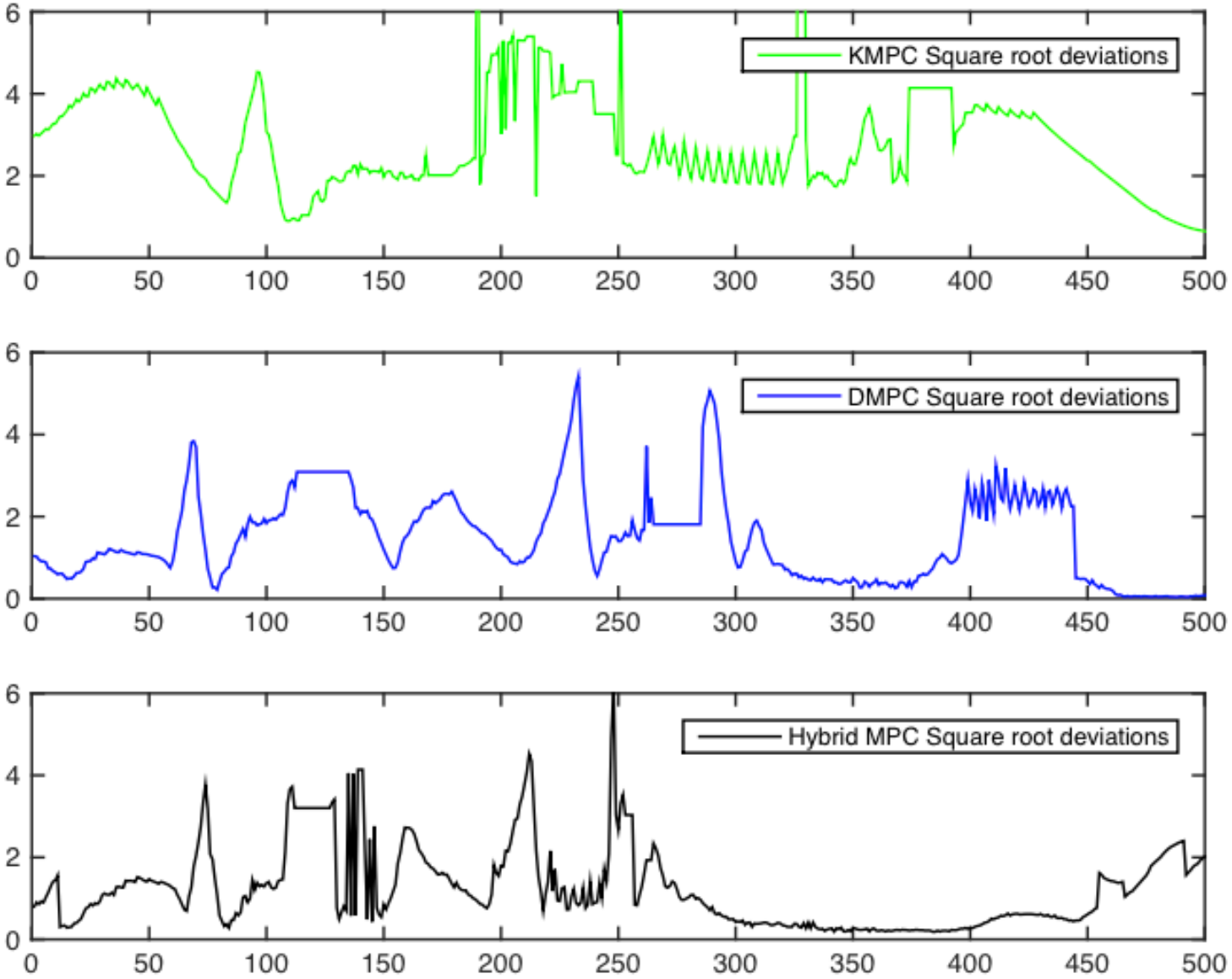
Simulation Result



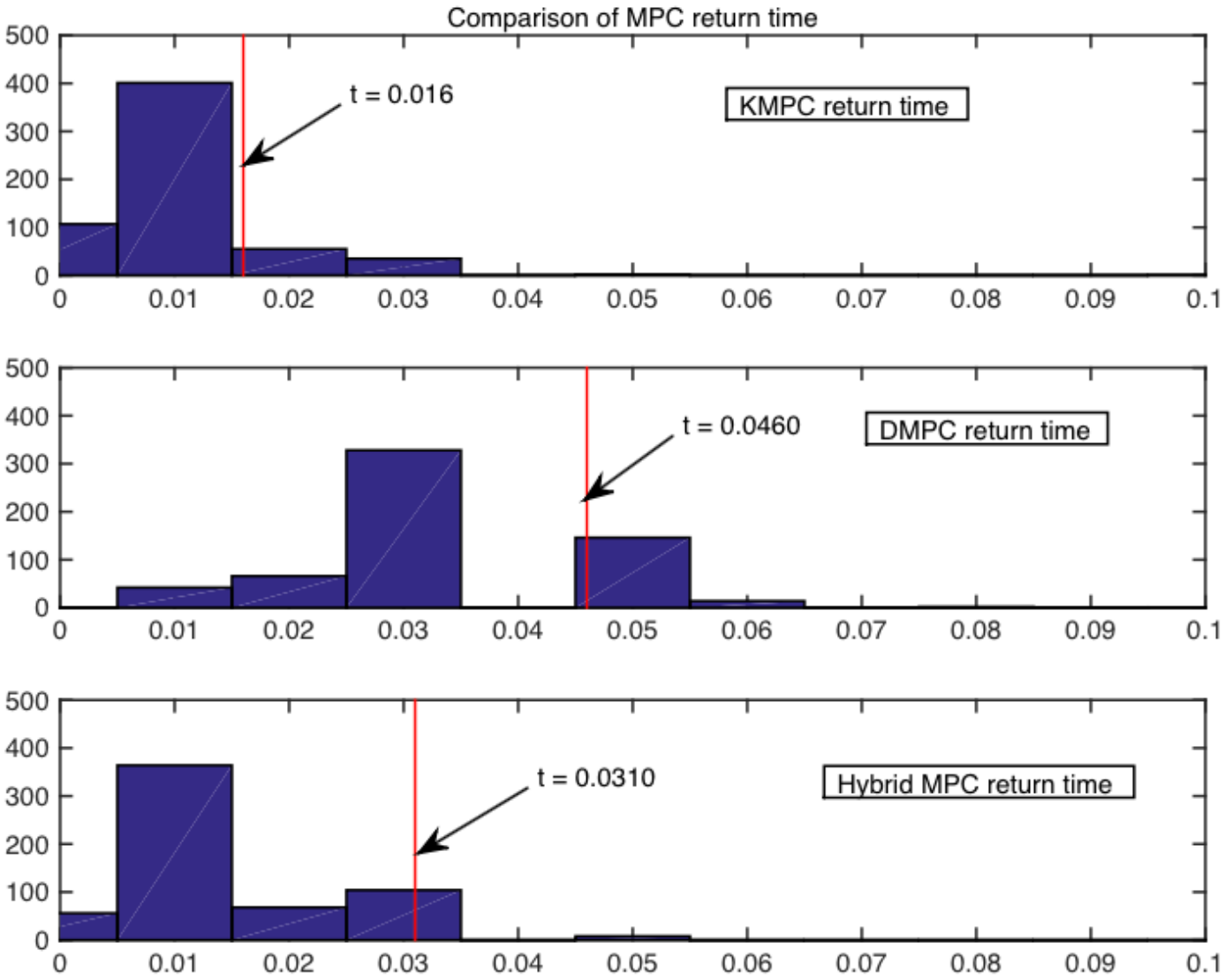
What about lots of obstacles?



