Dynamic Gaussian Process Models for Model Predictive Control of Vehicle Roll

by

David J. Broderick

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
May 7, 2012

Keywords: Gaussian Process, Model Predictive Control, Stability Control

Copyright 2012 by David J. Broderick

Approved by

John Y. Hung, Co-Chair, Professor of Electrical and Computer Engineering
David M. Bevly, Co-Chair, Professor of Mechanical Engineering
Bogdan M. Wilamowski, Professor of Electrical and Computer Engineering

Abstract

The machine learning method Gaussian process (GP) regression is used to learn the vehicle dynamics without any prior knowledge. The model formed with GP regression demonstrates characteristics that make it useful in model predictive control (MPC). Previous applications of model predictive control used linearized models to balance the need for fast computation and predictive accuracy. This work aims to make nonlinear predictions in a timely manner. A method of clustering the training data is also developed in order to further speed calculation of dynamic predictions necessary for MPC. An architecture to take advantage of the nature of GP regression calculations is then developed. The efficacy of the approach is examined through training and validation using data recorded from an instrumented all-terrain vehicle (ATV) and through a series of simulations.

The instrumented all-terrain vehicle allowed GPS, inertial, and encoder data to be used for training and validation of the GP-based dynamic model. A series of maneuvers approximating sinusoids of varying frequency and amplitude is used to excite the vehicle for the purpose of generating a training dataset. That training dataset allows GP regression to learn the functions describing a nonlinear state-space model of the vehicle dynamics. The data recorded during a separate set of maneuvers is used to validate the predictions generated from the GP-based model. The speed of computation is examined and methods of speeding the nonlinear portion of GP regression are presented. The result of this is a model that can provide accurate estimates in a timely fashion. The speed of computation is sufficient to allow for MPC to be implemented on a modest, contemporary desktop PC.

A method of further decreasing computation time is then presented. Clustering of training data has been mentioned in the literature though specifics regarding the choice of algorithm and parameter selection are conspicuously omitted. Clustering has been applied in the closely related method of support vector machine classification where similarity is easily defined. A definition of similarity is offered here for the purposes of regression. An algorithm is selected to leverage that definition of similarity. Parameter selection is addressed by basing the cluster size on the characteristic length scale of the function being regressed. Evaluation of the clustering algorithm and parameter is performed on two illustrative examples as well as the GP model formed from the recorded ATV data.

Acknowledgments

I would not have been successful in my studies without the help of many people. Even though it feels like I am finishing my time at Auburn with a wheel in the ditch and a wheel on the track I am grateful for the support of the faculty, my peers, and my family.

I would like to thank the members of my committee, Professors Hung, Bevly, and Wilamowski for their attention and guidance throughout my studies. Additionally, I must thank Professor Beale for his flexibility and kindness. I believe it is imperative that we be surrounded with people who help us improve ourselves. All of you have exceeded that standard through your commitment to my education.

From my early days with Nathan Loden to working on the Explorer with David Hodo, I learned as much from my peers as I learned from the faculty. I was always aided by my conversations with the boys in "The Deuce", Jon Ryan, Lowell Brown, and Ryan Hill, all of whom directly aided me in completing this work. I would be remiss to forget my work with Jordan Britt. It was the most productive distraction I had from graduating. Finally, Christopher Wilson had the dubious honor of sharing an office with me for too many years. Chris, for all of the chicken and conversation, thank you.

My family was also a source of support in a number of ways. I could not ask better parents. They taught me the value of learning and then supported me in every way they could. My brother Daniel offered his support in a number of unique ways as well. One notable effort on his part includes:

There once was a man from Connecticut
Who lacked Southern charm and etiquette.
    Watching models emerge,
    Drove him to the verge.
All for a post-nominal "doctorate".

With family like that, it’s hard to believe I could ever fail.

Finally, my wife Jennifer allowed me to drag her to Alabama and endured much of this long road with me. Jenn, I would not want anybody else by my side. I enjoy every nugget of time we share together.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ANN	Artificial Neural Network
ATV	All-Terrain Vehicle
BLAS	Basic Linear Algebra Subprograms
DLC	Double Lane Change
EKF	Extended Kalman Filter
ESC	Electronic Stability Control
FH	Fishhook
FHWA	Federal Highway Administration
GP	Gaussian Process
GPU	Graphical Processing Unit
IIHS	Insurance Institute of Highway Safety
IMU	Inertial Measurement Unit
LQR	Linear Quadratic Regulator
MAC	Model Algorithmic Control
MPC	Model Predictive Control
MPC	Model Predictive Control
NHTSA	US National Highway Traffic Safety Administration

QDMC	Quadratic Dynamic Matrix Control
RI	Rollover Index
RSC	Roll Stability Control™
SbW	Steer-by-Wire
SVM	Support Vector Machine
UAV	Unmanned Air Vehicle
UGV	Unmanned Ground Vehicle

Chapter 1

Introduction

Driving a vehicle is a complex task. It involves many of our senses and requires us to predict the behavior of our own vehicle as well as the behavior of others with whom we share the road. The majority of the time we negotiate these complexities without major incident. However, when we encounter unfamiliar situations advances in safety technology can act to preserve human life and mitigate financial damages. This work aims to further those advances using a unique modeling technique and associated controller.

Typical drivers are inexperienced with regards to executing emergency maneuvers and risk rolling the vehicle over. Electronic stability control (ESC) and roll prevention have emerged as crucial pieces of a passenger vehicle's safety system. Vehicle rollover accounted for 35% of crash fatalities in 2008 according to the US National Highway Traffic Safety Administration (NHTSA), a disproportionate amount considering rollover only occurs in 3.2% of accidents[1, 2]. In 2006, the Insurance Institute of Highway Safety (IIHS) estimated that one-third of fatal rollovers could be prevented. This would result in an estimated 10,000 fewer deaths annually[3]. This is roughly equivalent to saving the lives of a sold-out crowd in Auburn Arena or the entire population of Valley, Alabama each year. The implications of ESC technology, with regards to the preservation of human life, are profound.

Savings lives is important by itself but from the cold, mathematical perspective of the financial cost of rollover accidents, further motivation can be found. NHTSA estimated the total cost of vehicle accidents in the United States was 230.6 billion dollars[4]. Rollover accidents are among the most destructive to the vehicle and, as already discussed, the most deadly. An example of the destructive nature of rollover in shown in Figure 1.1. The Federal Highway Administration (FHWA) placed the cost associated with each fatality on the road at 2.6 million dollars[5]. We

Figure 1.1: The consequences of vehicle rollover.

must also note that these statistics describe the financial and human cost for rollover accidents in the United States but physics and tragedy do not recognize borders. Therefore, it is reasonable to conclude that the costs of rollover are greater when international accidents are included. It is also easy to conclude that, with rollover accounting for a large percentage of fatalities, the improvements to ESC technology will significantly impact the financial cost of rollover accidents.

1.1 Prior Works

1.1.1 Stability Control

These numbers paint a grim picture, but there is hope. ESC technology has already begun to reduce these costs and, with improvements, can mitigate further losses. Currently available ESC systems brake individual wheels to steer the vehicle when a loss of control is detected as described in [6]. This type of ESC system uses the steer angle to determine a desired yaw rate and then acts to control the actual yaw rate by braking individual wheels. Recently, Ford Motor Company presented a method to control the roll of the vehicle directly with the addition of an extra gyroscope to measure the vehicle roll rate[7]. That method has been named and trademarked by

Ford Motor Company as Roll Stability Control™(RSC). The authors of that work assert that the additional sensor is warranted to detect or predict rollover conditions directly rather than relying on the previous assumption that errors between desired and actual yaw rate will result in rollover. Such assumptions, they state, leads to “necessary trade-offs between control sensitivity and robustness”. This compromise can be avoided by controlling the roll angle of the vehicle directly. Asserting control by differential braking is an effective method to control vehicle yaw and roll, however, it also undesirably reduces vehicle speed.

