International Journal of Innovative Technology and Interdisciplinary Sciences
www.IJITIS.org
ISSN:2613-7305
Volume 4, Issue 1, pp. 560-603, 2021
DOI: https://doi.org/10.15157/IJITIS.2021.4.1.560-603
Received October 22, 2020; Accepted December 04, 2020

Model Predictive Control for Autonomous Vehicle Tracking

Vu Trieu Minh [*a1], **Resa Moezzi** [b2], **Klodian Dhoska** [c3], **John Pumwa** [d4]
[1] School of Engineering, Tallinn University of Technology, Tallinn, Estonia
[2] Institute for Nanomaterials, Advanced Technologies and Innovation, Technical University of Liberec, Czech Republic
[3] Department of Production and Management, Faculty of Mechanical Engineering, Polytechnic University of Tirana, Albania
[4] Department of Mechanical Engineering, Papua New Guinea University of Technology, Papua New Guinea
[*a] vutrieuminh@gmail.com; [b]rezamoezy@gmail.com; [c]kdhoska@upt.al;
[d]john.pumwa@pnguot.ac.pg

ABSTRACT

This study develops model predictive control (MPC) schemes for controlling autonomous vehicles tracking on feasible trajectories generated from flatness or polynomial equations. All of the vehicle online moving parameters including coordinate positions, body orientation angle, and steering angle are included into the MPC optimizer for calculating the real-time optimal inputs for the vehicle linear velocity and its steering velocity to minimize the errors between the desired and the actual course of travel. The use of MPC can simplify and eliminate the complexity of controller design since MPC can work itself as a system modelling controller. MPC can also handle on-line the constraints of any variables exceeding their limits. However the high computational demands are the main challenge for this method applying for the real applications.

Keywords: Model predictive control, autonomous vehicle, feasible path, optimal tracking.

1. INTRODUCTION

Autonomous vehicles have been received considerable attention in recent years and the needs are arising for the mechatronic systems to control the vehicle tracking from any given start points to any given destination points online generated from the global positioning system (GPS) and subject to the vehicle physical constraints.

This study develops a real-time control system for an autonomous ground vehicle directed online from the GPS maps or/and from unmanned aerial vehicles (UAVs) images. This system can be applied for auto traveling on road or off road for unmanned ground vehicles. The system can also be used for auto parking and auto driving vehicles.

Motivation for the use of MPC is its ability to handle the constraints online within its open-loop optimal control problems while many other control techniques are conservative in handling online constraints or even try to avoid activating them, thus, losing the best performance that may be achievable. MPC can make the close loop system operating near its limits and hence, produce much better performance.

However, MPC regulator is designed for online implementation, any infeasible solution of the optimization problems cannot be allowed. To improve the system's stability once some constraints are violated, some kinds of softened constraints or tolerant regions can be developed whereas the output constraints are not strictly imposed and can be violated somewhat during the evolution of the performance.

To deal with the system uncertainties and the model-plant mismatches, robust MPC algorithms can be built accounting for the modelling errors at the controller design. Robust MPC can forecast all possible models in the plant uncertainty set and the optimal actions then can be determined through the min-max optimization.

The reference feasible trajectories can be generated online using solver for ordinary differential equations (ODEs) with the flatness or polynomial equations presented in [1]. Algorithms for robust MPC tracking set points are referred in Minh V.T and Hashim F.B (2011) [2] where the system's uncertainties are demonstrated by a set of multiple models via a tree trajectory and its branches and the robust MPC problem is to find the optimal control actions that, once implemented, cause all branches to converge to a robust control invariant set.

Application of MPC in controlling vehicle speed and engine torque is referred to in Minh V.T and Hashim F.B (2012) [3] where a real time transition strategy with MPC is achieved for quick and smooth clutch engagements. Essential knowledge on vehicle handling and steering calculations is referred to in Minh V.T (2012) [4] in chapter 8 and chapter 9, where the vehicle dynamic behaviours are analysed and applied for designing a fee-error feedback controller for its autonomous tracking.

Robust MPC schemes for input saturated and softened state constraints are referred from Minh V.T and Afzulpurkar N (2005) [5] where uncertain systems are used with linear matrix inequalities (LMIs) subject to input and output saturated constraints. Nonlinear MPC (NMPC) algorithms are referred to in Minh V.T and Afzulpurkar N (2006) [6] where three NMPC regulators of zero terminal region, quasi-infinite horizon, and softened state constraints are presented and compared. In NMPC, all solution for the regulator is implemented for close-lope control by solving on-line the ODEs repeatedly.

Control of vehicle tracking with MPC can be referred to in some several latest research papers. However the idea of an MPC for online tracking optimal trajectories generated from flatness or polynomial equations is still not available. Some of MPC schemes for autonomous ground vehicle can be seen in Falcon P. *et al* (2008) [7] where an initial frame work based on MPC for a simplified vehicle is presented. However, the research has ignored the real-time solving of the vehicle ODEs equations and failed to generate the optimal controlled inputs for the vehicle linear velocity and its steering velocity. Similarly, another recent paper on optimal MPC for path tracking of autonomous vehicles by Lei L. *et al* (2011) [8] is presented where the vehicle's equations of motion are approximately linearized by the vehicle coordinates and the heading angle. The paper failed to include the steering angle in its equations.

Scheme for a robust MPC applied to mobile vehicle trajectory control can be seen also from Baharonian M. *el al* (2011) [9] with an assumption that there is a virtual reference moving according to the desired reference trajectory and then, the control problem becomes too simple and too trivial. An adaptive trajectory tracking control of wheeled mobile is considered by Wang J. *et al* (2011) [10], however the paper does not mention on how a feasible trajectory can be generated and how some optimal control actions can be achieved for the best trajectory tracking performance. Another reference by Shim T. *et al* (2912) [11] derives algorithms for MPC to control the front steering velocity and the wheel torque for autonomous ground vehicle. However, the paper failed to implement the on-line solving ODEs equations of NMPC. Another scheme of robust MPC to control fast vehicle dynamics with approximately linearized model is developed by Peni T and Bokor J (2006) [12] from some unrealistic assumptions that the vehicle velocity is a constant, and so that the system is always linear. The latest development of MPC and autonomous vehicle tracking are referred to in [14-34].

So, the idea of this paper is to generate comprehensive schemes for MPC to track reference trajectories generated online by ODEs from the vehicle kinematics. Vehicle location data can be collected and processed online from GPS or UAV. Then, the vehicle can automatically generate optimal feasible trajectory subject to feasible constraints on speed, steering, sideslip, obstacles, etc., and track exactly on these paths. The paper is constructed as follows: Section 2 describes the system kinematic equations; Section 3 develops MPC schemes; Section 4 presents MPC using linearized model; Section 5 develops MPC using nonlinear model; Section 6 presents the MPC performances comparison; Finally some study remarks are concluded in section 7.

2. SYSTEM MODELING

This part briefly presents concept of no holonomic system and definition of Lie bracket of two vector fields $X_1(q)$ and $X_2(q)$ in the matrix form corresponding to the Cartesian (x, y) coordinate system:

$$\left[X_1, X_2\right](p) = \frac{\partial X_2}{\partial q} X_1\Big|_p - \frac{\partial X_1}{\partial q} X_2\Big|_p = \left[X_2(f)X_1 - X_1(f)X_2\right] \tag{1}$$

where $\frac{\partial X_1}{\partial q}, \frac{\partial X_2}{\partial q}$ are Jacobian matrices, X_1 and X_2 are vector fields on a smooth m-dimensional manifold M of $(q^1, q^2, ..., q^m)$ around some point $p \in M$ and $\left[X_1, X_2\right]$ is the Lie bracket. The nonlinear motions of the vehicle can be presented via the following Lie bracket vector field.

Once a vehicle is rolling without slipping; the vehicle dynamic can be represented in a set of first-order differential constraints on its configuration variables. If the vehicle has the rear-wheel driving, the kinematic model can be derived in equation (2) and shown in figure 1:

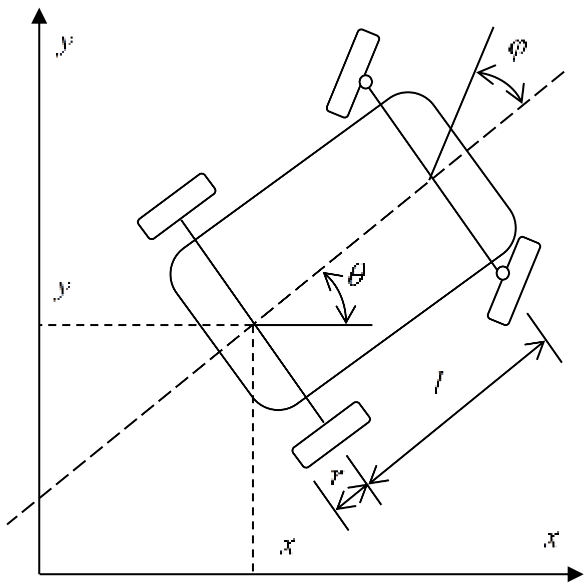


Figure 1. A simplified vehicle model

In figure 1, r is the vehicle wheel radius and l is the distance between the wheels; x and y are the Cartesian coordinates of the rear wheel, θ measures the orientation of the vehicle body with respect to the x axis, and φ is the steering angle.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\varphi} \end{bmatrix} = \underbrace{\begin{bmatrix} \cos\theta \\ \sin\theta \\ \dfrac{\tan\varphi}{l} \\ 0 \end{bmatrix}}_{X_1} u_1 + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}}_{X_2} u_2 \tag{2}$$

In equation (2), the vehicle motion is controlled by two inputs, u_1 is the linear driving velocity, and, u_2 is the angular steering velocity. There are four (4) coordinates or state variables, namely the position of the vehicle $x_1 = x$ and $x_2 = y$; the angle, $x_3 = \theta$, of the vehicle body orientation with respect to the x axis; and the steering angle, $x_4 = \varphi$.

A useful tool to test the controllability of this nonlinear system is the Lie brackets rank condition as referred to in De Luca *et al.* (1998) [13].

$$rank\left[X_1, X_2, X_3, X_4\right] = \left[X_1, X_2, [X_1, X_2], [X_1, [X_1, X_2]]\right] = 4 \tag{3}$$

The four components of function X_1 from equation (2) are: $X_1^1 = \cos\theta$, $X_1^2 = \sin\theta$, $X_1^3 = \dfrac{\tan\varphi}{l}$, and $X_1^4 = 0$.

For the feasible control of this dynamical system, the Lie brackets in (1) must be transferred and satisfied equations in (3). It can be seen that Jacobian matrix of the function X_1 is:

$$J_F(x, y, \theta, \varphi) = X_1(f) = \begin{bmatrix} \dfrac{\partial X_1^1}{\partial x} & \dfrac{\partial X_1^1}{\partial y} & \dfrac{\partial X_1^1}{\partial \theta} & \dfrac{\partial X_1^1}{\partial \varphi} \\ \dfrac{\partial X_1^2}{\partial x} & \dfrac{\partial X_1^2}{\partial y} & \dfrac{\partial X_1^2}{\partial \theta} & \dfrac{\partial X_1^2}{\partial \varphi} \\ \dfrac{\partial X_1^3}{\partial x} & \dfrac{\partial X_1^3}{\partial y} & \dfrac{\partial X_1^3}{\partial \theta} & \dfrac{\partial X_1^3}{\partial \varphi} \\ \dfrac{\partial X_1^4}{\partial x} & \dfrac{\partial X_1^4}{\partial y} & \dfrac{\partial X_1^4}{\partial \theta} & \dfrac{\partial X_1^4}{\partial \varphi} \end{bmatrix} = \begin{bmatrix} 0 & 0 & -\sin\theta & 0 \\ 0 & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & \dfrac{1}{l\cos^2\varphi} \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

The four components of function X_2 from equation (2) are: $X_2^1 = 0$, $X_2^2 = 0$, $X_2^3 = 0$, and $X_2^4 = 1$.

And Jacobian matrix of the function X_2 is:

$$J_F(x,y,\theta,\varphi) = X_2(f) = \begin{bmatrix} \frac{\partial X_1^1}{\partial x} & \frac{\partial X_1^1}{\partial y} & \frac{\partial X_1^1}{\partial \theta} & \frac{\partial X_1^1}{\partial \varphi} \\ \frac{\partial X_2^2}{\partial x} & \frac{\partial X_2^2}{\partial y} & \frac{\partial X_2^2}{\partial \theta} & \frac{\partial X_2^2}{\partial \varphi} \\ \frac{\partial X_2^3}{\partial x} & \frac{\partial X_2^3}{\partial y} & \frac{\partial X_2^3}{\partial \theta} & \frac{\partial X_2^3}{\partial \varphi} \\ \frac{\partial X_2^4}{\partial x} & \frac{\partial X_2^4}{\partial y} & \frac{\partial X_2^4}{\partial \theta} & \frac{\partial X_2^4}{\partial \varphi} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

From equation (1), the Lie bracket of vector field $X_3 = [X_1, X_2]$ is:

$$X_3 = [X_1, X_2] = [X_2(f)X_1 - X_1(f)X_2]$$

$$= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \cos\theta \\ \sin\theta \\ \frac{\tan\varphi}{l} \\ 0 \end{bmatrix} - \begin{bmatrix} 0 & 0 & -\sin\theta & 0 \\ 0 & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & \frac{1}{l\cos^2\varphi} \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -\frac{1}{l\cos^2\varphi} \\ 0 \end{bmatrix} \tag{4}$$

And the Lie bracket of vector field $X_4 = \left[X_1[X_1, X_2]\right]$ is:

$$X_4 = \left[X_1[X_1, X_2]\right] =$$

$$= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{2\tan\varphi}{l\cos^2\varphi} \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \cos\theta \\ \sin\theta \\ \frac{\tan\varphi}{l} \\ 0 \end{bmatrix} - \begin{bmatrix} 0 & 0 & -\sin\theta & 0 \\ 0 & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & \frac{1}{l\cos^2\varphi} \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ -\frac{1}{l\cos^2\varphi} \\ 0 \end{bmatrix} = \begin{bmatrix} -\frac{\sin\theta}{l\cos^2\varphi} \\ \frac{\cos\theta}{l\cos^2\varphi} \\ 0 \\ 0 \end{bmatrix} \tag{5}$$

Check the determinant of this 4x4 Jacobi-Lie bracket matrix from (3):

$$\det(X_1X_2X_3X_4) = \begin{pmatrix} \cos\theta & 0 & 0 & -\frac{\sin\theta}{l\cos^2\varphi} \\ \sin\theta & 0 & 0 & \frac{\cos\theta}{l\cos^2\varphi} \\ \frac{\tan\varphi}{l} & 0 & -\frac{1}{l\cos^2\varphi} & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} = \frac{1}{l^2\cos^4\varphi} \tag{6}$$

So, if $\varphi \neq \frac{\pi}{2}$, then $\det[X_1X_2X_3X_4]$ is well defined and the system in (2) is non-holonomic. This means that the dynamical system in (2) can be transformed from any given state to any other state or all of its position parameters are under controlled by the input vectors. Or it is possible to express all state variables as a function of the inputs.

In the next part, an approximate linearized system and its discretized form from continuous derivative equations in (2) will be developed. The discretized form of this system will be used for the MPC open loop optimization calculation.

The vehicle model in (2) is nonlinear and has the first order derivative form:

$$\dot{X} = f(x,u) \tag{7}$$

where the state variables are $x \triangleq [x, y, \theta, \varphi]'$, and the inputs are $u = [u_1, u_2]'$. The nonlinear equation in (7) can be expanded in Taylor series around the referenced point (x_r, u_r) at $\dot{X}_r = f(x_r, u_r)$, that:

$$\dot{X} \approx f(x_r,u_r) + f_{x,r}(x - x_r) + f_{u,r}(u - u_r) \tag{8}$$

where $f_{x.r}$ and $f_{r.x}$ are the Jacobean of f corresponding to x and u, evaluated around the referenced points (x_r, u_r).

Subtraction of (8) and $\dot{X}_r = f(x_r, u_r)$ results a linear approximation to the system at the reference points for a continuous time (t) model:

$$\dot{\tilde{X}}(t) = A(t)\tilde{X}(t) + B(t)\tilde{u}(t) \tag{9}$$

where $\tilde{X}(t) = X(t) - X_r(t) = \begin{bmatrix} x(t) - x_r(t) \\ y(t) - y_r(t) \\ \theta(t) - \theta_r(t) \\ \varphi(t) - \varphi_r(t) \end{bmatrix}$, and $\tilde{u}(t) = u(t) - u_r(t) = \begin{bmatrix} u_1(t) - u_{r1}(t) \\ u_2(t) - u_{r2}(t) \end{bmatrix}$,

$$A(t) = \begin{bmatrix} 0 & 0 & -u_{r1}(t)\sin\theta_r(t) & 0 \\ 0 & 0 & u_{r1}(t)\cos\theta_r(t) & 0 \\ 0 & 0 & 0 & \dfrac{u_{r1}(t)}{l\cos^2\varphi_r(t)} \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad B(t) = \begin{bmatrix} \cos\theta_r(t) & 0 \\ \sin\theta_r(t) & 0 \\ \dfrac{\tan\varphi_r(t)}{l} & 0 \\ 0 & 1 \end{bmatrix}$$

The continuous approximation of $\dot{\tilde{X}}(t)$ in (9) can be represented in the discrete-time (k) with time $k+1 = k + \Delta t$ and Δt is the length of sampling interval. The inputs $u(k)$ are held constant during the time interval $(k+1)$ and (k). The symbols of $x_k = x(k)$ and $u_k = u(k)$ are also used:

$$\begin{aligned} \tilde{X}(k+1) &= A(k)\tilde{X}(k) + B(k)\tilde{u}(k) \\ \tilde{Y}(k) &= C(k)\tilde{X}(k) \end{aligned} \tag{10}$$

where

$$A(k)=\begin{bmatrix}1 & 0 & -u_{r1}(k)\sin\theta_r(k)(\Delta t) & 0\\ 0 & 1 & u_{r1}(k)\cos\theta_r(k)(\Delta t) & 0\\ 0 & 0 & 1 & \dfrac{u_{r1}(k)}{l\cos^2\varphi_r(k)}(\Delta t)\\ 0 & 0 & 0 & 1\end{bmatrix},\ B(k)=\begin{bmatrix}\cos\theta_r(k)(\Delta t) & 0\\ \sin\theta_r(k)(\Delta t) & 0\\ \dfrac{\tan\varphi_r(k)}{l}(\Delta t) & 0\\ 0 & (\Delta t)\end{bmatrix},$$

$$C(k)=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix},\ \text{and,}$$

$$\tilde{X}(k)=X(k)-X_r(k)=\begin{bmatrix}x(k)-x_r(k)\\ y(k)-y_r(k)\\ \theta(k)-\theta_r(k)\\ \varphi(k)-\varphi_r(k)\end{bmatrix},\ \tilde{u}(k)=u(k)-u_r(k)=\begin{bmatrix}u_1(k)-u_{r1}(k)\\ u_2(k)-u_{r2}(k)\end{bmatrix}$$

In the above discretized model, the two control inputs are the difference in the actual and the desired vehicle linear velocity, $u_1(k)-u_{r1}(k)$, and the difference of the actual and desired steering angular velocity, $u_2(k)-u_{r2}(k)$. The four outputs, $y(k)=\tilde{Y}(k)=C(k)\tilde{X}(k)$, are totally measured and updated in each real-time scanning interval. It is important to note that the vehicle linearized model in (10) is a time variant system with its transfer function is depending on its positions and the scanning speeds.

The approximate linearized equations (10) are used to develop MPC algorithms in the next part.

3. MODEL PREDICTIVE CONTROL

This part presents the design of MPC algorithms for the discretized linearized model. MPC works out the optimal open-loop optimization problem that minimizes the difference between the predicted plant behaviour and the desired plant behaviour. MPC differs from other control techniques in that the optimal control problem is solved on-line for the current state of the plant, rather than off-line determined as the feedback policy. MPC has been widely applied in the robotic technologies because of its ability to handle input and output constrains in the optimal control problem.

MPC algorithms are now designed to control the two inputs of the vehicle driving velocity, $u_1(k)$, and, its steering velocity, $u_2(k)$, in order to achieve the desired outputs of the vehicle coordinate position, $x_1(k)=x(k)$, and $x_2(k)=y(k)$; the vehicle orientation body angle with respect to the x axis, $x_3(k)=\theta(k)$; and the steering angle, $x_4(k)=\varphi(k)$. All of these outputs are set to tract exactly on the given trajectory reference set points of $x_r(k)$, $y_r(k)$, $\theta_r(k)$, and $\varphi_r(k)$ at each discrete-time (k).