Simulation Result



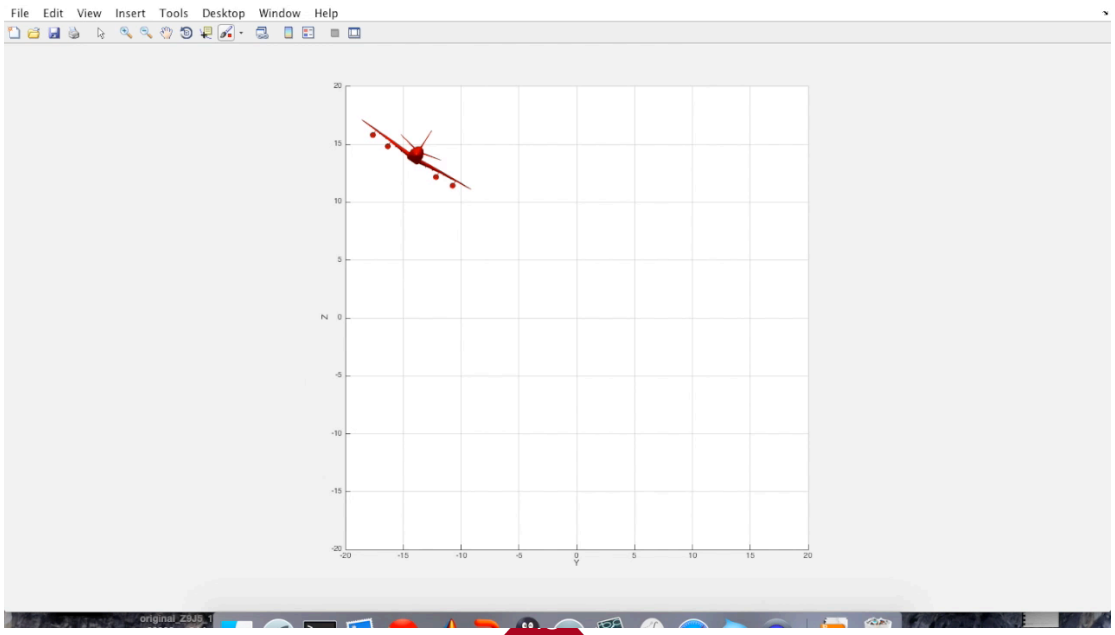
Simulation Result



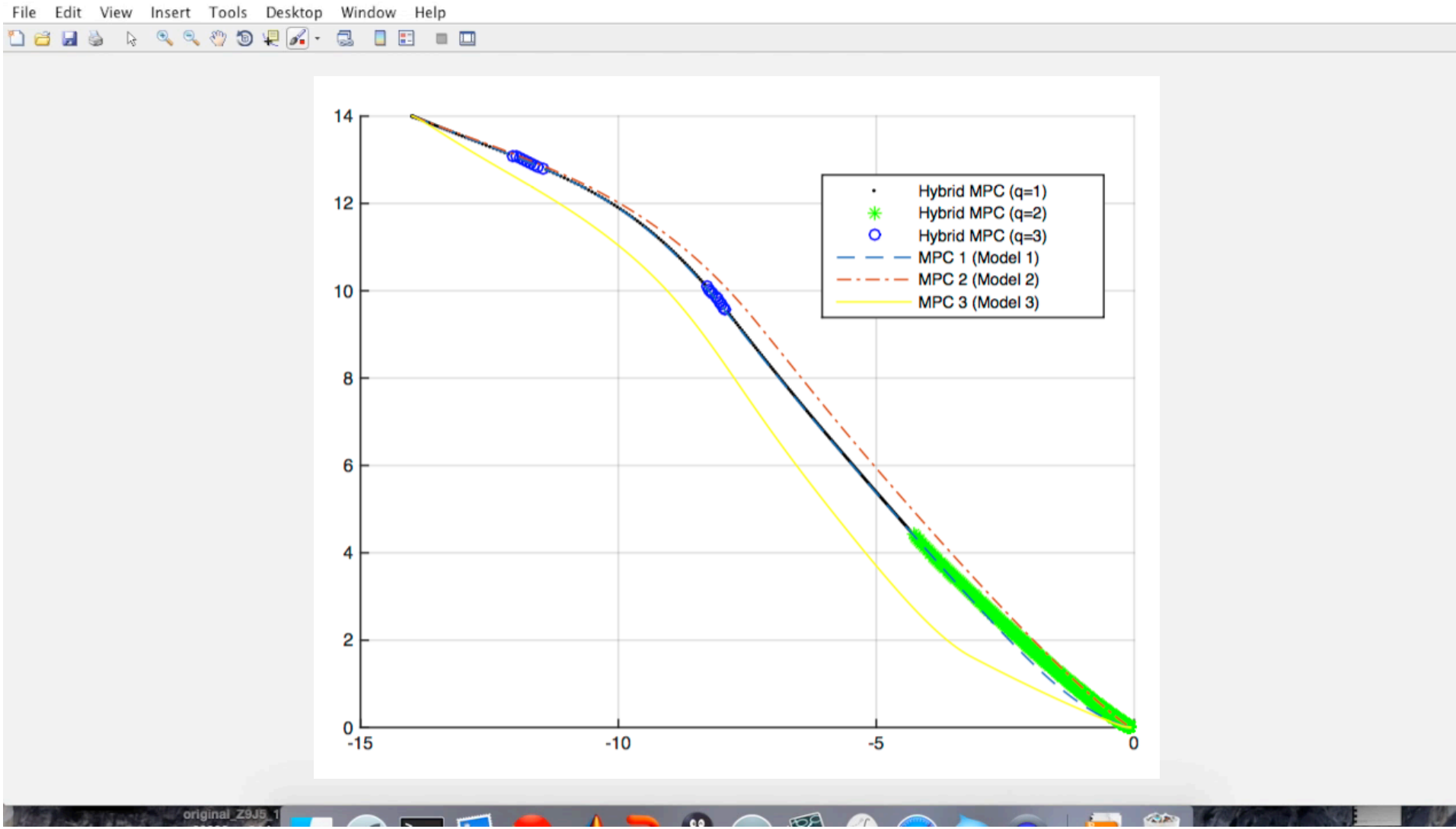
What if I don’t have a car?

