Trajectory Planning for Autonomous Vehicles using Sliding Mode Control

Harish N Sathishchandra

May. 03, 2018

Boston University

Department of Electrical and Computer Engineering

BOSTON UNIVERSITY

Trajectory Planning for Autonomous Vehicles using Sliding Mode Control

Harish N Sathishchandra

Boston University
Department of Electrical and Computer Engineering
8 Saint Mary's Street
Boston, MA 02215
www.bu.edu/ece

May. 03, 2018

Contents

List of Figures

1 Introduction

One topic of recent research in automobile industry is autonomous driving, which is especially challenging in urban environments. Automated control of vehicles is a challenging task due to the non-linearity of the dynamics, the influence of sensor noise, disturbances, and unknown parameters, as well as due to safety criticality. Most control schemes are tracking maneuvers generated by high-level planners.

While closely tracking the planned trajectory is important in all driving situations, it becomes especially crucial when considering extreme maneuvers in emergency situations. Only if the tracking controller is able to control the car despite all previously mentioned disturbances is it possible to execute safe emergency maneuvers.

Since vehicle control is an important topic, many different methods and control algorithms for this problem have been developed. Control methods include sliding mode control, flatness-based control, optimal linear-quadratic control, backstepping-based strategies, optimal preview control and optimization-based methods like model predictive control (MPC).

This report examines the use of Sliding Mode Control (SMC) techniques for tracking trajectories generated by a motion planner. Sliding Mode Control technique is very attractive, considering the stability of tracking control laws, the hardware computation load and the maneuverability in practice.

2 Sliding Mode Control

Sliding mode control evolved from pioneering work in the 1960's in the former Soviet Union. It is a particular type of Variable Structure System (VSS) which is characterized by a number of feedback control laws and a decision rule. The decision rule, termed the switching function, has as its input some measure of the current system behavior and

produces as an output the particular feedback controller which should be used at that instant in time. In sliding mode control, Variable Structure Control Systems (VSCS) are designed to drive and then constrain the system state to lie within a neighborhood of the switching function. One advantage is that the dynamic behavior of the system may be directly tailored by the choice of switching function - essentially the switching function is a measure of desired performance. Additionally, the closed loop response becomes totally insensitive to a particular class of system uncertainty. This class of uncertainty is called matched uncertainty and is categorized by uncertainty that is implicit in the input channels. Large classes of problems of practical significance naturally contain matched uncertainty. Hence the SMC is a very popular control technique.

2.1 Control Design

The design of a sliding mode controller can be explained by considering the following example of nonlinear system and it's homogenous liner time invariant approximation:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = u + f(x_1, x_2, t) \end{cases} \quad (1)$$

$$\dot{x}_1 + cx_1 = 0, \qquad c > 0 \quad (2)$$

Where u is the control force, and $f(x_1, x_2, t)$ is the nonlinear disturbance. Since $x_2(t) = \dot{x}_1(t)$, a general solution of Eq. (1.1) and it's derivative is given by:

$$x_1(t) = x_1(0)e^{(-ct)} \quad (3)$$

$$x_2(t) = \dot{x}_1(t) = -c\dot{x}_1(0)e^{-ct} \quad (4)$$

Both $\dot{x}_1$ and $\dot{x}_2$ converge to zero asymptotically. Note, no effect of the disturbance $f(x_1, x_2, t)$ on the state compensated dynamics is observed. To achieve these state compensated dynamics, a new variable is introduced in the state space of the system in equation (1) :

$$\sigma = \sigma(x_1, x_2) = x_2 + cx_1, \qquad c > 0 \tag{5}$$

In order to achieve asymptotic convergence of the state variables x_1 and x_2, with a given convergence rate as in equation (3) and (4), in the presence of the bounded disturbance $f(x_1, x_2, t)$, we have to drive the variable σ in equation (5) to zero in finite time by means of the control u. This task can be achieved by applying Lyapunov function techniques to the σ-dynamics that are derived from equation (1) and equation (5):

$$\dot{\sigma} = cx_2 + f(x_1, x_2, t) + u, \qquad \sigma(0) = \sigma_0 \tag{6}$$

For the σ-dynamics (6) a candidate Lyapunov function is introduced taking the form:

$$V = \frac{1}{2}\sigma^2 \tag{7}$$

In order to provide the asymptotic stability of equation (6) about the equilibrium point $\sigma = 0$, the following conditions must be satisfied:

a) $\dot{V} < 0$ for $\sigma \neq 0$

b) $\lim_{|\sigma| \to \infty} V = \infty$

Condition (b) is obviously satisfied by V in equation (7). In order to achieve finite-time convergence (global finite-time stability), condition (a) can be modified to be:

$$\dot{V} \leq -\alpha V^{\frac{1}{2}}, \qquad \alpha > 0 \tag{8}$$

Indeed, separating variables and integrating inequality (8), we get:

$$V^{\frac{1}{2}}(t) \leq -\frac{1}{2}\alpha t + V^{\frac{1}{2}}(0) \tag{9}$$

Consequently, $V(t)$ reaches zero in a finite time t_r that is bound by:

$$t_r \leq \frac{2V^{\frac{1}{2}}(0)}{\alpha} \tag{10}$$

Therefore, a control u that is computed to satisfy equation (8) will drive the variable σ to zero in finite time and will keep it at zero thereafter.