Removing mechanical linkage between the user controls and associated components of the vehicle is not unheard of. Throttle-by-wire, where the pedal position is sensed and electronically transmitted to the throttle, has been used on vehicles for a number of years. More recently, interest has begun to grow in steer-by-wire (SbW) systems. Actuating steering in this way allows for stability control to be performed without the need to slow the vehicle. SbW will therefore allow greater control to be retained by the driver, in regards to longitudinal speed, while still providing the safety realized by more conventional ESC systems. Some car manufactures are beginning to build SbW vehicles for experimentation and, as such, research has begun on implementing ESC with this new capability. Working ESC using SbW has been implemented in [8], [9], and [10]. The allocation of control effort is examined with regards to mixed systems, SbW and differential braking, in [11]. Yaw control of a vehicle for ESC was implemented in [12] using all-wheel braking and SbW. A study was conducted of what vehicle factors influence rollover in [13] and offered guidance in placing constraints on the MPC ultimately used.

The above approaches rely on the accurate detection of conditions that indicate roll over is imminent. There is evidence that suggests that this may not be the most effective control strategy. Expert drivers are better able to predict the behavior of the vehicle and make adjustments before conditions indicate that rollover is imminent[14]. This ability has been shown to benefit ESC systems as well as human drivers.

Use of model predictive control (MPC) has expanded in recent years given the corresponding increase in computation speed. MPC has been applied to ESC in [15], [16] and [17], all showing that a controller's ability to predict undesirable conditions early can allow for a more effective response. The ability of MPC to handle constraints explicitly allows a great deal of flexibility in how the controller is implemented. In addition, it is possible to dictate how control is allocated between driver and controller for different situations.

1.1.2 Modeling Issues

Model simplifications make MPC a viable control method for systems requiring faster sampling times, but these simplifications can degrade controller performance. Several works address model simplification beyond simply linearizing the model about an operating point. Linearizing the model during online operation is presented in [18] for a lateral MPC vehicle controller. This approach stabilized the vehicle's lateral dynamics but did not perform well outside the linear region of operation of the tire model as shown in [16, 19]. The difficulties introduced by the tire model were addressed differently in [20]. Rather than linearizing the tire model (offline or online) a simplified model was used in an effort to reach a comprise between controller performance and computation speed. Simulated comparison of this approach against a similar approach with the more complex Pacejka tire model in [21] shows that the more complex model outperforms the simplified model. Another method for simplifying the tire models used for MPC was demonstrated in [22]. The rear tire model was linearized online about the operating point while the nonlinearities in the front tire model were "extracted" to remove the computational burden placed on MPC during optimization of the control effort. This extraction was performed using MPC to find an optimal value of lateral force on the front tire. The front lateral tire force was then used, along with a nonlinear tire model, to find and command the associated steer angle. The resulting method preserves the convexity of the optimization problem solved by MPC allowing linear optimization methods to be applied. These works demonstrate that model fidelity and computation time remain a difficult trade-off to make.

1.1.3 The Gaussian Process Approach

The trade-off between the accuracy and speed of calculations is as old as computation itself. While it can not be avoided, there exists an interesting method of addressing it. Rather than using a parametric tire model, the nonlinearities required for calculation can be incorporated into a Gaussian process (GP) model regressed from recorded data as described in [23]. Three clear advantages to this method present themselves readily. First, nonlinearities can be modeled to an arbitrary accuracy. A GP model shares many similarities to a radial basis function network and can be rightfully described as a universal approximator. Cybenko introduced the idea that with the superposition of an arbitrarily large number of sigmoid functions a nonlinear function could be approximated to an arbitrary accuracy [24]. Hornik extended the use of Cybenko's work stating that it was the superposition of the functions that allow for universal approximation, not the nature of the sigmoid [25]. Where neural applications discuss this requirement in terms of an infinite amount of neurons, GP applications treat the requirement by speaking of an infinite number of regressors. While computation using this method seems infeasible, GP regression also provides a frame work for marginalizing regressors to make this a practical method. This greater accuracy, of course, can't be free of additional computation but the additional computation required is performed in a higher dimensional space where only linear operations are required.

An additional advantage also exists. The complexity of calculating predictions scales linearly making it feasible to calculate a large number of predictions very quickly [26]. If this still does not satisfy the requirements of a particular system, there are a number of works that offer further reduction of computational burden. The subset of data method, described in [27], suggests that some data points be removed prior to regression thus reducing computation. The method of selecting which points are retained has a great effect on model performance in this case. That work also suggest that, once a quantity of points is determined, the set of points that maximizes the differential entropy be selected which is a measure of average information content. Another method of selecting the points calls for the minimization of the Kullback-Leibler divergence which is a measure of information gain [28]. Both approaches discard data that may otherwise be used to improve model accuracy. The subset of regressors method retains all data points but only uses a subset to calculate each prediction. The matter of which regressors to use is again an issue and this method also inaccurately reports covariances [29].

Reduction of computation is treated with a clustering method in [30]. That work shows that clusters of data points can be described by averaging both inputs and outputs and using the cluster center points for GP regression. The computation versus accuracy compromise can then be managed directly through the adjustment of the clustering threshold. That work demonstrates the technique by modeling the lateral dynamics of a vehicle. While sensitivity to the cluster threshold is discussed no guidelines for its selections were offered.

Last, a GP model provides a measure of confidence in each prediction in the form of a variance estimate. The variance calculation does not scale as nicely as the prediction calculation, $\mathcal{O}(n^2)$, but offers a route to implement robust control. MPC requires multi-step prediction. The propagation of prediction uncertainty is addressed by [31] and [32] demonstrating that it is possible to determine prediction certainty for a distant horizon prediction.

Kocijan et al. discussed GP models for use in MPC [33]. That work lays the general framework for making the multi-step predictions needed by MPC as they apply to a chemical process. This is a typical application of MPC with slow sample rate placing little demands on the computation time. Several issues are left open by that work including method of computation reduction, optimization technique used to find the control effort, and stability of the closed-loop system. Some of these issues are addressed by [34]. That work incorporates the derivative of predictions and variances into the optimization technique, values that are easily calculated from the GP model [35]. However, that work stops short of using the GP model prior to considering computation-reducing measures and opts for local linear models instead. Simulated results for a simplified first-order vehicle model are solely used in that work.

1.2 Contributions

Gaussian process models provide an alternative to the simplified models previously used to implement nonlinear MPC. GP regression accurately approximates nonlinear functions and offers a variety of methods to reduce the computation required for this approximation. Therefore, the use of this model was investigated to determine its ability to this end and also to identify constraints appropriate to roll stability that allow for efficient computation. Further gains were made by considering a computation-reducing method and an appropriate optimization method was identified. This approach to MPC offers a path to remove the limitations, inherent to simplified models, on MPC-based roll stability of vehicles allowing the driver to retain as much control as possible.

Many issues were addressed to demonstrate the success of the method. MPC's sensitivity to model accuracy requires attention be paid to the formation and simplification of the model. The choice of optimization technique is affected by different model structures and was explored. The ability to implement constraints explicitly in an MPC opens discussions regarding what constraints are appropriate. With prediction and computation speed addressed using data recorded from a vehicle, a simulated controller is demonstrated to support to controller's ability to constrain the dynamics of the vehicle. Four specific contributions of this dissertation are detailed below:

i) Development, identification, and validation of GP model for vehicle Roll appropriate for use in MPC

Modeling of a vehicle's lateral dynamics in [30]. The same principles are used to model the roll dynamics of a vehicle in this dissertation. An autonomous all-terrain vehicle (ATV) is used to train and validate the model for use in the MPC controller. The maneuvers used in simulation were executed by the ATV autonomously allowing the dynamics to be excited near the unstable region of operation (i.e. close to vehicle to rollover). Identifying the model in this region is desirable to predict an imminent vehicle rollover. The vehicle, available from the GPS and Vehicle Dynamics Laboratory is shown in Figure 1.2. With an effective MPC, the vehicle will be prevented from

