ROB 590: Directed Study

Applications of Control Barrier Functions (CBFs) using Gaussian Processes (GPs) for Unknown Nonlinear Dynamics

Harikumar, Vaishnavi

A. Introduction

This report discusses the theory and applications of Control Barrier Functions (CBFs) on non-linear systems where the underlying dynamics of the system is deterministic but unknown. Using the data-driven approach of Gaussian Process Regression (GPR), the dynamics of an unknown system can be determined, which can then be used to build a controller using the Lyapunov Function and Control Barrier Function. The purpose of this directed study is to draw a comparison between the effects of two optimization algorithms on an unknown nonlinear system while ensuring safety and stability of the system. The mathematical formulations and implementation of two nonlinear systems have been discussed in this report. The results from two journal papers have been reproduced through implementation of the concepts discussed in them. The following sections describe in detail all concepts and results.

B. Theoretical Concepts

For this project, I was required to learn some very new theoretical concepts to reproduce the results from the papers. They have been briefly discussed as follows:

I. **Lyapunov Theory -** This theory devises a system energy function to analyze the stability of a system. Considering the dynamics of an uncontrolled autonomous system; to verify whether the trajectory converges to an equilibrium point, a Lyapunov function is necessary. It can be defined as:

$$Lyapunov\ Function\ \Longrightarrow V(x)$$

Such that,

$$V(x_e) = 0, \qquad V(x) > 0 \qquad \forall\, x \neq x_e$$
$$\dot{V}(X) = \frac{\partial \mathrm{V}}{\partial x}\, f(x) \,<\, 0 \quad \forall\, x \,\neq x_e$$

Here,

$V'(x) = derivative\ of\ Lyapunov\ Function\ w.r.t\ the\ dynamics$

$\frac{\partial \mathrm{V}}{\partial x} = Gradient\ of\ Lyapunov\ Function$

$f(x) = Vector\ Field$

This means that $V'(x)$ must be negative definite for asymptotic convergence. For exponential convergence, $V'(x)$ is given by,

$$\dot{V}(X) < -kV(x)\ ; \quad for\ k > 0$$

In a Lyapunov Function, exponential convergence is preferred since it gives a rate of convergence whereas in asymptotic convergence, there is no notion of how quickly the system will reach equilibrium.

II. **Control Lyapunov Function (CLF)** –
For a dynamical system in its control affine form given by,

$$\dot{X} = f(X) + g(X)\, u$$

Consider a Lyapunov Function $V(X)$,

$$\dot{V}(X) = \frac{dV}{dt} = \frac{dV}{dX} \cdot \frac{dX}{dt} = \frac{\partial V}{\partial x} . \dot{X} \leq -kV(x)$$

$$\dot{V}(X) = \frac{\partial V}{\partial x}\, [\, f(X) + g(X)\, u] \quad \leq -kV(x)$$

$$\dot{V}(X) = \frac{\partial V}{\partial x}\, f(X) + \frac{\partial V}{\partial x} g(X)\, u \quad \leq -kV(x)$$

$$\dot{V}(X) = L_f V(X) + L_g V(X)\, u \quad \leq -kV(x)$$

Therefore, the CLF derivative condition can be defined by the constraint,

$$L_f V(X) + L_g V(X)\, u \quad \leq -kV(x) \qquad (\boldsymbol{Eqn. 1})$$

Here,
$L_f V(X)\ and\ L_g V(X) = Lie\ Derivative\ Operators\ of\ the\ Lyapunov\ Function$
$u\ \ = control\ input$
$X\ \ = control\ state$

Based on Eqn. 1, the control inputs u must be found such that the CLF constraint is satisfied. For this, a Quadratic Program (QP) can be used. The CLF-QP can be mathematically represented as:

$$\min\ \ u^T u = u_1^2 + u_2^2 + \cdots + u_n^2$$
$$such\ that\text{:}\ \ L_f V(X) + L_g V(X)\, u \leq -kV(x)$$

This algorithm is used to ensure convergence to the goal state.

III. **Nagumo's Invariance Principle** – This principle evaluates whether the trajectory of a system remains within a region of interest. Given an obstacle, a function can be devised such that the trajectory of the system avoids collision. Here, we create a barrier function:

$$Barrier\ Function\ \Rightarrow h(x)$$

Such that,

$$h(x) = 0, \qquad \dot{h}(x) \geq\ 0, \qquad \forall\, x$$

Which means, at the boundary of the obstacle, the speed of the agent along the trajectory of the system must be 0 or tending towards 0.

Also,

$$h(x) \geq 0 \implies -\alpha h(x) \leq 0$$

Therefore, the Barrier function can be written as:

$$\dot{h}(x) \geq -\alpha h(x), \quad \forall\, \alpha \in R^{+}$$

IV. **Control Barrier Function (CBF) –**

For a dynamical system in its control affine form given by,

$$\dot{X} = f(X) + g(X)\, u$$

Consider a Barrier Function $h(X)$,

$$\dot{h}(X) = \frac{dh}{dt} = \frac{dh}{dX} \cdot \frac{dX}{dt} = \frac{\partial h}{\partial x} . \dot{X} \geq -\alpha h(x)$$

$$\dot{h}(X) = \frac{\partial h}{\partial x} \left[f(X) + g(X)\, u \right] \geq -\alpha h(x)$$

$$\dot{h}(X) = \frac{\partial h}{\partial x} f(X) + \frac{\partial h}{\partial x} g(X)\, u \geq -\alpha h(x)$$

$$\dot{h}(X) = L_f h(X) + L_g h(X)\, u \geq -\alpha h(x)$$

Therefore, the CBF derivative condition can be defined by the constraint,

$$L_f h(X) + L_g h(X)\, u \quad \geq -\alpha h(x) \qquad (\boldsymbol{Eqn. 2})$$

Here,

$Lfh(X)\ and\ Lgh(X) = Lie\ Derivative\ Operators\ of\ the\ Barrier\ Function$

$u\ = control\ input$

$X\ = control\ state$

Based on Eqn. 2, the control inputs u must be found such that the CBF constraint, which is the safety constraint, is satisfied. For this, like in the case of Lyapunov Functions, a Quadratic Program (QP) can be used. The CBF-QP can be mathematically represented as:

$$\min\ u^T u = u_1^2 + u_2^2 + \cdots + u_n^2$$
$$such\ that\!: \ L_f h(X) + L_g h(X)\, u \quad \geq -\alpha h(x)$$

This algorithm is used to ensure safety of the agent from along the system trajectory.