The derivative of V is computed as:

$$\dot{V} = \sigma\dot{\sigma} = \sigma(cx_2 + f(x_1, x_2, t) + u) \tag{11}$$

Assuming the control law $u = cx_2 + v$ and substituting it into equation 11, we obtain:

$$\dot{V} = \sigma(f(x_1, x_2, t) + v) = \sigma(f(x_1, x_2, t) + v) \leq |\sigma|L + \sigma v \tag{12}$$

Selecting $v = -\rho sign(\sigma)$, where

$$sign(x) = \begin{cases} 1 & if \;\; x > 0 \\ -1 & if \;\; x < 0 \end{cases} \tag{13}$$

And

$$sign(0) \in [-1 \;\; 1] \tag{14}$$

With $\rho > 0$ and substituting it into equation (12) we obtain:

$$\dot{V} \leq |\sigma|L - |\sigma|\rho = -|\sigma|(\rho - L) \tag{15}$$

Taking into account equation (17), condition (8) can be rewritten as:

$$\dot{V} \leq -\alpha V^{\frac{1}{2}} = -\frac{\alpha}{\sqrt{2}}|\sigma|, \quad \alpha > 0 \tag{16}$$

Combining equations (15) and (16), we obtain:

$$\dot{V} \leq -|\sigma|(\rho - L) = -\frac{\alpha}{\sqrt{2}}|\sigma| \tag{17}$$

Finally, the control gain ρ is computed as:

$$\rho = L + \frac{\alpha}{\sqrt{2}} \tag{18}$$

Consequently, a control law u that drives σ to zero in finite time (10) is:

$$u = -cx_2 - \rho sign(\sigma) \tag{19}$$

Definition 1: The variable (5) is called a sliding variable.
Definition 2: Equations (3) and (5) rewritten in a form:

$$\sigma = x_2 + cx_1 = 0, \qquad c > 0 \tag{20}$$

correspond to a straight line in the state space of the system (1) and are referred to as a sliding surface.

Condition (8) is equivalent to:

$$\sigma\dot{\sigma} \leq -\frac{\alpha}{\sqrt{2}}|\sigma| \tag{21}$$

and is often termed the reachability condition. Meeting the reachability or existence condition (21) means that the trajectory of the system in Eq. (1) is driven towards the sliding surface (20) and remains on it thereafter.

Definition 3: The control $u = u(x_1, x_2)$ in Eq. (19) that drives the state variables to the sliding surface (20) in finite time t_r and keeps them on the surface thereafter in the presence of the bounded disturbance $f(x_1, x_2, t)$, is called a sliding mode controller.

3 Sliding mode trajectory-tracking control

It is supposed that a feasible desired trajectory for the vehicle is pre-specified by a trajectory planner. The problem is to design a robust controller so that the vehicle will correctly track the desired trajectory under a large class of disturbances. Consider the following non holonomic motion model of a vehicle:

$$\dot{x}_r(t) = v_r(t) \cdot cos\theta_r(t) \tag{21}$$

$$\dot{y}_r(t) = v_r(t) \cdot sin\theta_r(t) \tag{22}$$

$$\dot{\theta}_r(t) = \frac{v_r}{l} \cdot tan\phi_r(t) \tag{23}$$

where (see Fig. 1), x_r and y_r are the Cartesian coordinates of the rear axle midpoint, v_r is the velocity of this midpoint, θ_r is the vehicle's heading angle, l is the inter-axle distance, and ϕ_r the front wheel angle, which is the control variable to steer the vehicle. Here, the kinematic bicycle model is considered (see Fig 2). The trajectory tracking errors can be described by (x_e, y_e, θ_e). The aim is to design a stable controller with commands (v_c, ϕ_c). The error vector for trajectory-tracking is easily obtained from Fig 2:

$$\begin{bmatrix} xe \\ ye \\ \theta e \end{bmatrix} = \begin{bmatrix} cos\theta d & sind\theta d & 0 \\ -sin\theta d & cos\theta d & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} xr - xd \\ yr - yd \\ \theta r - \theta d \end{bmatrix} \tag{24}$$