From (10), the prediction horizon for the outputs, $y_{k+i|k}$, and the input increments, $\Delta u_{k+i|k}$, can be rewritten as,

$$\begin{bmatrix} y_{k+1|k} \\ y_{k+2|k} \\ \vdots \\ y_{k+N_y|k} \end{bmatrix} = \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^{N_y} \end{bmatrix} x_{k|k} + \begin{bmatrix} CB \\ CAB+CB \\ \vdots \\ \sum_{i=1}^{N_y} CA^{i-1}B \end{bmatrix} u_{k-1} + \begin{bmatrix} CB & 0 & \cdots & 0 \\ CAB+CB & CB & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{N_y} CA^{i-1}B & \sum_{i=1}^{N_y-1} CA^{i-1}B & \cdots & \sum_{i=1}^{N_y-N_u+1} CA^{i-1}B \end{bmatrix} \begin{bmatrix} \Delta u_k \\ \Delta u_{k+1} \\ \vdots \\ \Delta u_{k+N_u-1} \end{bmatrix}$$

Then, the tracking set points MPC objective function with hard constraints is:

$$\min_{U \triangleq \{\Delta u_k,\dots,\Delta u_{k+N_u-1}\}} \left\{ J(U,x(k)) = \sum_{i=0}^{N_y-1} \left[(y_{k+i|k} - r_{k+i|k})' Q (y_{k+i|k} - r_{k+i|k}) + \Delta u'_{k+i|k} R \Delta u_{k+i|k} \right] \right\},$$

subject to:

$$u_k \in \mathcal{U}, \text{ and } u_{k+i} \in [u_{\min}, u_{\max}],\ \Delta u_{k+i} \in [\Delta u_{\min}, \Delta u_{\max}], \text{ for } i = 0,1,\dots,N_u-1, \tag{11}$$

$y_k \in \mathcal{Y}$, and $y_{k+i|k} \in [y_{\min}, y_{\max}]$, for $i = 0,1,\dots,N_y-1$,

$\Delta u_k = u_k - u_{k-1} \in \Delta\mathcal{U}$, and $\Delta u_{k+i} = 0$, for $i \ge N_u$,

$x_{k|k} = x(k)$, $x_{k+i+1|k} = A(k)x_{k+i|k} + B(k)u_{k+i}$, $u_{k+i|k} = u_{k+i-1|k} + \Delta u_{k+i|k}$, $y_{k+i|k} = C(k)x_{k+i|k}$

where $x(k)$ denotes the state variables at the current discrete time (*k*): $U \triangleq \{\Delta u_k,\dots,\Delta u_{k+N_u-1}\}$ is the solution of input increments, N_u is the inputs predictive horizon; N_y is the outputs predictive horizon; $y_{k+i|k}$ are the predictive outputs at the current discrete time (*k*), $r_{k+i|k}$ are the corresponding reference output setpoints; $\Delta u_{k+i|k}$ are the input increments prediction with $\Delta u_{k+i|k} = u_{k+i|k} - u_{k+i-1|k}$; $Q = Q' \ge 0$, $R = R' > 0$ are the weighting penalty matrices for predicted outputs and input increments, respectively.

The MPC regulator computes the optimal solution, $U^* \triangleq \{\Delta u_k^*,\dots,\Delta u_{k+N_u-1}^*\}$ and the new inputs $u_{k+i|k} = u_{k+i-1|k} + \Delta u_{k+i|k}$, from the objective function (11), then applies only the first element of the current inputs increment, Δu_k^*, and calculate the current optimal inputs, $u^*(k) = u_{k-1} + \Delta u_k^*$, and inserts this $u^*(k)$ into the system. After having inserted the current optimal inputs at time k, the MPC regulator repeats the optimization, $u^*(k+1)$, for the next interval time, $k+1$, based on the new calculation of the update state variables $x(k+1)$. This way, the closed loop control strategy is obtained by solving on-line the open loop optimization problem.

By substituting $x_{k+N_y|k} = A^{N_y}(k)x(k) + \sum_{i=0}^{N_y-1} A^i(k)B(k)u_{k+N_y-1-i}$, equation (11) can be rewritten as a function of only the current state $x(k)$ and the current set points $r(k)$:

$$\Psi\left(x(k), r(k)\right) = \frac{1}{2} x'(k) Y x(k) + \min_U \left\{ \frac{1}{2} U' H U + x'(k) r(k) F U \right\}, \tag{12}$$

subject to the hard combined constraints of $GU \le W + Ex(k)$, where the column vector $U \triangleq \left[\Delta u_k,\dots,\Delta u_{k+N_p-1}\right]' \in \Delta\mathcal{U}$ is the prediction optimization vector; $H = H' > 0$, and H, F, Y G, W and E are matrices obtained from Q, R and given constraints in (11). As only the optimizer U is needed, the term involving Y is usually removed from (12). Then, the optimization problem in (12) is a quadratic program and depends only on the current state $x(k)$ and the current set points $r(k)$ subject to the hard combined constraints. The implementation of MPC requires the on-line solution of this quadratic program at each time interval (*k*).

In reality, the system would have both input and output constraints and the difficulty will arise due to the inability to satisfy the output constraints due to the input constraints. Since MPC is designed for on-line implementation, any infeasible solution of the online optimization problem in (12) cannot be allowed. Normally the input constraints are based on the physical limits of the vehicle and can be considered as hard constraints. If the outputs constraints are the tracking positions which are not strictly imposed and can be violated somewhat during the evolution of the performance. To guarantee the system stability once the outputs violate the constraints, the hard constrained optimization in (11) can be modified to a new MPC objective function with softened constraints as:

$$\min_{U\triangleq\left\{\Delta u_k,\dots,\Delta u_{k+N_u-1}\right\}}\left\{J(U,x(k))=\sum_{i=0}^{N_y-1}\left[(y_{k+i|k}-r_{k+i|k})'Q(y_{k+i|k}-r_{k+i|k})+\Delta u'_{k+i|k}R\Delta u_{k+i|k}+\varepsilon'_i(k)\Lambda\varepsilon_i(k)\right]\right\} \qquad (13)$$

where $\varepsilon_i(k)\geq 0$ are the new penalty terms added to the MPC objective function, $\varepsilon_i(k)=\left[\varepsilon_y;\varepsilon_u\right]$, $y_{\min}-\varepsilon_y\leq y_{k+i|k}\leq y_{\max}+\varepsilon_y$ and $u_{\min}-\varepsilon_u\leq u_{k+i|k}\leq u_{\max}+\varepsilon_u$. And $\Lambda=\Lambda'\geq 0$ is the new penalty matrix (usually $\Lambda>0$ and set with small values). These terms, $\varepsilon_i(k)$, will keep the constrained violations at low values until the solution is returned. A new MPC algorithm for softened constraints to select the optimal inputs $u^*(k+i\,|\,k)$ can be conducted similarly to (12) with the new added penalty terms $\varepsilon'_i(k)\Lambda\varepsilon_i(k)$.

Furthermore, in order to increase the possibility of the MPC to find out online solution in critical time, some output set points can be temporally deleted because the deletion of some output set points can make the system looser and the probability that the MPC optimizer can find a solution will increase. Deletion of some output set points can be conducted via temporally assigning zeros in the penalty matrices Q and R. For example, the above MPC controller has four outputs $y=\left[y_1,y_2,y_3,y_4\right]'$, if we select the 4 by 4 penalty matrix $Q=diag\{1,1,1,1\}$, implying that all four outputs are required to reach set points. However, if we want to delete the output set points for y_3,y_4 or it is required that only the two outputs, y_1,y_2, to reach the set points, we can choose a new penalty matrix $Q=diag\{1,1,0,0\}$. In other words, the new controlled variables now become $y=\left[y_1,y_2\right]'$.

Robustness of MPC can be also increased if some set points can be relaxed into regions rather than in some specific values. Then, a new MPC algorithm can be developed if the set points $r(k)$ now can be changed into some regions. An output region is defined by the minimum and maximum values of a desired range. The minimum value is the lower limit, and the maximum value is the upper limit and satisfied $y_{lower}\leq y_{k+i|k}\leq y_{upper}$. The modified objective function for the MPC with output regions is:

$$\min_{U\triangleq\left\{\Delta u_k,\dots,\Delta u_{k+N_u-1}\right\}}\left\{J(U,x(k))=\sum_{i=0}^{N_y-1}\left[z'_{k+i|k}Qz_{k+i|k}+\Delta u'_{k+i|k}R\Delta u_{k+i|k}\right]\right\}, \qquad (14)$$

where $z_{k+i|k}\geq 0$; $z_{k+i|k}=y_{k+i|k}-y_{upper}$ for $y_{k+i|k}>y_{upper}$;

$$z_{k+i|k} = y_{lower} - y_{k+i|k} \text{ for } y_{k+i|k} < y_{lower};\ z_{k+i|k} = 0 \text{ for } y_{lower} \le y_{k+i|k} \le y_{upper}$$

As long as the outputs still lie inside the desired regions, no control actions are taken because none of the control objectives have been violated, all $z_{k+i|k} = 0$. But when an output violates the desired region, the control objective in the MPC regulator will activate and push them back to the desired regions. This modified MPC objective function can be applied for the autonomous tracking vehicle when the desired set points are changed to some desired regions. The tracking trajectory will become smoother and the controller tasks will be reduced to maintain the outputs in the desired regions.

Numerical experiments of the MPC schemes are presented in the following parts of this research.

4. MPC USING LINEARIZATION MODEL

This part presents the MPC performance for linearized vehicle model in (10). The diagram of the MPC control system is shown in figure 2.

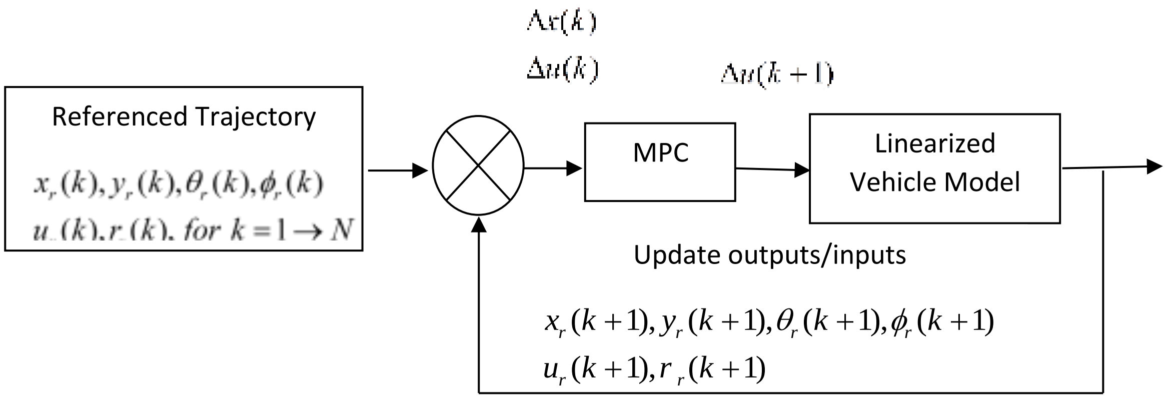


Figure 2. MPC control system

For the trajectory tracking, a reference trajectory is generated by solving the trajectory differential equations in (2). The difference of the reference trajectory parameters (set points) and the actual current vehicle parameters is provided to the MPC regulator. The MPC regulator will calculate the optimized control input horizon. Only the first element of this optimal solution is fed to the linearized vehicle model to generate the next outputs of the vehicle. The update system outputs are now compared with the update set points in the reference trajectory for the next MPC regulator calculation repetition.

4.1 MPC for tracking a full circle

For generating a full circle reference trajectory, the reference desired inputs are set at $u_1 = r\omega$ and $u_2 = 0$. The initial positions are selected as $\begin{bmatrix} x_0 & y_0 & \theta_0 & \varphi_0 \end{bmatrix}' = \begin{bmatrix} 0 & 0 & 0 & \arctan\frac{r}{l} \end{bmatrix}'$ (referred to figure 1). For this simulation, we use $r = 0.5m$, $l = 1.5m$, and $\omega = 1 rad/\sec$. The reference set points are generated using online ODE45 function in Matlab and shown in figure 3.

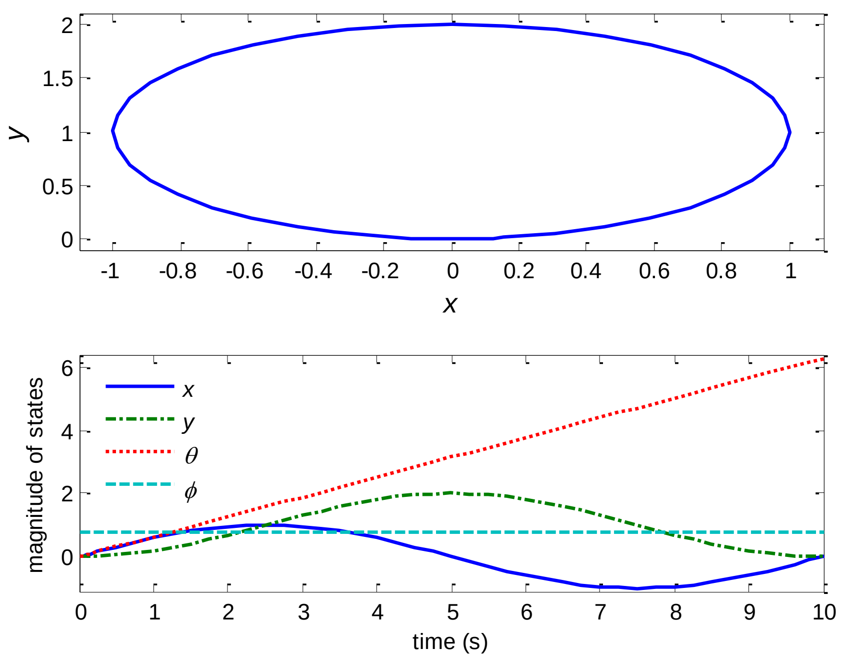


Figure 3. Circular reference set points

We now use the MPC control system in figure 2 to track the vehicle along the circular path in figure 3.

For this simulation, the initial positions of the vehicle are set at $X_0 = \begin{bmatrix} -0.5 & -0.5 & 0 & 0 \end{bmatrix}'$; The constraints are set at $u_{\min} = [-1,-1]'$, $u_{\max} = [1,1]'$, $\Delta u_{\min} = [-0.5,-0.5]'$, $\Delta u_{\max} = [0.5,0.5]'$, $y_{\min} = [-1,-1,-1,-1]'$, and $y_{\max} = [1,1,1,1]'$; The predictive horizons are set at $N_u = 10$ and $N_y = 10$; The penalty matrices are set at $Q = diag\{1,1,1,1\}$ and $R = diag\{1,1\}$. Performance of the MPC with linearized vehicle model to track the circular reference is shown in figure 4. The MPC optimizer is minimizing the tracking errors $y_{k+i|k} - r_{k+i|k}$ at each points (discrete time intervals) during its evolution performance from the initial position, $error_{initial} = y_{k|0} - r_{k|0} = \begin{bmatrix} -0.5 \\ -0.5 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0.7854 \end{bmatrix} = \begin{bmatrix} -0.5 \\ -0.5 \\ 0 \\ -0.7854 \end{bmatrix}$ to the final position, $error_{final} = y_{k|N} - r_{k|N} = \begin{bmatrix} -0.0202 \\ 0.0257 \\ 0.0094 \\ 0.7648 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0.7854 \end{bmatrix} = \begin{bmatrix} -0.0202 \\ 0.0257 \\ 0.0094 \\ -0.0206 \end{bmatrix}$, or very small errors are left at the end of the tracking trajectory.

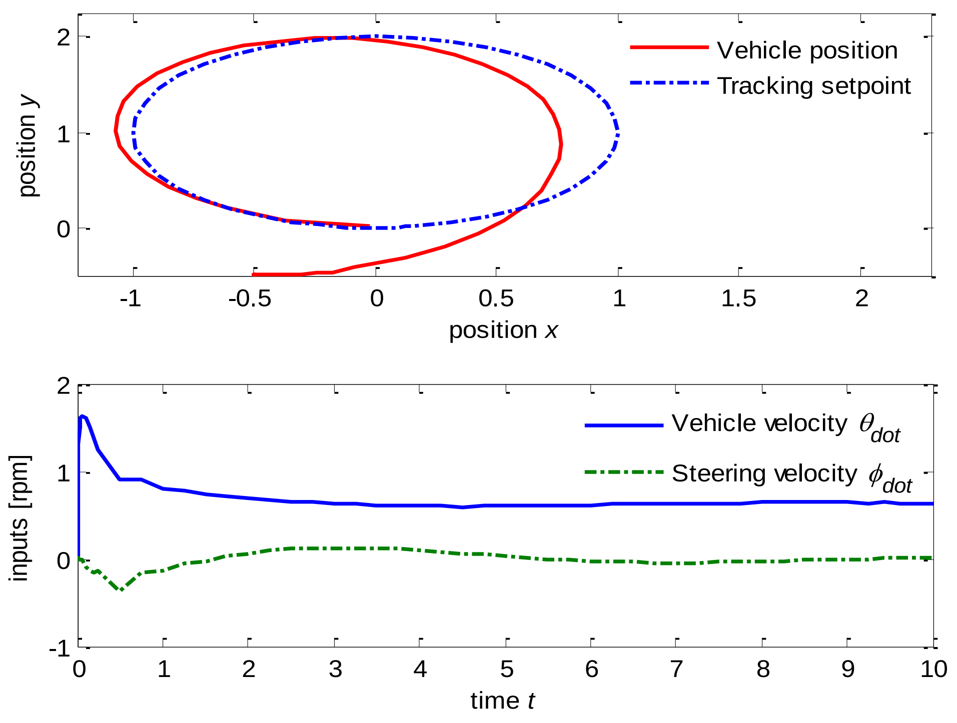


Figure 4. Tracking MPC linearized model

In MPC, if the prediction horizon is shortened, the calculation burden will be considerably reduced but will lead to incremental changes in the inputs and then, bad performance of the outputs. With shortened outputs and inputs predictions, the system may become instable. Figure 5 shows the MPC performance with shortened predictive horizons to $N_u = 4$ and $N_y = 4$.

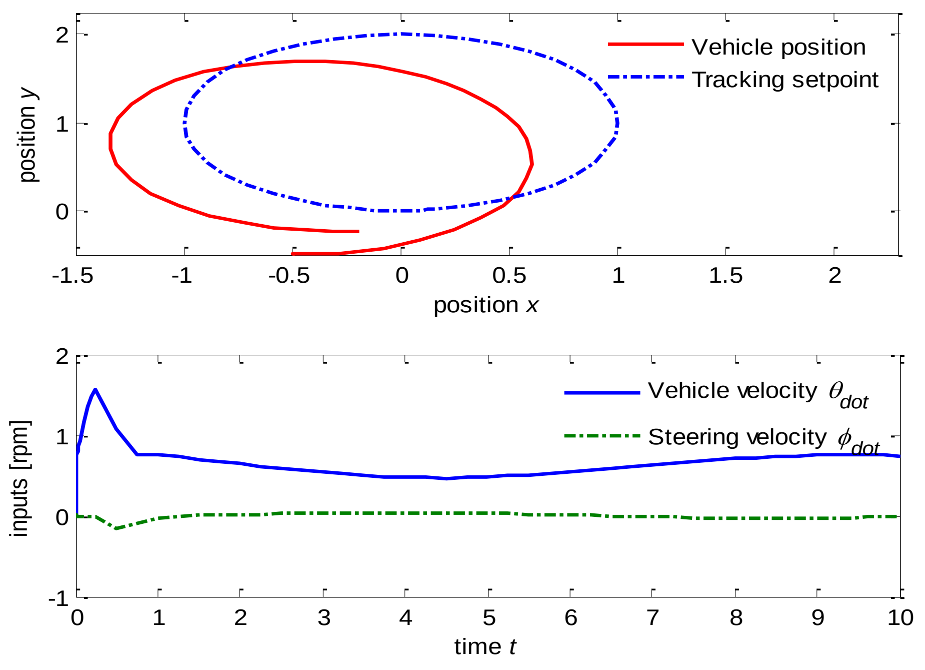


Figure 5. MPC linearized model with short horizon

Sufficient long prediction horizon will increase the MPC performance and its stability. However, the calculation burden will also be dramatically increased. The next simulation shown in figure 5 runs with $N_u = 20$ and $N_y = 20$. Performance of the tracking outputs is much improved as well as the inputs become smoother (easier to regulate the inputs to achieve the outputs).

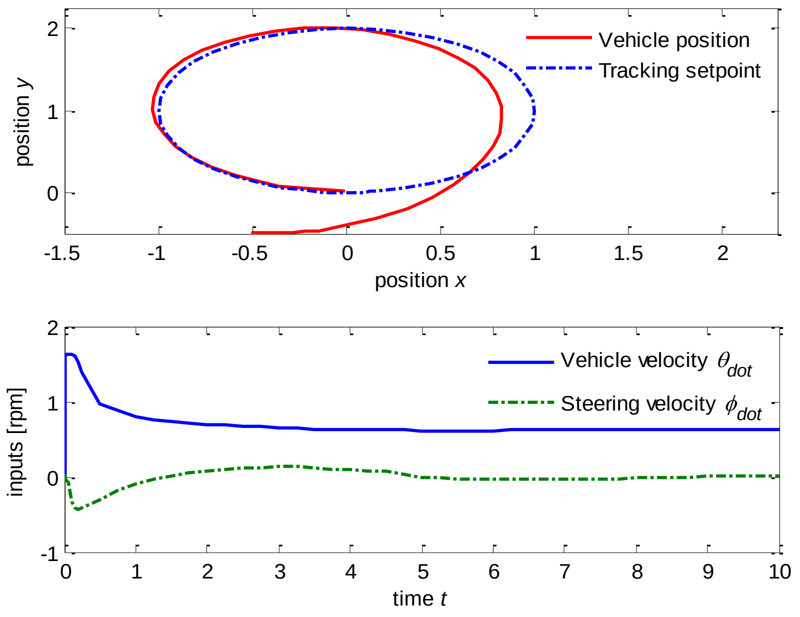


Figure 5. MPC linearized model with long horizon

However with too long horizon length, MPC will result too slow control increments and therefore deteriorate the controlled performance. The system becomes instable as shown in figure 6 with too long prediction horizon of $N_u = 23$ and $N_y = 23$.