To apply these algorithms to a given agent so that the safety of the agent is ensured along with the convergence of the agent to the goal, the two constraints must be applied to the optimization of the control input. This algorithm is defined as the CLF-CBF-QP, as shown:

$$\begin{aligned} \min\ & u^T u = u_1^2 + u_2^2 + \cdots + u_n^2 \\ \mathit{such\ that}:\ & L_f V(X) + LgV(X)\, u \le -kV(x) \\ & L_f h(X) + L_g h(X)\, u \ \ \ge\ -\alpha h(x) \end{aligned}$$

However, this algorithm is not complete. The safety constraint provided by the Barrier function $h(X)$, prevents the optimization from converging because the CBF being the safety constraint, is a strong function that does not allow quick convergence. Therefore, a slack variable δ is added to the objective function and the CLF constraint. The modified algorithm is:

$$\begin{aligned} \min\ & u^T u = (u_1^2 + u_2^2 + \cdots + u_n^2) + M\delta^2 \\ \mathit{such\ that}:\ & L_f V(X) + LgV(X)\, u \le -kV(x) + M\delta \\ & L_f h(X) + L_g h(X)\, u \ \ \ge\ -\alpha h(x) \end{aligned}$$

Eqn. 3

This algorithm is the CLF-CBF-QP which ensures both safety and stability of the system.

Note: The slack variable δ is also a design variable that needs to be optimized. It allows the weight of the CBF to decrease and hence assures convergence. δ needs to be optimized because choosing an arbitrary large δ defeats the purpose of a slack variable.

C. Journal Papers Implemented

For this directed study, two journal papers have been implemented.

The first paper, *Control Barrier Functions for Unknown Nonlinear Systems using Gaussian Processes* [1], discusses a more theoretical application of CBFs. Here, they use a Gaussian Process to learn the unknown nonlinear dynamics which are then used to synthesize a safety controller for a jet engine example. However, this paper focuses on developing a systematic approach to compute control barrier functions, which is out of the scope of this project. To build a simulation representing a controller that uses GP, CLF and CBF, I deviated from the case study used in this paper and adopted the motion of a robot around an obstacle for the simulation of a Gaussian Process based Control Lyapunov Function-Control Barrier Function-Quadratic Program (GP-CLF-CBF-QP). However, the concepts used for this simulation are very similar to those discussed in this paper.

The second paper, *Pointwise Feasibility of Gaussian Process-based Safety-Critical Control under Model Uncertainty* [2], builds upon the GP-CLF-CBF-QP to build a min-norm convex optimization-based controller called a GP-CLF-CBF-SOCP which is a Gaussian Process based Control Lyapunov Function-Control Barrier Function-Second Order Cone Program. The main theoretical results of this paper include a discussion and proof of the pointwise feasibility of min-norm convex optimization problem that is proposed. While a brief description of these results has been added in this report, the focus of the implementation of this paper is on the process of the optimization problem and the results obtained from applying these concepts on an Adaptive Cruise Control system.

I. Journal Paper 1

Control Barrier Functions for Unknown Nonlinear Systems using Gaussian Processes

This paper proposes a two-step approach for controller synthesis for unknown, nonlinear systems while ensuring safety constraints. The first step involves using Gaussian processes to learn the unknown control affine nonlinear dynamics, while the second step involves developing a systematic approach to compute control barrier functions that explicitly take into consideration the uncertainty of the learned model. The paper discusses the importance of synthesizing controllers that enforce safety in safety-critical applications. It highlights that conventional techniques for synthesizing such controllers require a precise mathematical model of the system, which is not always available. In such cases, data-driven approaches from machine learning can be used to identify unmodeled dynamics with high precision and complement the mathematical analysis from control theory. Gaussian processes have emerged as a data-driven approach providing a non-parametric probabilistic modeling framework to design controllers for unknown dynamical systems.

System Dynamics:

This paper considers a specific class of systems represented by unknown nonlinear control affine dynamics. These systems can be mathematically described as:

$$\dot{x} = f(x) + g(x)u$$

Here,

$x \in R^n$ = system state
$\dot{x} = derivative\ of\ x\ w.r.t\ time$
$u \in R^m$ = control input
$f(x), g(x) = functions\ that\ define\ the\ true\ dynamics\ of\ the\ system$

Three important assumptions have been made during the mathematical formulation of the system dynamics:

Assumption 1: The system is control affine and nonlinear. This means that the system dynamics can be expressed as a linear combination of the control inputs and a nonlinear function of the state variables.

Assumption 2: The system is subject to safety constraints given by a set of polytopic constraints. A polytope is a geometric object that is defined as the intersection of a finite number of half-spaces. In this case, the safety constraints are defined as a set of linear inequalities that define a polytope in the state space.

Assumption 3: The system is observable. This means that the state variables of the system can be measured or estimated from the available input-output data. Additionally, a Gaussian noise given by $w \sim N(0_n, \rho_f^2 \boldsymbol{I}_n)$ is added to the observed data.

These assumptions are important for the problem formulation and the development of the proposed approach for synthesizing a controller that ensures safety while accounting for the uncertainty in the system dynamics.

Gaussian Process Model:

A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution. A Gaussian process is completely specified by its mean function $m(x)$ and covariance function $k(x, x')$.

Given a set of input-output pairs $D = \{(x_1, y_1), (x_2, y_2), \dots \dots, (x_n, y_n)\}$, where x_i is the input and y_i is the output, the goal is to learn the underlying function $f(x)$ that maps the input to the output.

The Gaussian process model assumes that the output y is a random variable that follows a Gaussian distribution with mean $m(x)$ and variance $\sigma^2(x)$. The mean function $m(x)$ represents the expected value of the output given the input, while the covariance function $k(x, x')$ represents the degree of similarity between the outputs at different inputs.

The joint distribution of the output y at any set of inputs $X = \{ x_1, x_2, \dots., x_m\}$ is given by:

$$y \sim GP(m(X), K(X, X))$$

where $m(X)$ is the mean vector and $K(X, X)$ is the covariance matrix. The elements of the covariance matrix are given by:

$$K(X, X) = \left[k(x_i, x_j)\right] \quad for\ i, j = 1, 2, \dots m$$

The Gaussian process model can be used to make predictions at new input points x^* by computing the conditional distribution of the output y^* given the observed data D.