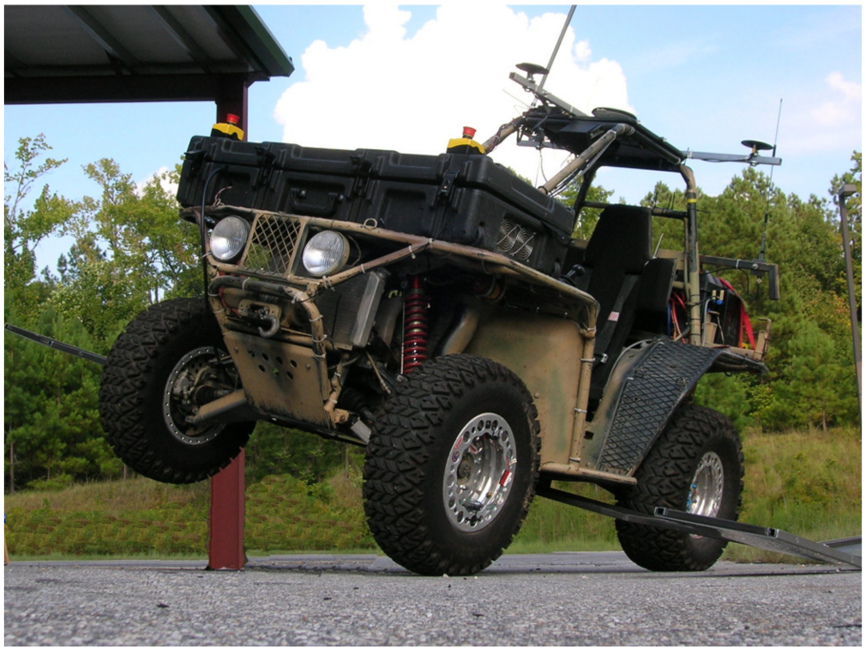


Figure 1.2: Autonomous Prowler ATV.

revisiting this state. Given the need for multi-step prediction, the GP model was evaluated for use in this way. Propagation of prediction uncertainty are implemented in the event that a robust implementation becomes desirable.

ii) **Architecture to incorporate GP dynamic model into MPC**

The GP dynamic model natural has strengths and weaknesses regarding its use in MPC. A method of performing the nonlinear constrained optimization as the heart of MPC is employed. It is shown that, due to the nature of the calculations necessary to generate predictions, it is feasible to perform an exhaustive search of the space defined by the input to the system. The details of required sensors are also described. A multi-antenna GPS receiver, from Septentrio, was used in conjunction with a Crossbow inertial measurement (IMU) to provide measurements of vehicle roll, roll rate, sideslip, and yaw rate. Steer angle was recorded from a potentiometer on the steering shaft of the vehicle during identification. The performance of the controller was significantly affected by the constraints placed on the dynamics and the method of optimization was suggested and evaluated. for demonstration and data collection.

iii) **Evaluation of large dataset approximation techniques as they pertain to GP based MPC**

As discussed previously there are many works addressing the reduction of computation in GP models. Clustering of training data has proven effective in classification problems where similarity is easily defined. However, defining similarity and using it to cluster in regression had not been addressed. The method used in [30] has been shown to be effective but little guidance is provided regarding the cluster threshold. A method of selecting this value based on the characteristics of the underlying function is presented here. The method of selection was evaluated for its ability to reduce computation while preserving accuracy. Results show that the selected threshold value is applicable to single and multiple dimensional problems as well as for models of dynamic systems.

iv) **Implementation and demonstration of GP based MPC roll stability controller on autonomous ATV**

Finally, the end product of this work was demonstrated using the CarSim suite of vehicle modeling tools. A series of maneuvers that are known to produce vehicle rollover were demonstrated without constraints. Then, with the GP based MPC being used to constrain the commanded steer angle, the same maneuvers were simulated. The results of these simulations show that the dynamics are indeed constrained for two common maneuvers used in ESC testing.

Chapter 3 addresses the model accuracy and speed of necessary calculations for the use of a GP model in MPC. Both of these aspects of the GP model are explored using data recorded from an ATV using GPS and IMU measurements. Chapter 4 examines clustering as a means to reduce the necessary computation for the GP model. A method of selecting a clustering method and associated algorithm parameters is presented. One such clustering method is applied to one and two dimensional problems to demonstrate its efficacy in an easily visualized manner. Next, Chapter 5 presents a manner of using the GP model in MPC in such a way that leverages the advantages of that model. Results of two evasive maneuvers known to induce vehicle rollover were gathered from simulation and are presented here. Finally, the final chapter includes a summation of results and conclusions are drawn given the performance of the GP model in MPC.

Chapter 2

Background

This work brings together two distinct areas of study and, as such, background is provided here for those topics in two separate sections. This chapter provides a brief historical perspective of model predictive control is given prior to a general statements of model predictive control (MPC) for the linear and nonlinear cases. Gaussian process regression is used in this work to learn the dynamics of a passenger vehicle. The method of learning a function with a GP is described and a number of prior applications in the fields of robotics and controls are provided.

2.1 Model Predictive Control

Prediction has long aided control systems in nature. A predator predicts the movement of its prey to affect its capture with the minimum necessary energy. A driver predicts traffic patterns to minimize travel time. A writer predicts the response of his readers to maximize the likelihood of his graduation. It is only since the 1960's that predictive control has begun to gain favor for Man-made control systems.

Distant horizon control is certainly not a new concept. Kalman's Linear Quadratic Regulator (LQR) considers controlling with an infinite horizon as early as 1960 [36]. Constraints were not considered at that time and the limitations of the linear formulation limited the scope of the method.

In 1978 Richalet et al. presented Model Algorithmic Control (MAC) [37] and in 1980 Cutler and Ramaker presented Dynamic Matrix Control [38]. These works used the impulse and step response of a system, respectively, to develop a predictive model for use in the controller. Stability was not considered so applications were limited to stable plants with very distant horizons. Garcia (1986) used quadratic programming to solve the optimal control problem [39] and handled constraints explicitly in Quadratic Dynamic Matrix Control (QDMC).

MPC had been adopted by industry for application in chemical process control but stability analysis proved elusive. Sistu proved stability of a discrete MPC in [40]. Stability over the finite-horizon has been addressed by a multitude of works and has been summarized nicely in [41]. Stability of a system controlled by MPC can be sufficiently proven using the axioms detailed in those work.

2.1.1 General Form of a Model Predictive Controller

A dynamic model is used to predict the system's open-loop response to control inputs over a finite horizon. MPC is defined from the control input that optimizes some cost function,

$$J = \sum_{k=1}^{N} w_{z_k}(\mathbf{r}_k - \mathbf{z}_k)^2 + \sum_{k=1}^{N} w_{u_k}\delta u_k^2 \tag{2.1}$$

with respect to a reference trajectory, r_k, and the predicted state, z_k, over the finite horizon, d. Weights w_{z_k} and w_{u_i} are left to tune the controller and may be vector quantities to account for multiple states or inputs. Constraints can be placed on the state and inputs explicitly defined by the inequality

$$C(\mathbf{z}_k, u_k) < 0 \tag{2.2}$$

and can be either linear on nonlinear.

The procedure implemented by the MPC at each time step can be generally stated as:

1. Generate reference trajectory, $\mathbf{r}_k$, over a finite horizon of N steps.

2. Find the optimal input to the system that minimizes the cost function in (2.1) as a function of the predicted state, $\hat{\mathbf{z}}_k$, error and control effort. This optimization is subject to the constraints described in (2.2).

3. Apply the first control effort in the set of inputs to the system.

While only open-loop predictions are computed, a closed-loop control is formed by recomputing the optimal input to the system at each time step. MPC is often referred to as receding horizon control given that the distance to the horizon is fixed and the location is relative to the current time step.

Given that the controller performs the optimization of (2.1) online, adoption of MPC has been limited to linear systems or those with dynamics which are sufficiently "slow". Linear optimization can be performed quickly but the accuracy of the model may be insufficient if the linear model is used to approximate some nonlinear system. This has led to the adoption of MPC in chemical plants and other systems where a low sampling frequency will suffice which allows time for an optimal solution to be calculated.

2.1.2 Nonlinear MPC

The predictive model can take a multitude of forms. A number of works have discussed the model forms appropriate for learning the dynamics of a system. Kocijan adopted an autoregressive from in [42] for MPC Control of a chemical process. A similar autoregressive approach was used in [30] for the prediction of lateral vehicle dynamics excluding any roll dynamics. Wang describes a state-space model structure in [43] and Ko adopted a similar model structure in [44] to model the dynamics of an autonomous blimp.

A generalized statement of the state-space model similar to those in [43, 44] is suitable to describe the system dynamics as nonlinear systems are considered:

$$\begin{aligned} \mathbf{z}_{k+1} &= G(\mathbf{z}_k, u_k) \\ y_k &= H(\mathbf{z}_k, u_k) \end{aligned} \tag{2.3}$$

and a block diagram can be found in Figure 2.1.