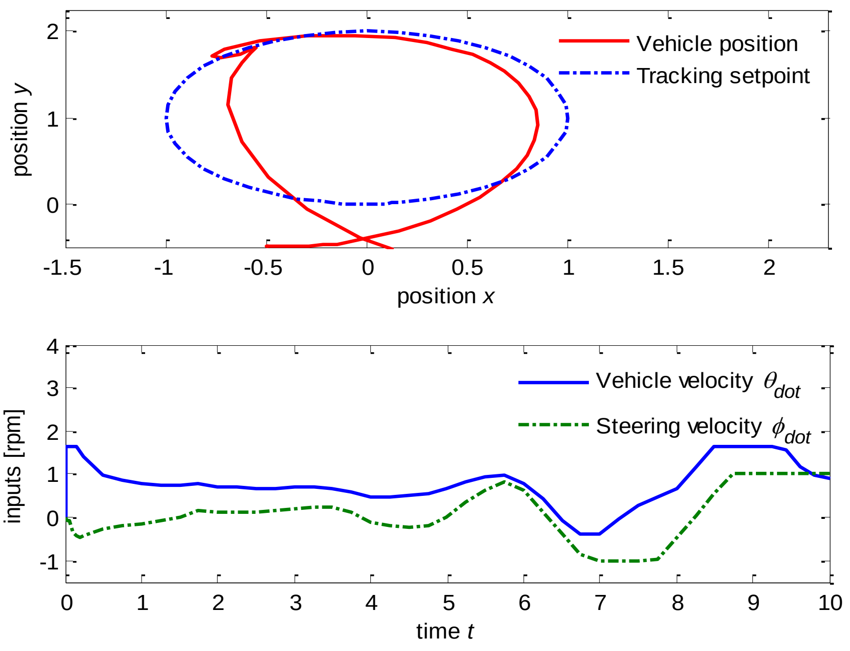


Figure 6. MPC linearized model with too long horizon and instability

Regulation of the penalty matrices can also affect the MPC performance. If we set $Q \gg R$ (Q is set much larger than R), then any small changes in the outputs will affect dramatically to the MPC objective function. It means that the inputs are set to be changed faster than the outputs. However, the vehicle inputs (speeds) are harder to be regulated or changed, so we can scarify some tracking output errors to gain some smoother inputs by setting $Q \ll R$. The next simulation runs with $N_u = 6$, $N_y = 6$, $Q = diag\{1,1,1,1\}$, and $R = diag\{10,10\}$. Figure 7 shows that the inputs become smoother (easier to control) but the output tracking errors become considerably larger.

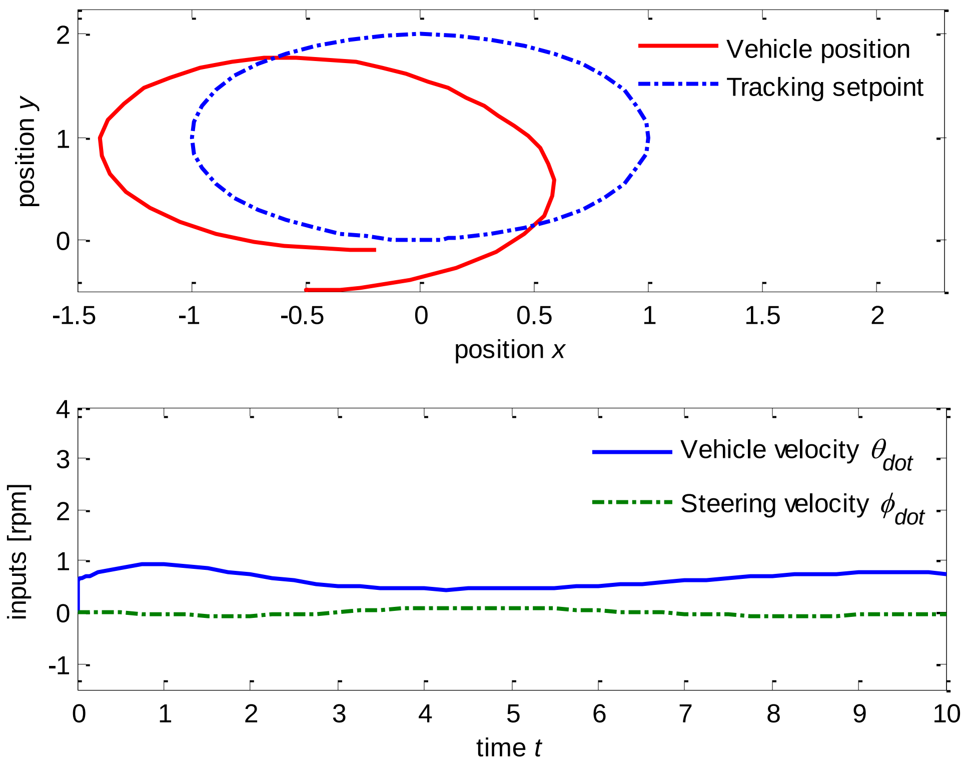


Figure 7. MPC linearized model with $Q \ll R$

For the case of $Q \gg R$, we set now $N_u = 6$, $N_y = 6$, $Q = diag\{10,10,10,10\}$, and $R = diag\{1,1\}$. Figure 8 shows the system becomes very sensitive to the input changes. Those faster input changes can be seen and resulted triangular in shape. These inputs shape is unrealistic since we cannot control the vehicle velocity on that shape. The conclusion is that that the system will be instability.

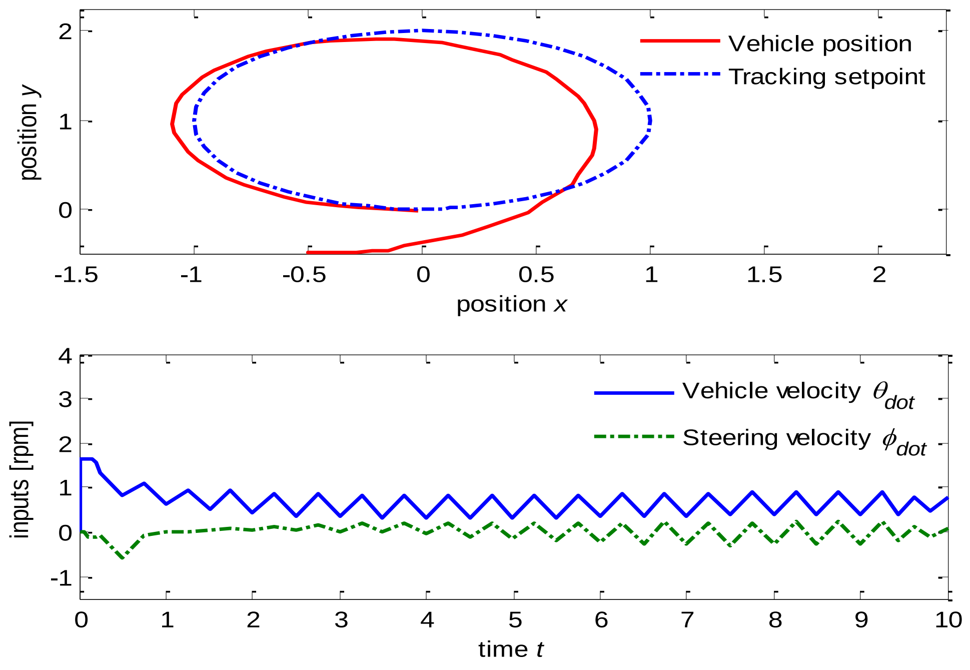


Figure 8. MPC linearized model with $Q \gg R$

Another way to regulate the system is to change the reference set points $(y_{k+i|k} - r_{k+i|k})$. In the previous simulations, we set the set point errors at zeros, $r_k = y_k - y_r = [0,0,0,0]'$, for all difference of the reference trajectory and the vehicle positions. To offset the vehicle sideslips or to compensate the model-plant mismatches, we can dynamically change these set point errors. For example, if we set $r_k = [0.1, 0.1, 0, 0]'$, the MPC performance is shown in figure 9, the final position of the vehicle becomes $[x_F, y_F] = [0.1, 0.1]$, but the vehicle tracks faster to the reference trajectory.

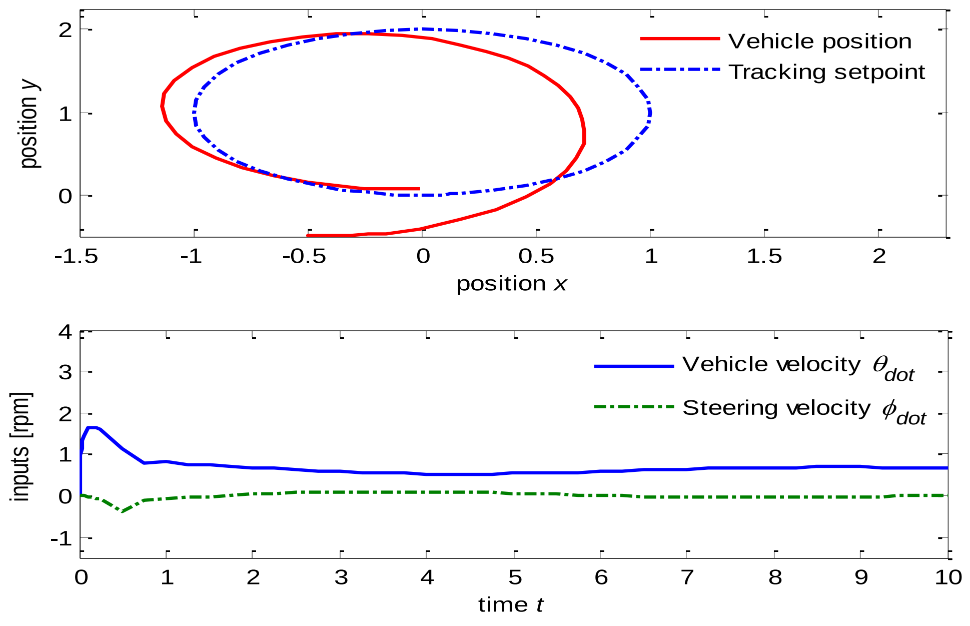


Figure 9. MPC linearized model with set point offsets variation

We can also assign the offsets to the vehicle orientation angle, θ, and steering angle, φ. For example, if we set the tracing errors at $r_k = [0, 0, 0.1, 0.1]'$, the MPC performance will be in figure 10. Due to the positive error offsets on the orientation and steering angles, the vehicle rotates in a smaller radius and also has the destination parameters of $[\theta_F, \varphi_F] = [0.1, 0.1]$.

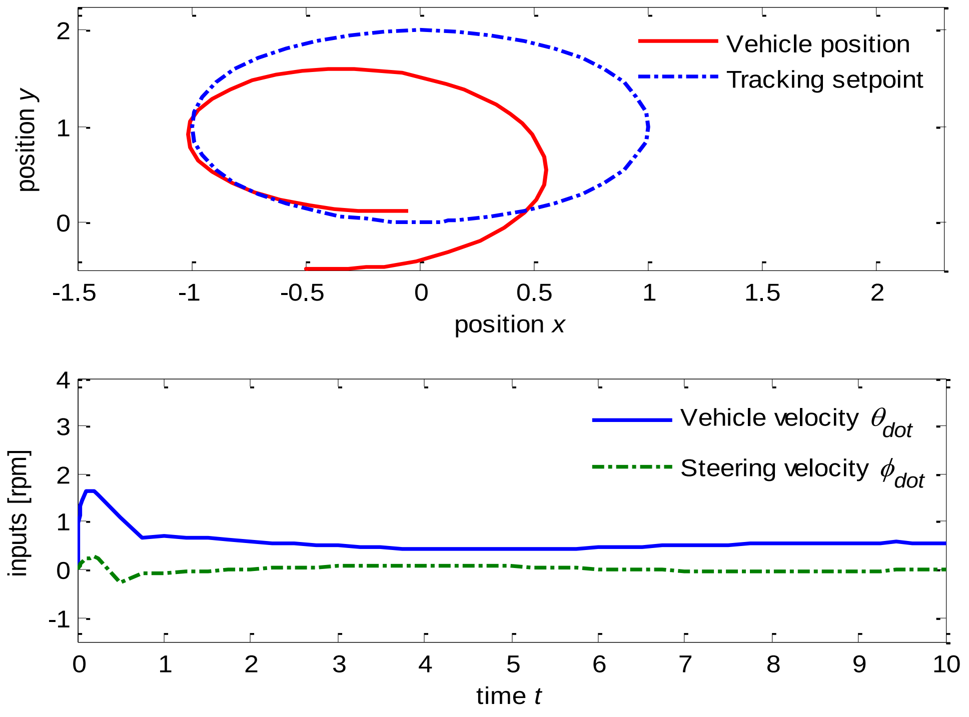


Figure 9. MPC linearized model with set point offsets in angles

All of the above MPC performances are set with the initial position of the vehicle, $X_0^{\text{Vehicle}} = [-0.5 \quad -0.5 \quad 0 \quad 0]'$. It is quite different to the initial position of the reference trajectory, $X_0^{\text{Reference}} = \left[0 \quad 0 \quad 0 \quad \arctan\frac{r}{l}\right]'$. This difference can be considered as the measured output errors or the model-plant mismatches. MPC regulator can gradually minimize these tracking errors during its evolution and drive the vehicle closer to its reference set points. In the next parts, we will investigate the ability of the MPC to track the vehicle on any feasible paths from any given start points to any given destination points generated directly from the kinematic differential equations in (2).

4.2. MPC for tracking flatness trajectory

Flatness trajectory generation is presented in [1]. Figure 10 shows a flatness trajectory for the vehicle from the initial position, $[x_0, y_0] = [0, 0]$, to the final position, $[x_F, y_F] = [10, 10]$, and the development of the orientation angle, θ, and steering angle, φ, during this travel. The time for completing this travel is set at $T = 100\,\text{sec}$;

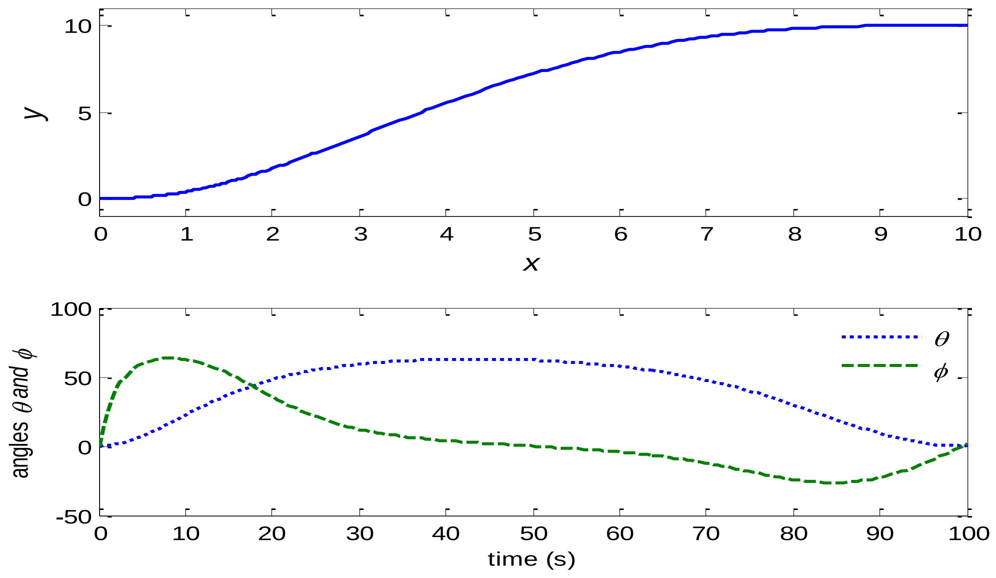


Figure 10. Flatness trajectory reference set points

For this MPC tracking, the initial positions of the vehicle are set at $X_0 = \begin{bmatrix} 0 & -0.5 & 0 & 0 \end{bmatrix}'$; The predictive horizons are set at short $N_u = 4$ and $N_y = 4$; Penalty matrices are set equally at $Q = diag\{1,1,1,1\}$ and $R = diag\{1,1\}$; The reference velocity inputs are $u_1 = 1$ (u_1 is set at 0 at the initial point, during the first 1/5 travelling time length, u_1 will gradually increase and maintain at 1 for 60 sec, during the final 1/5 traveling time length, u_1 will decrease back to 0), and $u_2 = 0$. Performance of this MPC is shown in figure 11. The vehicle starts from an initial velocity of $u_1 = 0$ and from its initial positions of $X_0 = \begin{bmatrix} 0 & -0.5 & 0 & 0 \end{bmatrix}'$, gradually tracks to the reference trajectory in 10 sec.

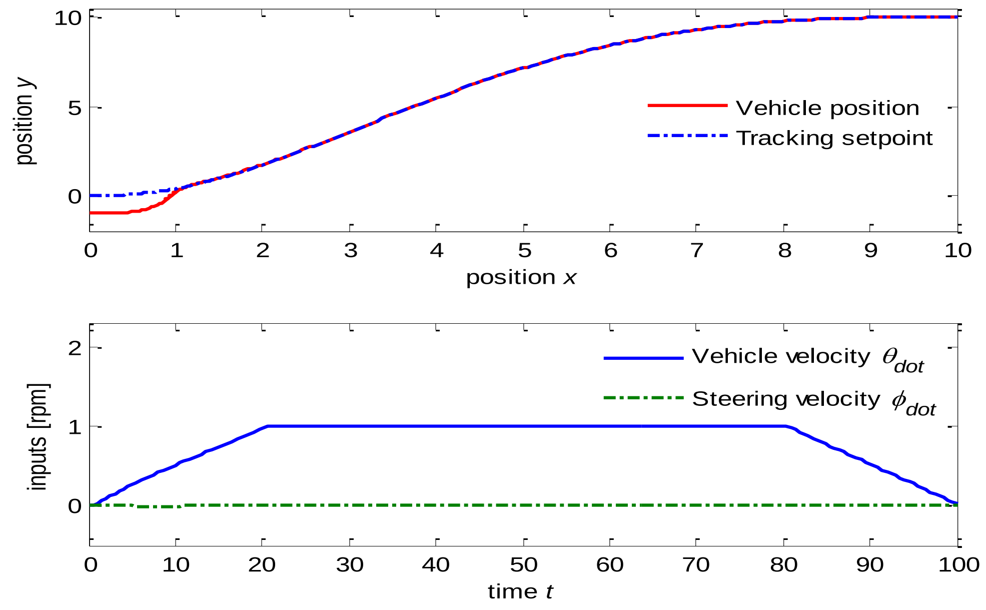


Figure 11. MPC for tracking flatness trajectory

Next, we lengthen the horizon prediction to $N_u = 8$ and $N_y = 8$, we now can see that the too long prediction horizon can degrade the MPC performance. In this example, the system becomes instable as shown in figure 12.

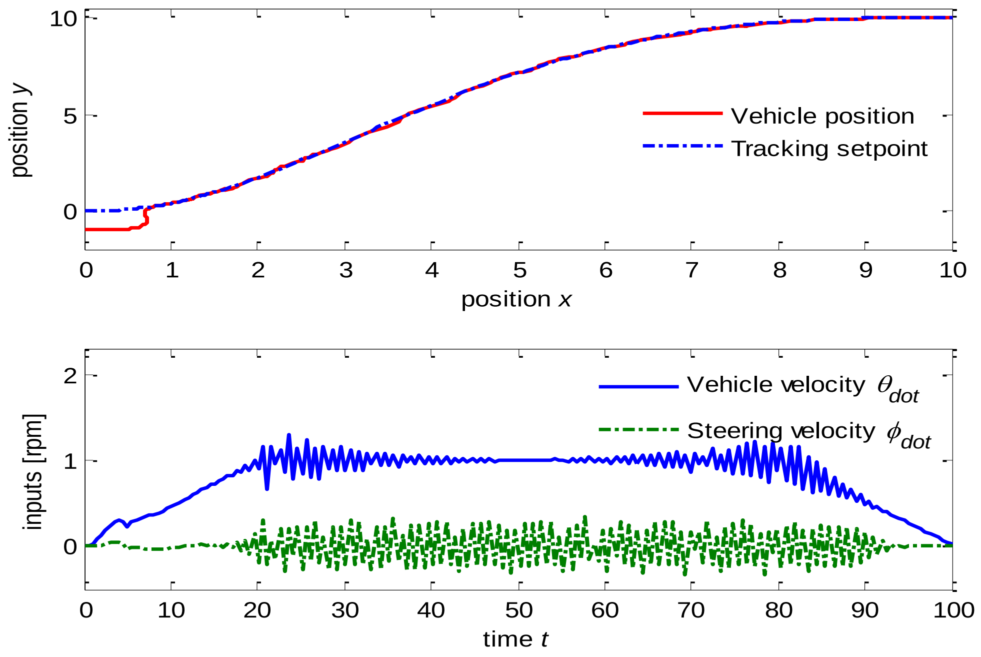


Figure 12. MPC with too long horizon

The system's instability shown in figure 12 is due to the too incentive changes of the vehicle input velocities. If we increase penalty matrix values, R, or set $Q < R$, the input increments will be slower and the inputs will become smoother. Next simulation runs with $Q = diag\{1,1,1,1\}$ and $R = diag\{3,3\}$. The system returns stable due to the slower increment of inputs as shown in figure 13.