Where, (x_d, y_d, θ_d) denotes the virtual car pose. The corresponding error derivatives are:

$$\dot{x}_e = -v_d + v_r \cdot cos\theta_e + y_e \cdot \frac{v_d}{l} \cdot tan\phi_d \tag{25}$$

(26)

$$\dot{y}_e = v_r \cdot sin\theta_e - x_e \cdot \frac{v_d}{l} \cdot tan\phi_d \tag{27}$$
$$\dot{\theta}_e = \frac{v_r}{l} \cdot tan\phi_r - x_e \cdot \frac{v_d}{l} \cdot tan\phi_d$$

Where, v_d and ϕ_d are the heading speed and desired front wheel angle, respectively. Here, it is assumed that $|\theta_e| < \frac{\pi}{2}$, which means that the vehicle orientation must not be perpendicular to the desired trajectory.

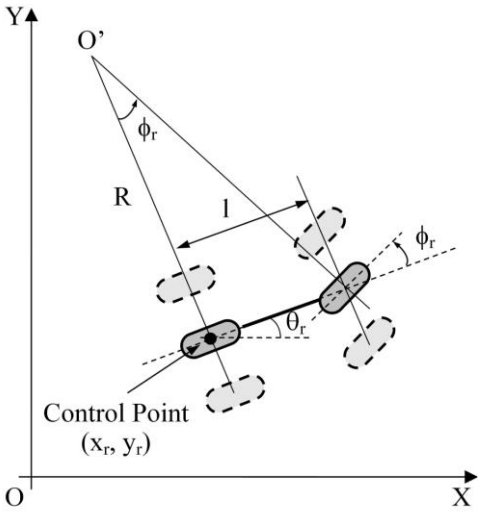

Figure 1 The Bicycle model

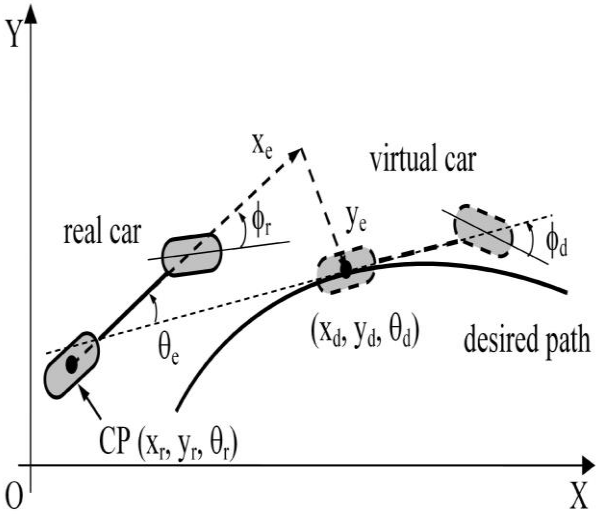

Figure 2 Lateral, longitudinal and orientation error (trajectory- tracking).

A new design of sliding surface is proposed, such that lateral error, y_e, and angular error, θ_e, are internally coupled with each other in a sliding surface leading to convergence of both variables. For that purpose, the following sliding surfaces are proposed:

$$\dot{s}_1 = \dot{x}_e + k_1 \cdot x_e \tag{28}$$
$$\dot{s}_2 = \dot{y}_e + k_2 \cdot y_e + k_0 \cdot sgn(y_e) \cdot \theta_e \tag{29}$$

In trajectory-tracking exist three variables (x_e, y_e, θ_e) and just two control variables, which implies that we have only two sliding surfaces. We chose to couple y_e and θ_e in one sliding surface. The condition under which the state will move toward and reach a

sliding surface is called a reaching condition. A practical general form of the reaching law is:

$$\dot{s} = -Q \cdot s - P \cdot sat(s, \epsilon) \tag{30}$$

Where,

$$Q = \begin{bmatrix} q1 & 0 \\ 0 & q2 \end{bmatrix},\ P = \begin{bmatrix} p1 & 0 \\ 0 & p2 \end{bmatrix},\ i = 1,2. \tag{31}$$

By adding the proportional rate term −Q · s, the state is forced to approach the switching manifolds faster when s is large. Various choices of Q and P specify different rates for s and yield different structures in the reaching law.