Unswitched

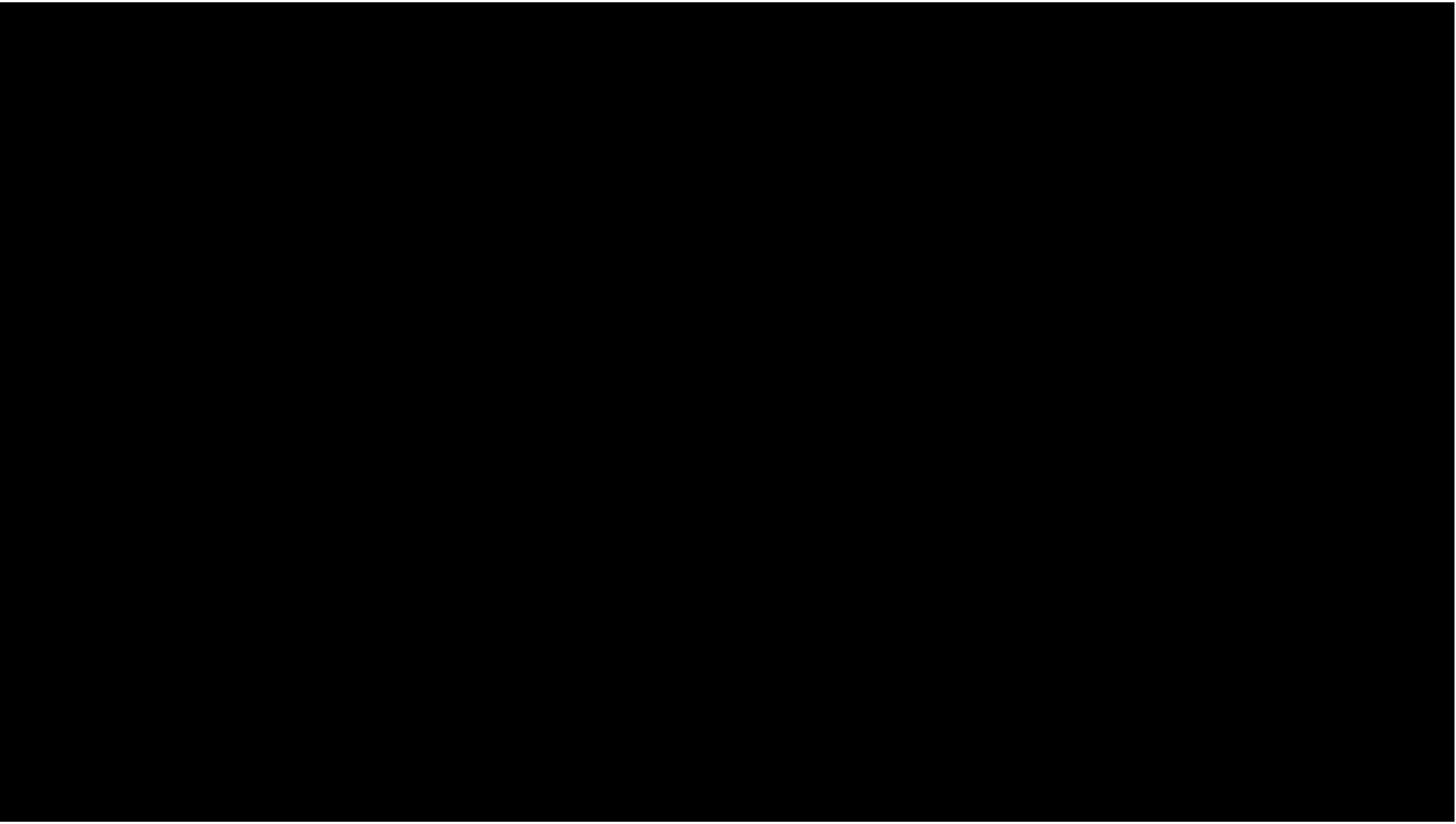
Switched



Application of switching MPC



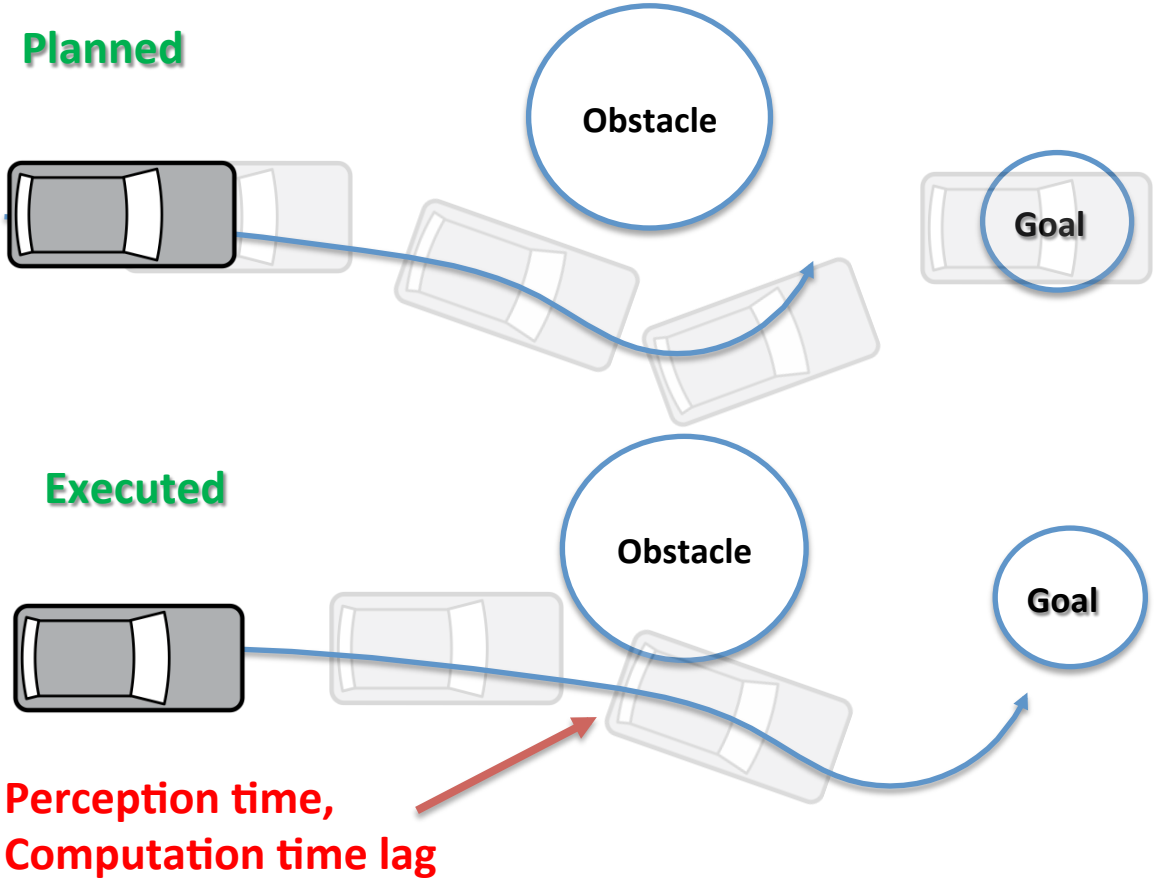
Application to Quadrotor Obstacle Avoidance



CACPS Current efforts: Design with time-awareness

1. Overcoming time-awareness challenges

- ➢ Ensure timeliness for critical applications
- ➢ Reliability for verification and composition



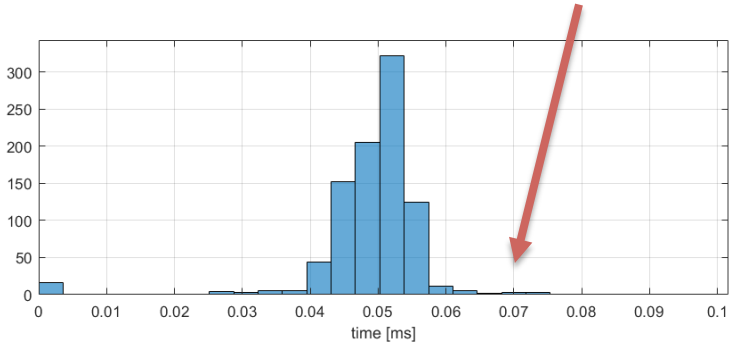
Approach

1. **Knowledge** of computational components time performance
2. **Design** to achieve computational constrains
3. **Self adaptation** to computational load, by trading accuracy with specification through an hybrid approach.

CACPS Current efforts: Design with time-awareness

2. Quantifying computational burden

WCET / WCRT



➢ **Quantifying time performance**

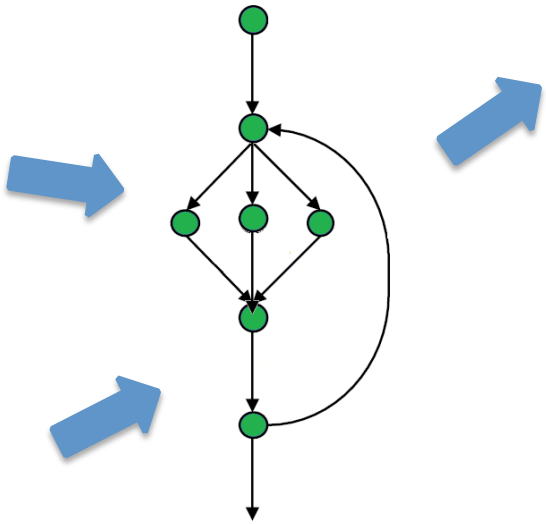
- Hardware: system's architecture
- Software
- Network induced

➢ **Approach**

```
function [bboxes, scores] = detectPeopleACF(I, varargin)
[params, detector] = parseInputs(I, varargin{:});
Iroi = vision.internal.detector.cropImageIfRequested(I, params.ROI, para
[bboxes, scores] = vision.internal.acf.detect(Iroi, detector, params);
bboxes = round(bboxes);
bboxes = clipBBox(bboxes, size(Iroi));
bboxes(:,1:2) = vision.internal.detector.addOffsetForROI(bboxes(:,1:2),
if params.SelectStrongest
    [bboxes, scores] = selectStrongestBbox(bboxes, scores, ...
        'RatioType', 'Min', 'OverlapThreshold', 0.65);
end
%--------------------------------------------------------------------
function m = getModel(params)
persistent model id name
if isempty(model)
    % first time initialization
    model = cell(1,2);
    [data, id, name] = loadModel(params.Model);
    model{id} = data;
  elseif ~strcmp(params.Model, name)

    % model changed
    [name, id] = getModelNameAndID(params.Model);

    if isempty(model{id})
        % model not loaded yet. load it now.
        model{id} = loadModel(params.Model);
```