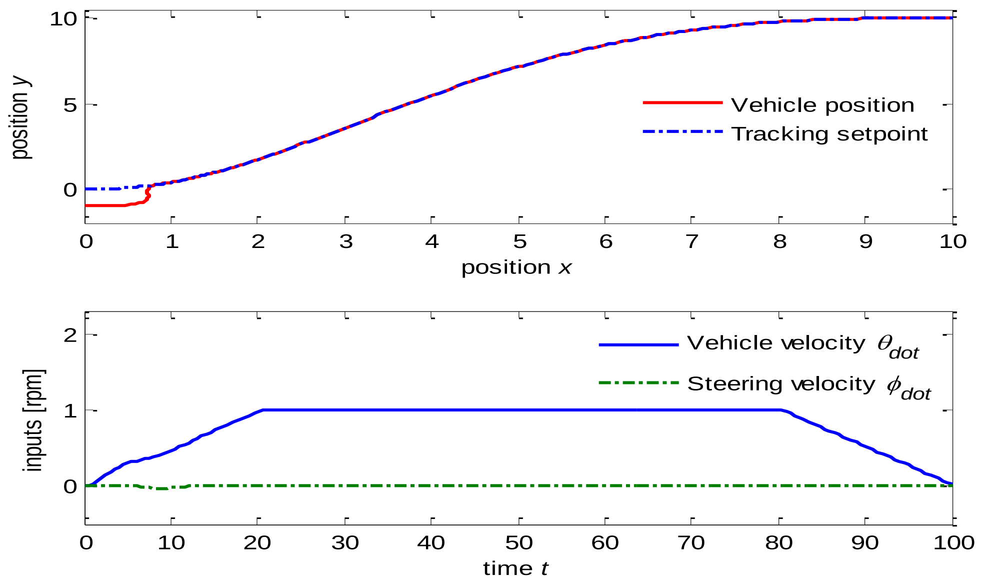


Figure 13. MPC with $Q < R$

Next, we will continue to test the MPC for tracking polynomial trajectory since the polynomial trajectories can be generated faster and smoother than the flatness trajectories.

4.3. MPC for tracking polynomial trajectory

Polynomial trajectory generation is presented in [1]. Figure 14 shows the polynomial trajectory for the vehicle from the initial position, $[x_0, y_0]=[0,0]$, to the final position, $[x_F, y_F]=[10,10]$, and the changes of the orientation angle, θ, and steering angle, φ, during the travel. Similarly, the time for completing this path is set at $T = 100\,\text{sec}$;

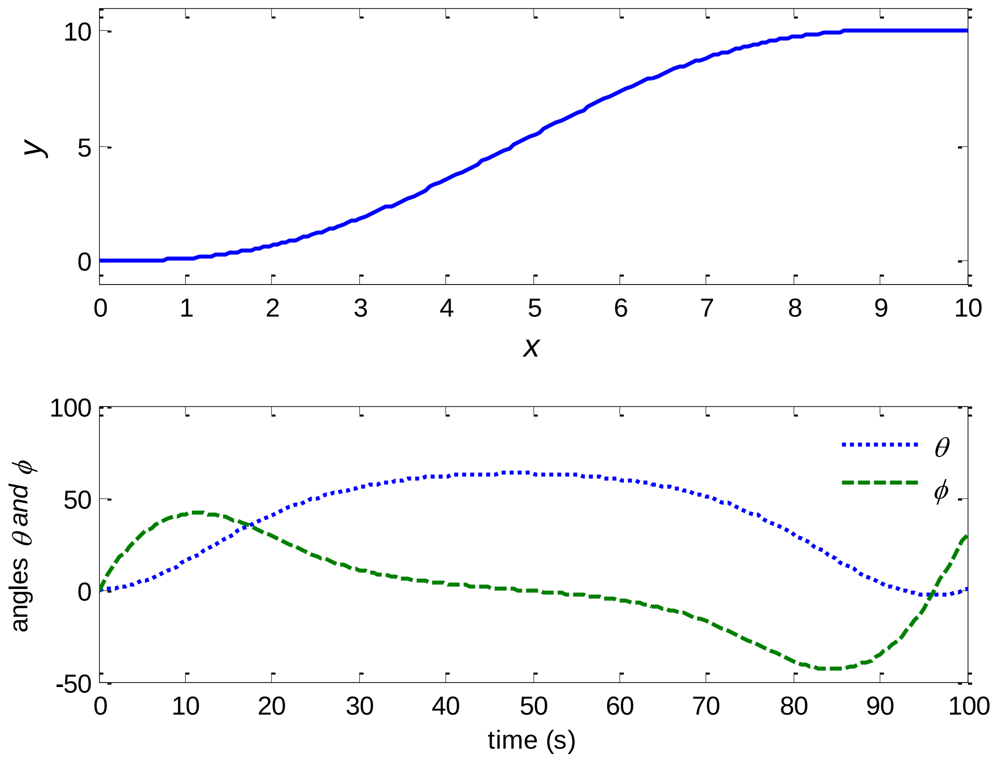


Figure 14. Polynomial trajectory reference set points

The initial positions of the vehicle are set at $X_0 = [0 \quad -0.5 \quad 0 \quad 0]'$; The predictive horizons are set at short $N_u = 4$ and $N_y = 4$; Penalty matrices are set equally at $Q = diag\{1,1,1,1\}$ and $R = diag\{1,1\}$; The reference velocity inputs are $u_1 = 1$ (u_1 is set at 0 at the initial point, during the first 1/5 traveling time length, u_1 will gradually increase and maintain at 1, during the final 1/5 traveling time length, u_1 will decrease back to 0), and $u_2 = 0$. Performance of this MPC is shown in figure 15. The vehicle gradually tracks the reference trajectory set points in 20 sec.

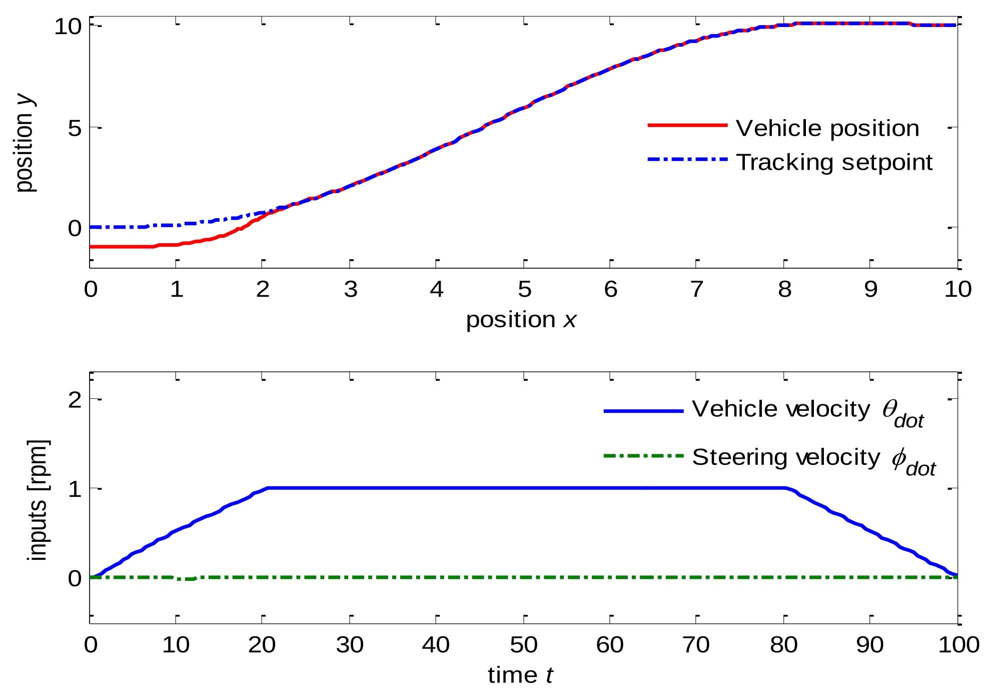


Figure 15. MPC for tracking polynomial trajectory

Next, we lengthen the horizon prediction to $N_u = 8$ and $N_y = 8$, we can see that the too long prediction horizon can degrade the MPC performance. The system becomes instable as shown in figure 16.

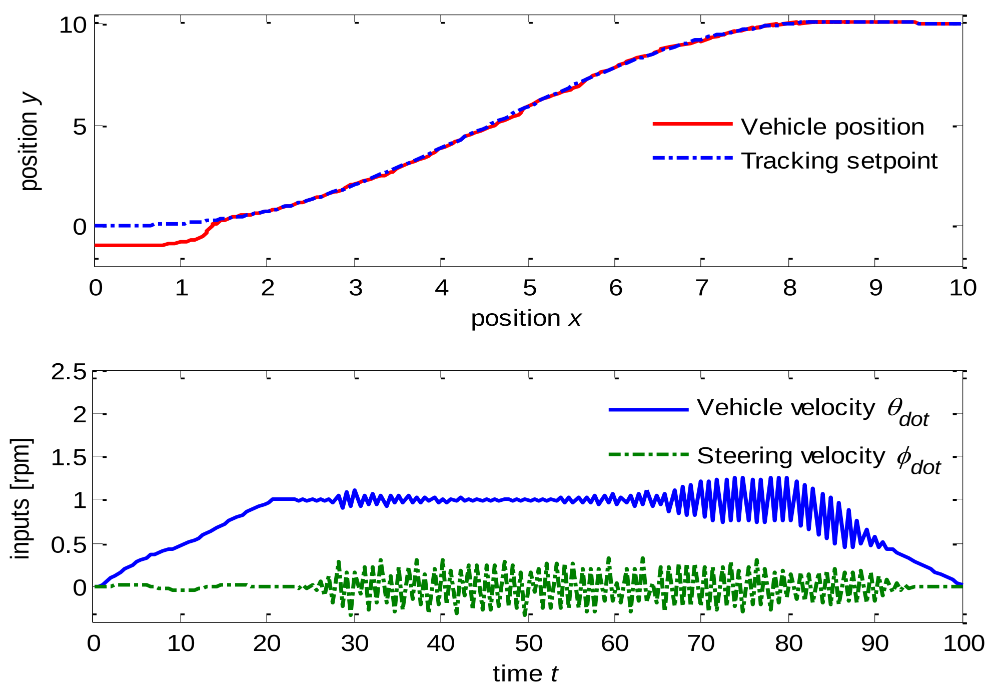


Figure 16. MPC with too long horizon

If we increase penalty matrix values, R, or set $Q < R$, the input increments will be slower and the inputs will become smoother. In the next simulation, we set the same horizon prediction lengths of $N_u = 8$ and $N_y = 8$, but increase the input penalty matrix values to $R = diag\{10,10\}$, and maintain the output penalty matrix at $Q = diag\{1,1,1,1\}$. The system returns back stable as shown in figure 17.

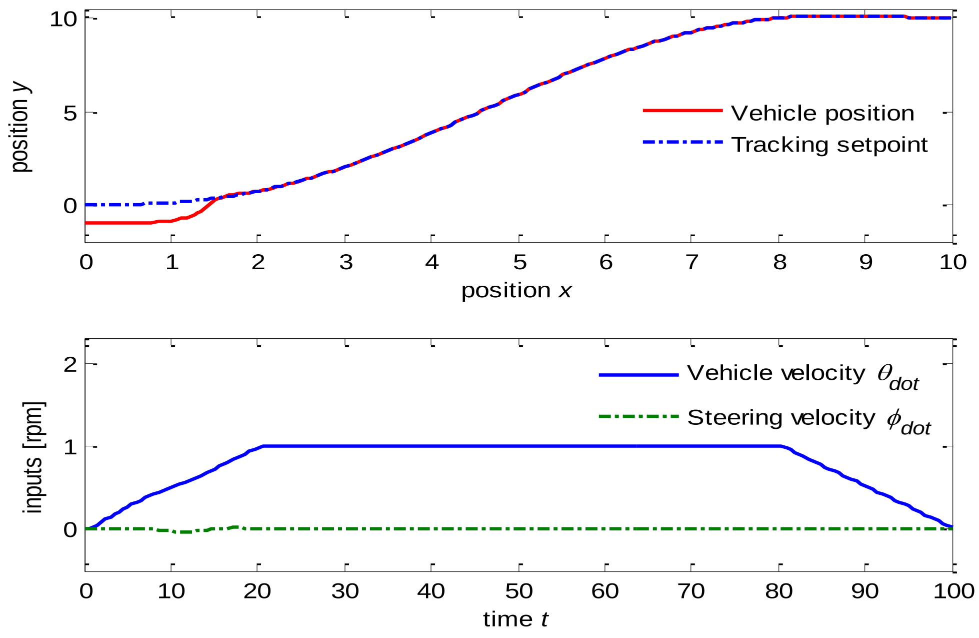


Figure 17. MPC with $R = 10Q$

Vehicle can track the given trajectory by reversed speed. In this case, we set the reference velocity inputs u_1 as minus values. Then, the vehicle now is moving backward tracking on the path. Figure 18 shows the MPC performance for this vehicle reversing on the trajectory with the velocity of $u_1 = -1$.

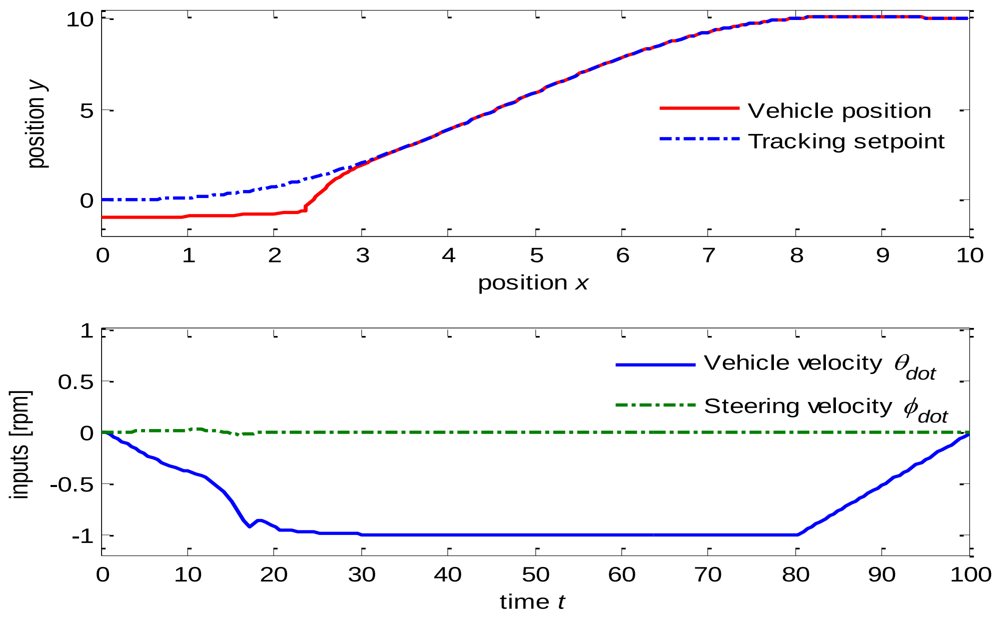


Figure 18 MPC for tracking in reverse direction

We have known that this system is very sensitive to the vehicle velocity changes. If we double the vehicle reference speed, $u_1 = 2$, the system will become instable as shown in figure 19. The too fast velocity input can lead to bad performance or the system becomes instable.

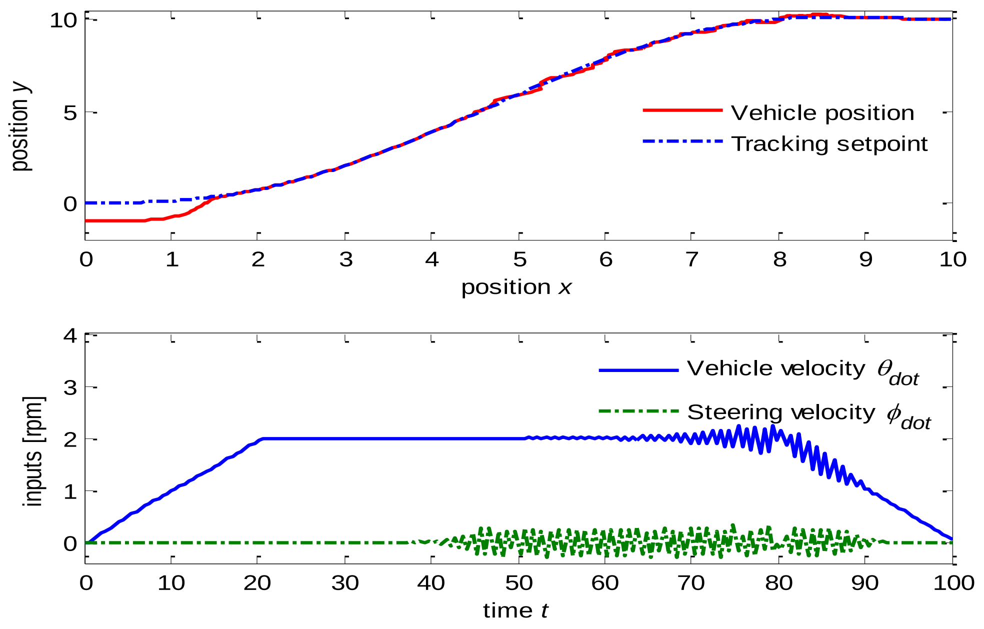


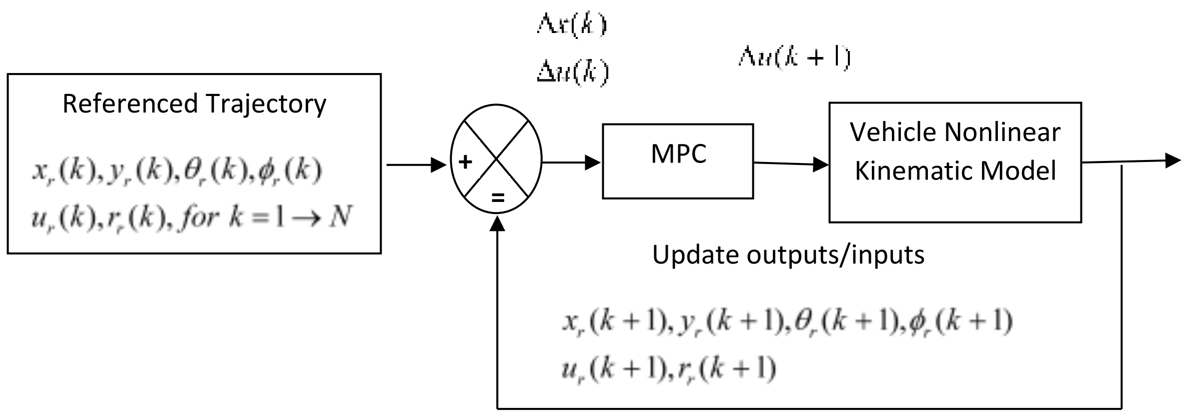
Figure 19 MPC with velocity $u_1 = 2$

In the next part, we will investigate the MPC performance using directly the nonlinear vehicle model.

5. MPC USING NONLINEAR MODEL

This part presents the MPC performance for the original nonlinear vehicle model in (2) since the linearized model can only approximate the dynamics of the true system. MPC schemes can guarantee the stability for nonlinear system by imposing the stabilized conditions on the open loop optimal regulator. These conditions take a terminal constrained region to the origin or with the terminal penalty of the softened constraints.

The diagram of the MPC control using nonlinear kinematic model is shown in figure 20.



$\Delta x(k)$
$\Delta u(k)$
$\Delta u(k+1)$
Referenced Trajectory
$x_r(k), y_r(k), \theta_r(k), \phi_r(k)$
$u_r(k), r_r(k), for\ k = 1 \rightarrow N$
+
=
MPC
Vehicle Nonlinear Kinematic Model
Update outputs/inputs
$x_r(k+1), y_r(k+1), \theta_r(k+1), \phi_r(k+1)$
$u_r(k+1), r_r(k+1)$

Figure 20. NMPC control system

The on line optimization problem for this NMPC is taken place in real time. The MPC regulator determines an optimal future input trajectory that brings the system from its current estimated state to the state and input targets via an quadratic objective function subject to constraints. It is noted that, the nonlinear system in (2) has been approximated to the linearized system at the reference points in (9) and discretised time variant system in (10). For this NMPC stability, we apply the zero terminal equality or zero terminal region at the end of the prediction horizon as per Minh V.T and Afzulpurkar N (2006) [5], *i.e.* adding the zero constraint for the terminal prediction state $x_{k+i|k} = A^i(k)x_{k|k} + \sum_{i\geq N_u}^{N_y-1} A^i(k)B(k)u_{k+N_y-1-i|k} = 0$ in the MPC objective function (11). Inputs solution for the NMPC regulator is implemented for close-loop control by solving directly on-line the ordinary differential equations (ODEs) for the vehicle nonlinear kinematic model in (2) repeatedly.

5.1. NMPC for a full circle trajectory

For generating a full circle reference trajectory using in the above model, the reference inputs $u_1 = r\omega$ and $u_2 = 0$ are used. The initial positions are selected as $[x_0 \quad y_0 \quad \theta_0 \quad \varphi_0]' = \left[0 \quad 0 \quad 0 \quad \arctan\frac{r}{l}\right]'$. For the simulation, we use $r = 0.5m$, $l = 1.5m$, and $\omega = 1rad/\sec$. The reference set points generated using ODE45 function in Matlab are shown in figure 21.