From the time derivative of Equations (28) and (29) and using (30), results in:

$$\dot{s}_1 = \ddot{x}_e + k1 \cdot \dot{x}_e = -q_1 \cdot s_1 - p_1 \cdot sat(s_1, \epsilon) \tag{32}$$

$$\dot{s}_2 = \ddot{y}_e + k_2 \cdot \dot{y}_e + k_0 \cdot sat(y_e) \cdot \dot{\theta}_e = -q_2 \cdot s_2 - p_2 \cdot sat(s_2, \epsilon) \tag{33}$$

From equations (25), (26), (27), (32) and (33), and after some mathematical manipulation, we get the commands:

$$v_c = \frac{1}{cos\theta_e} \cdot (-q_1 \cdot s_1 - p_1 \cdot sat(s_1, \epsilon) - k_1 \cdot \dot{x}_e - \dot{\omega}_d \cdot y_e - \omega_d \cdot \dot{y}_e + v_r \cdot \dot{\theta}_e \cdot sin\theta_e + \dot{v}_d \tag{34}$$

$$\phi_c = \arctan(\frac{l}{v_r} \cdot \omega_d + \frac{l}{vr \cdot (v_r \cdot cos\theta_e + k_0 \cdot sat(y_e, \epsilon))} \cdot (-q_2 s_2 - p_2 sat(s_2) - k_2 \cdot \dot{y}_e - \dot{v}_r \cdot sin\theta_e + \dot{\omega}_d \cdot x_e + \omega_d \cdot \dot{x}_e)) \tag{35}$$

Let $V = \frac{1}{2} \cdot s^T \cdot s$ be a Lyapunov function candidate, therefore its time derivative is:

$$\dot{V} = -s^T \cdot Q \cdot s - p_1 \cdot |s_1| - p_2 \cdot |s_2| \tag{36}$$

For $\dot{V}$ to be negative semi-definite, it is sufficient to choose q_i and p_i such that $q_i, p_i > 0$.

3 Simulation results

Here, simulation results of the proposed methods are presented. The simulation model block diagram is shown in Fig. 3. The following transfer functions were employed:

$$H_\phi = \frac{\phi_r}{\phi_c} = \frac{\omega_n^2}{s_2 + 2 \cdot D \cdot \omega_n \cdot s + \omega_n^2} \tag{37}$$

$$H_v = \frac{v_r}{v_c} = \frac{1}{0.25 \cdot s + 1} \tag{38}$$

With $D = 0.7$ and $\omega_n = 2\pi \cdot 5Hz$. The reference trajectory was generated using a trajectory planner. Design parameters of equations (28), (29), (32) and (33) are $k_0 = 0.05$, $k_1 = 0.25, k = k_2 = 0.5, p_1 = 1, p_2 = 1,\ q_1 = 1,\ q_2 = 1$. The saturation function, used in the control equations have a threshold of $\pm 0.5,$ to reduce the chattering phenomenon.

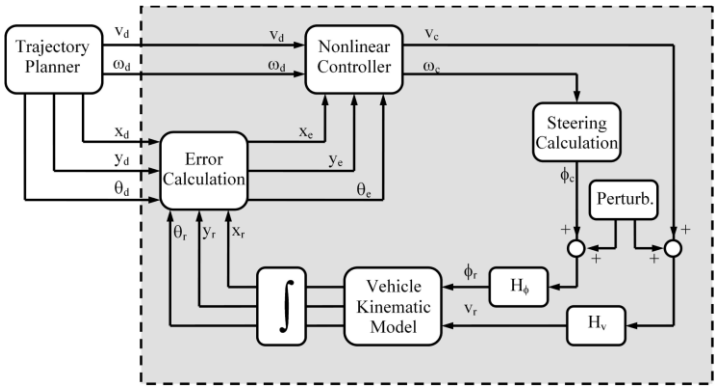

Figure 3 Simulation model block diagram

Figs. 4-5 show the trajectories used in the simulation. Figs. 6–8 show the results for trajectory-tracking control without initial pose error where it is shown that the achieved linear velocity follows the reference velocity, both accelerations (lateral and longitudinal) follow the reference accelerations, and the tracking errors converge to zero.

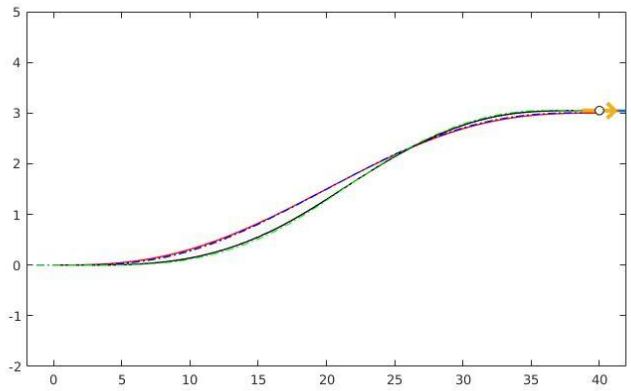

Figure 4 double lane change (red is the trajectory to be followed)