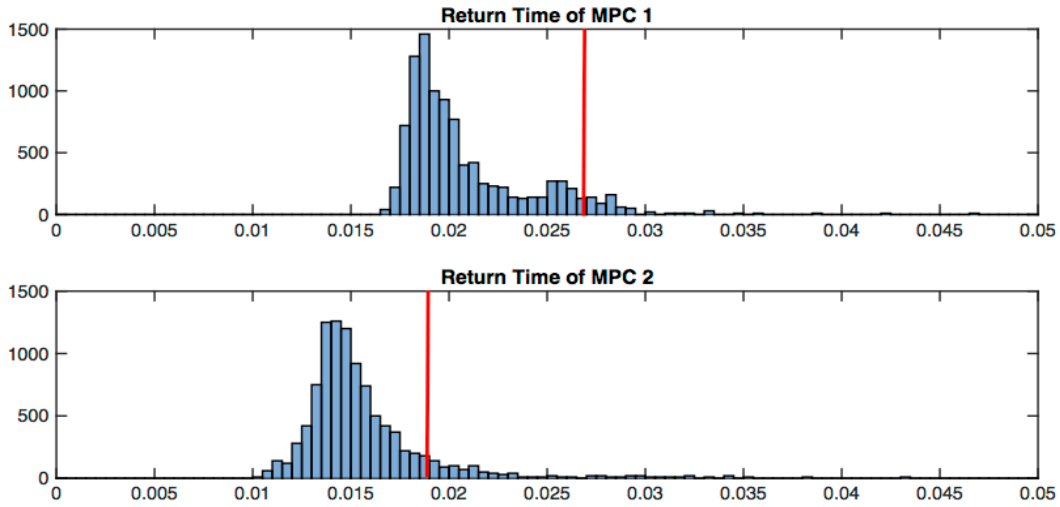


CACPS Current efforts: Design with time-awareness

2. Quantifying computational burden

➢ **Quantifying time performance**

- Characterizing return time and defining boundaries for algorithms

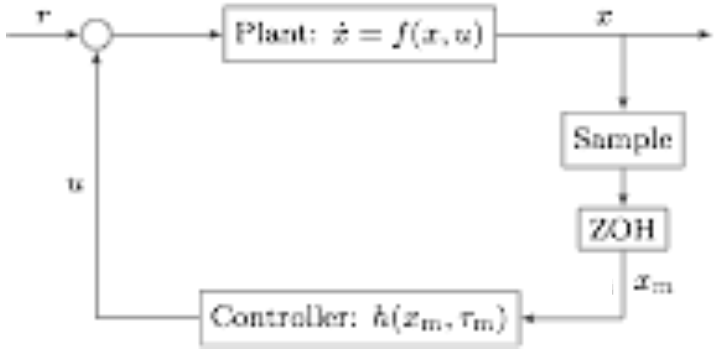


➢ **Reactive and time-adaptive controller design**

Incorporating time characterization in the design process

CACPS Current efforts: CA Predictive Control for CPS

3. Predictive control that is robust to computation time uncertainty



➢ **Computation time T_{comp} is uncertain due to**

- Hardware variability
- Delays and data losses
- Computational complexity

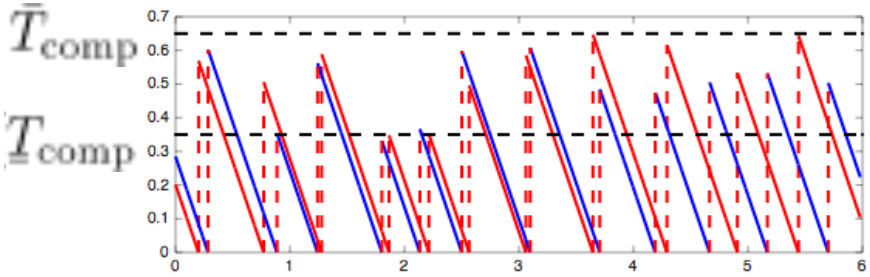
➢ **Approach**

Hybrid system modeling

At sampling instants, T_{comp} is updated via the *difference inclusion*:

$$T_{\text{comp}}^{+} \in [\underline{T}_{\text{comp}}, \bar{T}_{\text{comp}}]$$

while in between sampling events, it decreases to zero linearly.



Design control algorithm to compensate for all possible values of T_{comp}

Accomplishments during first year of performance and ongoing efforts

Summary of milestones reached (first period of performance):

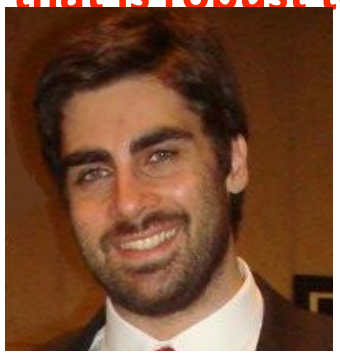
- **Formulation of model of computationally-aware CPS (CACPS)**
- **Derivation of uncertainty divergence (UD)**
- **Application to ground and aerial vehicle applications**

Ongoing efforts:

- **Overcoming time-awareness challenges**
- **Quantifying computational burden**
- **Predictive control that is robust to computation time uncertainty**

CACPS Student Team:

+ 2 Undergrad students
+ 2 High School inters



Berk Altin
(postdoc, UCSC)

Nathalie Risso
(grad, UofA)

Yegeta Zeleke
(grad, UCSC)

Seventh Annual
Cyber-Physical Systems Principal Investigators' Meeting
Arlington, VA | October 31 – November 1, 2016

CPS: Synergy: Collaborative Research

Computationally Aware Cyber-Physical Systems

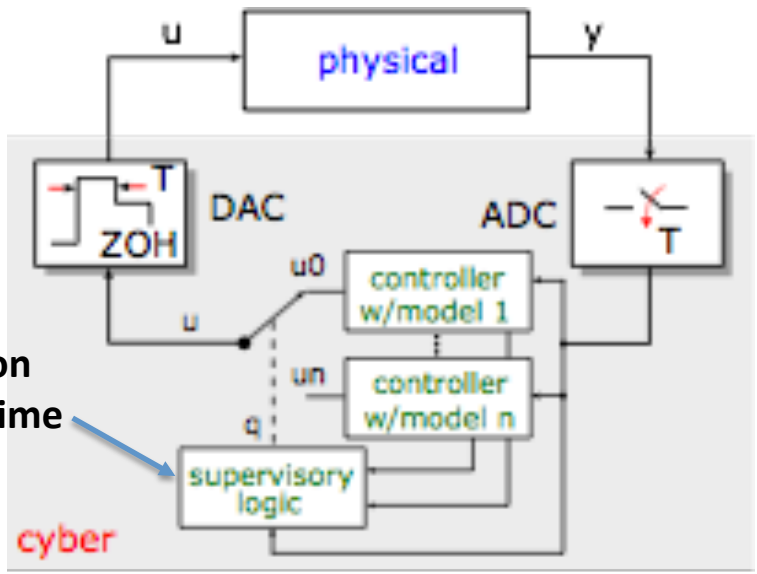
Ricardo Sanfelice (UC Santa Cruz NSF 1544396) , Jonathan Sprinkle (University of Arizona, NSF 1544395)

Challenge:

- Include computation time information at runtime as part of the controller's design for switching criteria
- Do not give up accuracy to improve computation time

Solution:

- Metric for switching criteria
- Basis in accumulated error for predictive controller
- Hybrid control approach
- Trade off between time to compute and accuracy



UAVs in the National Air Space (courtesy NASA)

Self-driving Cars

Scientific Impact:

- More specific controllers made available, based on available computation power
- Increased confidence in runtime controller, if switched

Broader Impact:

- Less conservative controllers mean closer flight patterns
- Benefits to ground and aerial vehicle control systems
- Applications to undergraduate team test beds
- ~20% reduction in average return time for dynamical models without compromising stability

Thank you for your attention!