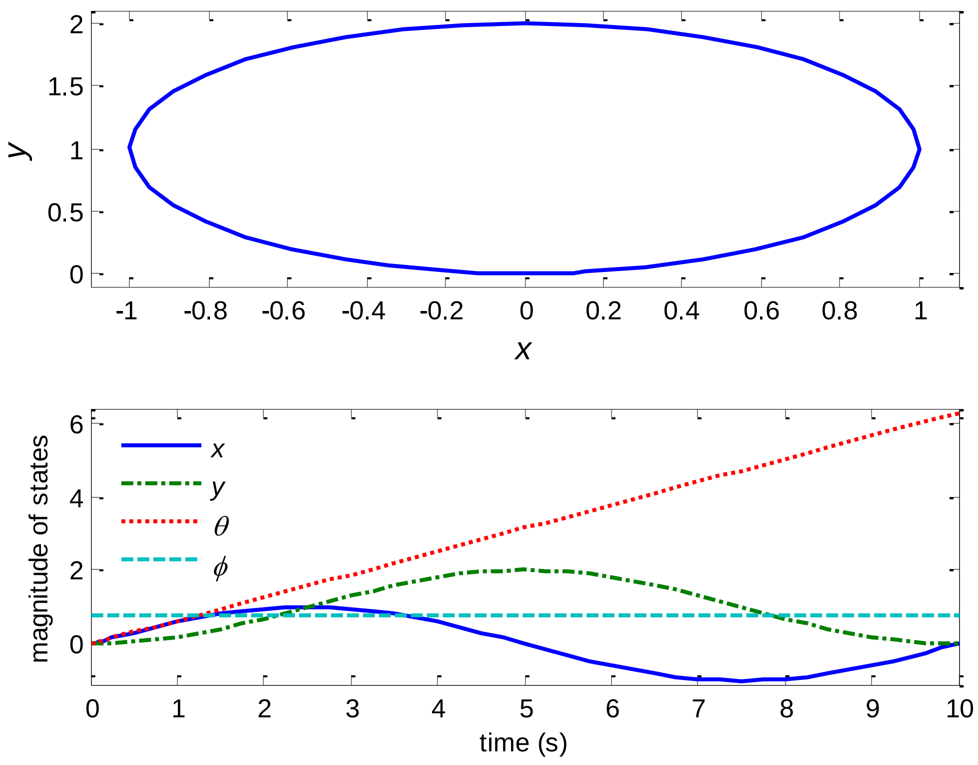


Figure 21. Circular reference set points

We now use the on-line MPC regulator in (11) to run the nonlinear vehicle kinematic model in (2) to track these circular reference set points. For this simulation, the initial positions of the vehicle are set at $X_0 = \left[-0.5 \quad -0.5 \quad 0 \quad 0\right]'$; The constraints are set at $u_{\min} = \left[-1, -1\right]'$, $u_{\max} = \left[1, 1\right]'$, $\Delta u_{\min} = \left[-0.5, -0.5\right]'$, $\Delta u_{\max} = \left[0.5, 0.5\right]'$, $y_{\min} = \left[-1, -1, -1, -1\right]'$, and $y_{\max} = \left[1, 1, 1, 1\right]'$; The predictive horizons are set at $N_u = 10$ and $N_y = 10$; Penalty matrices are set at $Q = diag\{1,1,1,1\}$ and $R = diag\{1,1\}$. Performance of the NMPC vehicle model to track the circular reference set points is shown in figure 22. The NMPC optimizer minimizes the tracking errors at each points and tracks the vehicle with very small errors left at the end of the trajectory. The inputs look good since they are physically smooth enough for controlling this vehicle.

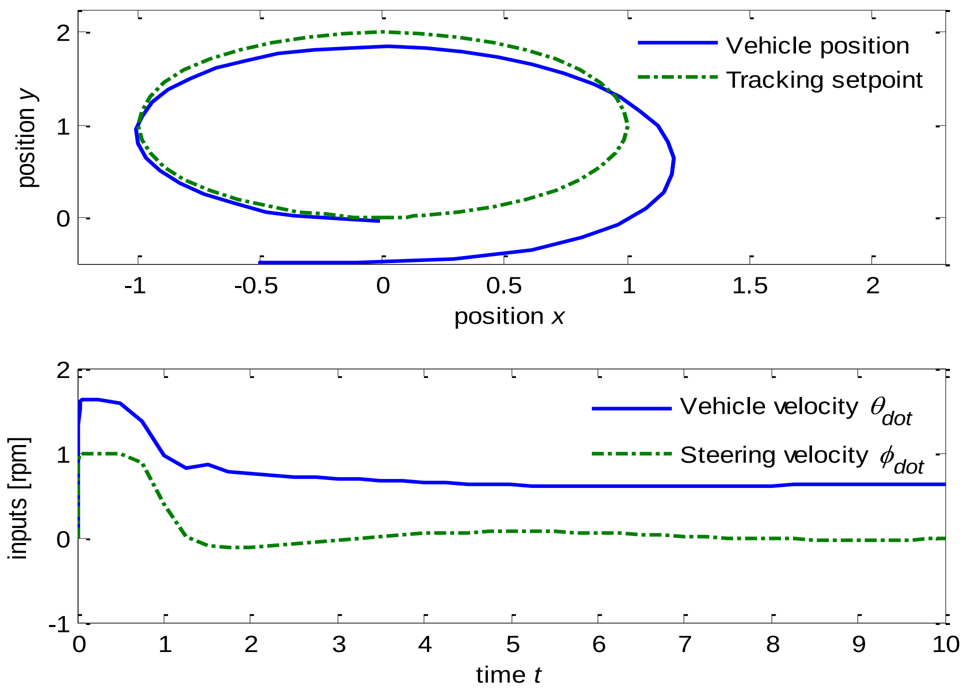


Figure 22. NMPC tracking

In NMPC, if the prediction horizon is shortened, the calculation burden will be considerably reduced but will lead to sharp and faster changes in the inputs, then, causing bad performance of the outputs. With short prediction horizons, the system may become instable. Figure 23 shows the NMPC performance with shortened horizons to $N_u = 4$ and $N_y = 4$. We can see the worse performance from the final tracking errors and the sharp inputs movement at the starting time.

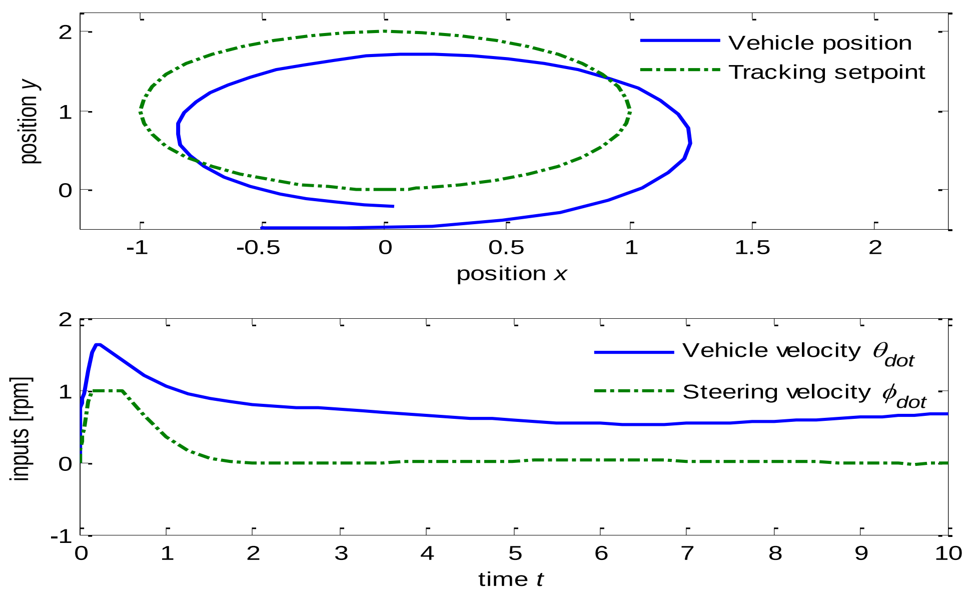


Figure 23. NMPC with shortened horizon

In NMPC, we can regulate the control performance by changing the predictive horizon length, penalty matrices, softened constraints or time scanning intervals. We can also regulate the system by changing reference set point errors $(y_{k+i|k} - r_{k+i|k})$ to offset the vehicle sideslip or to compensate the model-plant mismatches. In the previous simulations, we have set the set point errors at $r_k = y_k - y_r = [0,0,0,0]'$. To offset the vehicle sideslips or the model-plant mismatches, we can dynamically change these set point errors. For example, if we now set the set point errors at $r_k = [-0.1, 0.3, 0, 0]'$, the NMPC performance is shown in figure 24 and we can see some better tracking performances:

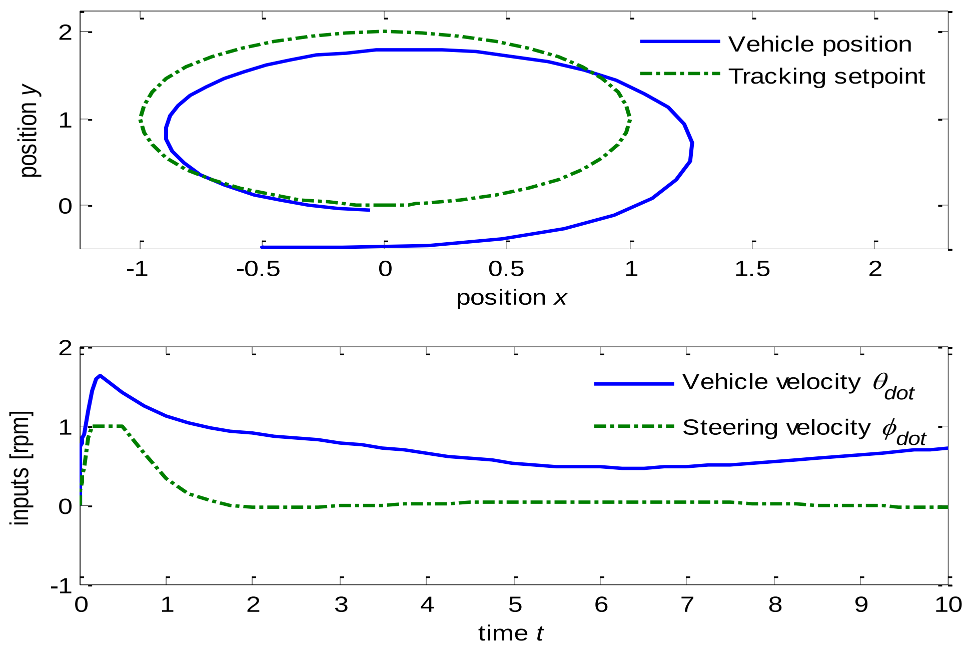


Figure 24. NMPC with new set point offsets

The above NMPC performances show that the vehicle can track a full circle with the initial positions of $X_0^{\text{Vehicle}} = \begin{bmatrix} -0.5 & -0.5 & 0 & 0 \end{bmatrix}'$ other than the reference initial positions of $X_0^{\text{Reference}} = \begin{bmatrix} 0 & 0 & 0 & \arctan\frac{r}{l} \end{bmatrix}'$. This difference can be considered as the possible errors of the measured outputs or the initial model-plant mismatches. NMPC regulator can overcome those errors and track the vehicle exactly along the given reference set points. In the next part, we will investigate the ability of the NMPC to track the vehicle on any feasible generated directly from the original vehicle kinematic differential equations in (2).

5.2. NMPC for tracking flatness trajectory

Flatness trajectory equations are presented in [1]. Figure 25 shows a flatness trajectory for the vehicle from the initial position, $[x_0, y_0] = [0,0]$, to the final position, $[x_F, y_F] = [10,10]$, and the development of the orientation angle, θ, and steering angle, φ, during the travel. The time for completing this travel is set for $T = 100\,\text{sec}$;

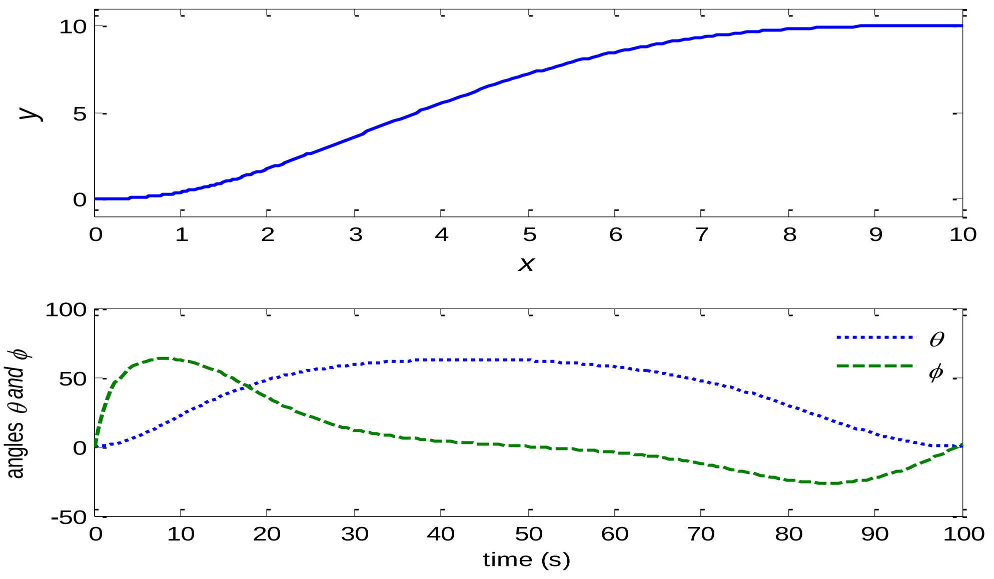


Figure 25. Flatness trajectory reference setpoints

For this NMPC tracking, the initial positions of the vehicle are set at $X_0 = \begin{bmatrix} 0 & -0.5 & 0 & 0 \end{bmatrix}'$; The predictive horizons are set at $N_u = 10$ and $N_y = 10$; Penalty matrices are set at $Q = diag\{1,1,1,1\}$ and $R = diag\{60,60\}$; The reference velocity inputs are set at, $u_1 = \frac{1}{3}$ (u_1 is set at 0 at starting point, during the first 1/5 time length, u_1 will gradually increase and maintain at 1/3 for 60 sec, during the final 1/5 time length, u_1 will decrease back to 0), and, $u_2 = 0$. Performance of this NMPC tracking is shown in figure 26. The vehicle starts with an initial velocity of $u_1 = 0$ and from the initial position of $X_0 = \begin{bmatrix} 0 & -0.5 & 0 & 0 \end{bmatrix}'$ gradually tracks to the reference tracking trajectory in 15 sec.

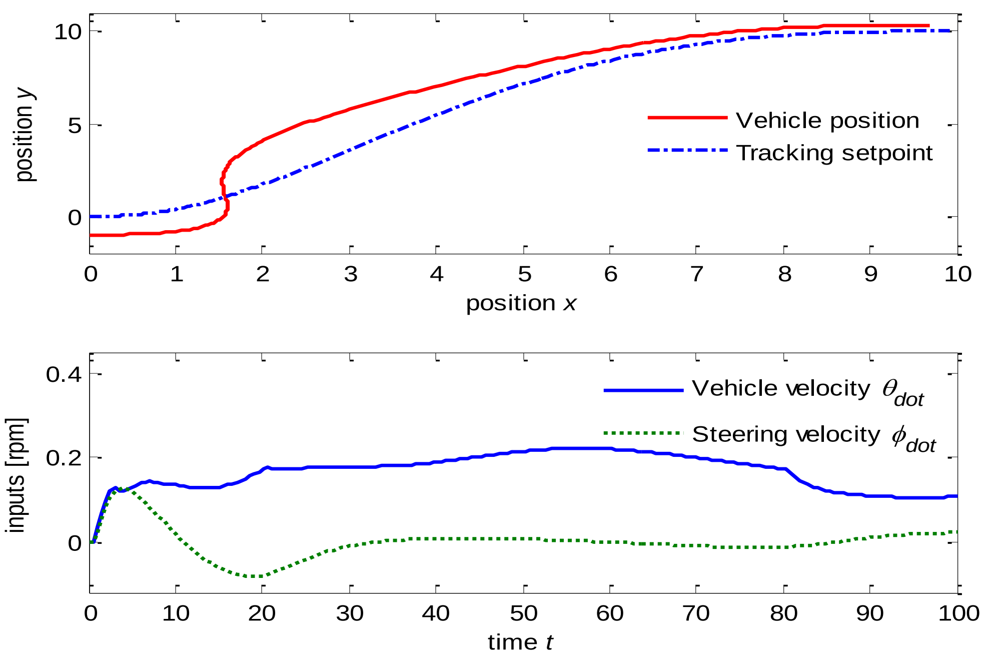


Figure 26. NMPC for tracking flatness trajectory

Next, we shorten the horizon prediction length to $N_u = 6$ and $N_y = 6$ while maintain other parameters unchanged. We can see that this shortened prediction horizon can degrade the performance because it causes the deterioration of the inputs. In this example, the system becomes infeasible for the inputs as shown in figure 27.

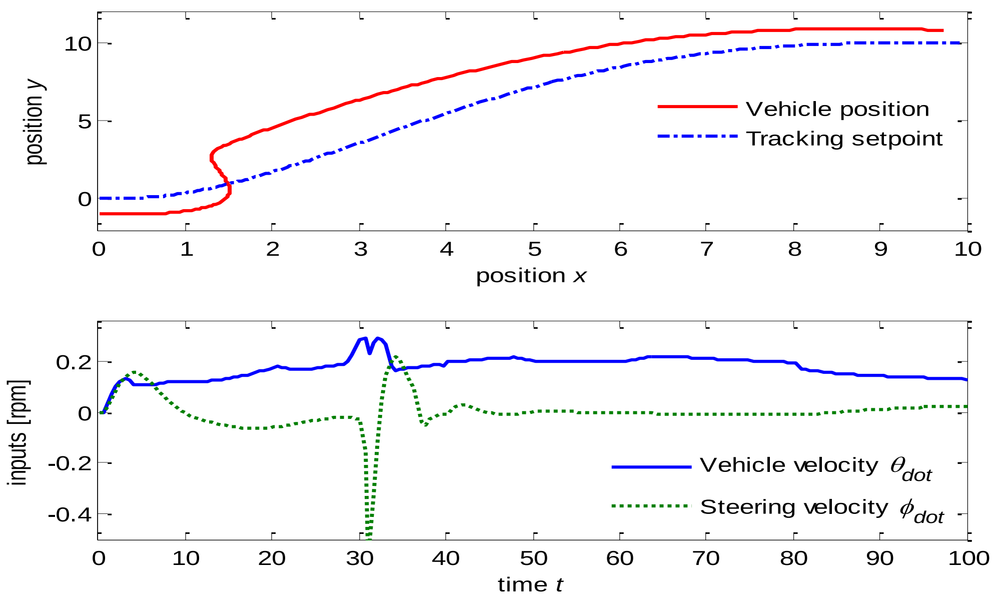


Figure 27. NMPC with too short horizon

The above instable system is due to the too heavy penalty values imposed on the input matrix ($R = diag\{60, 60\}$). This heavy penalty causes too slow and too small changes in the inputs. If we release this penalty on the inputs to $R = diag\{1,1\}$, the system returns stable as shown in figure 28.

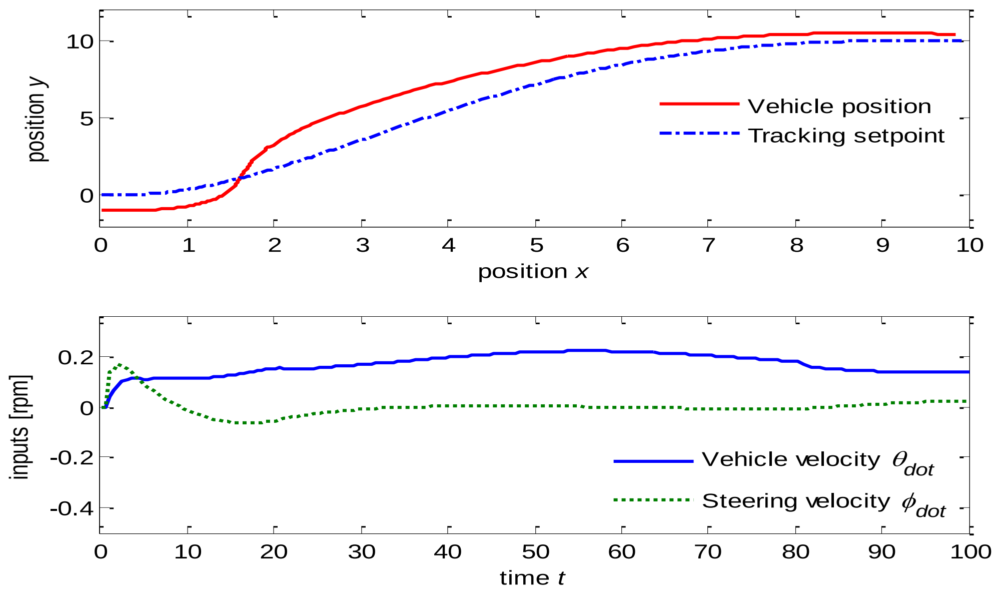


Figure 28. NMPC with $R = diag\{1,1\}$

The above NMPC can run with longer predictive horizon and achieve better performance. Figure 29 shows this NMPC performance with $N_u = 16$ and $N_y = 16$. We can see the very small tracking errors at the end of the travel.

