Development of a Genetic-Algorithm-Based Nonlinear Model Predictive Control Scheme on Velocity and Steering of Autonomous Vehicles

Arjun Srinivasan Ambalam - 116911528

Rajeshwar N S - 116921237

University of Maryland, College Park

ENPM667 -Control of Robotic Systems-Fall 2019

Dr. Waseem A. Malik

Dr. Olusesan Iwarere

Abstract

The selected paper proposes a non-linear MPC controller for predicting Velocity and Steering angle using Genetic algorithm as the Optimization algorithm. With a given trajectory representing the driver intent, the controller has to autonomously track the desired trajectory by controlling front steering angle and velocity of the car. The paper also discusses the cost function, which takes into account of various terms like safety constraints which offers safe and smooth travel for the passenger avoiding potential outcomes like motion sickness or frequent braking. Moreover, the Results suggest that a nonlinear MPC control could offer a better solution to Autonomous vehicles over the conventional PID control mechanisms. Simulation results show the benefits of the systematic control methodology used. In particular we show how very effective steering maneuvers are obtained as a result of the MPC feedback policy. Moreover, we highlight the tradeoff between the vehicle speed and the required preview on the desired path in order to stabilize the vehicle.

Table of Contents

List of Figures

1.Introduction

A number of recent technological advancements have been made in the Self-driving industry in the field of Control theory, Machine Learning and Computer Vision. Control Schemes like PID, Adaptive PID, fuzzy logic control. However, MPC has shown better results when it comes to handling nonlinearities, uncertainties and various constraints. It uses the model of plant to predict future plant output behavior. The optimization algorithm (genetic algorithm in this case) tracks the predicted output with desired reference. But, Limitation of MPC is Computational complexity in solving non-convex optimization problem which may require Powerful processors/GPUs with larger memory. One method to compensate for this is to linearize the system by choosing multiple operating points and thus obtaining multiple linear regions throughout the nonlinear function. However, this approach is only limited to SISO i.e., either steering angle or the velocity is considered as the input whereas in real scenario both the inputs are being controlled. Other Approaches of simplifying MPC computation are Model Order Reduction ie. removing state vectors which does not contribute, making prediction horizon shorter, shorter control horizon, taking reduced number of constraints into account, explicit MPC. The most popular method is Explicit MPC where optimizations are done for certain range of values of x. The linear functions are evaluated region wise where current state lies. GA maintains a population of chromosomes—a set of potential solutions for the problem. The idea is that evolution will find an optimal solution for the problem after a number of successive generations—similar to natural selection.

2. Background

2.1 Model Predictive Control

MPC is a popular advanced control process which were used earlier in Industrial plants where a lot of constraints must be met. For Most cases, they require a dynamic model (kinematic vehicle model in our case). The main advantage of MPC over other controllers like PID, LQR is that they optimize the current input while also taking future inputs into account by optimizing a timeslot of the finite time horizon. The prediction horizon keeps shifting forward which is why MPC is a type of Receding control model. It uses the current plant measurements, dynamic state of process, process target variables to calculate changes in dependent variable. At time step k, current state is sampled and cost minimizing strategy is computed for a short time in the future and an optimal control sequence is computed by solving an open-loop, constrained problem over the prediction horizon H_p. Only the first step of control input is implemented, sampled again and repeated from new current state resulting new control and predicted path. In principle, MPC is a three-step algorithm that uses: a. Dynamic model of system b. Past control moves c. Optimization cost function over receding prediction horizon.

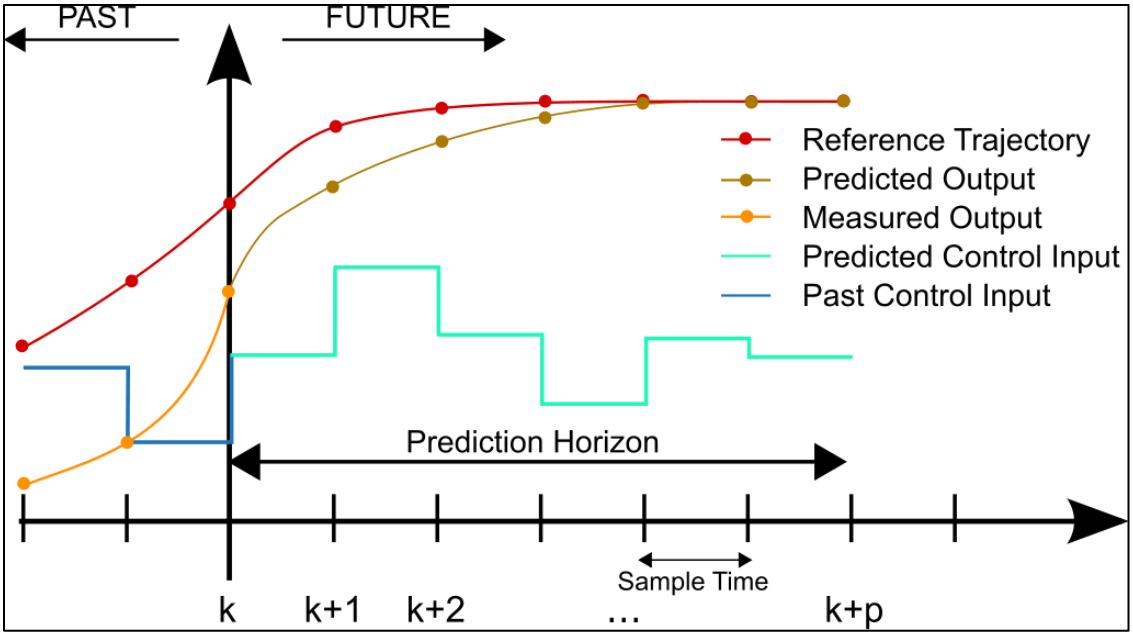

Fig-1: Sample MPC scheme

2.2. Vehicle Dynamic Model

For simplifying the model, only the front wheel can be steered. Also, there is an assumption that vehicle does not slip, any slippage is considered as external disturbance. Thus, slip angle is zero, meaning velocity is directed along the vehicle.