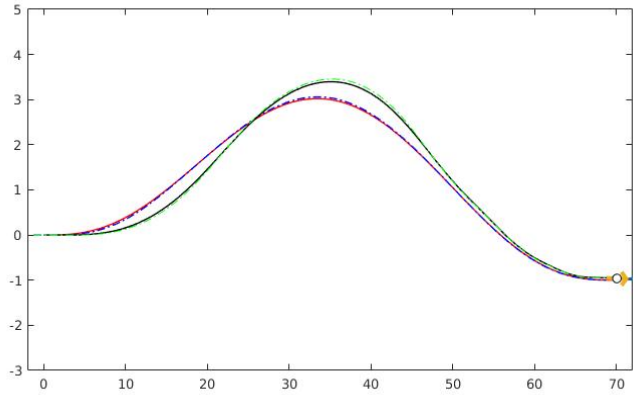

Figure 5 single lane change (red is the trajectory to be followed)

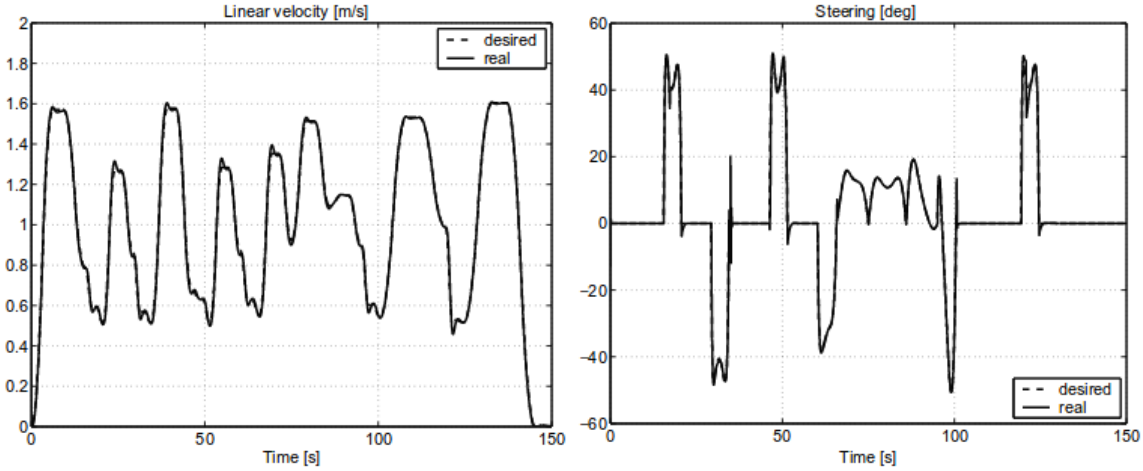

Figure 6 Linear velocity and steering angle for trajectory-tracking controller without initial pose error.

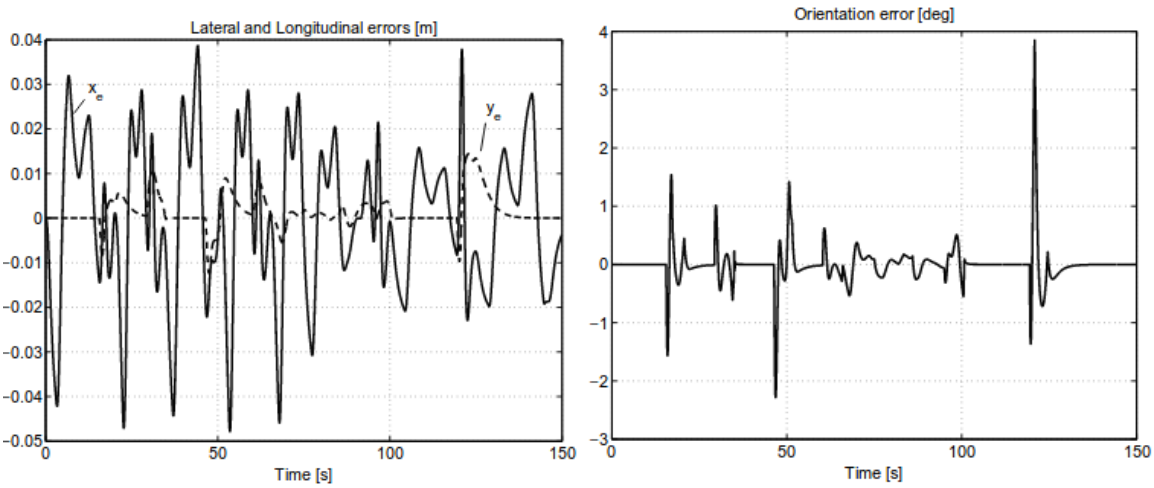

Figure 7 Lateral, longitudinal and orientation errors for trajectory-tracking controller without initial pose error