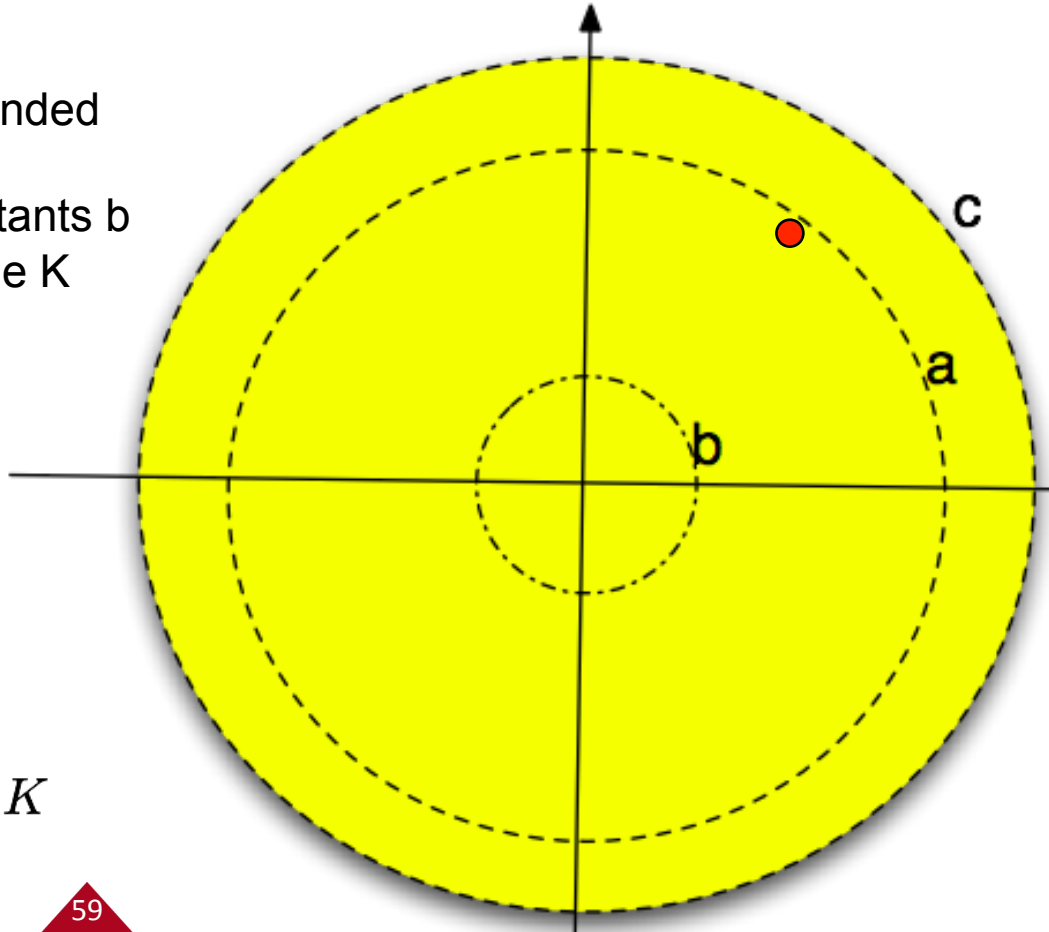
Back up slides

Sufficient conditions for asymptotically ultimately bounded

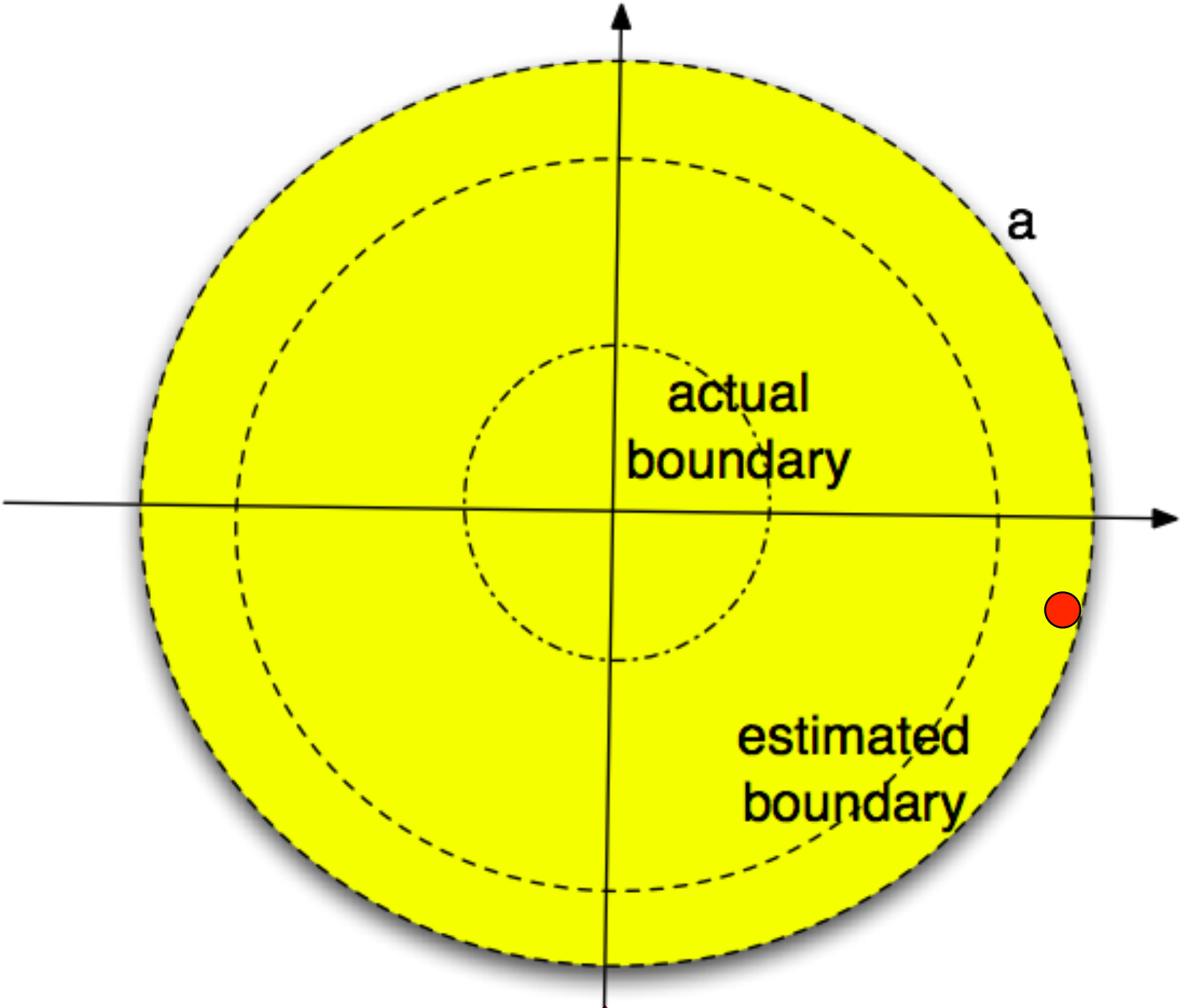
Why consider boundedness?
(1) Minimizing UD does not automatically ensure stability.
(2) An overall system consisting of stable subsystems is not necessarily stable.

These sufficient conditions are switching constraints.

A system is asymptotically ultimately bounded if the system evolves asymptotically to a bounded set, i.e., there are positive constants b and c, and for every 0<a<c, there is a time K such that



$$\|\xi_0 - \xi_r\| \leqslant a \Rightarrow \|\xi_k - \xi_r\| \leqslant b, \forall k > K$$


a
actual
boundary
estimated
boundary

If the following assumptions are satisfied by a candidate CPS, then this work can be applied:

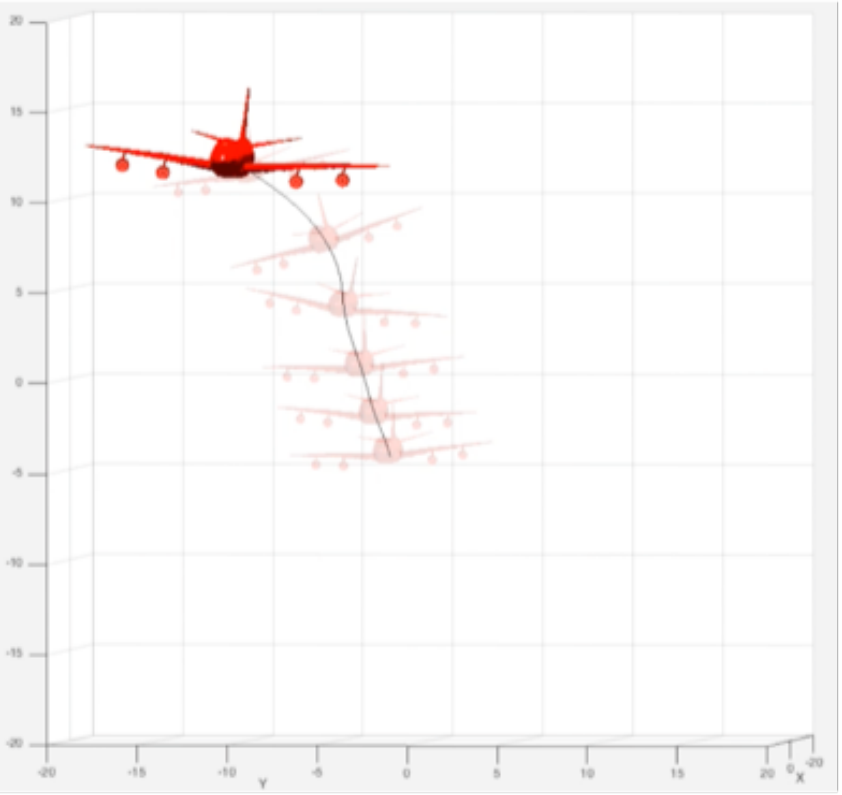
- The target state $\hat{\xi}_r$ for the system $\hat{f}(\hat{\xi}, \hat{u})$ and the predictive model $\hat{f}^q(\hat{\xi}, \hat{u})$, $q \in \mathbb{Q}$ is an equilibrium; namely, $\hat{\xi}_r \in \mathcal{X}$, and $\hat{f}(\hat{\xi}_r, \kappa^q(\hat{\xi}_r, 0)) = \hat{\xi}_r$ and $\hat{f}^q(\hat{\xi}_r, \kappa^q(\hat{\xi}_r, 0)) = \hat{\xi}_r$.

- The horizon is long enough; $\forall q, q' \in \mathbb{Q}$, $S^q_{\max} + S^{q'}_{\max} \leqslant N$. This prevents the depletion of control inputs while waiting for MPC return.