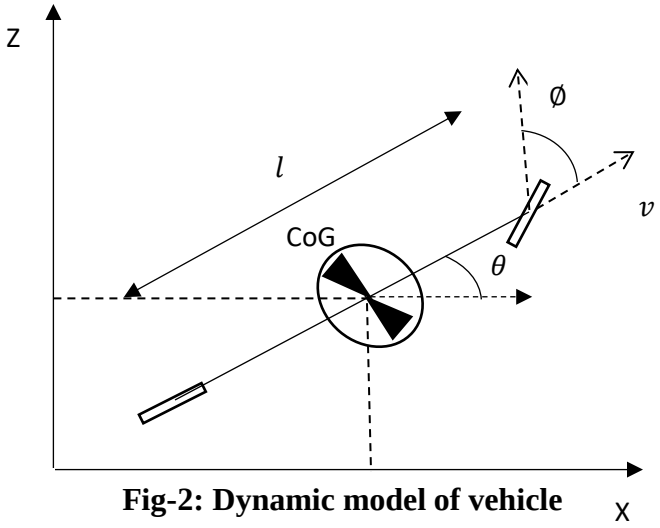

Fig-2: Dynamic model of vehicle

$$\dot{x} = v\cos\theta$$
$$\dot{z} = v\ \sin\theta$$
$$\dot{\theta} = \tan\emptyset\left(\frac{v}{l}\right)$$

Where l is the length between axes of the vehicle, θ is the heading angle of the vehicle and speed v and the steering angle $\emptyset$ are the inputs to the system.

2.3 Vehicle Safety Constraints

The Constraints represent the hard requirements of avoiding collision and ensuring vehicle's dynamic safety. These requirements are hard in sense that their violations are not allowed under any circumstances. An important constraint ensuring vehicle's dynamic safety is translated to avoiding tire lift off. This requirement can always be taken as an inequality constraint to enforce a nonzero vertical load on four tires. However, this would increase the computational load for MPC. Therefore, another way is to enforce an upper bound both for steering angle and the velocity of the car expressed by the inequality constraint

$$-\delta v_{max} \leq \delta v_{k+i} \leq \delta v_{max} \quad \text{- (C1)}$$

$$-\emptyset_{max} \leq \emptyset_{k+i} \leq \emptyset_{max} \quad \text{- (C2)}$$

We penalize the model if the car does not maintain at target speed. Ideally, there should not be any steering or zero acceleration. But since it is not practical, keep the rate of change of these as low as possible. These will prevent the motion sickness of passenger and drifting of the vehicle.

$$-\delta\emptyset_{max} \leq \delta\emptyset_{k+i} \leq \delta\emptyset_{max} \quad \text{- (C3)}$$

$$-\delta\omega_{max} \leq \delta\emptyset_{k+i} - \delta\emptyset_{k+i-1} \leq \delta\omega_{max} \quad \text{- (C4)}$$

The vehicle centrifugal and tangential accelerations are defined as acc_{cen} and acc_{tan} respectively. The Overall acceleration is given by $acc_{k+i} = \sqrt{acc_{cen}{}^2 + acc_{tan}{}^2}$. High vehicle acceleration is penalized by setting an upper bound so that resulting acceleration is less than maximum acceleration

$$w_{acc_{k+i}} = \begin{Bmatrix} 0, & if\ acc_{k+i} \leq acc_{max} \\ w_{acc_0}, & otherwise \end{Bmatrix} \quad \text{- (C5)}$$

The vehicle slows down when the acceleration is too large or deviations to the path is very large For this $w_{in_{k+i}}$ will be ON to to penalize $-\delta v_{k+i}$ and w_{vk+i} will be OFF so that J will not be penalized even if current speed is less that cruise speed.

$$w_{in_{k+i}} = \begin{Bmatrix} w_{in_0}, & if\ \delta v_{k+i} > 0\ and\ (acc > acc_{max}) \\ 0, & otherwise \end{Bmatrix} \text{- (C6)}$$

$$w_{v_{k+i}} = \begin{Bmatrix} 0, & if\ \delta v_{k+i} > 0\ and\ (acc > acc_{max}) \\ w_{v_0}, & otherwise \end{Bmatrix} \quad \text{- (C7)}$$

2.4. Genetic Algorithm

A genetic algorithm (GA) is a method for solving both constrained and unconstrained optimization problems based on a natural selection process that mimics biological evolution. The algorithm repeatedly modifies a population of individual solutions. At each step, the genetic algorithm randomly selects individuals from the current population and uses them as parents to produce the children for the next generation. Over successive generations, the population "evolves" toward an optimal solution. Various Selection Algorithms can be used in this work Deterministic Tournament selection algorithm has been used which has several benefits over alternative selection methods for genetic algorithms (for example, fitness proportionate selection and reward-based selection): it is efficient to code, works on parallel architectures and allows the selection pressure to be easily adjusted.

The genetic algorithm is used to solve problems that are not well suited for standard optimization algorithms, including problems in which the objective function is discontinuous, nondifferentiable, stochastic, or highly nonlinear.

The genetic algorithm differs from a classical, derivative-based, optimization algorithm in two main ways, as summarized in the following table.

Classical Algorithm	**Genetic Algorithm**
Generates a single point at each iteration. The sequence of points approaches an optimal solution.	Generates a population of points at each iteration. The best point in the population approaches an optimal solution.
Selects the next point in the sequence by a deterministic computation.	Selects the next population by computation which uses random number generators.