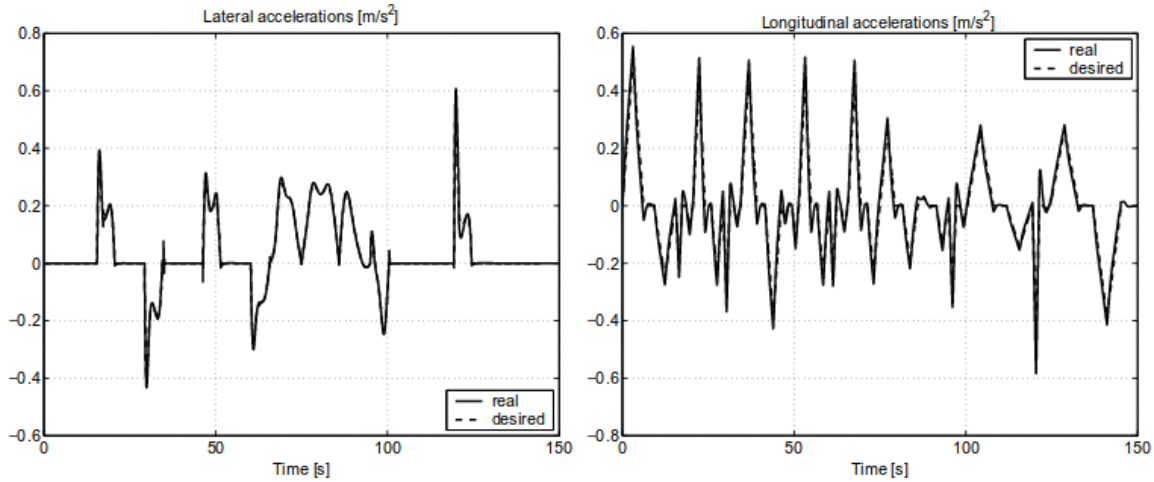

Figure 8 Lateral and longitudinal accelerations for trajectory-tracking controller without initial pose error

Figures 9–11, for trajectory-tracking control with initial pose error ($x_r(0) = 1,\ y_r(0) = 9, \theta_r(0) = \frac{\pi}{8}$), show that the car retrieve the initial difference ($x_e = -2, y_e = 2$) quickly ($\Delta t \approx 20s$), and that the tracking errors converge to zero. The trajectory-tracking task must perform not only the tracking of the path segment (spatial dimension) but also doing it following a specified speed profile (temporal dimension).

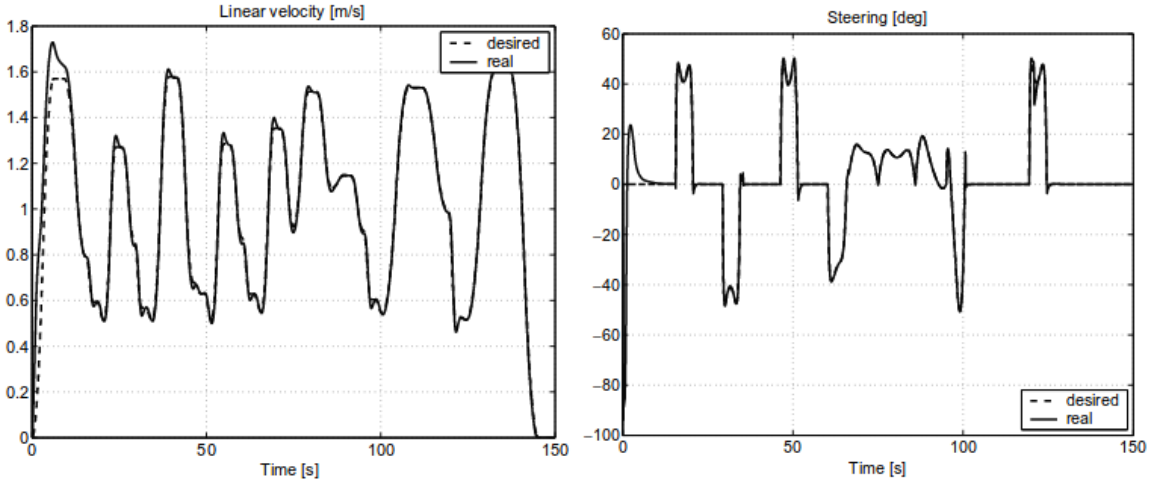

Figure 9 Linear velocity and steering angle for trajectory-tracking controller without initial pose error