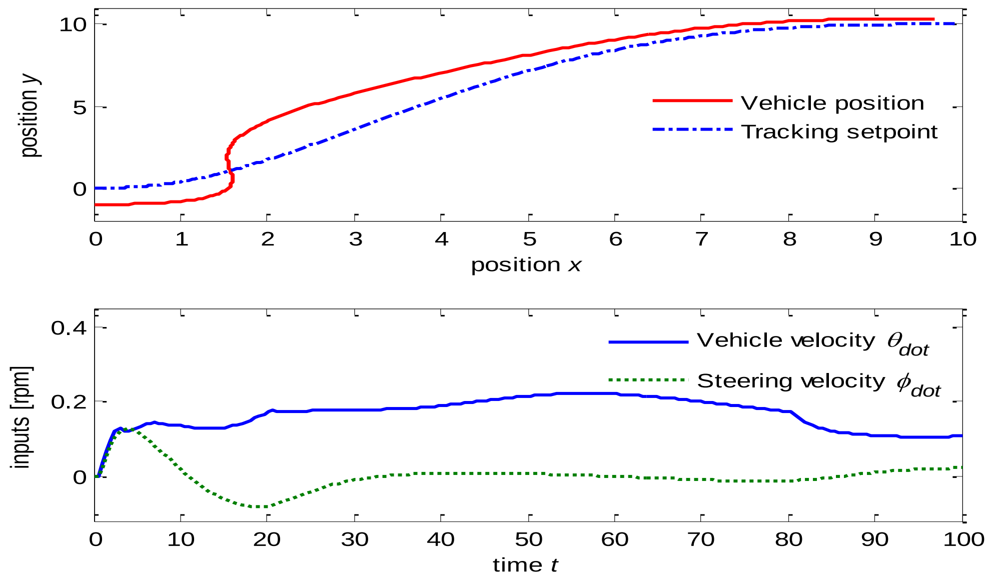


Figure 29. NMPC with $N_u = 16$ and $N_y = 16$

Next, we continue to test the NMPC for tracking the polynomial trajectory since the polynomial trajectories can be generated faster and smoother than the flatness trajectories.

5.3. NMPC for tracking polynomial trajectory

Polynomial trajectory equations are presented in [1]. Figure 30 shows a polynomial trajectory for the vehicle from the initial position, $[x_0, y_0] = [0, 0]$, to the final position,

$[x_F, y_F] = [10,10]$, and the development of its orientation angle, θ, and steering angle, φ , during the travel. Similarly, the time for completing this travel is set for $T = 100\,\text{sec}$;

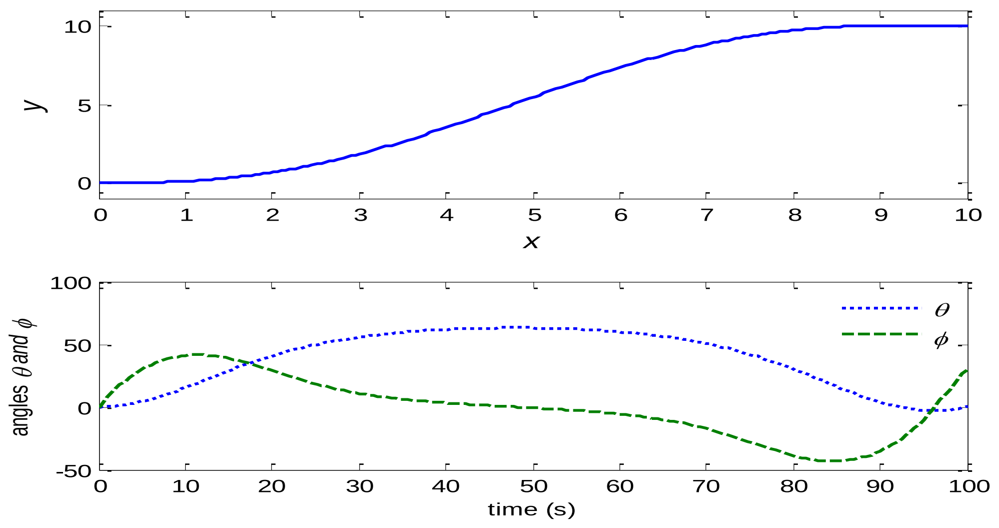


Figure 30. Polynomial trajectory reference set points

The initial positions of the vehicle are set at $X_0 = [0 \quad -0.5 \quad 0 \quad 0]'$; The predictive horizons are set at $N_u = 10$ and $N_y = 10$; Penalty matrices are set at $Q = diag\{1,1,1,1\}$ and $R = diag\{60,60\}$; The reference velocity inputs are set at $u_1 = 1$ (u_1 is set at 0 at the starting point, during the first 1/5 time length, u_1 will gradually increase and maintain at 1, during the final 1/5 time length, u_1 will decrease back to 0), and, $u_2 = 0$. Performance of this NMPC tracking is shown in figure 31. The vehicle gradually tracks exactly the reference set points in 28 sec.

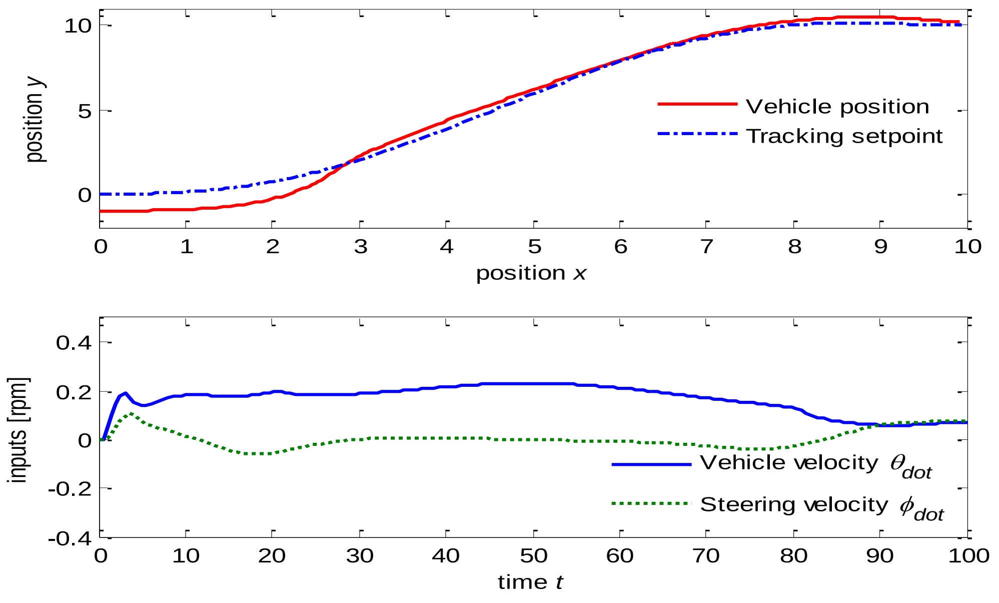


Figure 31. NMPC for tracking polynomial trajectory

Next, we shorten the horizon prediction length to $N_u = 5$ and $N_y = 5$, we can see that the too short prediction horizon can degrade the performance. The system becomes instable as shown in figure 32. The performance is worse due to the sensitiveness of the inputs and the vehicle cannot reach the output set points.

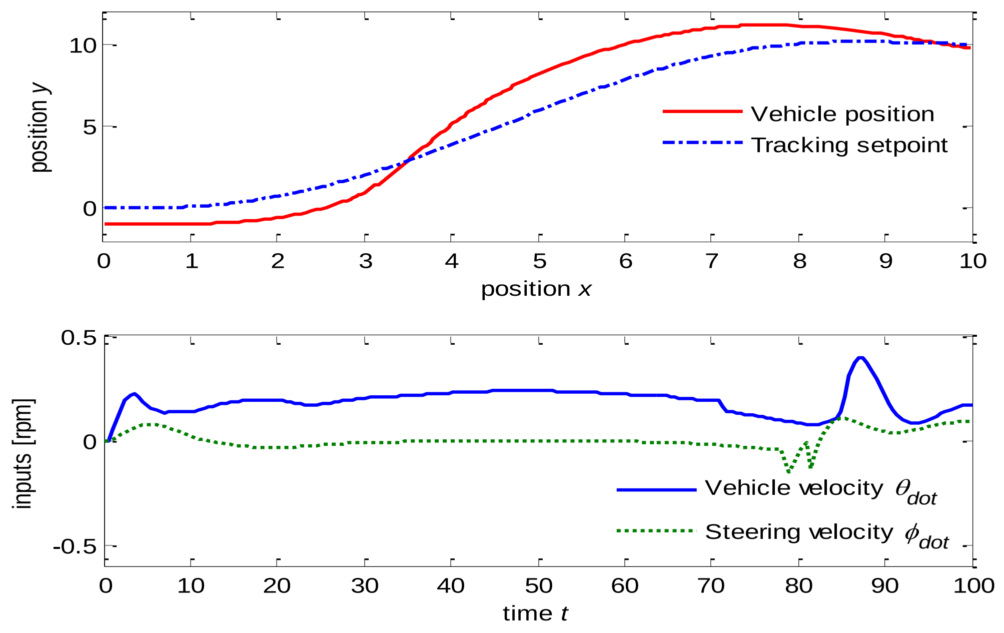


Figure 32. NMPC with shorten horizon

The above shortened horizon NMPC becomes instable at the end of the trajectory since the input increments are too slow and too small due to the too heavy penalty imposed on the inputs matrix, $R = diag\{60,60\}$. If we release this penalty and the inputs can variate more freely, the system will return stable with $R = diag\{1,1\}$ as shown in figure 33.

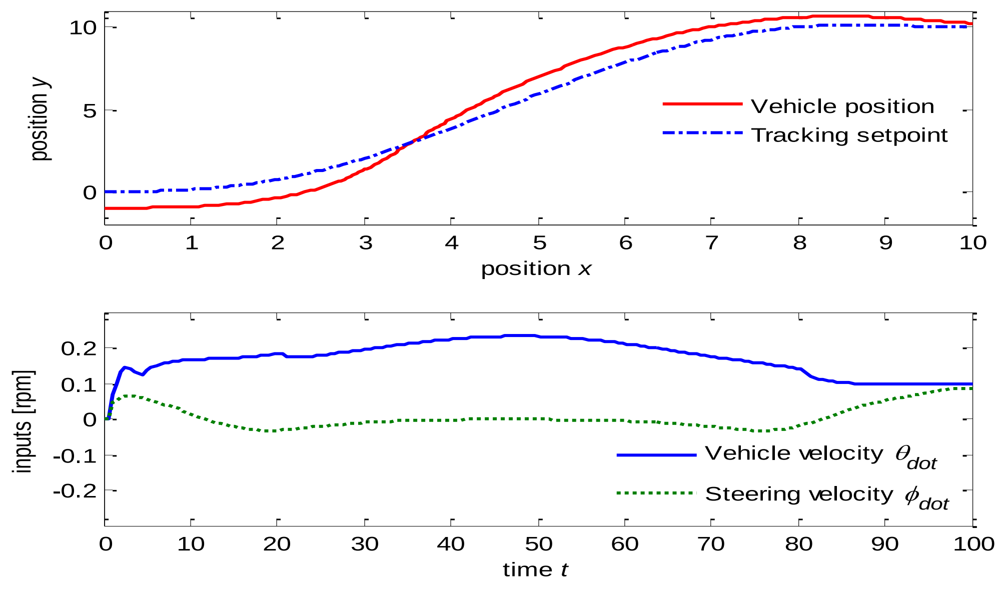


Figure 33. NMPC with $R = diag\{1,1\}$

Now we can lengthen the predictive horizon for the above NMPC to $N_u = 16$ and $N_y = 16$. The system performance becomes much better as shown in figure 34. The vehicle tracks rapidly and exactly on the reference set points in 28 sec.

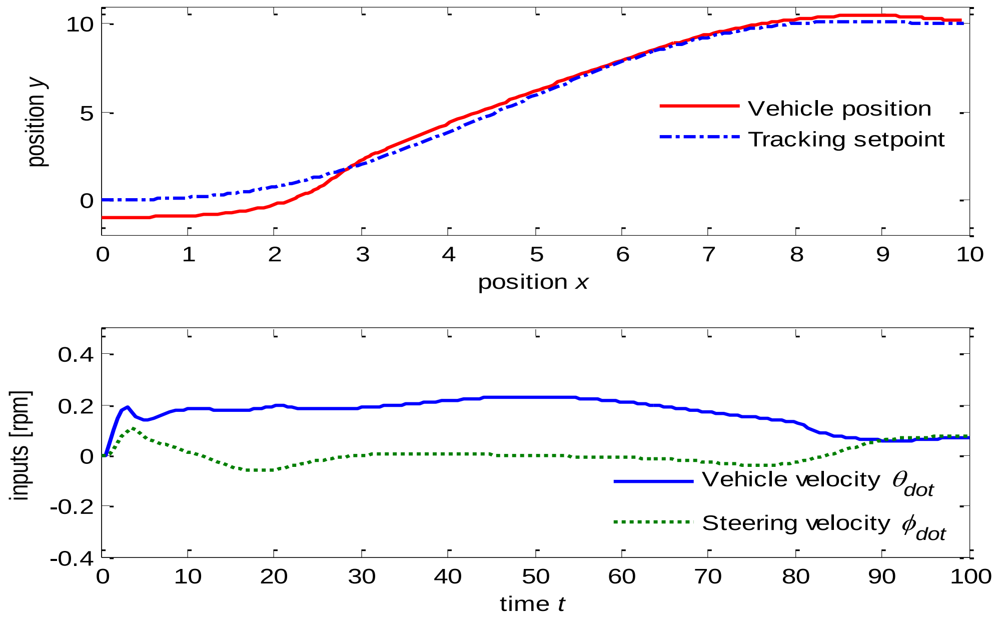


Figure 34. NMPC with lengthen horizon

In the next part, we compare the performances of linearized MPC and NMPC performances applying for the above different reference trajectories.

6. MPC PERFORMANCES COMPARISON

6.1. Compared for tracking a full circle trajectory

The two MPC schemes are compared for tracking a full circle trajectory. For this simulation, the initial positions of the vehicle are set at $X_0 = \left[-0.5 \quad -0.5 \quad 0 \quad 0\right]'$; The constraints are set at $u_{\min} = \left[-1, -1\right]'$, $u_{\max} = \left[1, 1\right]'$, $\Delta u_{\min} = \left[-0.5, -0.5\right]'$, $\Delta u_{\max} = \left[0.5, 0.5\right]'$, $y_{\min} = \left[-1, -1, -1, -1\right]'$, and $y_{\max} = \left[1, 1, 1, 1\right]'$; The predictive horizons are set at $N_u = 10$ and $N_y = 10$; Penalty matrices are set at $Q = diag\{1,1,1,1\}$ and $R = diag\{1,1\}$. Performances of the two MPC schemes are shown in figure 35.

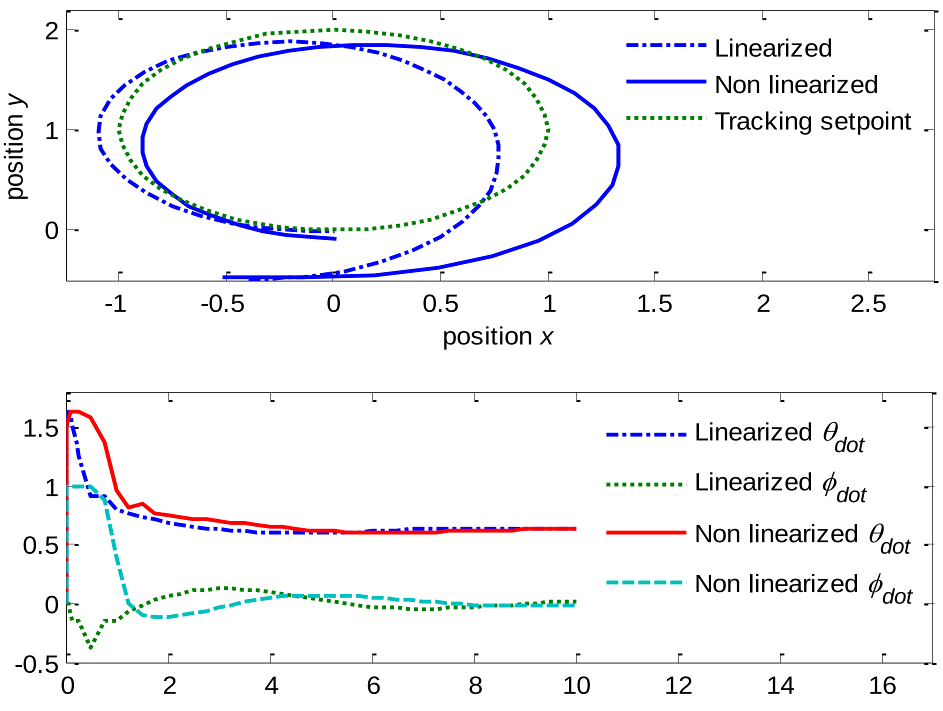


Figure 35. MPCs tracking a full circle

The linearized MPC scheme drives the vehicle with a smaller rotating radius and goes faster and closer to the reference set points. However the inputs the NMPC are smoother and really better in term of stability for the system performance. The CPU elapsed time for the two MPC schemes is almost the same. Elapsed CPU time for the linearized MPC is **0.74** sec and elapsed CPU time for the NMPC is **0.89** sec.

6.2. Compared for tracking a flatness trajectory

The two MPC schemes are compared for tracking a flatness trajectory. The initial positions of the vehicle are set at $X_0 = \begin{bmatrix} 0 & -0.5 & 0 & 0 \end{bmatrix}'$; The predictive horizons are set at short of $N_u = 6$ and $N_y = 6$; Penalty matrices are set at $Q = diag\{1,1,1,1\}$ and $R = diag\{2,2\}$; The reference velocity inputs are set at $u_1 = \frac{1}{3}$ (u_1 is set at 0 at the starting point, during the first 1/5 time length, u_1 will gradually increase and maintain at 1/3, during the final 1/5 time length, u_1 will drop back to 0), and $u_2 = 0$. Performance of these MPCs is shown in 36.

The linearized MPC scheme drives the vehicle faster to track the reference trajectory as well as its inputs come closer to the input trajectory (easier to control) since the vehicle model is not real. It is only an approximation via its linearized model. In reality, the movement of the vehicle must be online calculated through its real nonlinear derivative equations. And then, the NMPC is slower and more difficult to track the trajectory because that its plant-model mismatches are more significant. The NMPC scheme becomes more difficult to control in term of its lower stable inputs. The CPU elapsed time for these two schemes is now becoming a big challenge since the elapsed CPU time for the NMPC is **4.27** sec, almost doubles the elapsed CPU time for the linearized MPC of only **2.45** sec for the whole travel calculation.

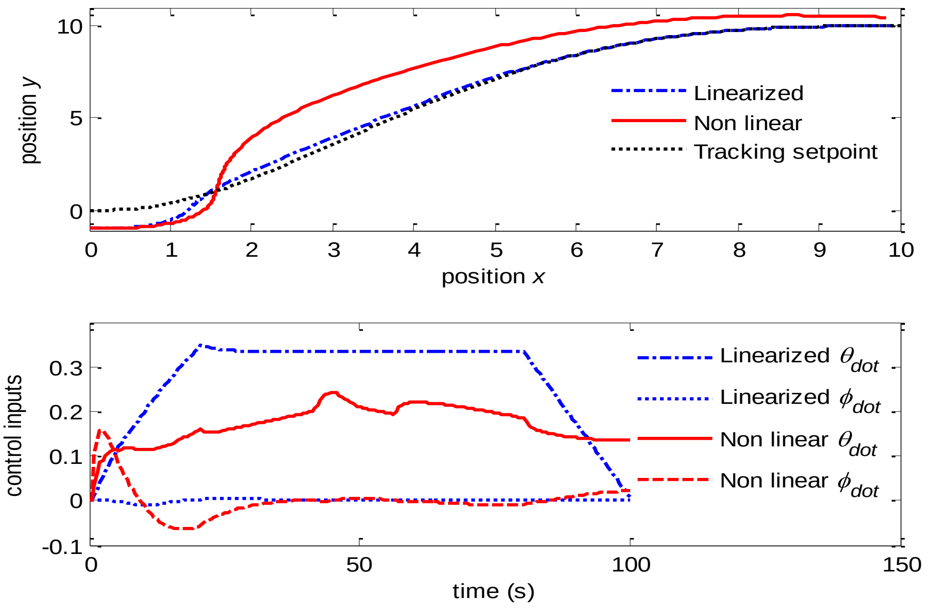


Figure 36. MPCs tracking a flatness trajectory

The NMPC scheme will become more difficult to control if we impose high values on the input penalty matrix since with too slow and small input increments, the system will become instable as shown in figure 37 if we set the input penalty matrix of $R = diag\{60,60\}$. The CPU elapsed time for the NMPC in this simulation is **3.61** sec, the elapsed CPU time for the linearized MPC is **1.61** sec for the whole travel calculation

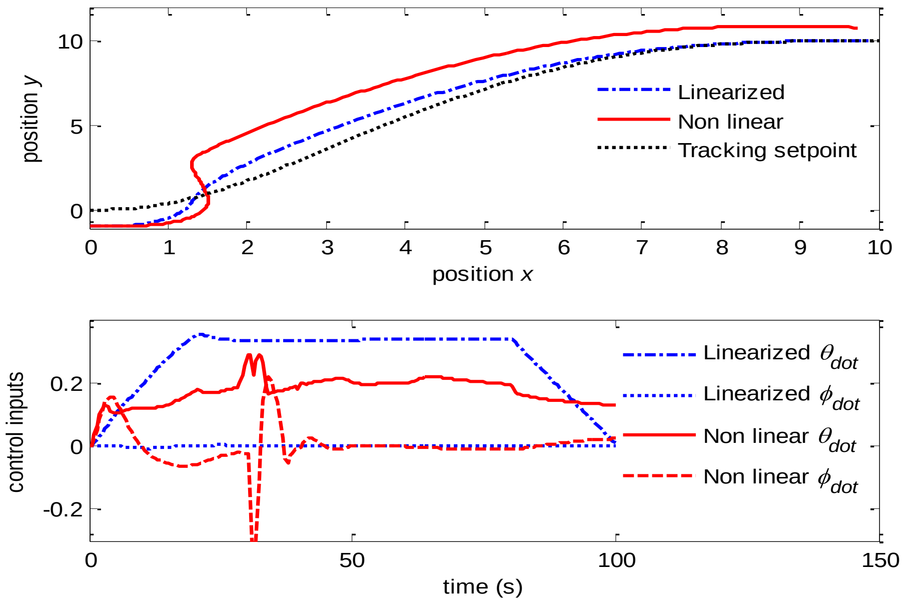


Figure 37. MPCs tracking a flatness trajectory with $R = diag\{60,60\}$

6.3. Compared for tracking a polynomial trajectory

The two MPC schemes are compared for tracking a polynomial trajectory. The initial positions of the vehicle are set at $X_0 = \begin{bmatrix} 0 & -0.5 & 0 & 0 \end{bmatrix}'$; The predictive horizons are set at short of $N_u = 6$ and $N_y = 6$; Penalty matrices are set at $Q = diag\{1,1,1,1\}$ and $R = diag\{2,2\}$; The reference velocity inputs are set at, $u_1 = \frac{1}{3}$ (u_1 is set at 0 at the initial point, during the first 1/5 time length, u_1 will gradually increase and maintain at 1/3, during the final 1/5 time length, u_1 will drop back to 0), and $u_2 = 0$. Performance of these MPCs tracking is shown in figure 38.