2.4.1 Deterministic q-tournament selection

Tournament Selection is a Selection Strategy used for selecting the fittest candidates from the current generation in a Genetic Algorithm. These selected candidates are then passed on to the next generation. In a q-way tournament selection, we select q-individuals and run a tournament among them. Only the fittest candidate amongst those selected candidates is chosen and is passed on to the next generation. In this way many such tournaments take place and we have our final selection of candidates who move on to the next generation. It also has a parameter called the selection pressure which is a probabilistic measure of a candidate's likelihood of participation in a tournament. If the tournament size is larger, weak candidates have a smaller chance of getting selected as it has to compete with a stronger candidate. The selection pressure parameter determines the rate of convergence of the GA. More the selection pressure more will be the Convergence rate. GAs can identify optimal or near-optimal solutions over a wide range of selection pressures. Tournament Selection also works for negative fitness values

3.Methodology

3.1. Non-Linear Model Predictive Control

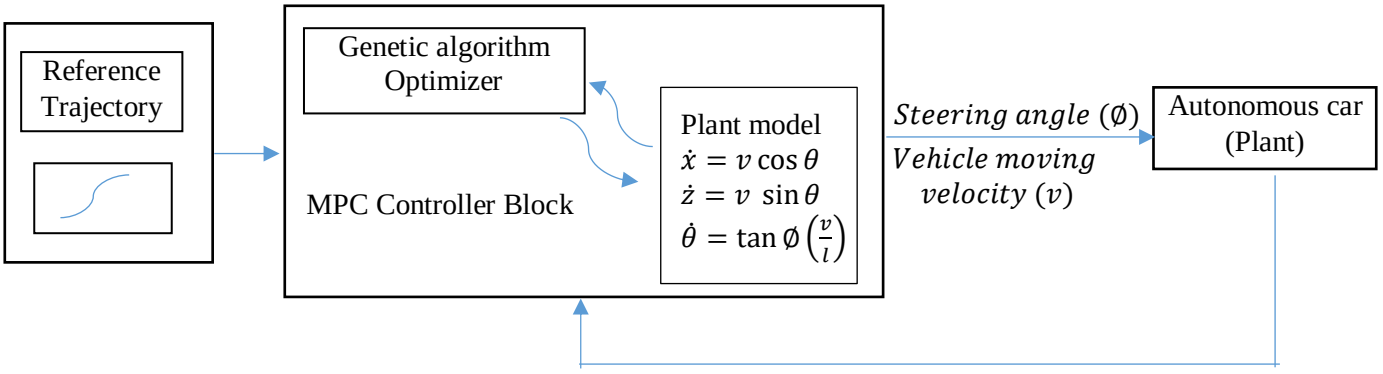

Fig-3: Overview of MPC architecture

$\varepsilon_k = [x_k, z_k, \theta_k]^T$ is the pose estimation of the vehicle with respect to the global coordinate system and T_s is the sampling time. The Controlled inputs to the system namely Steering angle $(\emptyset)$ and Velocity (v) are given by

$$v_k = v_{k-1} + \delta v_k$$
$$\emptyset_k = \emptyset_{k-1} + \delta\emptyset_k$$

These equations equivalently can be represented as

$$U_k = U_{k-1} + \delta U_k.$$

$$\begin{bmatrix} v_k \\ \emptyset_k \end{bmatrix} = \begin{bmatrix} v_{k-1} \\ \emptyset_{k-1} \end{bmatrix} + \begin{bmatrix} \delta v_k \\ \delta\emptyset_k \end{bmatrix}$$

Now, the state space representation can be written as

$$\begin{bmatrix} \dot{x} \\ \dot{z} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\ \sin\theta \\ \tan\emptyset\left({}^{v}/_{l}\right) \end{bmatrix}$$

Where state space vector is $\dot{x(t)} = f_c(x(t), u(t))$
Applying Euler Discretization with sampling time T_s ,

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ z_k \\ \theta_k \end{bmatrix} + T_s \begin{bmatrix} v_k\cos\theta_k \\ v_k\sin\theta_k \\ \tan\emptyset_k\ ({}^{v}/_{l}) \end{bmatrix}$$

Now, our reference trajectory path input to the optimizer function can be constructed using a second order degree polynomial:

$$X = aZ^2 + bZ + c$$

Based on the above Polynomial, we have calculated the co-efficient values and calculated pose of the vehicle at different time instances.

Steering angle ∅ can be calculated as follows:

Consider a tangent line AB drawn to the Reference path as shown below:

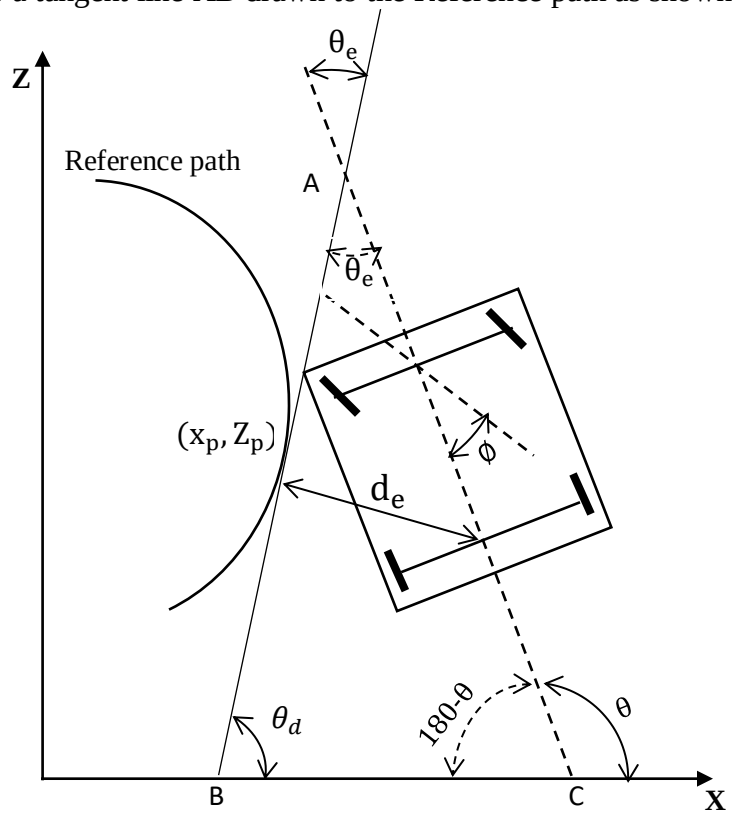

Fig-4: Non-holonomic vehicle model and error position

In the figure, consider the ΔABC,
Sum of interior angles, $(180^\circ - \theta_k) + \theta_e + \theta_d = 180^\circ$