In this paper, the Gaussian process (GP) technique is used to approximate a nonlinear mapping function, $f: X \to R^n$. As f is n-dimensional, each component, f_j is approximated using a Gaussian process, where $f_j(x) \sim GP(m_j(x), k_j(x, x_0))$ for $j = \{1, \dots., n\}$.

The approximation of f using n independent GPs is given as:

$$f(x) = \begin{cases} f_1(x) \sim GP(m_1(x), k_1(x, x_0) \\ \quad\quad\quad\quad\quad\quad . \\ \quad\quad\quad\quad\quad\quad . \\ \quad\quad\quad\quad\quad\quad . \\ f_n(x) \sim GP\big(m_n(x), k_n(x, x_0)\big) \end{cases}$$

Given a set of N measurements, $\{y^{(1)}, \dots.. y^{(N)}\}$ and corresponding input states, $\{x^{(1)}, \dots.. x^{(N)}\}$, where $y^{(i)} = f\big(x^{(i)}\big) + w^{(i)},\ i \in \{1, \dots., N\}$, the posterior distribution for $f_j(x)$ where $j = \{1, \dots., n\}$, at an arbitrary state $x \in X$, is computed as a normal distribution $N(\mu_j(x)\ , \rho_j(x))$ with the mean and covariance given by:

$$\mu_j(x) = \overline{k}_j^T \left(K_j + \rho_f^2 I_N\right)^{-1} y_j \quad (\boldsymbol{Eqn. 4})$$

$$\rho_j^2 = k_j(x,x) - \overline{k}_j^T \left(K_j + \rho_f^2 I_N\right)^{-1} \overline{k}_j \quad (\boldsymbol{Eqn. 5})$$

Here,

$$\overline{k}_j = \left[k_j\left(x^{(1)}, x\right), \ldots . k_j\left(x^{(N)}, x\right)\right]^T \in R^N$$

$$y_j = \left[y_j^{(1)}, \ldots ., y_j^{(N)}\right]^T \in R^N$$

$$K_j = \begin{bmatrix} k_j\left(x^{(1)}, x^{(1)}\right) & \cdots & k_j\left(x^{(1)}, x^{(N)}\right) \\ \vdots & \ddots & \vdots \\ k_j\left(x^{(1)}, x^{(N)}\right) & \cdots & k_j\left(x^{(N)}, x^{(N)}\right) \end{bmatrix} \in R^{N \times N}$$

By utilizing these expressions, the GP framework enables the computation of posterior distributions and facilitates the approximation of the unknown nonlinear system dynamics.

Case Study – Robot Motion:

To implement the concepts discussed, the adopted model for implementation is the motion of a robot whose underlying dynamics are unknown.

Assume a control affine non-linear system:

$$\dot{x} = f(x) + g(x)u$$

Here, x is the state given by $x = [x_1, x_2]$. The function $g(x)$ is known and $f(x)$ is the unknown function that needs to be found using GP. And u is the control input that determines the motion of the robot. To simulate the motion of the robot and train the GP, the true dynamics is given as:

$$f(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \end{bmatrix} = \begin{bmatrix} \dfrac{-x_2 - \frac{3}{2}x_1^2 - \frac{1}{2}x_1^3}{6} \\ \dfrac{x_1}{6} \end{bmatrix} \quad and \quad g(x) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

To implement the GP, first a set of 100 random states for x is generated by sampling from a uniform distribution. Using these random states, the corresponding $f(x)$ values are generated using the given dynamics to use as true data for both training the GP as well as comparing the results from GP to the true dynamics of the robot.

The random states generated were then divided into training and testing data. The first six states from the randomly generated states and the corresponding true values were taken to train the Gaussian Process. Python has an inbuilt function to perform Gaussian Process Regression which is fundamentally built upon Eqn. 4 and 5. This inbuilt function called *GaussianProcessRegressor* imported from the *sklearn* library allows the user to choose the optimal kernel for the given problem. For this simulation, the code uses a squared exponential kernel called the RBF kernel. The training data and the true values are used to train and fit the GP. The remaining 94 randomly generated states are used to test the GP which then outputs the mean and standard deviation. Fig.1 shows two subplots comparing the true values and the GP results individually for the functions $f_1(x)$ and $f_2(x)$.