Similarly, the NMPC performance is slower and more difficult to track the trajectory due to the larger plant-model mismatches. The elapsed CPU time for the linearized MPC is **2.67** sec and the elapsed CPU time for the NMPC is **4.69** sec.

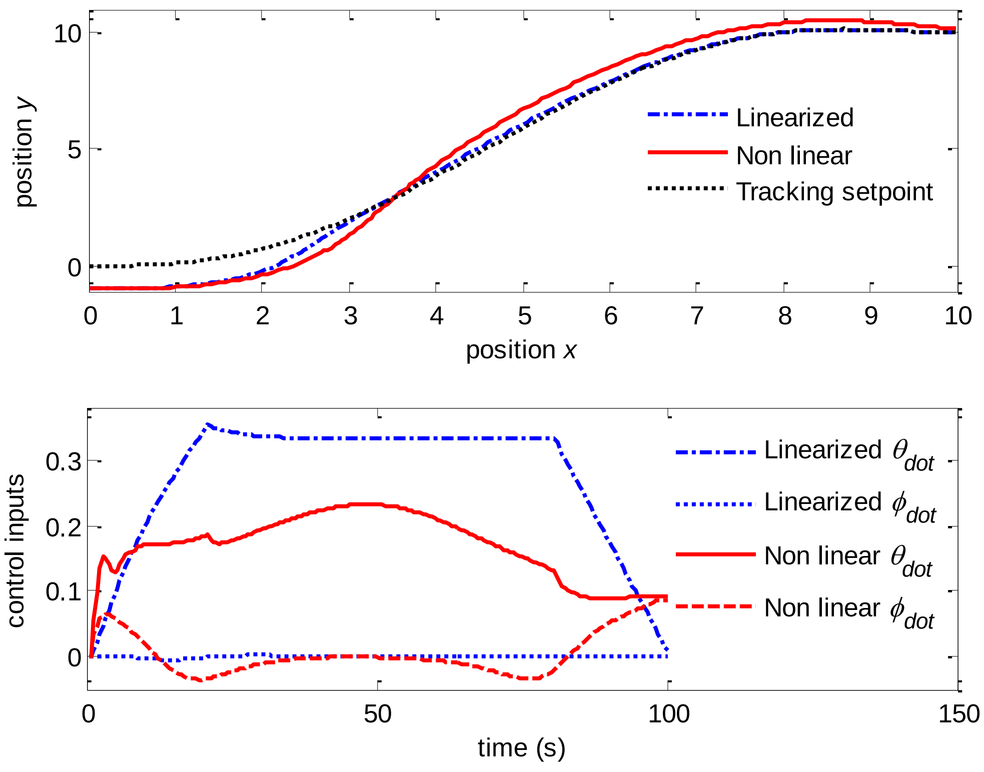


Figure 38. MPCs tracking a polynomial trajectory

The inputs of the NMPC also are more sensitive to its stability. If we shorten the control horizon to $N_u = 5$ and $N_y = 5$, the NMPC scheme becomes instable dues to the sensitiveness of its inputs as shown in figure 39. In this simulation, the elapsed CPU time for the linearized MPC is **1.84** sec and the elapsed CPU time for the NMPC is **3.23** sec.

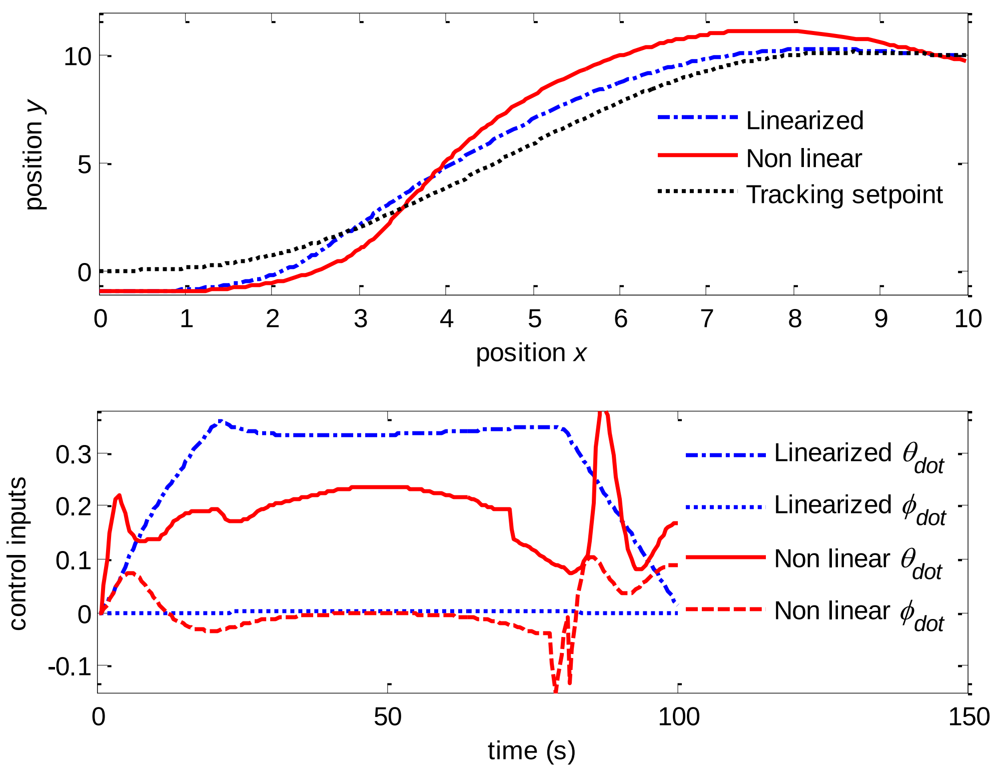


Figure 38. MPCs tracking a polynomial trajectory with shorten horizon

6.4. NMPC with Varying the Scanning Time Intervals

The amount of time between each measurement, called the sampling time interval, is one of critical factors for this type of discretized control system. If the scanning time is too short (too fast), the computer may not complete the calculation yet. Or due to significant plant model mismatches, the too small and slow input increments can also deteriorate the system instability. On contrary, if the scanning time is too long, then the vehicle dynamics can also lead to undesirable performance. Therefore, appropriate scanning time length much be chosen via real experiments and depending on each real systems. In this part, we investigate some different sampling time intervals they may affect the performances of NMPC.

For flatness trajectory:

The initial positions of the vehicle are set at $X_0 = \begin{bmatrix} 0 & -0.5 & 0 & 0 \end{bmatrix}'$; The predictive horizons are set at short of $N_u = 10$ and $N_y = 10$; Penalty matrices are set at $Q = diag\{1,1,1,1\}$ and $R = diag\{60,60\}$; The reference velocity inputs $u_1 = \frac{1}{3}$ and $u_2 = 0$. The scanning time interval now is shortened to 0.1 sec. The system becomes instable due to the sensitiveness of the inputs as shown in figure 40. The elapsed CPU time of this simulation is 25.0482 sec (considerably increasing).

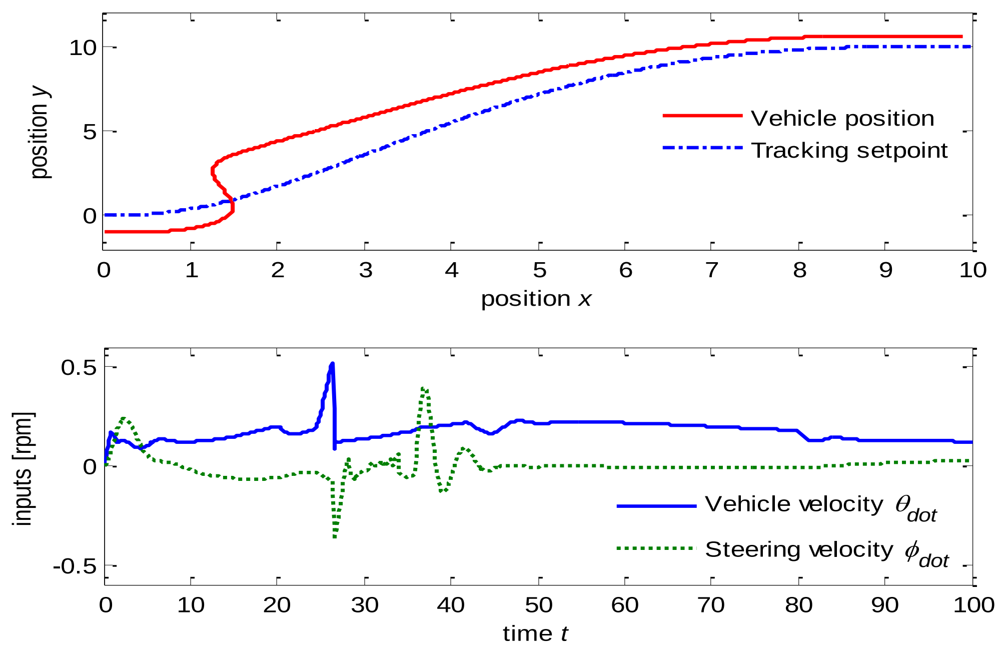


Figure 40. NMPC tracking flatness trajectory with a short scanning time (0.1 sec)

The above system returns stable if we lengthen the scanning interval to 0.5 sec as shwon in figure 41. The elapsed CPC time now for this simulation is reduced to 4.1465 sec.

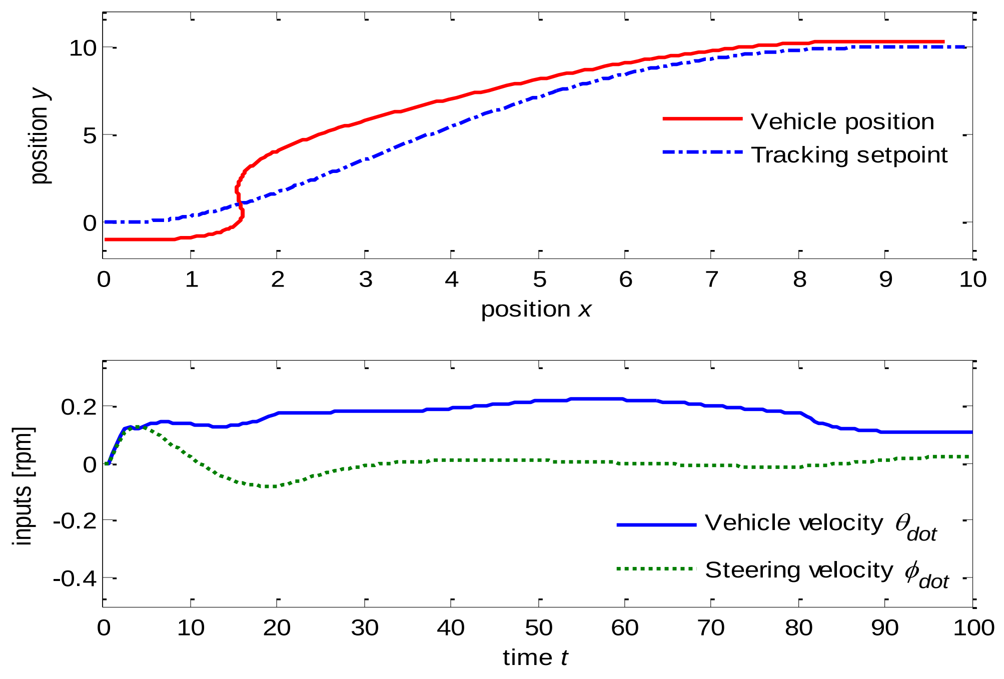


Figure 41. NMPC tracking flatness trajectory with a long scanning time (0.5 sec)

For this short time scanning interval (0.1 sec) but if we lengthen the prediction horizon to $N_u = 30$ and $N_y = 30$, the system becomes instable due to the too slow and too small input increments amid the significant model-plant mismatches as shown in figure 42. The elapsed CPU time for this simulation is 7.6514 sec.

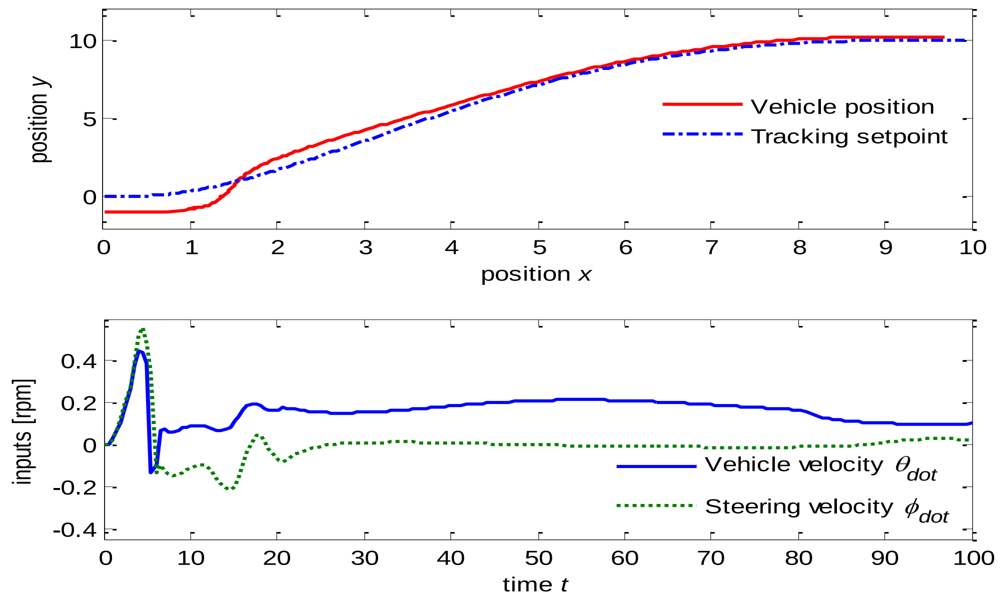


Figure 42. NMPC tracking flatness trajectory with lengthened prediction horizon

Comparison of the two NMPC performances with short scanning time (0.1 sec) and long scanning time (0.5 sec) is shown in figure 43. The longer scanning time NMPC scheme tracks faster with smaller output errors but the shorter scanning time NMPC scheme requires smoother inputs and better stability.

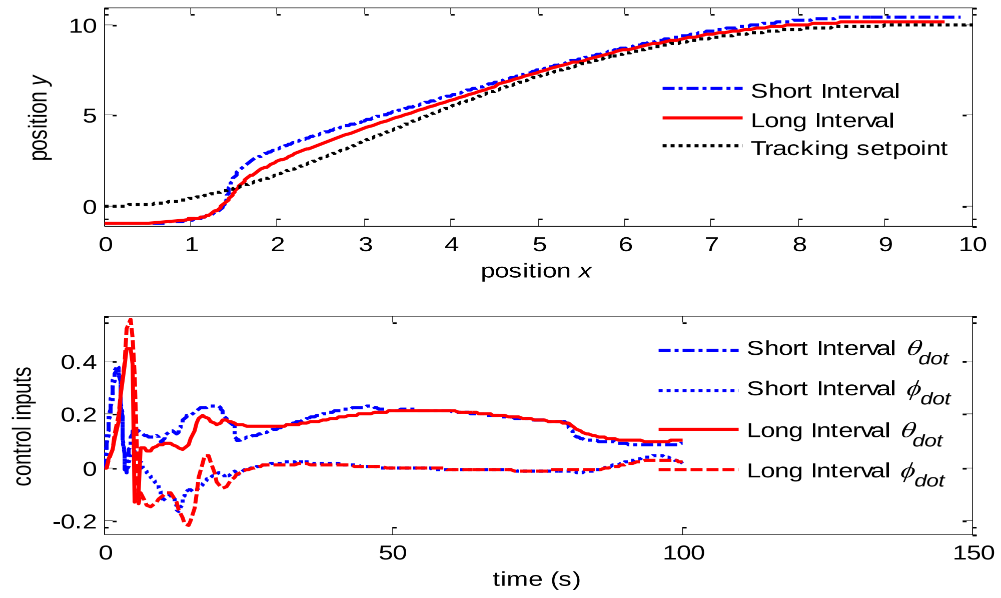


Figure 43. NMPC tracking flatness trajectory with long and short prediction horizon

For Polynomial trajectory:

The initial positions of the vehicle are set at $X_0 = \begin{bmatrix} 0 & -0.5 & 0 & 0 \end{bmatrix}'$; The predictive horizons are set at short of $N_u = 6$ and $N_y = 6$; Penalty matrices are set at $Q = diag\{1,1,1,1\}$ and $R = diag\{60,60\}$; The reference velocity inputs are set at $u_1 = \frac{1}{3}$

and $u_2 = 0$. The scanning time interval now is set at short of 0.1 sec. The system becomes instable due to the high sensitiveness of the inputs as shown in figure 44. The elapsed CPU time of this simulation is 21.7409 sec.

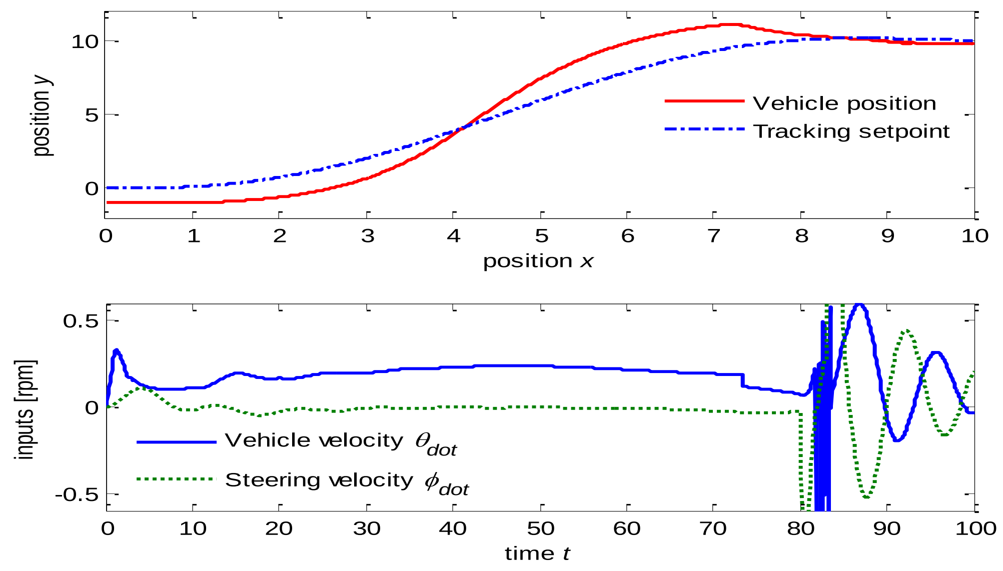


Figure 44. NMPC for polynomial trajectory with short scanning time (0.1 sec)

The above system returns stable if we lengthen the scanning interval to 0.5 sec as seen in figure 45. The elapsed CPC time for this simulation is reducing to 4.1465 sec.

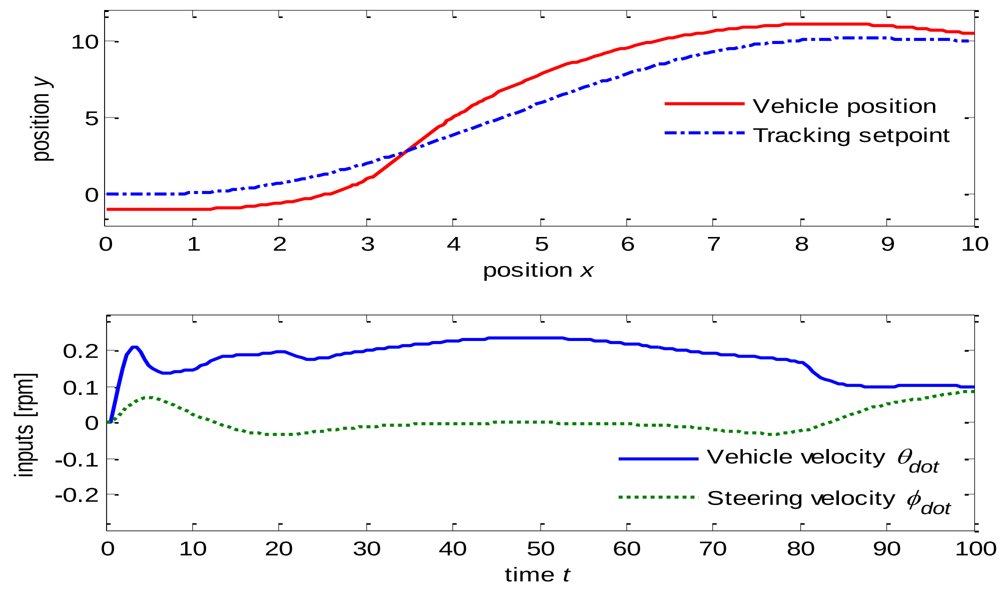


Figure 45. NMPC for polynomial trajectory with long scanning time (0.5 sec)

For this short scanning interval (0.1 sec) but if we lengthen the prediction horizon to $N_u = 15$ and $N_y = 15$, the system is still maintaining stable with the elapsed CPU time of 24.5552 sec as shown in figure 46. However, the system will become no longer stable if we lengthen the prediction horizon for $N_u > 15$ and $N_y > 15$.