$$\theta_e = \theta_k - \theta_d$$

Let the equation of the tangent be $Ax+By+C=0$

Slope of the curve is given by $\frac{dz}{dx}$

Differentiate (1) with respect to x,

$$1 = 2aZ\,\frac{dZ}{dx} + b\frac{dZ}{dx}$$

The slope is given as

$$\frac{dZ}{dx} = \frac{1}{2aZ+b}$$

From point slope form, we have

$$\frac{Z-Z_p}{x-x_p} = \frac{1}{2aZ_p+b}$$

$$Z(2aZ_p + b) - 2aZ_p^{\,2} - bZ_p = x - x_p$$

$$Z(2aZ_p + b) - x + \left(-2aZ_p^{\,2} - bZ_p + x_p\right) = 0 \qquad (2)$$

Now, the shortest distance between a point (x_k,z_k) and the line $Ax+By+C=0$ is

$$d = \frac{Ax_k + Bz_k + C}{\sqrt{A^2 + B^2}}$$

therefore, d_e is shortest distance between (x_k,z_k) and line (2) which is given by:

$$d_e = \frac{-x_k + Z_k(2aZ_p + b) + (-2aZ_p^{\,2} - bZ_p + x_p)}{\sqrt{(2aZ_p + b)^2 + 1^2}}$$

3.2. Optimization Problem

Now that we have derived variables which are our controlled inputs, we compute the cost function J which our Genetic algorithm optimizes for every single iteration the path with given trajectory. In this case, the cost function consists of three terms: Minimizing path tracking error ($\mathrm{d_e}, \theta_e$), Control input smoothening $(\delta\emptyset, \delta v)$ and it also includes safety constraints like keeping the acceleration within a limit (acc), preventing increasing in velocity (δv) and control the vehicle to travel at some cruise speed $(v_{\max})$

Thus, cost function can be constructed as:

$$J = \sum_{i=1}^{H_p} \left(w_1 \left\| d_{e_{k+i}} \right\|^2 + w_2 \left\| \theta_{e_{k+i}} \right\|^2 \right) + \sum_{i=0}^{H_c-1} \left(w_3 \| \delta\emptyset_{k+i} \|^2 + w_4 \| \delta v_{k+i} \|^2 \right)$$

$$+ \sum_{i=0}^{H_p} \left(w_{acc_{k+i}} \| acc_{k+i} \|^2 + w_{in_{k+i}} \| \delta v_{k+i} \|^2 + w_{v_{k+i}} \| v_{max} - v_{k+i} \|^2 \right)$$

Where w_1, w_2, w_3, w_4 are the weighing factors for path tracking error and control inputs whereas $w_{acc_{k+i}}, w_{in_{k+i}}, w_{v_{k+i}}$ are on-off weighting factors varying with vehicle states. H_p and H_c are prediction and control horizon respectively. Here the controller needs to simulate tunings for duration of prediction horizon. The controller must execute the simulation and run optimization at every sampling time T_s. So, if our microprocessor is not that capable, one technique is to reduce the computation time. One technique for this is to reduce the control horizon H_c which is the number of moves for duration of simulation phase. For this reason, we choose $H_c < H_p$

3.3. Genetic Algorithm Design

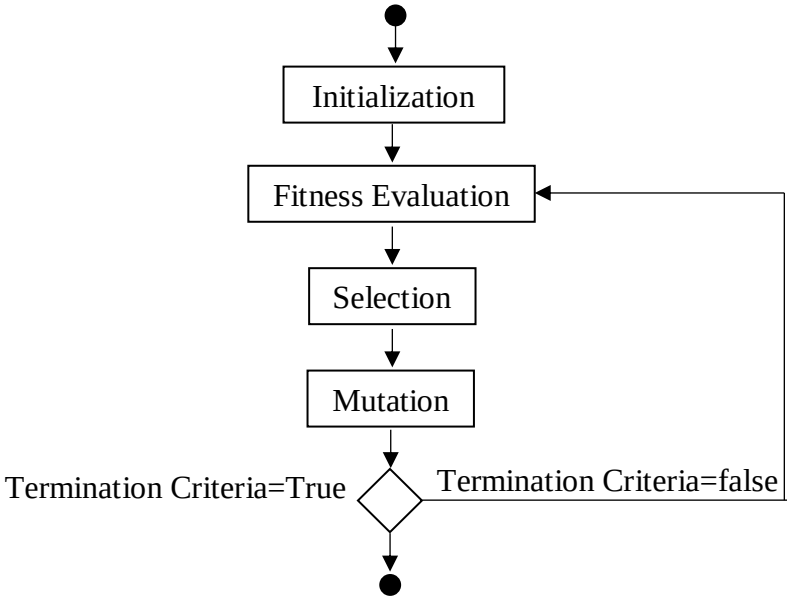

Fig-5: Genetic Algorithm Flow

3.3.1. Chromosome Representation

$$O_p = \begin{bmatrix} \delta v_1 & \delta v_2 & . & . & . & \delta v_{H_c} \\ \delta \emptyset_1 & \delta \emptyset_2 & . & . & . & \delta \emptyset_{H_c} \end{bmatrix}, p = 1,2,3\ldots\ N_{pop}$$

N_{pop} is total number of chromosomes in the population
Each Gene in chromosome δv_i or $\delta \emptyset_i$ follows constraint $C_1,\ C_2,\ C_4$

3.3.2. Initialization

80% of the initialization population is randomly generated satisfying the safety constraints. The remaining 20% are inherited from the previous best chromosome by shifting the best chromosome to the left by one position and patching the last position randomly.

3.3.3. Fitness Evaluation

The fitness F of an individual is defined based on the cost function J. The safety constraints $C_2,\ C_4$ enforced here to save computation time.

$$F = \frac{1}{(1+J)}$$

3.3.4. Selection

Two stages of selection are required, one is the mutation selection and the other is survivor selection. Deterministic q-tournament selection is implemented. Survivor selection selects N_{pop} chromosomes out of the $2N_{pop}$ after mutation which consists of the original ones before mutation selection (or parents) and the mutated ones (or children). In this selection process, all parents are replaced by children.