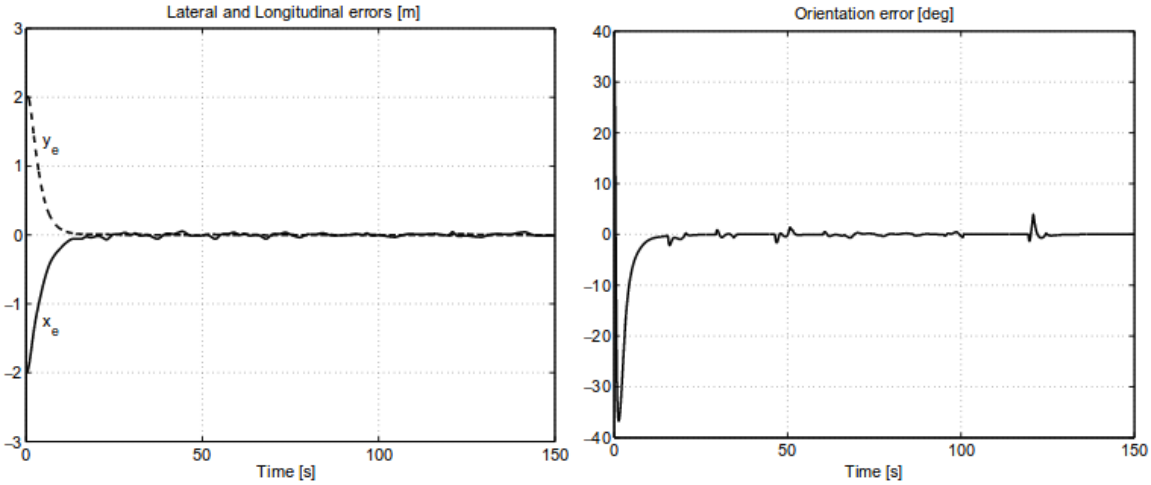

Figure 10 Lateral, longitudinal and orientation errors for trajectory-tracking controller without initial pose error

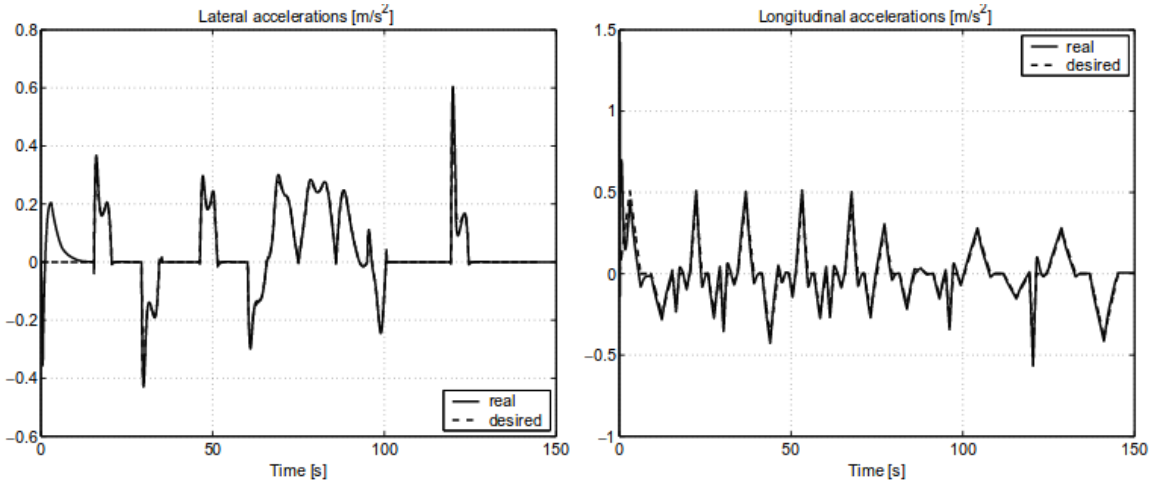

Figure 11 Lateral and longitudinal accelerations for trajectory-tracking controller without initial pose error

4 Conclusion

A new design of sliding surface is proposed, such that lateral and angular errors are internally coupled with each other in a sliding surface leading to convergence of both variables. The main advantages of using SMC include fast response, good transient and robustness with respect to system uncertainties and external disturbances. Simulations are performed for with and without initial pose error. The simulations show that despite it's many advantages, as stated above, Sliding mode controllers are by design aggressive: tracking performances are greatly degraded due to the rapid changes of the sliding control input which quickly causes saturation of the tire forces, and might not be as comfortable to the humans as other controllers.