Linear optimization is no longer an option to determine the optimal control input once a non-linear is considered. Many works have focused on how optimization can be performed quickly with nonlinear dynamic models. Some approaches include online linearization about the current operating point (x_i) [18], making piece-wise linear approximations of the system dynamics [20],

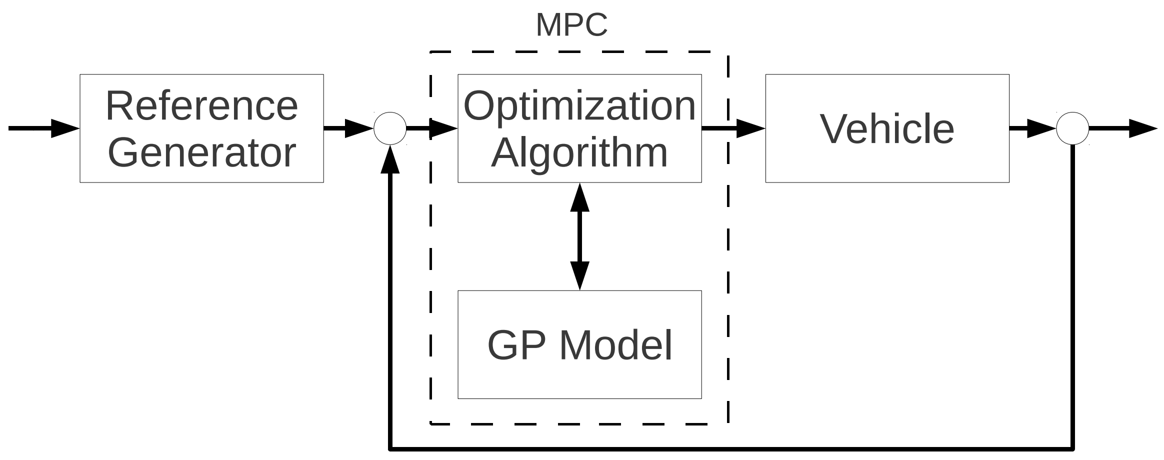


Figure 2.1: MPC controller architecture.

and separating the linear and non-linear parts of a model to perform optimization on only the linear part [22]. All of these contrivances serve to speed up the optimization of the cost function with respect to the control input. However, each method has the undesirable side-effect of compromising model accuracy. The success of MPC control is closely tied to the internal models ability to accurately predict the response of the system. Compromising model accuracy will therefore result in suboptimal control and therefore degrade system performance.

The trade-off between speed of optimization and model accuracy will determine whether a particular system can be controlled by MPC. The open-loop predictive capability of the model must be shown to be sufficiently accurate over the finite horizon. However, accurate predictions and the optimization associated with them come at a cost, namely computation time. An accurate model will not be sufficient if optimization can not be performed within the sampling period of the controller. Therefore, a model must make accurate predictions and allow for optimization within the sampling period of the controller to be suitable for use in an MPC controller.

2.2 Gaussian Process Regression

The field of machine learning is beneficial to developing advanced autonomous robotic platforms. Using machine learning techniques in the design of robotic platforms can improve a robots ability to navigate accurately, learn a specific motion or task, or infer missing information about the environment it is in. The use of Gaussian process regression is an emerging technique in machine learning that can aid in making each of these improvements and surpasses the ability of more conventional techniques.

While functionally mimicking neural networks, Gaussian process regression surpasses the ability of a neural approach in many respects. A Gaussian process approach simplifies model selection, copes with measurement noise, and avoids many pitfalls common to neural network training. In addition to addressing these common difficulties of a neural approach, GP regression also provides a variance associated with each estimate without much additional computation.

Here, an overview of the fundamental calculations is provided highlighting the benefits and difficulties of using Gaussian process regression. Three applications of Gaussian process regression are provided as a review of GP applications in related fields.

2.2.1 Calculation of Estimates and Variances

GP regression was introduced in [23] and a more thorough treatment of GP models can be found in [26]. Gaussian process (GP) regression has been shown to accurately model nonlinearities in the presence of measurement noise allowing for an estimate to be developed for a nonlinear function including the noise characteristics inherent to the sampling method. GP regression removes the need to be familiar with the structure of the function by learning the nonlinearities from sampled data. Additionally, GP regression incorporates the noise present on the sampled data to develop a confidence interval for the inferred output.

GP models outperform traditional closed-form models and more widely accepted machine learning techniques such as artificial neural networks (ANN). Much attention has been paid to ANNs to learn functions but they fail to account for measurement noise and suffer from over-fitting resulting in a neural model that poorly describes the general behavior of the function. GP regression mitigates the risk of over-fitting by adopting a Bayesian approach to learning. Additionally, a simpler model structure makes the learning method simple to implement and, therefore, able to consistently outperform a comparable neural network models. ANN models also require model selection to be performed much in the way a closed-form model would be selected though ANN models are less sensitive to errors in selection. Neural architecture requires a great deal of attention where as the model structure adopted by GP model allows for the most likely model, in a Bayesian sense, to be selected thereby making the solution reliable with respect to generally modeling the function.

2.2.2 Estimating a Function with a Gaussian Process Model

The GP models the function from input-output pairs recorded from the function, the input being defined as x and the output as y. The input-output pairs are used to construct a covariance matrix using the function

$$k(\mathbf{x},\mathbf{x}') = \sigma_f^2 exp\left[\frac{-(\mathbf{x}-\mathbf{x}')^2}{2l^2}\right] + \sigma_n^2 \Delta(\mathbf{x},\mathbf{x}') \tag{2.4}$$

where the values $\left[\sigma_f, \sigma_n, l\right]$ are termed the hyperparameters to distinguish them from parameters of the function being modeled. Here, Δ indicates the Kronecker delta function.

The matrix is constructed by iterating the values for i and j from 1 to n as in:

$$K(i,j) = k(\mathbf{x}_i, \mathbf{x}_j) \tag{2.5}$$

where n is the number of input-output pairs in the training data set.

The optimal values of the hyperparameters are determined by maximizing the marginal likelihood

$$\log p(\mathbf{y}|X) = -\frac{1}{2}\mathbf{y}^T K_y^{-1}\mathbf{y} - \frac{1}{2}\log\left|K_y\right| - \frac{n}{2}\log 2\pi \tag{2.6}$$

over the training data set using a line search with a maximum of one hundred function calls. The selection of hyperparameters constitutes the problem of model selection for a GP model. Neither, the closed-form equation nor the neural architecture needs to be determined. The search for these hyperparameters is considered the training of the model equivalent to determining parameters in a closed-form model or weights in a neural model. A closed-form model will be more sensitive to a poorly selected model structure. A neural model requires the unreliable search of a high-dimensional space to determine neuron weights. By reducing the dimensions of the search space during training GP models do not suffer from problems caused by local minima as GP training reliably converges to hyperparameters representing an acceptable model.

Estimates can be generated once the optimal hyperparameters are known. Given a point within the input-space, $\mathbf{x}_*$ indicating the location of a desired estimate, a vector is generated as

$$K_* = [k(\mathbf{x}_*, \mathbf{x}_1), k(\mathbf{x}_*, \mathbf{x}_2), ..., k(\mathbf{x}_*, \mathbf{x}_m)] \tag{2.7}$$

and the auto covariance is also found using,

$$K_{**} = k(\mathbf{x}_*, \mathbf{x}_*) \tag{2.8}$$

The estimate for y is then assumed to be the mean of the output distribution and is calculated as

$$\bar{y}_* = K_* K^{-1} \mathbf{y} \tag{2.9}$$

The variance of the distribution can be found with

$$var(y_*) = K_{**} - K_* K^{-1} K_*^T \tag{2.10}$$

to provide a measure of confidence in the generated estimate. The matrix inversion is calculated using the Cholesky decomposition since the covariance matrix is known to be symmetric positive definite. If the estimate's input is close to a measurement from the training data, the variance will be small. As the Euclidean distance from the training data increases, the variance will grow as the region of the input-space has not been sufficiently sampled.

The resulting model provides accurate nonlinear estimates of the function output with an associated variance to be used as a measure of confidence in the estimates. In this way, accurate regression of any function can be developed including nonlinearities and noise characterization.

The benefits introduced above have helped Gaussian process regression to gain acceptance and wider use. In summary, these advantages are:

1. **Simplified Model Selection:** Closed-form equation or neural architecture are not needed for this non-parametric method. Model selection is reduced to determination of hyperparameters through training.