3.3.5. Mutation

Each gene, a randomly picked number in [0, 1] is compared with the mutation rate p_m'. If it is < p_m', then gene undergoes mutation, meaning the gene is replaced by a new one satisfying the constraints.

$$p_m' = \left(1 + (1/p_m - 1)e^{-\gamma N(0,1)}\right)^{-1}$$

$N(0,1)$ represents a random number picked from the normal distribution (mean of 0 and s.d. of 1).
γ - Learning rate

3.3.5. Termination Criteria

The termination condition is a tradeoff between the best optimal solution and computation time. In order to make sure that the MPC works properly, a suboptimal solution will be returned if the time taken is too long. In particular, the GA will terminate when the optimal solution is found or the time elapsed is reaching $1.1T_s$ whichever is earlier.

4 Results and discussions

w_1	0.8	H_c	15	$\delta_{v_{max}}$	0.05 m/s	w_{acc_0}	0.5
w_2	1.5	H_p	20	$\delta_{\emptyset_{max}}$	0.02 rad/s	w_{v_0}	5
w_3	2.0	l	1.28m	$\delta_{\omega_{max}}$	.015 rad/s^2	N_{pop}	40
w_4	2.0	T_s	0.1s	$\emptyset_{max}$	0.40 rad/s	p_{m_0}	0.1
v_{max}	20 m/s	$\mathbf{acc}_{max}$	1.5m/s^2	w_{in_0}	2.0	λ	2.0

Fig-6: Parameters in MPC setup

The above parameters values are used in simulations. The simulator takes steering and velocity inputs from MPC and outputs the vehicle real velocity, steering angle, location and orientation in map to MPC. The MPC controller takes feedback from simulator and locates the vehicle on the map. It then fits a curve approximately close to the reference path. The cost function is computed, and this is passed to GA function to search for the optimal control sequence.

Since we do not have a road map available to us, we have created a path similar to the one in the paper and used part of the trajectory from map which is shown in the Fig 7.We have run simulations on the blue colored box region A as highlighted in the Fig-7 to show the working of our Control algorithm.