Appendix

Below is listed Matlab source code developed for this project. Note that I only wrote the KinSliding.m, TEST.m and startup.m functions. The rest of the functions required for simulating the lane changes were given in the framework that I found online for trajectory simulation. I will reproduce the above mentioned 3 functions below.

KinSliding.m:

```
function [controller] = KinSliding()


controller.name='KINSM';
controller.control_point='REAR';
controller.init=@(model,options)init(controller,model,options);
end

function controller = init(controller,model,options)

n=length(model(0,zeros(10,1),zeros(5,1),options.p));

switch func2str(model)
    case options.types{1}
        mapInput=@AXController_pacejka;
    case options.types{2}
        mapInput=@map0;
end
controller.compute_input=@(varargin)compute_input(mapInput,n,varargin);
end

function [u] = compute_input(mapInput,n,varargin)

%reference trajectory
[state,tau,t,p] = parseInput(varargin{:});

% trajectory values
pd = [tau.X(t);tau.Y(t)];
vd = [tau.dX(t);tau.dY(t)];
thetad = tau.theta(t);
omegad = tau.dtheta(t);
domegad= tau.ddtheta(t);
lad = tau.a(t);


[e,de]=getLocalError(state,pd,vd,thetad,omegad,-p.l_R,2);

%velocity of rear tire
vr=state(4); % == vx
```

```
%gains
k0=0.05;
k1=.25;
k2=.5;

%sliding surfaces
s1=de(1)+k1*e(1);
s2=de(2)+k2*e(2)+k0*sgn(e(2))*e(3);


p1=1;
p2=3;

q1=1;
q2=3;

%control law
dv_c=1/cos(e(3))*(-q1*s1-p1*sgn(s1)-k1*de(1)-domegad*e(2)-omegad*de(2)+...
    vr*de(3)*sin(e(3))+lad);
delta=atan(p.L/vr*omegad+p.L/(vr*(vr*cos(e(3))+...
    k0*sgn(e(2))))*(-q2*s2-p2*sgn(s2)-k2*de(2)-...
    dv_c*sin(e(3))+domegad*e(1)+omegad*de(1)));



delta=saturate( delta,-p.MAX_delta,p.MAX_delta);


u=mapInput(dv_c,state,delta,p);
u(end+1)=0;
end
```

TEST.m:

```
% addpath( genpath( cd ) );
function [ benchmark ] = TEST()
%TEST

%specify scenario: 01_single_lane_change or 02_double_lane_change
scenario = '02_double_lane_change';
controller = KinSliding;
model = @vmodel_A;
options = default_options;
options.p.mu0=1;
options.pc.mu0=1;

tau = Trajectory();
tau.load(['Scenarios/' scenario]);

if strcmp(controller.control_point,'CO')
```

```
    tau.solveID(options.pc);
    tauD = tau.transform(options.p.J/options.p.l_R/options.p.m).make_static(100);
else if strcmp(controller.control_point,'REAR')
        tau.solveID(options.pc);
        tauD = tau.transform(-options.p.l_R).make_static(100);
    else
        tauD=tau;
    end
end

%scale cf,cr
if (options.pc.mu0<1)
    options.p.cf=options.p.cf*options.p.mu0;
    options.p.cr=options.p.cr*options.p.mu0;
    options.pc.cf=options.pc.cf*options.pc.mu0;
    options.pc.cr=options.pc.cr*options.pc.mu0;
end

% options.p.m=1.3*options.p.m;
% options.p.J=1.3*options.p.J;
% options.p.l_R=1.3*options.p.l_R;
% options.p.L=options.p.l_R+options.p.l_F;

%tau=load_path(scenario,options.duration);
%initial state
options.n=int32(length(model(0,zeros(10,1),zeros(5,1),options.p)));
x0=getVehicleX0(model,tau,options)+0*[0 -0.2 degtorad(-3) 0 0 0 0]';

%initialize controller;
controller=controller.init(model,options);

%run
%profile on
benchmark=simulate(tau,tauD,controller,model,x0,options);
%profile off
%profile viewer

%visualize
if options.DISPLAY_OUTPUT
    disp(benchmark)
    % draw real time plot
    visualize(benchmark)
end
end
```

Startup.m:

```
addpath( genpath( cd ) );
```

References

[1] R. Solea, U. Nunes, “Trajectory planning and sliding mode control based trajectory tracking for cybercars”

[2] J.M.Snider, “Automatic steering methods for autonomous automobile path tracking”

[3] S.Spurgeon, “Sliding mode control : a tutorial”

[4] J.M.Yang, J.H.Kim, “Sliding Mode Control for Trajectory Tracking of Nonholonomic Wheeled Mobile Robots”

[5] H.K.Khalil, Nonlinear Systems, 3rd edition