2. **Reliable Training Method:** Closed-form models suffer from sensitivity to poor model selection. Neural models require a high-dimensional search for neuron weights and may not converge. Search for hyperparameters is of relatively low-dimensional space and selects the most-likely model in a Bayesian sense.

3. **Estimate Confidence:** Quality of estimates is not immediately available from closed-form and neural models. Posterior variance is easily calculated for a GP regression estimate and can be interpreted as a measure of estimate quality.

2.2.3 Examples of Gaussian Processes Regression in Controls and Robotics

The ease of designing for nonlinear systems offered by this technique has driven its use to address a variety of problems faced in designing autonomous robotic systems. Training has proven reliable over the three examples summarized here as well as many others. Additionally, the ability of GP regression to provide confidence intervals is attractive to those interested in robust applications.

Robotic systems often operate in unknown environments. The first example addresses the construction of an occupancy map to aid in collision avoidance and path planning. Many sensors are nonlinear and provide noisy measurements of the environment. The second example addresses the nonlinear, noisy case of determining vehicle pitch and roll from LiDAR measurements to the roadway. State estimation of nonlinear system dynamics requires great familiarity with the system being modeled. The final example examines the case of modeling the lateral vehicle dynamics of a passenger vehicle for predictive purposes.

Inference in Occupancy Maps

Learning about an environment is a critical task for a robotic system. The formation of an occupancy map allows the design of path planning and obstacle avoidance behaviors. Grid-based maps, such as the one in Figure 2.2 [45], are typically used requiring the environment to be sectioned off into square locations and an occupancy-state to be stored for each grid location. A trade-off is required between grid resolution and computation. A grid that is more coarse requires less computation but struggles to accurately represent non-uniform obstacles. Storing an individual state for each location can be burdensome and does not scale as the number of dimensions representing the environment grows. A method of mapping that is effective in two dimensions may not operate well in three dimensions as required by unmanned air vehicles (UAV). Finally, a margin of safety is typically implemented around obstacles to improve traversability. Each grid location along the path from a sensor to an object must be updated with a new state for each measurement taken. The addition of a safety margin surrounding an obstacle increases the number of grid locations that must be updated for a given measurement.

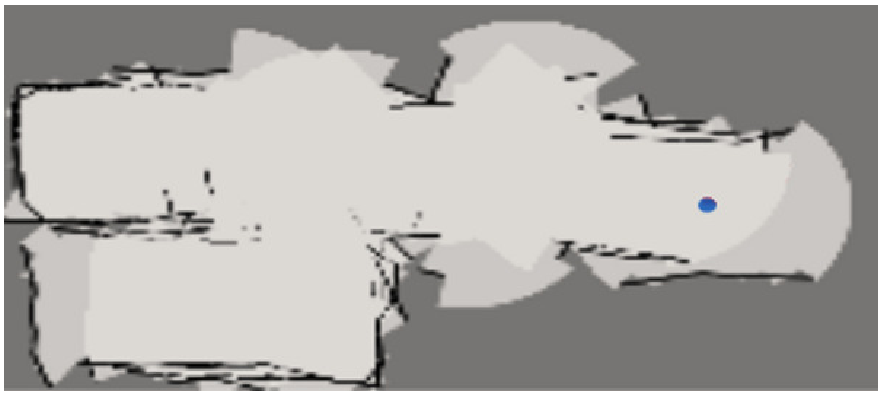


Figure 2.2: A grid-based occupancy map.

Seng Keat Gan et al. of the Australian Center for Field Robotics have published work addressing these difficulties by using a GP to classify locations as occupied or unoccupied [46]. Rasmussen details a method of altering the output of the GP model described above to act as a binary classifier [47]. An occupied location can be assigned a value of +1 and an unoccupied location a value of -1. The GP is trained to output these values. The output of the GP is then passed through a sigmoidal "squashing" function,

$$p(y_i|x, y_{-i}, \theta) = \Phi\left(\frac{y_i(\alpha\mu_i + \beta)}{\sqrt{1+\alpha^2\sigma_i^2}}\right) \tag{2.11}$$

to calculate the probability of a new location being occupied. The values of α and β are hyperparameters of the classifier and are identified through additional training. Calculating the probability of occupancy for a new location, $\mathbf{x}_*$, is described in Equations (2.9) and (2.11). To form the map, a LiDAR was used to scan the environment. For each scan the position of the obstacle was recorded, if it was present, along with the appropriate state for that scan. If the scan did not reflect off of an obstacle the correct classification, unoccupied, was associated with that location. In this way the classifier now operates using the sensor measurements as input directly rather than requiring the update of individual grid locations along the path of the LiDAR measurement. This approach scales more easily to three dimensions than a grid-based approach since only a position and classification are recorded for each scan making the grid unnecessary.

The problem of updating each grid location to implement a safety boundary is addressed directly by the GP model. A value can be set indicating a threshold probability of occupancy that is acceptable for the robot to traverse. The level of risk for a particular path can therefore be set explicitly in the path planning algorithm. It is also possible to vary the amount of acceptable risk during operation of the system without revisiting each location as required by a grid-based map.

Grid-based occupancy maps have limitations that are addressed by the GP-based occupancy map. The cumbersome record keeping of an occupancy grid, the poor scalability, and a simplistic approach to safety margins are removed as difficulties to a robotic system learning about its environment. The work shows great potential as it was integrated into that research group's pre-existing UAV path planning algorithms.

Sensor Reduction and Calibration

Effective sensor calibration is a key component in contemporary autonomous systems. An accurate nonlinear sensor model allows focus to be shifted to the use of the sensor output rather than accommodating inaccuracies in the sensor model. Important issues that must be addressed to improve the performance and reliability of modern autonomous systems include identification of nonlinearities and characterization of noise in sensor output as well as developing a measure of confidence in the measurement provided by the sensor. As nonlinear systems are considered in more depth the range of mathematical structures the sensor model can adopt becomes a limiting factor to our ability to calibrate sensors. Traditionally, complete knowledge of the sensor model must be known to effectively calibrate the sensor, however, Gaussian process regression offers an alternative that removes this burdensome requirement.

GP techniques have been previously used to estimate the pitch and roll of a passenger vehicle relative to the road surface using a sensor already present on a test vehicle [48, 49]. The test vehicle was equipped with a four-layer LiDAR for the purposes of identifying lane markings on the roadway. A GP model was used to calibrate this sensor to estimate relative pitch and roll of the vehicle. This model removed the need for an additional sensor to estimate these values, typically a multi-antenna GPS receiver. A Septentrio multi antenna GPS receiver was used to provide training data and a comparative measure of the LiDAR/GP estimates. Data was recorded covering the expected range of pitch and roll of the vehicle and paired with measurements from the GPS receiver. This data was then used to form the training data set. The measurements from an individual LiDAR scan, ρ_i , were formed into an input row vector:

$$\mathbf{x} = [\rho_1 \; \rho_2 \ldots \rho_k] \tag{2.12}$$

and associated with the Septentrio measurement, *y* for the purpose of training. Accurate estimates of the vehicle pitch and roll were then provided by the LiDAR while the Septentrio could be removed during operation of the vehicle. The vehicle was put through a number of induced pitch and roll maneuvers and the estimates provided by the GP model were compared to the Septentrio measurements. The plots accompanying these values are seen in Figure 2.3

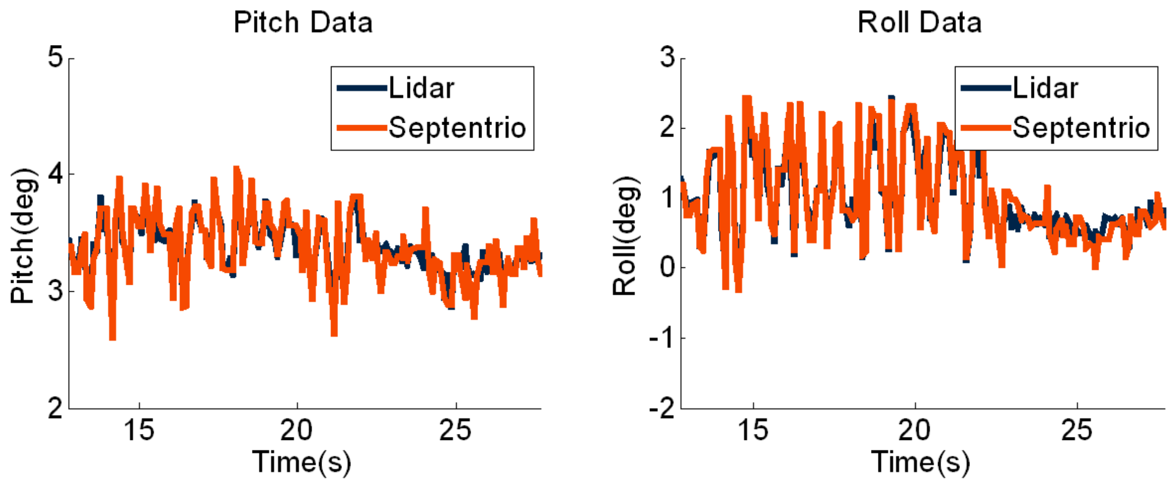


Figure 2.3: Comparing GP model pitch and roll estimates to Septentrio measurements.