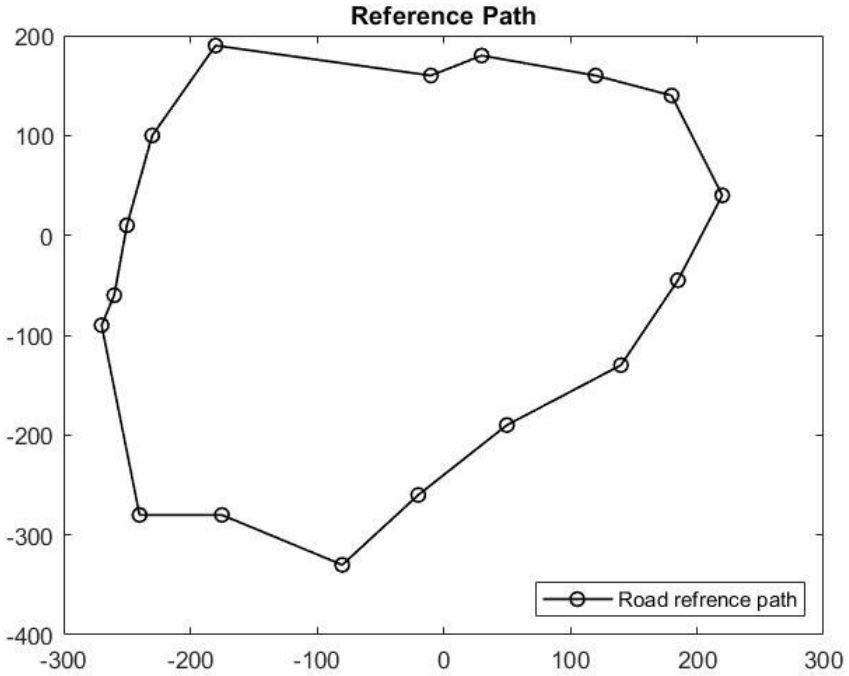

Fig 7: Reference Path

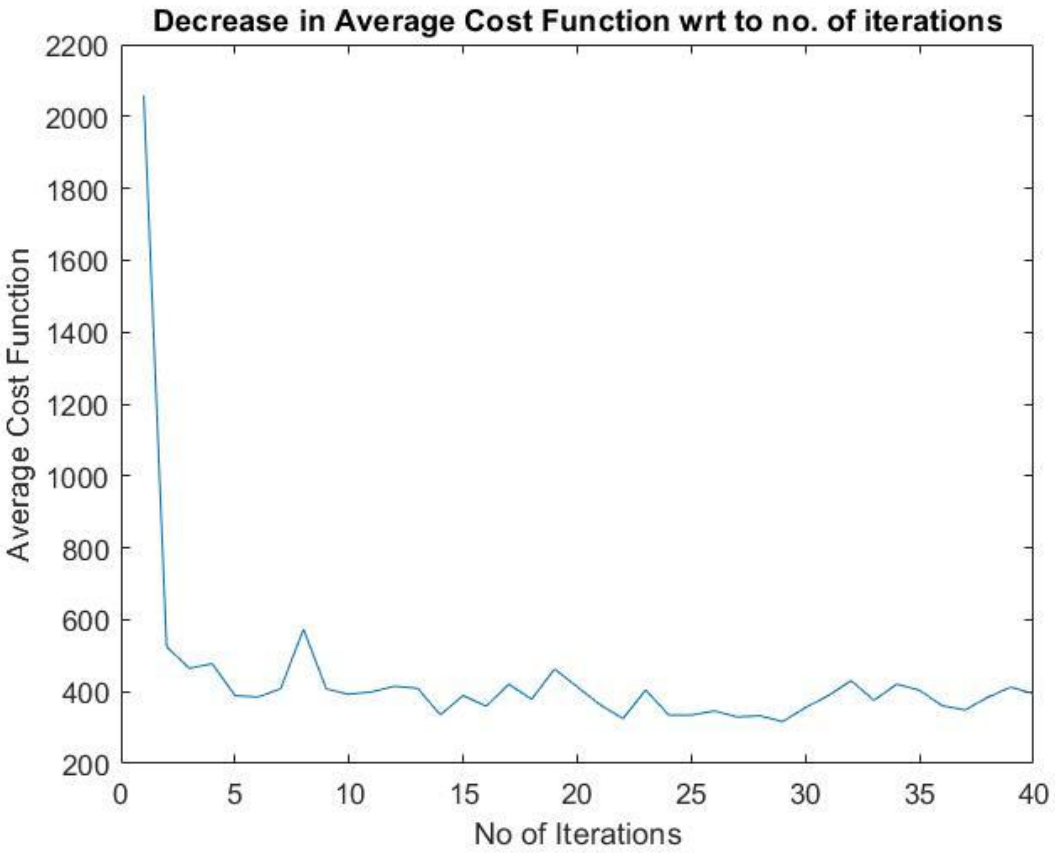

Fig 8: Average Cost Function (J)

Here we observe that GA is optimizing the cost function for every iteration. Initially, the predicted output differed a lot from the Actual path. But as GA algorithm optimizes, cost function drastically reduces from 2000 to 400 which shows that GA is performing well in finding the optimal control sequence. The path tracking by the MPC is shown in the below figure. We observe that, the predicted trajectory is close to the reference trajectory though it is irregular in the beginning. Theta error (orientation of vehicle) and distance error (distance of vehicle with respect to actual path) has been measured and fed into the cost function which is minimized by GA.

The Fig 9 and Fig 10 shows our GA solver is able to use the vehicle model to predict the future path and produce the necessary control inputs to steer the vehicle

The Fig 11 shows the comparison between x and z coordinates of reference path and simulated path. Our algorithm is able to follow th given path with minimal distance and orientation errors