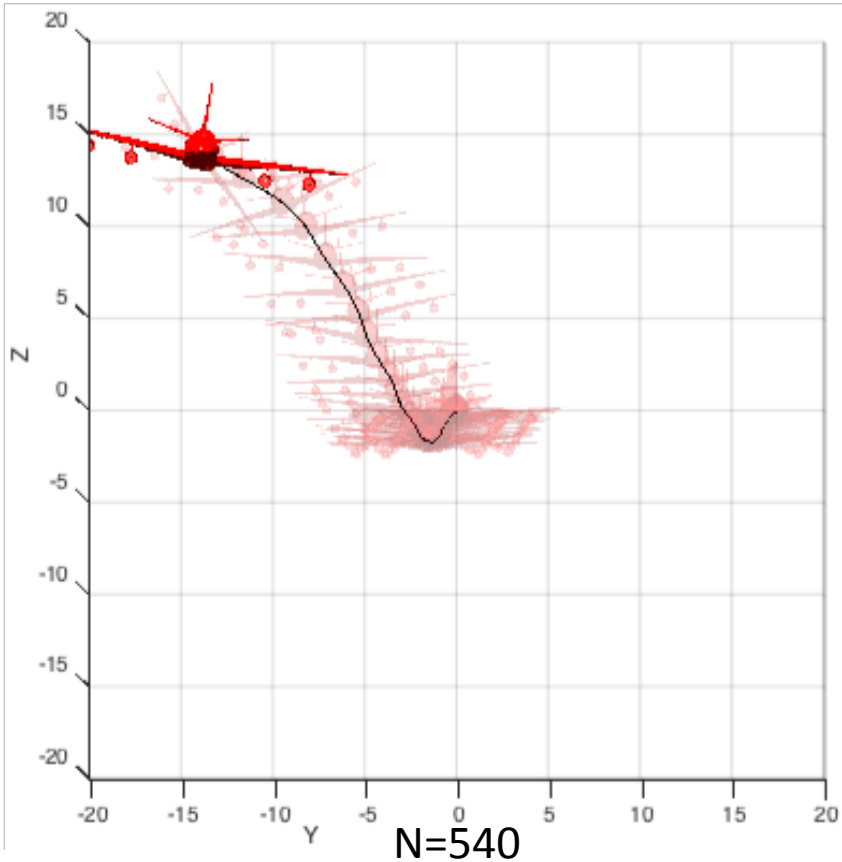
- In order to ensure that UD is bounded, $\frac{\partial \hat{f}^q}{\partial \hat{\xi}}$, $\frac{\partial \hat{f}^q}{\partial \hat{u}}$, $\nabla \kappa^q$, and $\hat{\Gamma}^q$ are required to be bounded $\forall \hat{\xi} \in \mathcal{X}$, $\forall \hat{u} \in \mathcal{U}$ and $\forall q \in \mathbb{Q}$.

- Implicit MPC control through $\kappa^q(\cdot, \cdot)$ will always drive the plant within the feasible state space $\mathcal{X}$, and $\kappa^q(\hat{\xi}, i) \in \mathcal{U}$, $\forall \hat{\xi} \in \mathcal{X}$ and $\forall i \in \{0, 1, \cdots, N-1\}$.

Long horizon examples



N=36



N=540

This permits the use of a “dwell-time” approach to determination of stability.

It doesn't always work.

Reference Plant: The reference plant $\hat{f}$ is formulated by

$$\ddot{x} = -g\sin(\theta) + u_1\cos(\theta)/m - u_2\sin(\theta)/m - c\dot{x}/m$$
$$\ddot{y} = g(\cos(\theta) - 1) + u_1\sin(\theta)/m + u_2\cos(\theta)/m - c\dot{y}/m$$
$$\ddot{\theta} = ru_1/J$$

linearized around 0, but multiplied by rho

remove the last term

Predictive Model 3 ($q = 3$): The third predictive model is a linearized version of the reference plant about the origin, but the equations are then multiplied by ρ:

$$\ddot{x} = (u_1/m + (-g - u_2/m) - c\dot{x}/m\theta)\,\rho$$
$$\ddot{y} = (u_2/m + u_1/m\theta - c\dot{y}/m)\,\rho$$
$$\ddot{\theta} = (ru_1/J)\rho$$

where $\rho = 0.95$ is deliberately selected to adjust the model's accuracy.

Predictive Model 1 ($q = 1$): The first predictive model is derived fr reference plant by removing the last terms:

$$\ddot{x} = -g\sin(\theta) + u_1\cos(\theta)/m - u_2\sin(\theta)/m$$
$$\ddot{y} = g(\cos(\theta) - 1) + u_1\sin(\theta)/m + u_2\cos(\theta)/m$$
$$\ddot{\theta} = ru_1/J$$

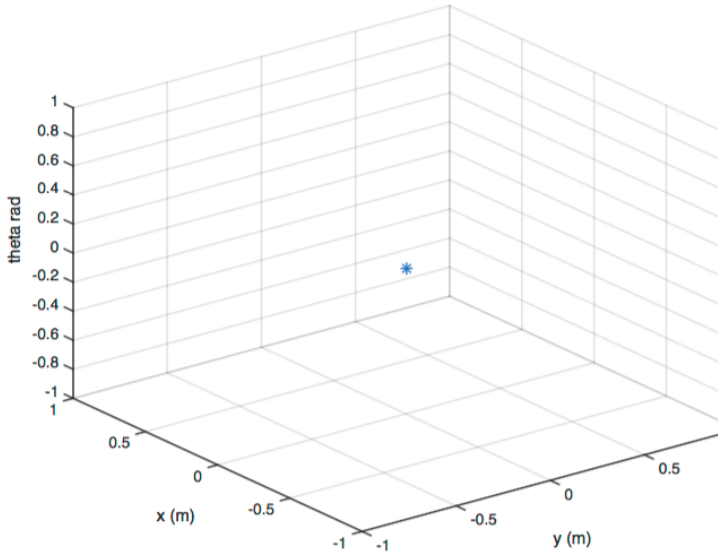
linearized around 0

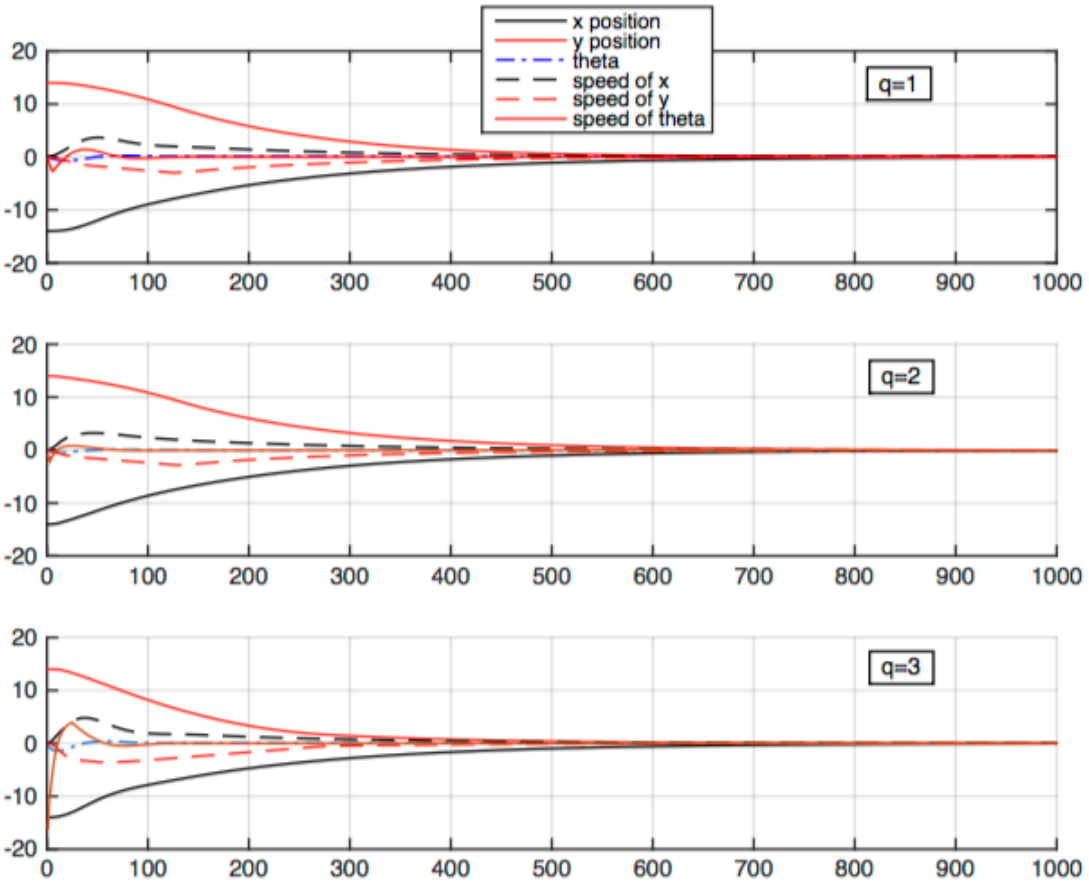
Predictive Model 2 ($q = 2$): The second predictive model is a linearized version of the reference model about the origin:

$$\ddot{x} = u_1/m + (-g - u_2/m)\theta$$
$$\ddot{y} = u_2/m + u_1/m\theta$$
$$\ddot{\theta} = ru_1/J$$