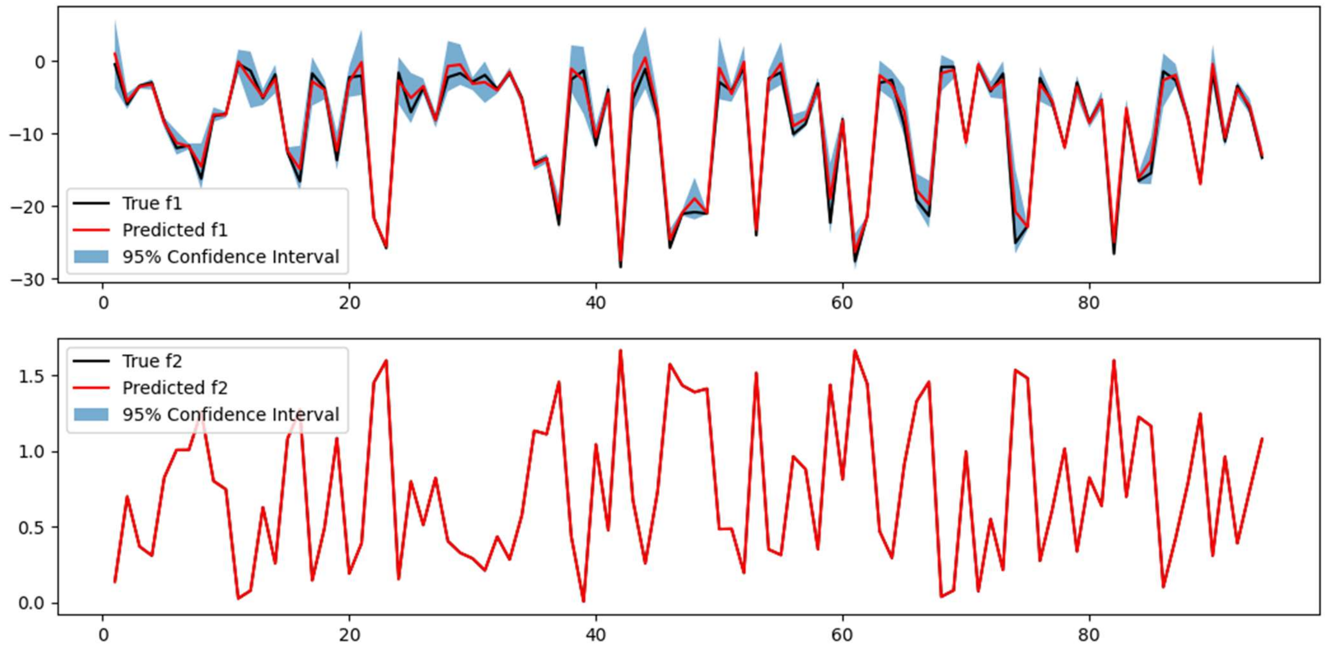

Fig. 1: Comparison of true values to results from Gaussian Process Regression

From Fig.1, it is evident that the GPR was successful in fitting the training data. The plot on top shows the comparison of true values to GP mean for function f_1 which also has a slight variance due since f_1 is a non-linear function of the state x. The plot on the bottom shows the comparison of true values to GP mean for function f_2 which does not seem to have a variance since the GP fits perfectly. This is because the function f_2 is a linear function of the state x.

After training the GP, to simulate the dynamics of the robot, a controller was built using the python library *cvxpy*. This library allows the optimization of a quadratic function given a set of linear inequality constraints. To build the controller, the CLF-CBF-QP as per Eqn. 3 was used.

For implementation, a Control Lyapunov Function and Control Barrier Function was adopted.

- Control Lyapunov Function: $V(x) = \left(x - x_g\right)^2$where, x is the current state of the robot and x_g is the goal state.
- Control Barrier Function: $h(x) = (x - x_{obs})^2 - d^2$ where, x is the current state of the robot and x_{obs} is position of the obstacle and d is the minimum distance to be maintained from the obstacle by the robot.

The obstacle was placed at $x_{obs} = [1\,, 1]$ and the goal of the robot is set to $x_g = [2.5\,, 2.5]$. The robot motion was simulated for 300 timesteps where the Gaussian Process provides the dynamics of the robot for its motion. The optimization process chooses a valid control input u based on current state and the function $f(x)$ which is defined as:

$$f_{min}(x) = \left[f_{\min\,(1)}(x)\,, \quad f_{\min\,(2)}(x)\right] = [\mu_1 - \sigma_1\,, \quad \mu_2 - \sigma_2]$$
$$f_{min}(x) = \left[f_{\max\,(1)}(x)\,, \quad f_{\max\,(2)}(x)\right] = [\mu_1 + \sigma_1\,, \quad \mu_2 + \sigma_2]$$

The minimum and maximum values of $f(x)$ are used to find the maximum value of $L_f V$,

$$L_f V = \ max \left[\frac{dV}{dx} f_{min}(x), \quad \frac{dV}{dx} f_{ma}\ (x)\right]$$

to satisfy the constraint,

$$L_f V(X) + L_g V(X)\ u \leq -kV(x) + M\delta$$

and to choose the minimum value of $L_f h$,

$$L_f h = \ min \left[\frac{dh}{dx} f_{min}(x), \quad \frac{dh}{dx} f_{max}(x)\right]$$

to satisfy the constraint,

$$L_f h(X) + L_g h(X)\ u \quad \geq \ -\alpha h(x)$$

The constants were assigned values as such: $k = 0.3, \ \alpha = 0.5, \ M = 1000$

This system is called the GP-CLF-CBF-QP which was simulated on a moving robot. A video of the simulation is provided in the zip folder along with the code and this report. From the simulation, it is evident that the robot moves around the obstacle and tends towards the goal state in majority of the cases.

In some cases, the GP does not train optimally when the randomly chosen states used to train the GP are too close to each other. In these cases, the controller seems to manipulate the robot to entirely avoid the obstacle even if the goal is not reached. However, this simulation proves that for a non-linear system with unknown dynamics, a controller can be built that considers both safety and stability of the agent while a Gaussian Process estimates the dynamics of the system.

II. Journal Paper 2

Pointwise feasibility of gaussian process-based safety-critical control under model uncertainty

This paper proposes a Gaussian Process-based approach to tackle the problem of model uncertainty in safety-critical controllers that use Control Barrier Functions (CBFs) and Control Lyapunov Functions (CLFs). It also derives probabilistic bounds on the effects of model uncertainty on the dynamics of the CBF and CLF and use these bounds to build safety and stability chance constraints that can be incorporated in a min-norm convex optimization-based controller called the GP-CLF-CBF-SOCP. Additionally, the paper presents necessary and sufficient conditions for pointwise feasibility of the proposed optimization problem.

Finally, the proposed framework is validated with numerical simulations of an adaptive cruise controller for an automotive system.

To consider the effects of model uncertainty, this paper introduces the concept of a nominal model of a system. The difference between the true and nominal dynamics is used to train a Gaussian Process Regressor, which is then used to build an SOCP controller.

System Dynamics:

This paper considers a specific class of systems represented by unknown nonlinear control affine dynamics. These systems can be mathematically described as:

$$\dot{x} = f(x) + g(x)u$$

Here,

$x \in R^n$ = system state
$\dot{x} = derivative\ of\ x\ w.r.t\ time$
$u \in R^m$ = control input
$f(x), g(x) = functions\ that\ define\ the\ true\ dynamics\ of\ the\ system$

The CLF-CBF-QP is defined as the following optimization problem:

$$\begin{aligned} \min\ u^T u &= (u_1^2 + u_2^2 + \cdots + u_n^2) + M\delta^2 \\ such\ that\!: L_f V(X) + LgV(X)\,u &\leq -kV(x) \quad + M\delta \\ L_f h(X) + L_g h(X)\,u &\geq -\alpha h(x) \end{aligned}$$

Problem Formulation:

First a nominal model is assumed,

$$\dot{x} = \tilde{f}(x) + \tilde{g}(x)u$$

$\tilde{f}(x), \tilde{g}(x) = functions\ that\ define\ the\ nominal\ dynamics\ of\ the\ system$

Then design a locally exponentially stabilizing CLF (V) and CBF (h) based on the nominal model. The derivates of V and h for both true and nominal models are taken.