The GP model estimates were well within the half degree accuracy of the Septentrio measurements having a mean square error of 0.0172 deg and 0.0398 deg for pitch and roll respectively. The estimates were also calculated quickly enough, 0.003s per estimate, to allow for real-time implementation given the 20Hz scan rate of the LiDAR. The ability to estimate vehicle attitude using sensors that were already present for other reasons allows a reduced sensor set to be used.

Modeling of Dynamic Systems

Vehicle parameter and state estimation is a significant part of modern vehicle control systems. Specifically, estimation has become an integral part of controlling unmanned ground vehicles (UGV). GP regression has been shown to be an effective technique to describe the dynamics of non-linear systems. Furthermore, GPs address the difficulties encountered in previous attempts to develop a flexible model and provide a means of control for the system. One such work considers the efficacy of modeling a four-wheeled vehicle with GP models. That effort investigated the usefulness of these methods on a system that has proven to be complex and has relatively faster dynamics requiring fast calculations [30].

The form of the inputs and outputs for the GP must be determined. The model describes the lateral behavior of a vehicle and therefore accepts the steer angle, δ, as input and provide estimates of the side slip angle, $\hat{\beta}$, and yaw rate, $\hat{r}$, as outputs. While this describes the input and outputs to the system, an autoregressive structure was adopted to present the system input and outputs to the GP for regression. A key parameter in forming the GP models is the number of previous input and outputs to feedback as input to the model, also referred to as the lag space.

Two previous inputs and two previous outputs are fed back in addition to the current input to form the lag space. Therefore, each case of input data will have the form

$$[\delta_{\tau-2}, \delta_{\tau-1}, \delta_{\tau}, \beta_{\tau-2}, \beta_{\tau-1}] \tag{2.13}$$

The output of the GP model is the estimate of the current output or $\hat{\beta}_{\tau}$ in this case keeping in mind that the sampling frequency of the training set, $100Hz$, is preserved. A similar autoregressive approach was taken to model and predict vehicle yaw rate. This results in the input shown here:

$$[\delta_{\tau-2}, \delta_{\tau-1}, \delta_{\tau}, r_{\tau-2}, r_{\tau-1}] \tag{2.14}$$

and the output of the GP model is $\hat{r}_{\tau}$.

GP regression was used to map the two five-dimensional input-spaces to the estimates of β_{τ} and r_{τ}. It is important to note that the two estimators have two distinct input-spaces and therefore are separated into two separate GP models. Each will have its own unique hyperparameters even though they are modeling one system.

The simulated vehicle was excited using a chirp signal,

$$\delta_f = \frac{1}{2}\sin(0.1\pi t^2) \tag{2.15}$$

sampled at 100 Hz for $t = 1$s to 10s and corrupted by Gaussian white noise to mimic the noise present should these values be measured directly from a vehicle. This noise had a standard deviation of 0.5 deg.

A double lane change maneuver was used for validation. A test of model generalization is provided by using a validation maneuver distinct from any of the training maneuvers. The data recorded during the validation maneuver was then processed into input-output pairs as described earlier with regards to the training data. Figure 2.4 shows that the estimates provided are accurate

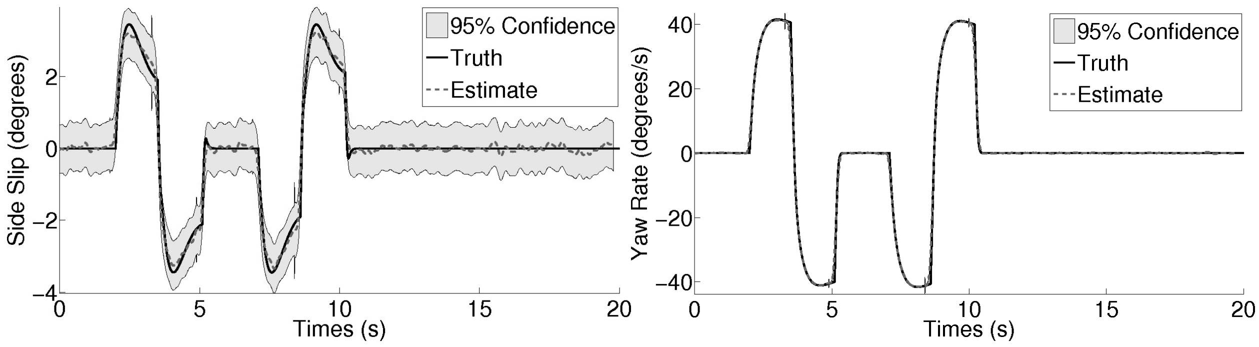


Figure 2.4: Side slip angle estimates, $\hat{\beta}$ (Left). Yaw rate estimates, $\hat{r}$ (Right).

as compared against the truth data. It also reveals that the 95% bounds are near ± 0.5 deg. Similar analysis was performed with the yaw rate, r, estimates as both are required to estimate that position of the vehicle.

2.2.4 Summary

GP regression offers simplified model selection in nonlinear problems, reliable training methods, and indication of estimate confidence. These attributes make the use of GP regression a compelling answer to many of the questions faced in designing robotic systems. GPs have already been used to improve obstacle avoidance and path planning, sensor calibration, and dynamic system modeling. Results such as these demonstrate the effectiveness of GP regression in learning the complex relationships inherent in all robotic systems. It is the ability to learn nonlinear dynamics that makes GP regression a technique worthy of investigation for use in MPC.

Chapter 3

Dynamic Gaussian Process Model for Vehicle Roll

3.1 Introduction

Use of model predictive control (MPC) has expanded in recent years given the corresponding increase in computational power but still requires a trade-off between model accuracy and computation speed. MPC has been applied to ESC in [15], and [16],both showing that a controller's ability to predict undesirable conditions early can allow for a more effective response even with simplifications necessary to accommodate available processing capacity.

As discussed in Chapter 1, a number of works explore the use of simplified models for use in MPC-based vehicle stability control [16, 18, 19, 20, 21, 22]. linearized online about the operating point while the nonlinearities in the front tire model were "extracted" to remove the computational burden placed on MPC during optimization of the control effort. This extraction was performed using MPC to find an optimal value of lateral force on the front tire. The front lateral tire force was then used, along with a nonlinear tire model, to find and command the associated steer angle. The resulting method preserves the convexity of the optimization problem solved by MPC allowing linear optimization methods to be applied. These works demonstrate that model fidelity and computation time remain a difficult trade-off to make. An alternative approach to calculating an optimal control effort is taken in this work. To begin, an alternative to the parametric model form used in previous work is adopted. Gaussian process(GP) regression, a non-parametric machine learning technique, is used to learn the vehicles dynamics rather than assuming a form and identifying parameters. The criteria necessary for this model to be successfully used in MPC are described and the model's performance will be evaluated against those criteria. A GP-based dynamic model is presented incorporating GPS and inertial data recorded from a vehicle to demonstrated the speed and accuracy of calculations.

3.2 Gaussian Process Predictive Model

Four states are of interest in this application. The roll angle (ϕ) and roll rate ($\dot{\phi}$) are the first two states considered as the aim is to constrain these values to prevent vehicle rollover. Figure 3.1 shows that the vehicle can be approximated by a sprung mass with ϕ describing the deviation of the sprung mass from the vertical axis. The pair of states, ϕ and $\dot{\phi}$, are referred to as the roll states in this work.