Each model converges

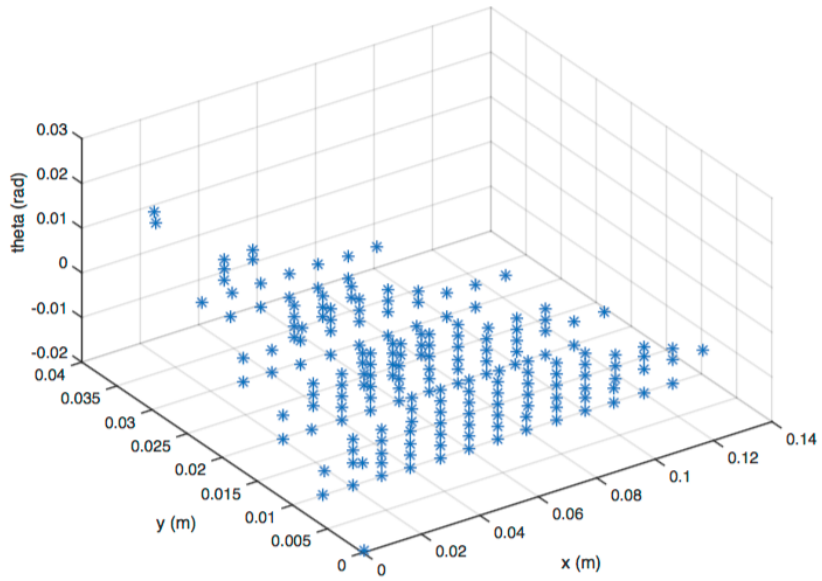
$$
\begin{aligned}
R^{\mathrm{MPC}} &= \max\left(\max_q\left(\frac{b^q}{a^q}\right)^{\frac{1}{\sigma}}, \max_q\left(\frac{b^q}{a^q}\frac{b^{\bar{q}}}{a^{\bar{q}}}\gamma\right)^{\frac{1}{\sigma}}\right)\underline{\mathrm{R}}^{q+} \\
&= \max_q\left(\left(\frac{350}{250}\right)^{\frac{1}{2}}, \left(\frac{350}{250}\frac{350}{250}0.95\right)^{\frac{1}{2}}\right)0 \\
&= 0
\end{aligned}
$$



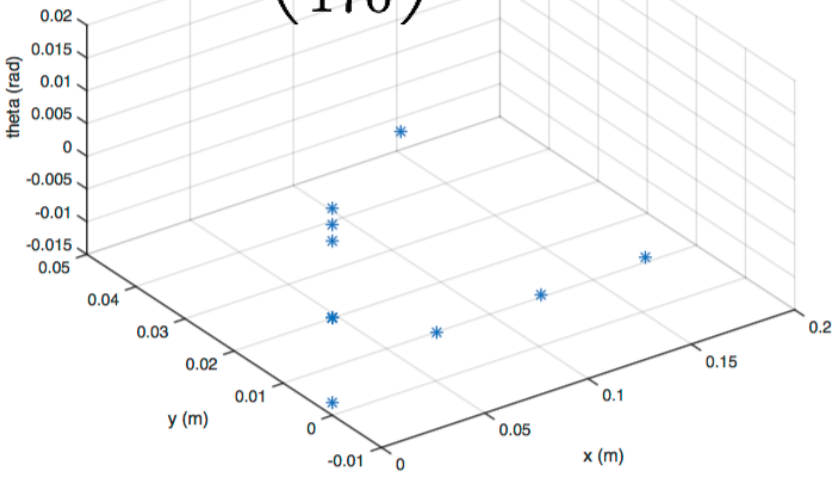


Numerical results of the superlevel sets of all three models suggest that each of these three MPCs can bound the aircraft to the origin, or at least a small ball containing the origin with a radius smaller than the discrete spatial steps.

Make model-3 less accurate

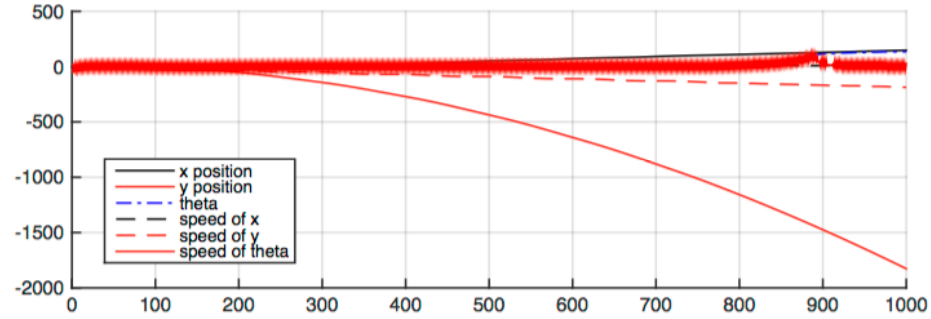


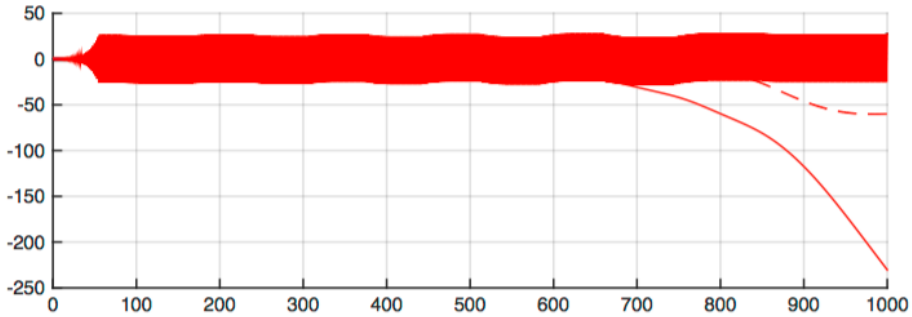
$$\left(\frac{400}{170}\right)^{\frac{1}{2}} \cdot 0.2 \approx 0.31$$

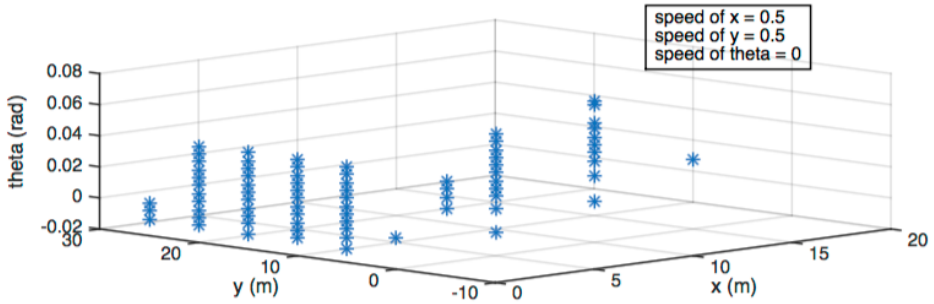


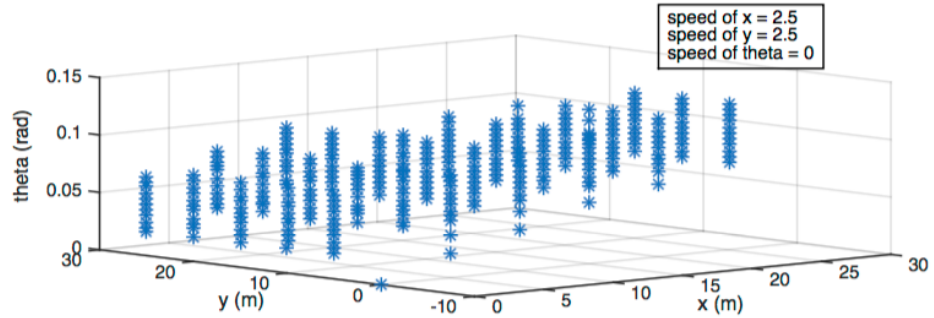
As d decreases: although the superset $L_d(\Delta V^{q=3}(\hat{\xi}))$ has fewer points, the points in these two figures cover a similar area, and the set $L_d(\Delta V^{q=3}(\hat{\xi}))$ will not shrink in an obvious way by decreasing d.