True Model:

$$\dot{V}(x,u) = L_f V(X) + LgV(X)\,u \quad and \quad \dot{h}(x,u) = L_f h(X) + L_g h(X)\,u$$

Nominal Model:

$$\dot{\tilde{V}}(x,u) = L_{\tilde{f}} V(X) + L_{\tilde{g}} V(X)\,u \quad and \quad \dot{\tilde{h}}(x,u) = L_{\tilde{f}} h(X) + L_{\tilde{g}} h(X)\,u$$

The errors of the nominal model-based estimates are found as:

$$\Delta v(x,u) = \dot{V}(x,u) - \dot{\tilde{V}}(x,u) \quad and \quad \Delta h(x,u) = \dot{h}(x,u) - \dot{\tilde{h}}(x,u)$$

Then the CLF-CBF constraints become:

$$\begin{aligned} L_{\tilde{f}}V(X) + L_{\tilde{g}}V(X)\,u + \Delta v(x,u) + kV(x) &\leq M\delta \\ L_{\tilde{f}}h(X) + L_{\tilde{g}}h(X)\,u + \Delta h(x,u) + \alpha h(x) &\geq 0 \end{aligned}$$

Both Δv and Δh have control affine structures, which requires an affine dot product compound kernel while performing Gaussian Process Regression, as explained in the upcoming section.

Gaussian Process Regression:

The Gaussian Process Regression in this paper uses very similar concepts from those discussed in the previous paper. However, here a new kernel is used to perform GPR called the Affine Dot Product Compound kernel, given by:

$$kc\left(\begin{bmatrix} x \\ y \end{bmatrix}, \begin{bmatrix} x' \\ y' \end{bmatrix}\right) := y^T Diag([k_1(x,x'), \ldots\ldots, k_{m+1}(x,x')])y'$$

Here, $kc\left(\begin{bmatrix} x \\ y \end{bmatrix}, \begin{bmatrix} x' \\ y' \end{bmatrix}\right)$ is the compound kernel of m+1 individual kernels $k_1, \ldots. k_{m+1}$ and y is given as $[1\ u^T]^T$. This compound kernel captures the appropriate structure of the target functions which results in a much better regression fit as compared to arbitrary kernels.

Using this compound kernel, the mean and variance of the GP prediction at a given query point (x^*, y^*) can be written as:

$$\mu_* = \underbrace{z^T (K_c + \sigma_n^2 I)^{-1} K_{*Y}^T}_{=:\, b_*^T}\ y_*$$

$$\sigma_*^2 = y_*^T \underbrace{\left(Diag\left(\begin{bmatrix} k_1(x_*, x_*) \\ \cdot \\ \cdot \\ k_{m+1}(x_*, x_*) \end{bmatrix}\right) - K_{*Y}^T (K_c + \sigma_n^2 I)^{-1} K_{*Y}^T \right)}_{=:\, C_*} y_*$$

Here, $K_c = \begin{bmatrix} K_{1*} \\ \cdot \\ \cdot \\ K_{(m+1)*} \end{bmatrix} \circ Y\ \ and\ K_{i*} = [k_i(x_*, x_1), \ldots\ldots, k_i(x_*, x_N)]$

Controller:

The GPR is used to find the mean and standard deviation of Δv and Δh. Finally, the GP-CLF-CBF-SOCP min-norm optimization problem looks like:

$$\begin{aligned} u^*(x) &= \text{argmin}\ \left|\left|\mathrm{u}\right|\right|_2^2 + M\delta^2 \\ \dot{\tilde{V}}(x,u) + \mu_v(x,u) + \beta\sigma_v(x,u) + kV(x) &\leq \delta \\ \dot{\tilde{h}}(x,u) + \mu_h(x,u) - \beta\sigma_h(x,u) + \alpha h(x) &\geq 0 \end{aligned}$$

To write these constraints in SOCP form, first the mean and variance of Δv and Δh is found from the GP:

$$\mu_V(x,u) = b_V(x)^T \begin{bmatrix}1\\u\end{bmatrix} \quad and \quad \sigma_V^2(x,u) = [1, u^T]C_V(x)\begin{bmatrix}1\\u\end{bmatrix}$$

$$\mu_h(x,u) = b_h(x)^T \begin{bmatrix}1\\u\end{bmatrix} \quad and \quad \sigma_h^2(x,u) = [1, u^T]C_h(x)\begin{bmatrix}1\\u\end{bmatrix}$$

The standard deviation can be written as:

$$\sigma_V = \left\| G_V(x)\begin{bmatrix}1\\u\end{bmatrix} \right\| \quad and \quad \sigma_h = \left\| G_h(x)\begin{bmatrix}1\\u\end{bmatrix} \right\|$$

Here, $G_V(x)$ is the matrix square root of $C_V(x)$ and $G_h(x)$ is the matrix square root of $C_h(x)$.

The standard form of SOC constraints for CLF is:

$$\|Q_V(x)u + r_V(x)\| \leq w_V(x)u + v_v(x)$$

Here,

$Q_V(x) := \beta G_{V[2:(m+1)]} \in R^{(m+1)\times m}$
$r_v(x) := \beta G_{V[1]} \in R^{(m+1)\times 1}$
$w_v(x) = \widehat{L_g V(x)} = L_{\tilde{g}}V(X) + b_V(x)^T_{2:(m+1)} \in R^{1\times m}$
$v_v(x) = \widehat{L_f V(x)} = L_{\tilde{f}}V(X) + b_V(x)_1 + kV(x) \in R$

The standard form of SOC constraints for CBF is:

$$\|Q_h(x)u + r_h(x)\| \leq w_h(x)u + v_h(x)$$

Here,

$Q_h(x) := \beta G_{h[2:(m+1)]} \in R^{(m+1)\times m}$
$r_h(x) := \beta G_{h[1]} \in R^{(m+1)\times 1}$
$w_h(x) = \widehat{L_g h(x)} = L_{\tilde{g}}h(X) + b_h(x)^T_{2:(m+1)} \in R^{1\times m}$
$v_h(x) = \widehat{L_f h(x)} = L_{\tilde{f}}h(X) + b_h(x)_1 + \alpha h(x) \in R$

Case Study – Adaptive Cruise Control:

To validate the proposed controller, the model of an Adaptive Cruise Control System is used, given by:

$$\dot{x} = f(x) + g(x)u$$

Here, $x = \begin{bmatrix}v\\z\end{bmatrix} \in R^2,\ f(x) = \begin{bmatrix}-F_r(v)/m\\ v_0 - v\end{bmatrix},\ and\ g(x) = \begin{bmatrix}1/m\\0\end{bmatrix}$

The state x is defined by v being the forward velocity of the ego car, and z being the distance of ego car to the front car. $u \in R$ is the control input which is the wheel force of the ego car. The front car is assumed to be at a constant velocity, $v_0 = 14\ m/s$, m is the mass of ego car, and $F_r(v) = f_0 + f_1 v + f_2 v^2$ is the rolling resistance acting on the ego car. The control objective is for the ego car to reach a goal speed while maintaining a safe distance from the front car. To implement the controller, a CLF is taken as $V(x) = (v - v_d)^2$ where $v_d = 24\ m/s$ is the goal velocity of ego car, and CBF is taken as $h(x) = z - T_h v$ where $T_h = 1.8\ s$ is the lookahead time.

The nominal model has the following parameters:

$$m = 1650\ kg, f_0 = 0.1, f_1 = 5, f_2 = 0.25$$

The true model has the following parameters:

$$m = 3300\ kg, f_0 = 0.2, f_1 = 10, f_2 = 0.5$$

First, the true model and the nominal model was simulated using a CLF-CBF-QP where the dynamics for both models was generated using the given functions $f(x)$ and $g(x)$. The result from this simulation compares the trend of the state $x = \begin{bmatrix} v \\ z \end{bmatrix}$ with time for the true model and the nominal model. Fig.2 shows the velocity of the ego car in m/s and the distance between the ego car and the front car in m, against the time.

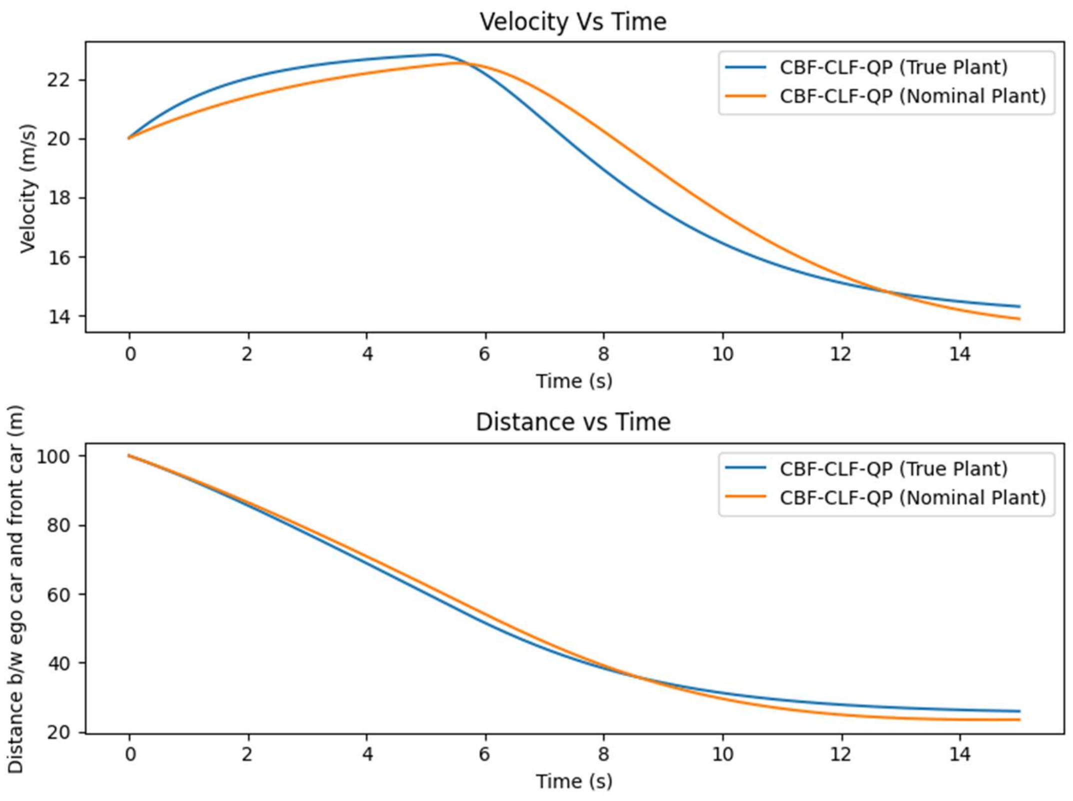

Fig. 2: Comparison of trends of True and Nominal Plant w.r.t time

From the plots in Fig. 2, the trend of the state in the nominal plant slightly defers from that of the true plant. In this simulation, the Gaussian Process is used to estimate the error between the nominal and true plant which is then used to build a controller.

To build a GP-CLF-CBF-SOCP controller, Gaussian Process Regression must be performed on Δv and Δh. For this, first random data must be sampled for velocity, v, distance, z, as well as control input u, since in this scenario, both the state and control input of the system is unknown and needs to be learnt using a GP. A set of 100 datapoints were uniformly sampled for $v, z,$ and u, out of which 50 points were used to train the GP and the remaining 50 were used to test it. Fig. 3 shows the how similar the true values are to the GP predicted values at the testing points. A confidence interval of 95% is also shown on the plot.