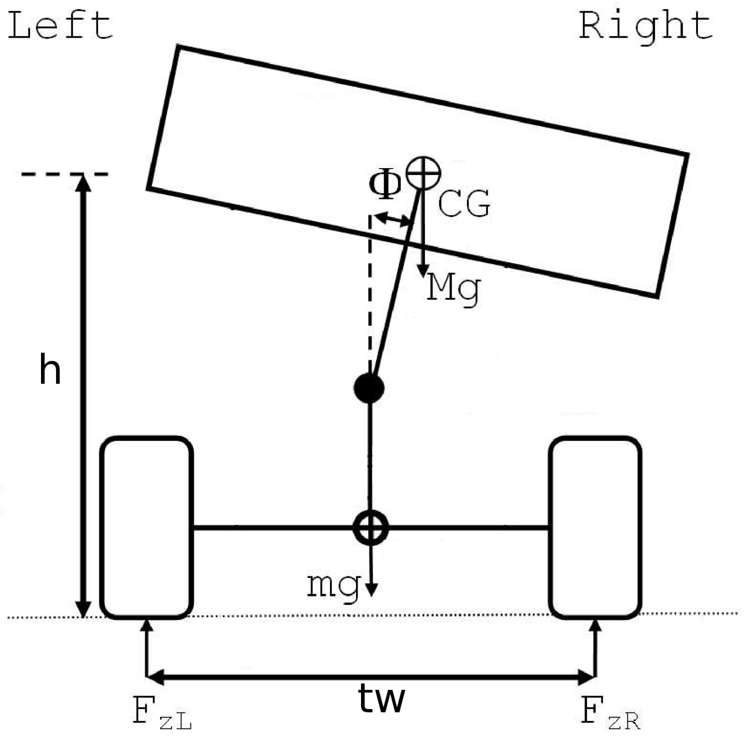


Figure 3.1: Body roll free body diagram.

In addition to the states describing vehicle roll, the lateral dynamics are also included in the state vector, **z**. The vehicle sideslip (β) and yaw rate (r) can be used to reconstruct the path of the vehicle and are coupled to the roll dynamics described above. Figure 3.2 shows β as describing the angle between vehicle course and vehicle heading. The yaw rate is shown as the angular rate of the vehicle around the vertical axis as it passed through the center of gravity.

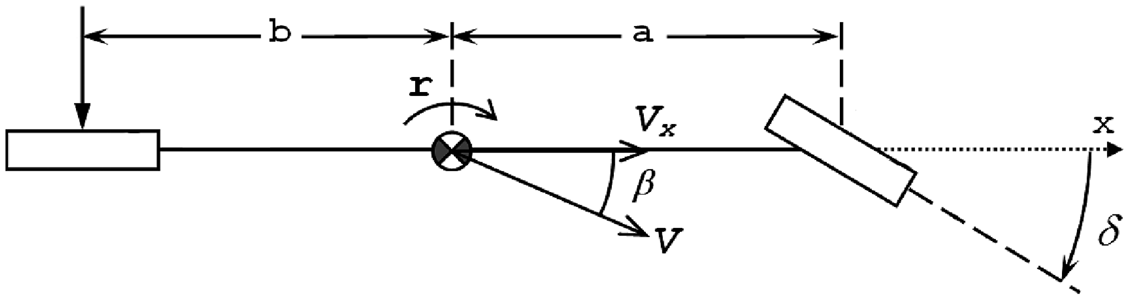


Figure 3.2: Bicycle model.

The complete state vector is defined by,

$$\mathbf{z} = \begin{bmatrix} \beta \\ r \\ \phi \\ \dot{\phi} \end{bmatrix} \tag{3.1}$$

Adopting the discrete state-space form introduced in (2.3) the problem of model identification can then be restated as the search for the function $G(z_k, u_k)$. Note that the full state vector is assumed to be observable. This work uses a nonparametric method of describing these functions rather than assuming a model form and identifying model parameters. The method used to learn the two functions here is Gaussian process regression as described in Section 2.2.

The input to each GP is comprised of the state vector $\mathbf{z}$ and additional inputs. Longitudinal velocity (V_x) of the vehicle is also included in the input vector, $\mathbf{x}$. However coupling between longitudinal acceleration and the other vehicle states was ignored for this work and omitted from the state vector. V_x still affects the dynamics of the vehicle and can not be ignored entirely. V_x is assumed constant and, while the V_x dimension of the input does not select between discrete models, vehicle speed can be thought of as a type of model scheduling in this context. Where a discrete value V_x does not exist for the prediction being made a value is inferred based on the prediction of closely adjacent models.

The steer angle at the steering wheel serves as the input to the system and is therefore also included in the input to the GP. The entire vector fed to each GP as input is

$$\mathbf{x} = \begin{bmatrix} \beta \\ r \\ \phi \\ \dot{\phi} \\ V_x \\ \delta_{SW} \end{bmatrix} \tag{3.2}$$

3.2.1 Propagation of Uncertainty

The variance of the estimate given the input in (3.2) was addressed in (2.10) given that $\mathbf{x}$ was considered certain. However the case where $\mathbf{x}$ is not certain was not address as part of GP regression originally. Girard et al. addressed the propagation of estimate uncertainty in dynamic GP models in [31].

The variance of the posterior distribution for a single-step estimate is easily calculated with (2.10) however MPC depends on a multi-step estimate. The multi-step estimate can be found iteratively though it becomes necessary to accumulate the prediction variance as each new step is dependent on the previous estimate.

In [31] it is shown that while the mean of the distribution may be propagated naively in the case of small σ_{x^*}, the variance must be corrected at each iteration. Building from (2.10) using the law of iterated conditional variance and the second order Taylor series expansion the corrected variance was found as

$$var(y) = \sigma_y^2(\mu_{x*}) + Tr\left\{ \Sigma_{x*}\left(\frac{1}{2}\left|\frac{\partial^2\sigma_y^2(x^*)}{\partial x_d^*\partial x_d^*}\right|_{\mu_{x^*}} \right.\right.$$
$$\left.\left. + \left|\frac{\partial\mu_y(x^*)}{\partial x^*}\right|_{\mu_{x^*}}\left|\frac{\partial\mu_y(x^*)}{\partial x^*}\right|_{\mu_{x*}}^T \right)\right\} \tag{3.3}$$

where,

$$\left|\frac{\partial\mu_y(x^*)}{\partial x_d^*}\right|_{\mu_{x^*}} = \left|\frac{\partial\mathbf{k}(x^*)}{\partial x_d^*}\right|_{\mu_{x^*}}^T K^{-1}\mathbf{t} \tag{3.4}$$

and,

$$\left|\frac{\partial\mu_y(x^*)}{\partial x_d^*}\right|^{\mu_{x^*}} = \left|\frac{\partial^2 K(x^{*p},x^{*q})}{\partial x_d^{*p}\partial x_{d'}^{*q}}\right|_{\mu_{x^*}}$$
$$- \left|\frac{\partial K(x^*,\mathbf{x})}{\partial x_d^*}\right|_{\mu_{x^*}}^T K(\mathbf{x},\mathbf{x})^{-1}\left|\frac{\partial K(\mathbf{x},x^*)}{\partial x_{d'}^*}\right|_{\mu_{x^*}} \tag{3.5}$$

This type of correction was made in this work given that the predictions in the MPC are iterative in nature and must be calculated to a distant but finite horizon. Any error bounds presented in this dissertation for multi-step predictions were calculated with this method.