For rho=0.5, and d=1





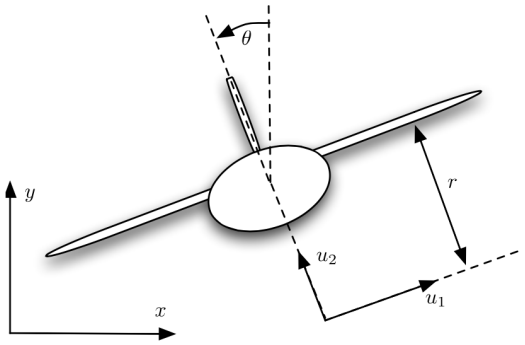


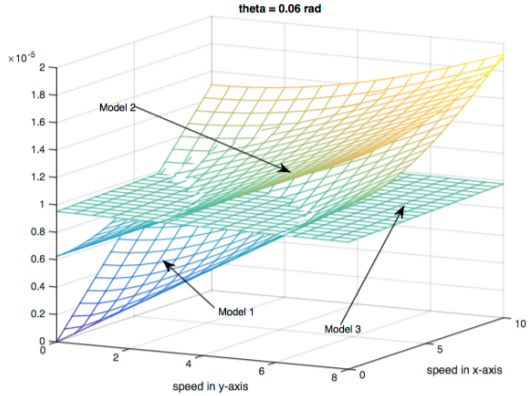


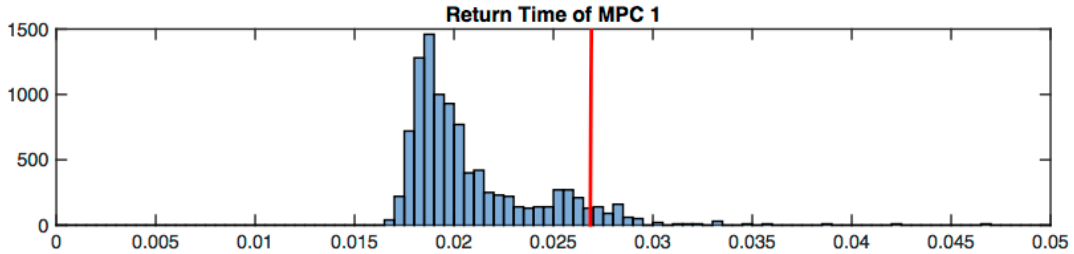
The boundary is not found even when the state reaches x=30 m and y=30 m.

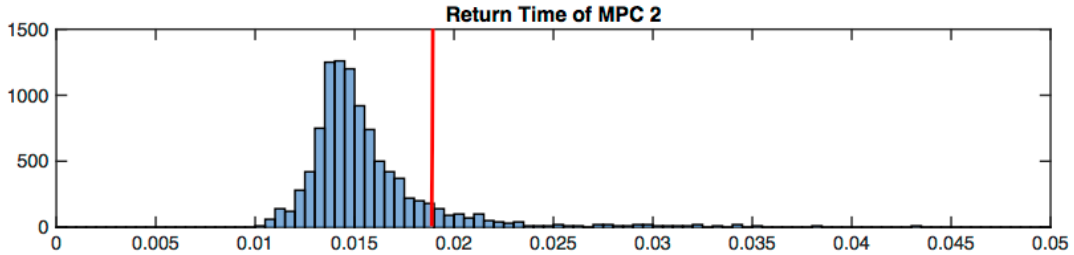
In this situation, the models do not satisfy the necessary conditions in order to use the uncontrollable divergence as the switching metric. It confirms that the approach requires some analysis in order to use the design methodology with confidence.

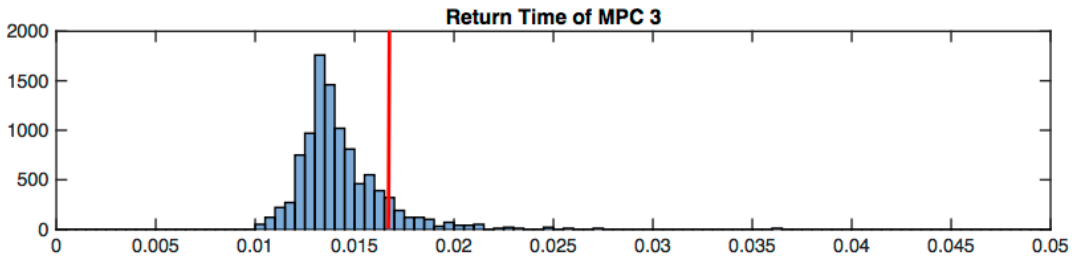
Return time


θ
y
x
r
u2
u1


theta = 0.06 rad
×10^-5
Model 2
Model 1
Model 3
speed in y-axis
speed in x-axis

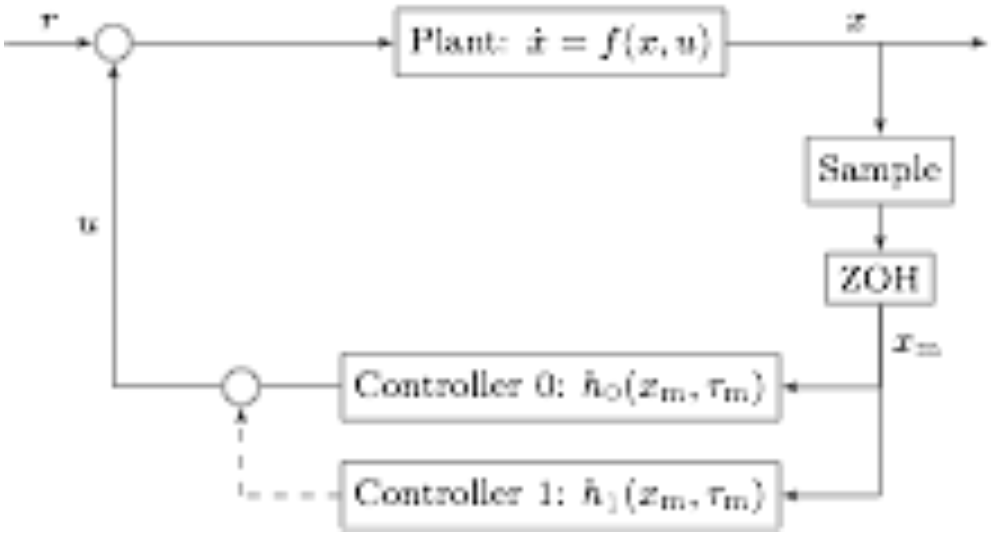

Return Time of MPC 1
0
500
1000
1500
0
0.005
0.01
0.015
0.02
0.025
0.03
0.035
0.04
0.045
0.05


Return Time of MPC 2
0
500
1000
1500
0
0.005
0.01
0.015
0.02
0.025
0.03
0.035
0.04
0.045
0.05


Return Time of MPC 3
0
500
1000
1500
2000
0
0.005
0.01
0.015
0.02
0.025
0.03
0.035
0.04
0.045
0.05

CACPS Current efforts: Real Time Predictive Control

Functions $f_q(x,u)$, $q = 0,1,\ldots,N$ of varying fidelity.

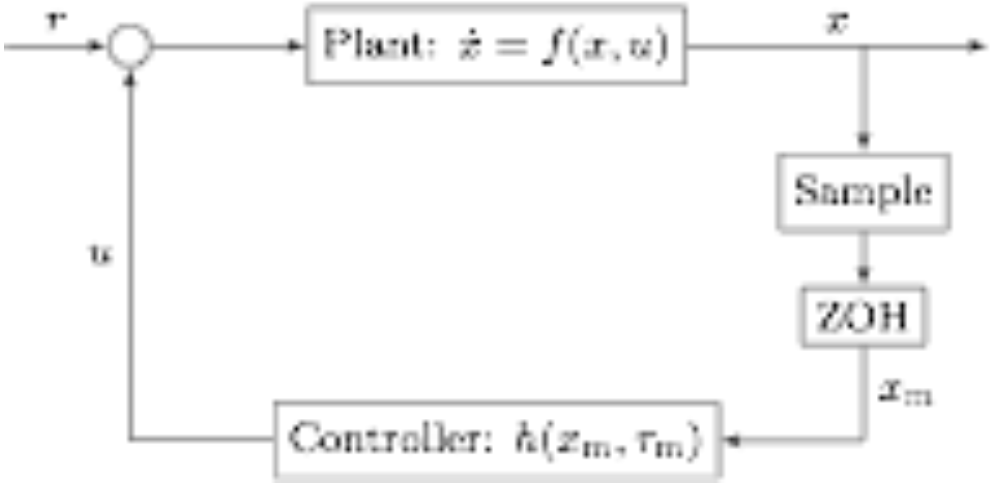


Switching MPC

Find a rule $S : x \mapsto q$ to switch between prediction models f_q, with respect to a safety-accuracy trade-off.

Hybrid Model of RHC

Basic model with measurement state and timer : τ_m



$h(x_m,$ is the minimizer of

$$J(x_m, u) = \int_0^T \ell(x(s), u(s))\, ds + V_f(x(T))$$

with initial condition

Current Work: RHC with varying Horizons

Closed loop state:

$$\underbrace{z \triangleq (x, x_{\mathrm{m}}, \tau_{\mathrm{m}}, T, \tau_{\mathrm{c}})}$$

(plant state, sample, sampling timer, prediction horizon, controller timer)

Difference *inclusion* model with:

$$\dot{z} = f_{\mathrm{cl}}(z), \quad \tau_{\mathrm{m}} \in [0, T_{\mathrm{m}}],$$
$$z^{+} \in G_{\mathrm{cl}}(z), \quad \tau_{\mathrm{m}} \in \{0\},$$

The *set valued* map allows

- intermittent sampling (due to delays, losses etc.), and
- varying horizons for flexible control to compensate .

Stable provided is a control Lyapunov function!