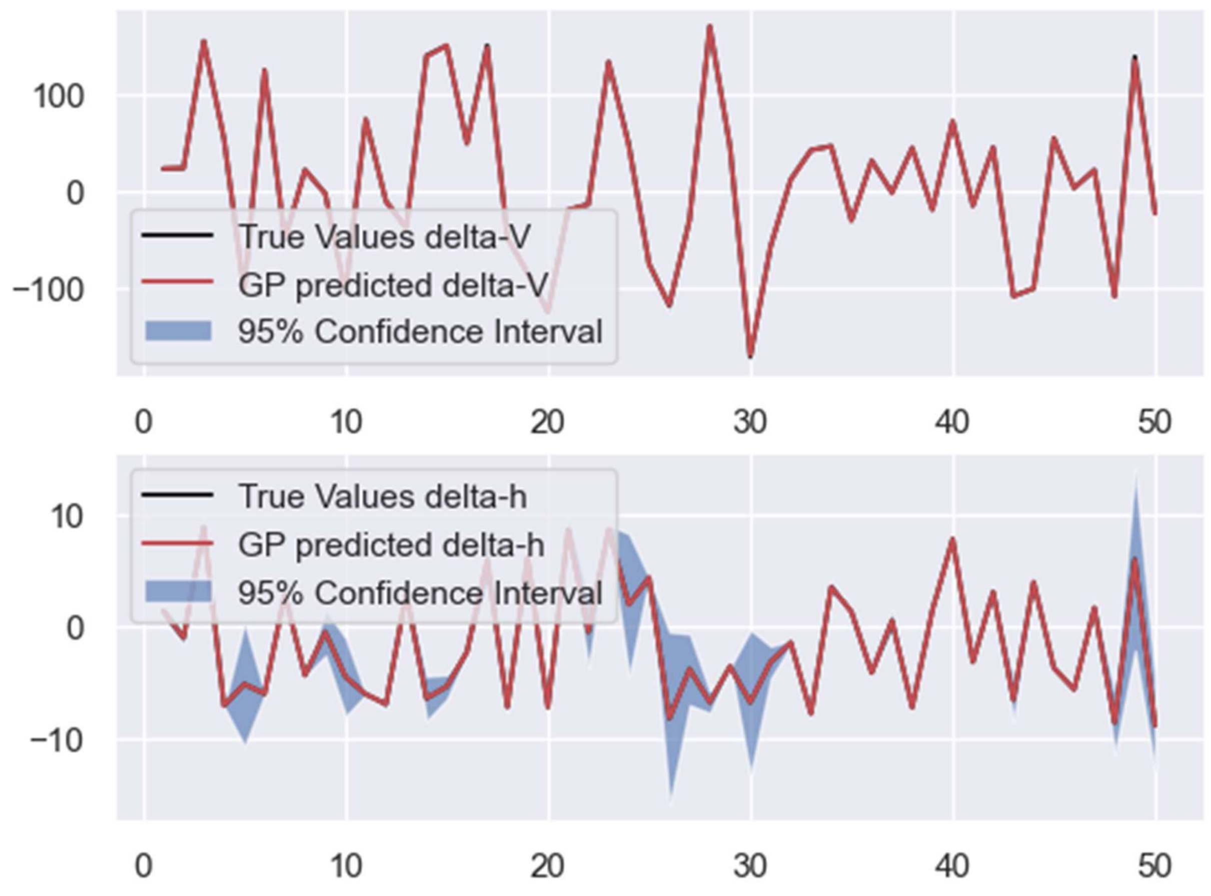

Fig. 3: Comparison of true values to prediction from Gaussian Process Regression

From Fig.3, it is evident that the GP predictions are very close to the true values for both functions. This means that the training and fit of the GPR was successful. Using this trained GPR, the controller was built for the Adaptive Cruise Control.

For the controller, first a GP-CLF-CBF-QP was built, and the simulation was run for 4000 timesteps. Fig. 4 shows trend of the state $x = \begin{bmatrix} v \\ z \end{bmatrix}$ with time for this algorithm.

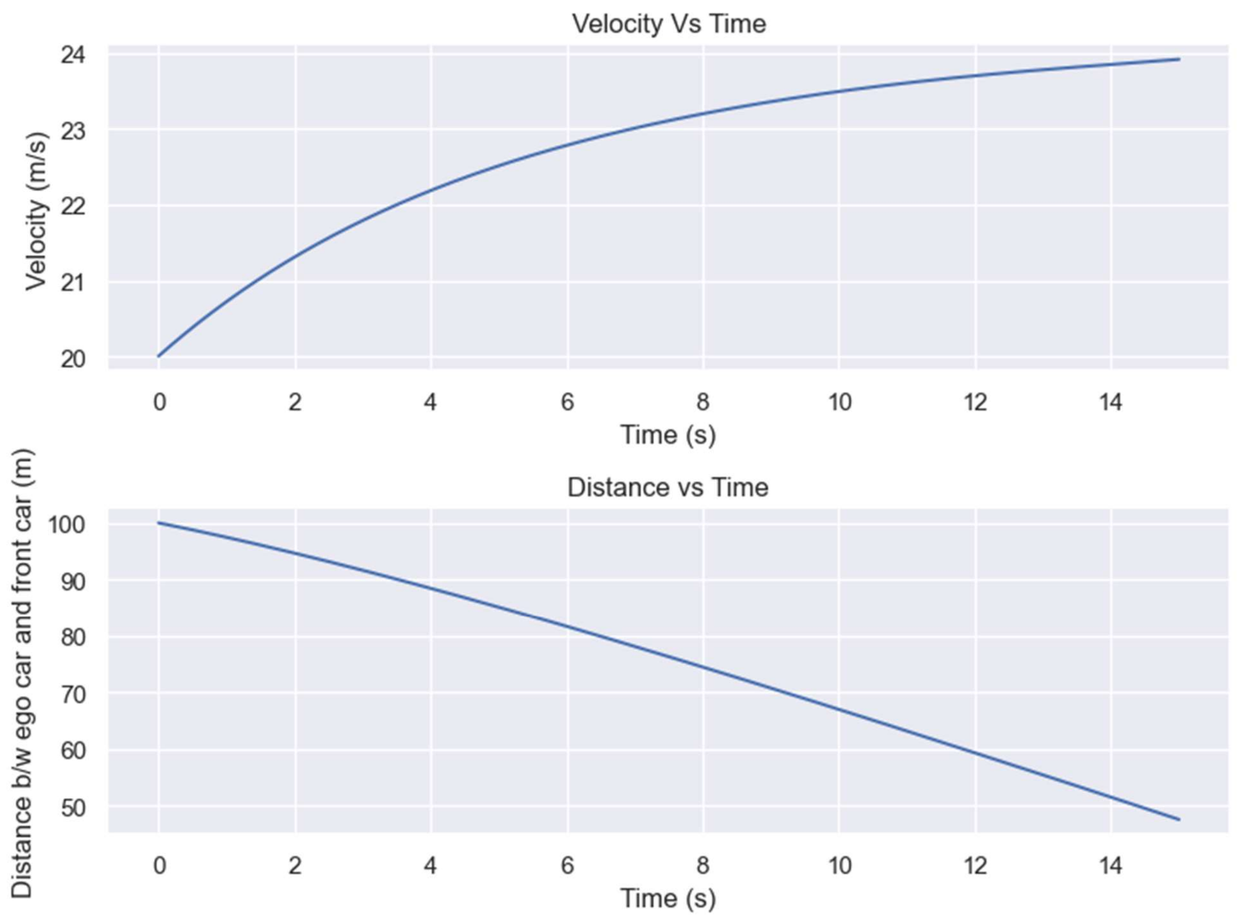

Fig. 4: Gaussian Process based CLF-CBF-QP controller results for state x

From Fig.4, the plot seems to be increasing until the goal velocity of 24 m/s while the distance between the ego car and the front car decreases. However, the time taken to reach the goal velocity seems to be significantly larger than the controller without GP.