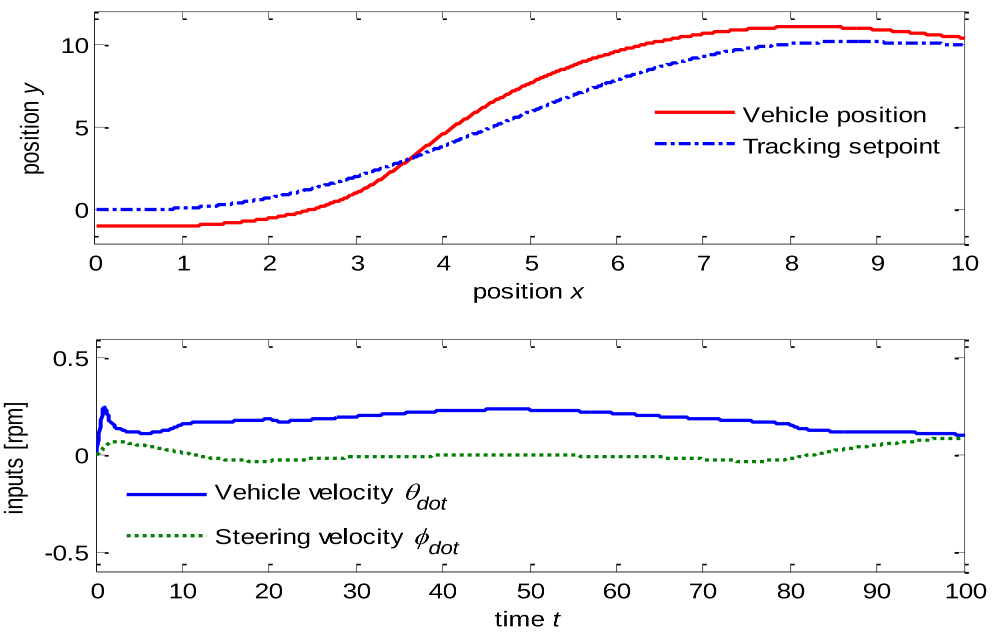


Figure 46. NMPC for polynomial trajectory with $N_u = 15$ and $N_y = 15$

Comparison of the two NMPC performances for short scanning time (0.1 sec) and long scanning time (0.5 sec) with the same other parameters is shown in figure 47. The longer scanning time NMPC scheme tracks the reference trajectory faster but the shorter scanning time NMPC scheme is required much smoother inputs and thus, more stable.

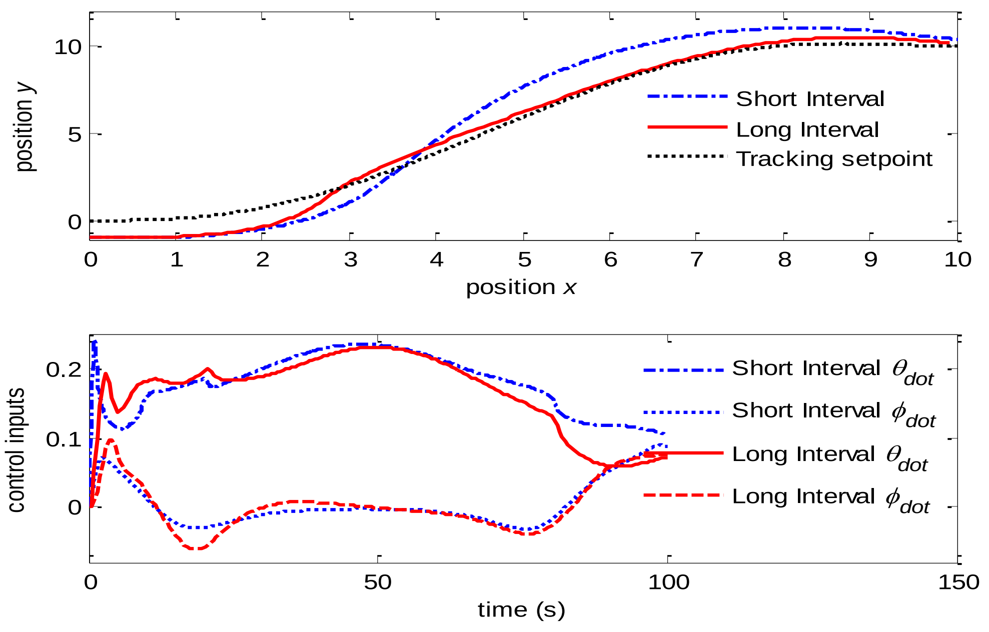


Figure 47. NMPC for polynomial trajectory with long and short scanning interval

In the above simulation, since we have set the equal heavy penalty on the inputs with $R = diag\{60, 60\}$ on both input u_1 and input u_2. As we know that the vehicle velocity, u_1, is harder to control (regulate) than the steering velocity, u_2. Now we can test the system performances with a new input matrix of $R = diag\{60,1\}$. It means that we set the penalty for any changes of u_1 is 60 times greater than the any changes of u_2. In other words, the steering velocity, u_2, is set to move much more freely than the vehicle velocity, u_1. Simulation for this example is shown in figure 48. Both MPC performances have been significantly improved and both systems are stable and required much smoother inputs.

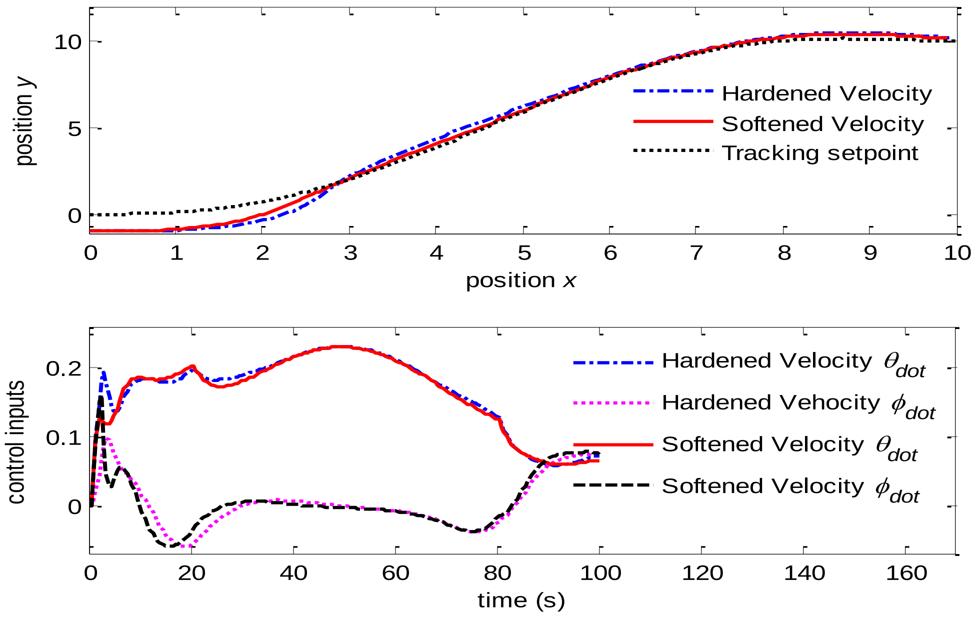


Figure 48. NMPC tracking polynomial trajectory with $R = diag\{60,1\}$

However if we set the penalty on the steering velocity with too small values, the bad consequence will appear. When the input penalties are set at $R = diag\{60, 0.1\}$, or the penalty on the steering velocity is released 600 times more freely than the vehicle velocity, the steering velocity becomes instable and leads to the bad vehicle velocity as shown in figure 49.

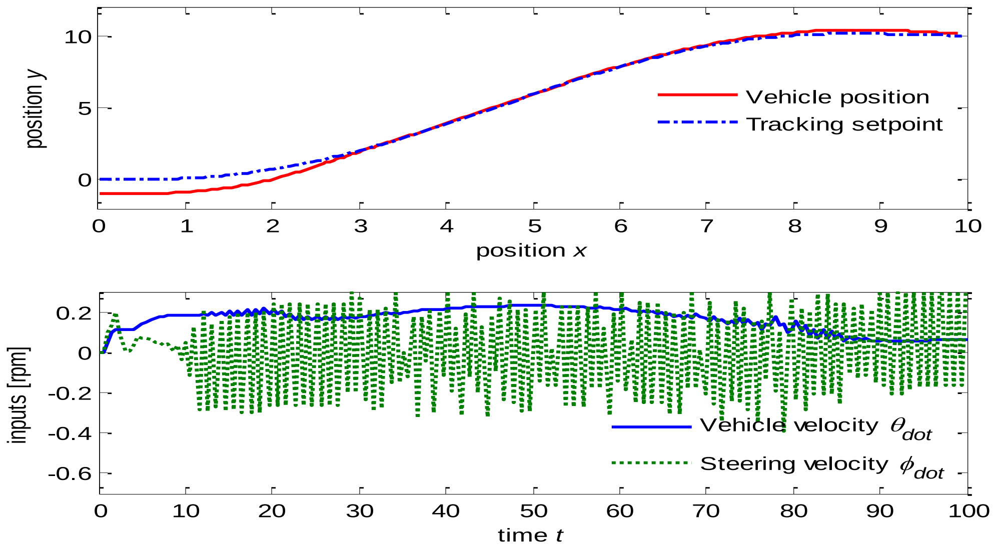


Figure 49. NMPC tracking polynomial trajectory with $R = diag\{60, 0.1\}$

This NMPC system also becomes instable if we delete the penalty on the steering velocity. In this case, the steering velocity can be freely moved and lead to the free movement of the vehicle velocity since these two inputs are very highly correlated. The vehicle tracking performance is poor since the inputs become instable as shown in figure 50.

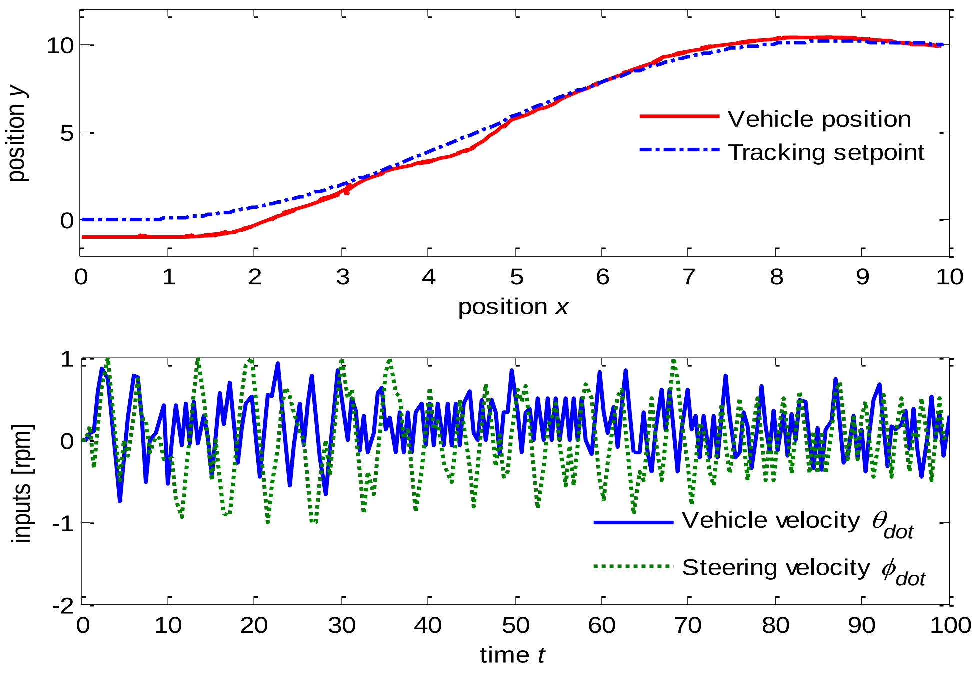


Figure 50. NMPC tracking polynomial trajectory with $R = diag\{1, 0\}$

Comparison of the NMPC schemes for tracking flatness and polynomial trajectory shows that the polynomial trajectory is more stable and better performance than the flatness trajectory since the polynomial trajectory has a smoother path and, thus, easier for the vehicle to track on with higher stability. This is also out recommendation for the use of polynomial trajectory in the application of autonomous ground vehicles.

7. CONCLUSION

MPC schemes for linearized and nonlinear have been developed and tested for controlling the vehicle tracking on different trajectories. Simulations show that MPC algorithms can control very well the tracking vehicle since it can solve on line the optimal control actions subject to constraints. The performance, the stabilization as well as the robustness of the MPC controller can be regulated by varying its parameters and modifying its objective functions to softened constraints or to constraint regions. MPC schemes are able to guarantee the system stability when the initial conditions lead to violations of some constraints.

Even though the examples show that modified MPC algorithms are successful in controlling the vehicle tracking, model of uncertainty and the model-plant mismatches that may affect the closed loop stability are still open issues. Further analysis is needed for the effectiveness of the modified MPC schemes to softened constraints and to output regions. Real experiments and further validations for this proposed controller are also needed in the next step of the project.

CONFLICT OF INTERESTS

The authors would like to confirm that there is no conflict of interests associated with this publication and there is no financial fund for this work that can affect the research outcomes.

REFERENCES

[1] Minh V.T. (2013) Trajectory Generation for Autonomous Vehicles". In: Březina T., Jabłoński R. (eds) *Mechatronics*. Springer, Cham.

[2] Minh V.T. and Hashim F.B. "Tracking setpoint robust model predictive control for input saturated and softened state constraints“ *International Journal of Control, Automation and Systmes*, 2011; 9(5); 958–965. .

[3] Minh V.T. Hashim F.B. and Awang M. "Development of a Real-time Clutch Transition Strategy for a Parallel Hybridelectric Vehicle“. *Proc IMechE, Part I: J Systems and Control Engineering*, 2012; 226(2); 188–203.

[4] Minh V.T., (2012) *Advanced Vehicle Dynamics*, 1st edition, Malaya Press, Pantai Valley, 50603. Kuala Lumpur, Malaysia, p. 127-144.

[5] Minh V.T. and Afzulpurkar N. "Robust Model Predictive Control for Input Saturated and Softened State Constraints", *Asian Journal of Control*, 2005; 7(3) 323-329.

[6] Minh V.T. and Afzulpurkar N. "A Comparative Study on Computational Schemes for Nonlinear Model Predictive Control", *Asian Journal of Control*, 2006; 8(4); 324-331.

[7] Falcone P., Borrelli F., Tseng H., Asgari J. and Hrovat D. "A Hierarchical Model Predictive Control Framework for Autonomous Ground Vehicles", *American Control Conference* (ACC), Seattle 2008, pp. 3719-3724.

[8] Lei L., Zhurong J., tingting C. and Xinchung J. "Optimal Model Predictive Control for Path Tracking of Autonomous Vehicle", *3rd International Conference on Measuring Technology and Mechatronics Automation* (ICMTMA), Shanghai 2011, p. 791-794.

[9] Bahadorian M., Savkovic B., Eston R. and Hesketh T. "Toward a Robust Model Predictive Controller Applied to Mobile Vehicle Trajectory Tracking Control", *Proceedings of the 18th IFAC World Congress*, Milano 2011, p. 552-557.

[10] Wang J., Lu Z., Chen W., and Wu X. "An Adaptive Trajectory Tracking Control of Wheeled Mobile Robots", *Industrial Electronics and Applictions* (*ICIEA*), Beijing 2011, p. 1156-1160.

[11] Shim T., Adireddy G. and Yuan H, "Autonomous Vehicle Collision Avoidance System using Path Planning and Model Predictive Control Base Active Front Steering and Wheel Torque Control", *Part D: Journal of Automobile Engineering*, 2012; 226(6); 767-778.

[12] Peni T. and Bokor J. "Robust Model Predictive Control for Controlling Fast Vehicle Dynamics" *14th Mediterranean Conference on Control and Automation*,

Budapest 2006, p. 1-5.

[13] Laumond J. (1998) Robot Motion Planning and Control, chapter 4: Feedback Control of a Nonholonomic Car-like Robot, by De Luca A, Oriolo G, and Samson C, Springer-Verlag.

[14] Ovchinnikov I., Kovalenko P. "Predictive Control Model to Simulate Humanoid Gait", *International Journal of Innovative Technology and Interdisciplinary Sciences*, 2018; 1(1); 9-17.

[15] Tamre M., Hudjakov R., Shvarts D., Polder A., Hiiemaa M., Juurma M. "Implementation of Integrated Wireless Network and MATLAB System to Control Autonomous Mobile Robot", *International Journal of Innovative Technology and Interdisciplinary Sciences*, 2018; 1(1); 18-25.

[16] Abouelkheir M. "Development of Dual Clutch Transmission model for Hybrid Vehicle", *International Journal of Innovative Technology and Interdisciplinary Sciences*, 2018; Vol 1(1); 26-33.

[17] Moezzi R., Minh V.T., Tamre M., "Fuzzy Logic Control for a Ball and Beam System", *International Journal of Innovative Technology and Interdisciplinary Sciences*, 2018; 1(1); 39-48.

[18] Pumwa J. "Time Variant Predictive Control of Autonomous Vehicles", *International Journal of Innovative Technology and Interdisciplinary Sciences*, 2019; 2(1); 62-77.

[19] Parman S. "Fuzzy Logic Control of Clutch for Hybrid Vehicle", *International Journal of Innovative Technology and Interdisciplinary Sciences*, 2019; 2(1); 78-86.

[20] Wan Muhamad W.M. "Vehicle Steering Dynamic Calculation and Simulation", *International Journal of Innovative Technology and Interdisciplinary Sciences*, 2019; 2(1); 87-97.

[21] Coskunturk Y. "Design of a Robust Controller Using Sliding Mode for Two Rotor Aero-Dynamic System", *International Journal of Innovative Technology and Interdisciplinary Sciences*, 2019; 2(2); 119-132.

[22] Israel O. "Study on Modeling and Simulation of HEV for Optimal Fuel Economy", *International Journal of Innovative Technology and Interdisciplinary Sciences*, 2019; 2(2); 133-146.

[23] Katusin N. "Glove for Augmented and Virtual Reality" *International Journal of Innovative Technology and Interdisciplinary Sciences*, 2019; 2(2); 147-159.

[24] Afzulpurkar N. "New Approach for a Fault Detect Model-Based Controller", *International Journal of Innovative Technology and Interdisciplinary Sciences*, 2019; 2(2); 160-172.

[25] Suebsomran A. "Stabilizability Analysis of Multiple Model Control with Probabilistic", *International Journal of Innovative Technology and Interdisciplinary Sciences*, 2019; 2(2); 173-180.

[26] Poulaiin T. "Path Generation and Control of Autonomous Robot", *International Journal of Innovative Technology and Interdisciplinary Sciences*, 2019; 2(3); 200-211.

[27] Yasin H. "Modelling and Control of Hybrid Vehicle", *International Journal of*

Innovative Technology and Interdisciplinary Sciences, 2019; 2(3); 212-222.

[28] Minh V.T., Tamre M., Musalimov V., Kovalenko P., Rubinshtein I., Ovchinnikov I., Moezzi R., "Simulation of Human Gait Movements", *International Journal of Innovative Technology and Interdisciplinary Sciences*, 2020; 3(1); 326-345.

[29] Vu Trieu Minh , M. Tamre , V. Musalimov , P. Kovalenko , I. Rubinshtein , I. Ovchinnikov , D. Krcmarik , R.Moezzi, J. Hlava, "Model Predictive Control for Modeling Human Gait Motions Assisted by Vicon Technology", *Journal Européen des Systèmes Automatisés* (*JESA*); 2020, 53(5); 589-600.

[30] Minh, V.T., Tamre, M., Safonov, A., ...Kovalenko, P., Monakhov, I., "Design and implementation of a mechatronic elbow orthosis", *Mechatronic Systems and Control*; 2020; 48(4); 231-238.

[31] Minh, V.T., Katushin, N., Pumwa, J., "Motion tracking glove for augmented reality and virtual reality", *Paladyn, Journal of Behavioral Robotics*; 10(1); 160-166.

[32] VT Minh' "Conditions for stabilizability of linear switched systems"; *In proceedings:* AIP Conference Proceedings 1337 (1), 108-112.

[33] Vu Trieu Minh, Wan Mansor Wan Muhamad, "Model Predictive Control of a Condensate Distillation Column", International Journal of Systems Control; 2010; Vol. 1; Issue 1; pp 4-12.

[34] Vu Trieu Minh, John Pumwa, "Simulation and control of hybrid electric vehicles", *International Journal of Control, Automation and Systems*, 2012; 10 (2); 308-316.