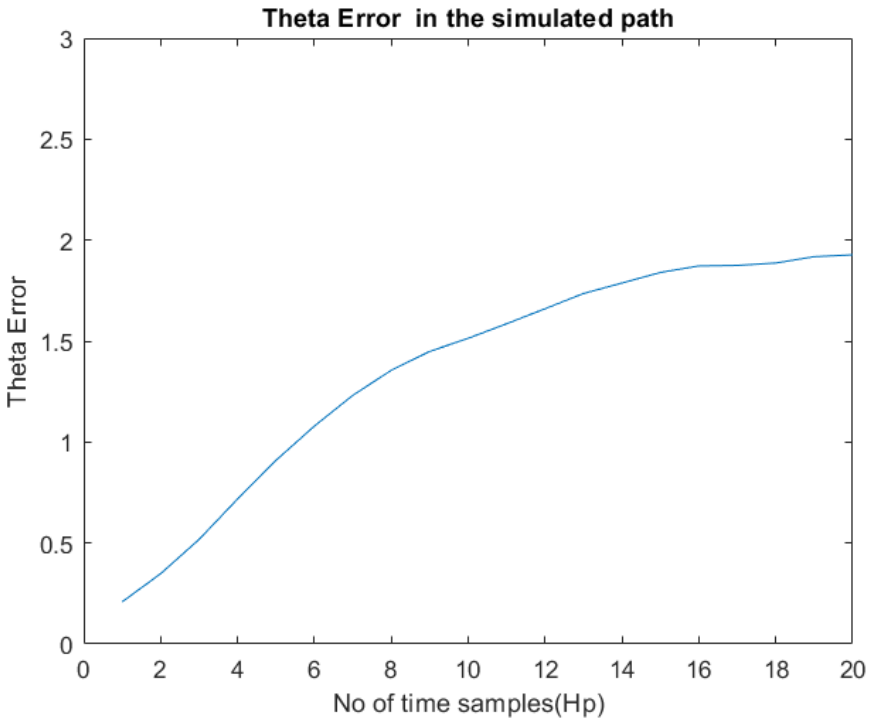

Fig-9: Orientation error with respect to Reference

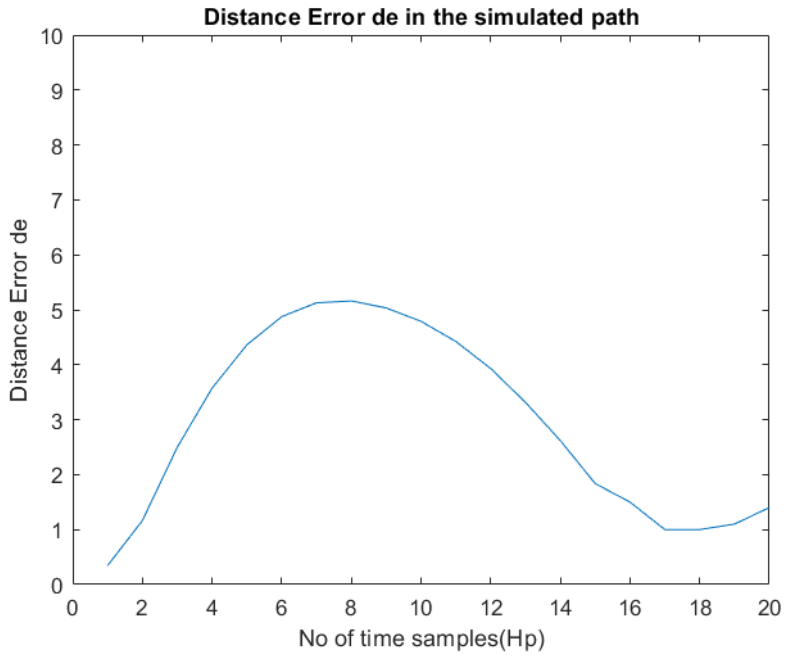

Fig-10: Distance error with respect to Reference

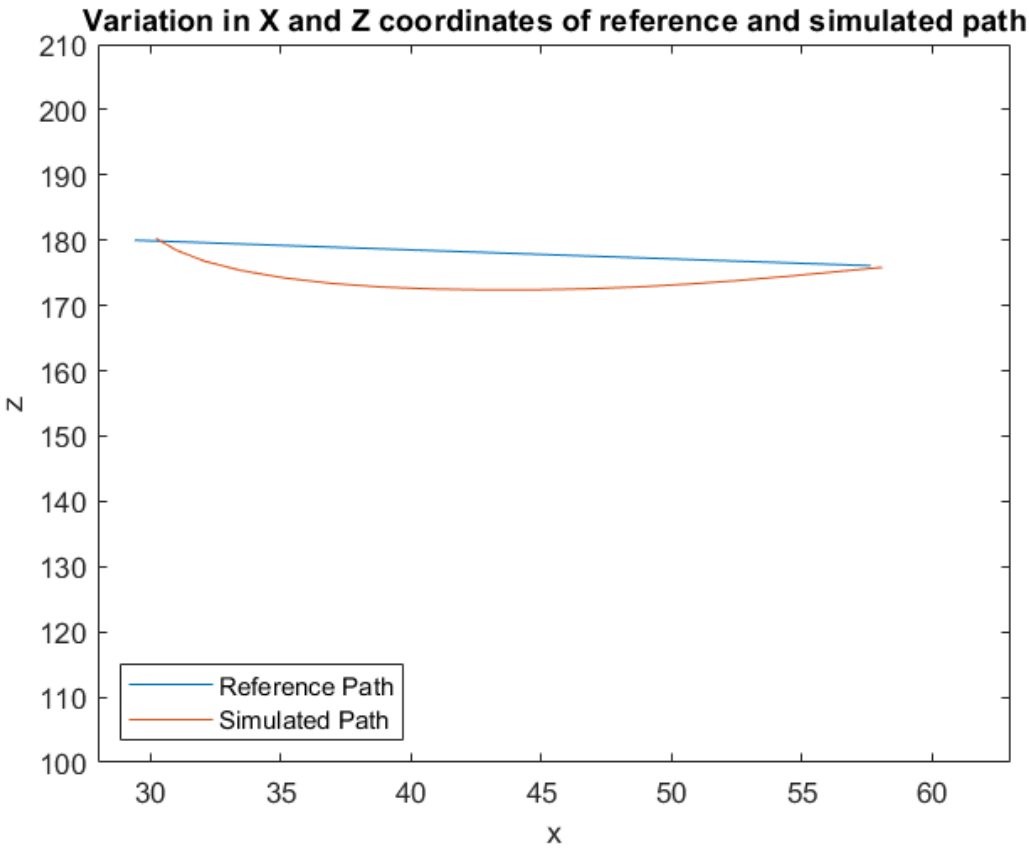

Fig-11: Simulated Path Vs Reference Path

5. Conclusions

We observe that the both the steering angle and velocity are being controlled simultaneously in which the cost function is continuously being optimized by the Genetic algorithm. Also, most importantly various safety constraints have been considered to ensure comfort and safety for the passengers. The simulation results show that the MPC is performing better in minimizing the offset and tends to steer the car. The vehicle can maneuver to center of the lane and met the safety constraints/ boundaries which shows that controlled inputs to the system (velocity and steering angle) are smoothened and does not cause any rollover or jerking effect.

6. References

[1] - Ming, Lei, Guan Zailin, and Yang Shuzi. "Mobile robot fuzzy control optimization using genetic algorithm." *Artificial intelligence in engineering* 10.4 (1996): 293-298.

[2] - Liu, Kai, et al. "Model Predictive Stabilization Control of High-Speed Autonomous Ground Vehicles Considering the Effect of Road Topography." *Applied Sciences* 8.5 (2018): 822.

[3] - Thilén, Emma. "Robust Model Predictive Control for Autonomous Driving." (2017).

[4] - Lin, Ching-Fu, Jyh-Ching Juang, and Kun-Rui Li. "Active collision avoidance system for steering control of autonomous vehicles." *IET Intelligent Transport Systems* 8.6 (2014): 550-557.

[5] - Miller, Brad; Goldberg, David (1995s). "Genetic Algorithms, Tournament Selection, and the Effects of Noise" (PDF). *Complex Systems*. **9**: 193–212.

[6] - Anderson, Sterling J., et al. "An optimal-control-based framework for trajectory planning, threat assessment, and semi-autonomous control of passenger vehicles in hazard avoidance scenarios." *International Journal of Vehicle Autonomous Systems* 8.2-4 (2010): 190-216.

[7] - Sharma, S. K., and R. Sutton. "A genetic algorithm based nonlinear guidance and control system for an uninhabited surface vehicle." *Journal of Marine Engineering & Technology* 12.2 (2013): 29-40.

[8] - Sarimveis, Haralambos, and George Bafas. "Fuzzy model predictive control of non-linear processes using genetic algorithms." *Fuzzy sets and systems* 139.1 (2003): 59-80.

[9] - F. Manenti, “Considerations on nonlinear model predictive control techniques,” Comput. Chem. Eng., vol. 35, no. 11, pp. 2491–2509, Nov. 2011.

APPENDIX A
MATLAB CODE
Main file

```
O=GA_init1();
[x_ref,z_ref,theta_ref,a,b,c]=refpath();
x_init=x_ref(1);
z_init=z_ref(1);
theta_init=theta_ref(1);
Javg=zeros(40,1);
[F,AvgJ,J]=Fitness(O,x_ref,z_ref,theta_ref,a,b,c,x_init,z_init,
theta_init);
Onew=tournament(O,F);
Onew=mutation(Onew);
Javg(1)=AvgJ;
for l=1:39
[F,AvgJ]=Fitness(Onew,x_ref,z_ref,theta_ref,a,b,c,x_init,z_init,
theta_init);
Onew=tournament(Onew,F);
Onew=mutation(Onew);
Javg(l+1)=AvgJ;
End
```