Finally, to incorporate a probabilistic bound to the mean values obtained from the GP-CLF-CBF-QP, the SOCP constraints are introduced where the standard deviation from obtained from the GP is used in the controller. For this part of the simulation, the python library *cvxopt* was used to optimize a Second Order-Cone-Program. Fig. 5, shows trend of the state $x = \begin{bmatrix} v \\ z \end{bmatrix}$ with time for this algorithm. However, the results did not seem promising. The solver was not able to solve the SOCP problem. This could be due to an issue with the implementation of the solver with respect to the problem.

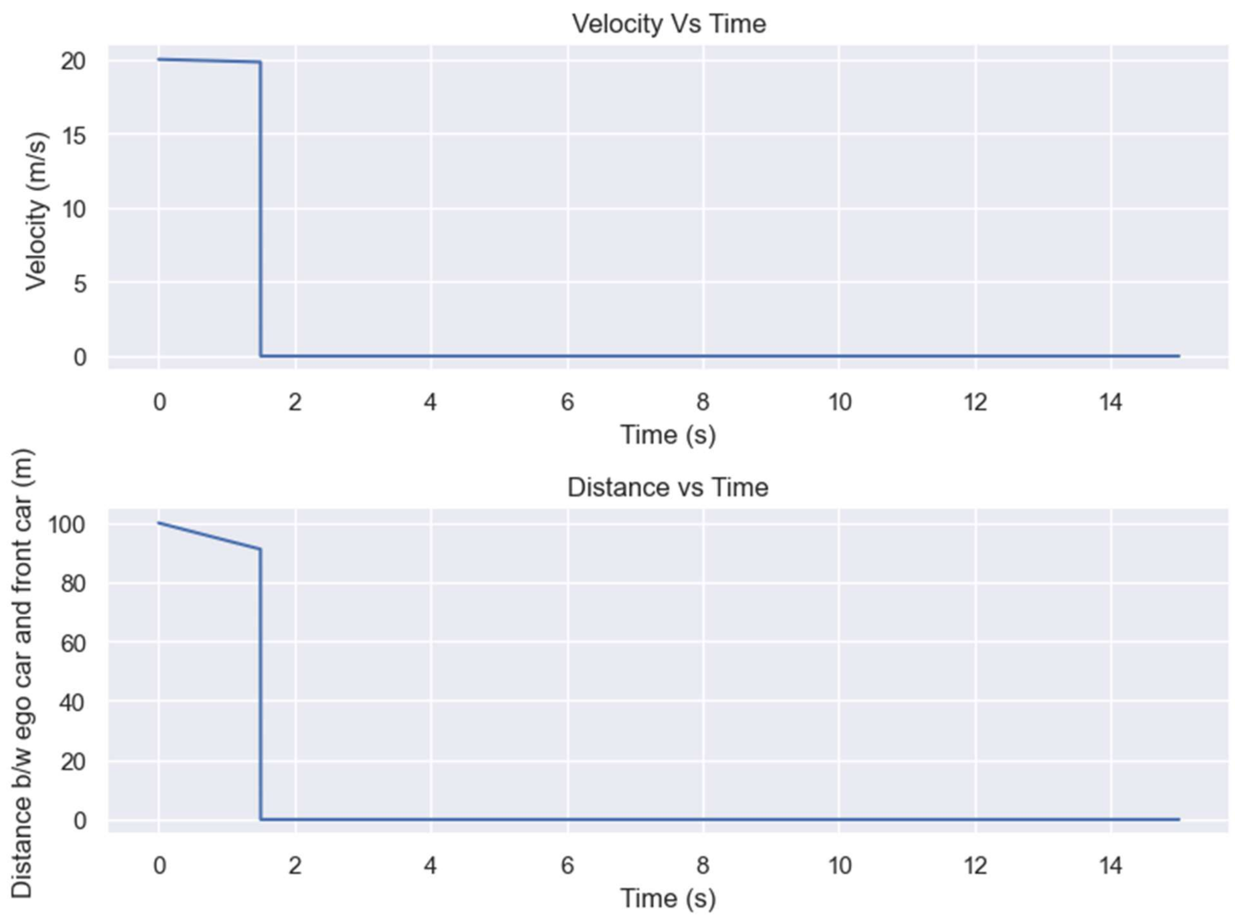

Fig. 5: GP-CLF-CBF-SOCP for Adaptive Cruise Control System

Ideally, the plot in Fig. 5 is supposed to match exactly with the CLF-CBF-QP plot for the true plant. However, implementing an SOCP controller seems to be more complicated than expected and due to lack of time, this issue could not be explored further.

D. Conclusion

For the scope of this project, I would like to focus on the implementation of the following concepts:

- Controller using a Quadratic Program with linear inequality constraints
- Control Lyapunov Function: mathematical formulation and application in controller
- Control Barrier Function: mathematical formulation and application in controller
- Gaussian Process Regression using inbuilt python libraries
- Gaussian Process Regression using Affine Dot Product Compound Kernel
- Mathematical formulation of Second Order-Cone-Program

Most of the above-mentioned concepts were very new to me when I first started this project and I believe working on this directed study has helped me improve my knowledge in these areas significantly.

E. References

[1] Jagtap, P., Pappas, G.J. and Zamani, M. (2020) ‘Control barrier functions for unknown nonlinear systems using Gaussian Processes’, *2020 59th IEEE Conference on Decision and Control (CDC)* [Preprint]. doi:10.1109/cdc42340.2020.9303847.

[2] Castaneda, F. *et al.* (2021) ‘Pointwise feasibility of gaussian process-based safety-critical control under model uncertainty’, *2021 60th IEEE Conference on Decision and Control (CDC)* [Preprint]. doi:10.1109/cdc45484.2021.9683743.