Functions

```
function[Onew]=tournament(O,F)
Npop=40;
Sel=randi([1,Npop],20,1);%
[m,i]=max(F(Sel));
Sel(i);
Onew=O(Sel(i));
for l=2:Npop
    Sel=randi([1,Npop],10,1);%
    [m,i]=max(F(Sel));
    Sel(i);
    Onew=[Onew;O(Sel(i))];
End

function[Onew]=mutation(Onew)
pmo=0.1;%initial mutation rate
gamma=0.2;%learning rate
pm=pmo;Npop=10;
n=20;%prdiction horizon or no of time samples
delvmax=(0.05); % m/sdelphimax=0.02;% rad/s
delomegamax=0.015;% rad/s2
for i=1:Npop
     for j=1:n
```

```
          if(rand<pm)
     Onew(i).o(1,j)=(randi(10000*[-delvmax delvmax],1))/10000;
          if Onew(i).o(2,j)>0
              sign=1;
          else
              sign=-1;
          end
  Onew(i).o(2,j)= (randi(10000*[-delphimax delphimax],1))/10000;
          if(j>1)
     while ~(((-delomegamax) <=(Onew(i).o(2,j)-Onew(i).o(2,(j-
      1)))&&(Onew(i).o(2,j)-Onew(i).o(2,(j-1)))<=delomegamax))
  Onew(i).o(2,j)= (randi(10000*[-delphimax delphimax],1))/10000;
            end
          end
          if((sign)*(Onew(i).o(2,j))<0)
              Onew(i).o(2,j)=-Onew(i).o(2,j);
          end
          pm=(1+((1/pm)-1)*exp(-gamma*rand))^(-1);
          end
      end
end


function[F,AvgJ,J]=Fitness(O,x_ref,z_ref,theta_ref,a,b,c,x_init,
z_init,theta_init)
w1=0.8;w2=1.5;w3=2.0;w4=2.0;Ts=0.1;l=1.28;
Npop=40;
n=20;%prdiction horizon or no of time samples
delvmax=(0.05); % m/s
delphimax=0.02;% rad/s
delomegamax=0.015;% rad/s2
F=zeros(Npop,1);%fitness evaluation
J=zeros(Npop,1);
theta_veh=zeros(n,1);%path angle
x_veh=zeros(n,1);
z_veh=zeros(n,1);
for i=1:Npop
% i=1;
V=20;Phi=0.05;
V=V+O(i).o(1,1);%v is v+ delv
Phi=Phi+O(i).o(2,1);%Phi is Phi+ delPhi
theta_veh(1)=[theta_init+(V*tan(Phi)*Ts/l)];
x_veh(1)=[x_init+V*cos(theta_veh(1))*Ts];
z_veh(1)=[z_init+V*sin(theta_veh(1))*Ts];
for j=2:n
V=V+O(i).o(1,j);
```

```
Phi=Phi+O(i).o(2,j);
theta_veh(j)=[theta_veh(j-1)+(V*tan(Phi)*Ts/l)];
x_veh(j)=[x_veh(j-1)+V*cos(theta_veh(j))*Ts];
z_veh(j)=[z_veh(j-1)+V*sin(theta_veh(j))*Ts];
end
theta_err=theta_veh-theta_ref;%to minimisde in cost func
% d_err
d_err=zeros(n,1);
for k=1:n
 d_err(k)=abs((z_veh(k)*(2*a*z_ref(k)+b) -x_veh(k)+(x_ref(k)-
2*a*((z_ref(k))^2)-
b*(z_ref(k))))/(sqrt(((2*a*z_ref(k))+b)^2)+1));
end
J(i)=(w1*((norm(d_err))^2))
+(w2*((norm(theta_err))^2))+(w3*((norm(O(i).o(2,1:5)))^2))+(w4*(
(norm(O(i).o(1,1:3)))^2));
F(i)=(1/(1+J(i)));
end
AvgJ=sum(J)/Npop
end


function[x_ref,z_ref,theta_ref,a,b,c]=refpath()

x_ref=[30,31.5,33,34.5,36,37.5,39,40.5,42,43.5,45,46.5,48,49.5,5
1,52.5,54,55.5,57,58.5,60,61.5,63,64.5,66,67.5,69,70.5];

z_ref=[180,179.5,179.4,179.3,179.2,179,178.6,178.5,178,178,177.9
,177.8,177.6,177,176.9,176.8,176.7,176.3,176.1,176.2,176,175.8,1
75.6,175.4,175.2,175,174.5,174.1];

x_ref=x_ref(1:20)';z_ref=z_ref(1:20)';

p=polyfit(z_ref,x_ref,2);

a=p(1);b=p(2);c=p(3);

x_ref=polyval(p,z_ref);

n=size(z_ref);

theta_ref=zeros(5,1);

for i=1:n

theta_ref(i)=atan((1/((2*p(1)*z_ref(i)))+p(2)));

end


function[O]=GA_init1()
Npop=40;
n=20;%prdiction horizon or no of time samples
```

```
pmo=0.1;%initial mutation rate
delvmax=(0.05); % m/s
delphimax=0.02;% rad/s
delomegamax=0.015;% rad/s2
delv=(randi(10000*[-delvmax delvmax],1,n))/10000;
delphi= (randi(10000*[-delphimax delphimax],1,n))/10000;
i=1;
while(i<(n))
    if ~(((-delomegamax) <=(delphi(i+1)-
delphi(i)))&&((delphi(i+1)-delphi(i))<=delomegamax))
        delphi= (randi(10000*[-delphimax delphimax],1,n))/10000;
        i=1;
    else
        i=i+1;
    end
end
O1=[delv;delphi];%chromosome
O=struct('o',O1);
for j=2:Npop
    delv=(randi(10000*[-delvmax delvmax],1,n))/10000;
    delphi= (randi(10000*[-delphimax delphimax],1,n))/10000;
    %%%%%%%%%%%%%
    i=1;
    while(i<(n))
        if ~(((-delomegamax) <=(delphi(i+1)-
delphi(i)))&&((delphi(i+1)-delphi(i))<=delomegamax))
          delphi= (randi(10000*[-delphimax
delphimax],1,n))/10000;
          i=1;
        else
          i=i+1;
        end
     end
    %%%%%%%%%%%%%%%%
    O1=[delv;delphi];
    O=[O;struct('o',O1)];
end
end
```