3.3 Model Identification and Excitation

The Prowler all-terrain vehicle from ATV Corporation shown in Figure 3.3 was instrumented and used to perform experimental validation of the model. A Septentrio 3 antenna GPS receiver was used to measure vehicle speed, attitude, course, and heading during training and validation. A Crossbow 440 6-DOF inertial measurement unit (IMU) was used to measure yaw and roll rates as well as refine the GPS data when filtered together in an extended Kalman filter(EKF). Details of the GPS/INS EKF can be found in [50]. Steer angle, δ_{SW}, was measured at the steering wheel

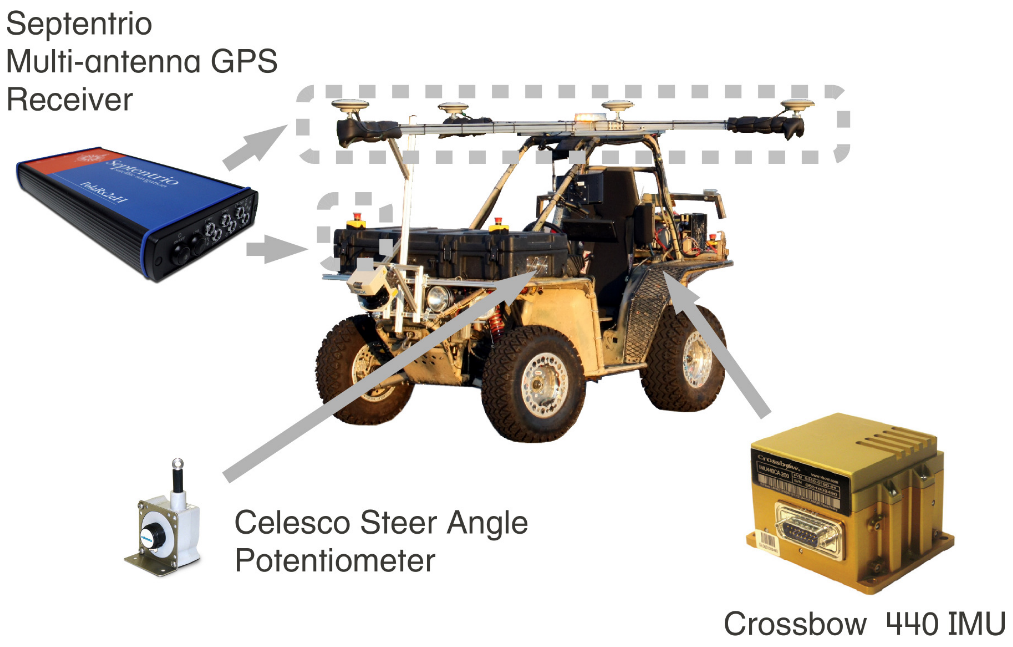


Figure 3.3: ATV Corporation Prowler vehicle instrumented for data collection.

using a Celesco string potentiometer connected to the steering wheel shaft and measured by a 10-bit analog to digital converter.

Dynamic system identification requires that input to the system during identification be sufficiently rich for parameter estimates to converge. The condition extends to nonparametric models such as the GP model. Lack of persistency of excitation affects the nonparametric case by degrading the training data's ability to sufficiently describe the system's behavior with regards to a variety of states and inputs. Taking the machine learning perspective, the function that is regressed by the GP must be sufficiently sampled within the region which inference will be performed. The space defined by the dynamic system's state and input forms the input-space to the GP. Ideally, the function would be sampled uniformly throughout this space however this can prove difficult with a dynamic system. The dynamics of the system can limit sampling within the input-space to trajectories through that space. Exciting the input ensures that the training data that will result from sampling the function along these trajectories will be sufficiently rich to infer values of the function and predict the next state of the system given the current state and input. The need for persistently exciting input is also necessary for thorough model validation. A trajectory, or set of trajectories, that occupy a greater area in the input-space will evaluate the model's accuracy over a greater variety of states and inputs.

Generally, in system identification signals such as an impulse, step, or white noise are used to persistently excite the system for identification. While effective, these signals can be difficult in practical identification given the inherent discontinuities. However, they all share the attribute of exciting the system over a variety of frequencies. A more practical signal that excites the system over a variety of frequency is a chirp, or sinusoid with sweeping frequency. It was this signal that was used to collect data and train the dynamic GP model of the ATV Corp Prowler. The vehicle was manually steered while state and input data was recorded. A series of sinusoidal maneuvers were performed with varying frequencies to approximate a chirp input. The steer angle (system input) used during training is shown in Figure 3.4 along with the longitudinal velocity during data collection. This data was recorded over a variety of longitudinal velocities around which the validation maneuver was expected to be performed. The bandwidth of possible inputs is limited

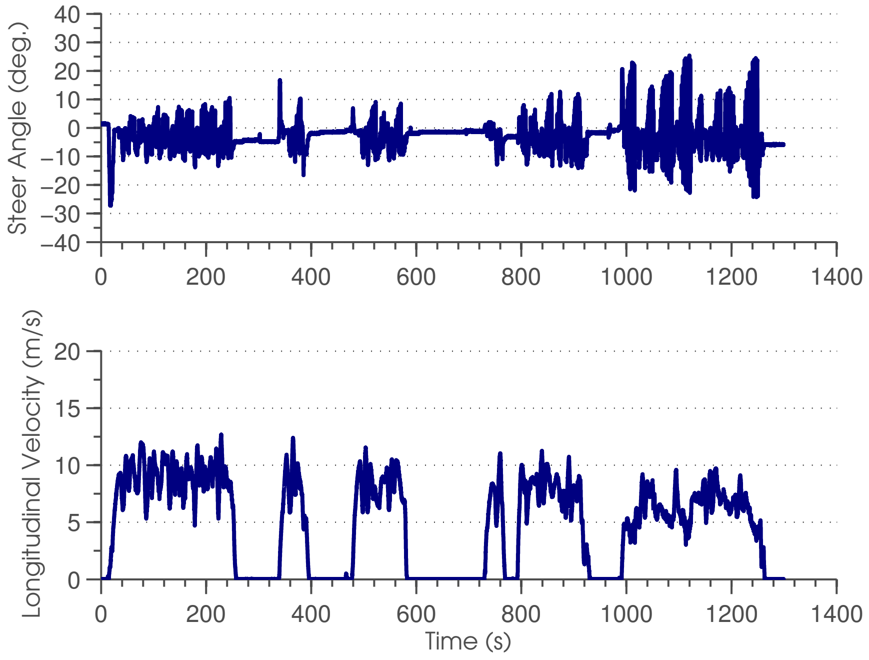


Figure 3.4: Steer angle and velocity during training maneuver.

by mechanical design and driver strength. Examining the frequency content of the input during the training maneuver in Figure 3.5, it becomes clear that the input is of a low frequency. However,

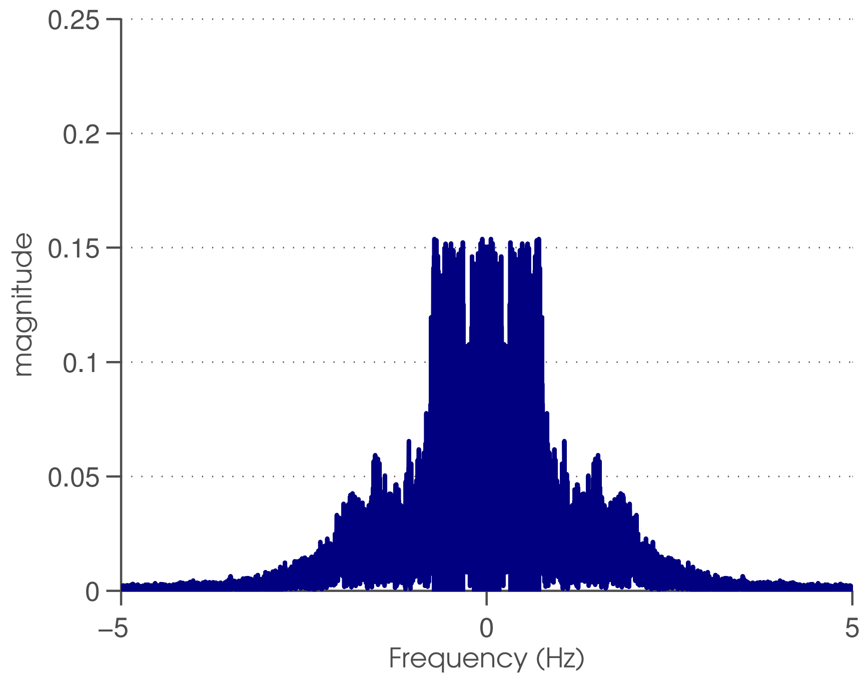


Figure 3.5: Frequency content of steer angle during training maneuver.

the frequency content is distributed over the bandwidth indicated rather than concentrated on a single frequency. It should also be noted that wheel lift occurred multiple times during the training maneuver thus exciting the dynamics near the bound of stability.

Validation of both parametric and nonparametric models for lateral and roll dynamics of vehicles is often performed with maneuvers that induce vehicle roll and sideslip, both of which are phenomena that lead to vehicle rollover. Two maneuvers are commonly used to induce vehicle roll and sideslip and can be performed on the experimental platform. Lambert focused on the NHTSA fishhook in [13]. This maneuver was excluded from experimental validation to avoid completely rolling the vehicle and destroying the instrumentation though it is studied later in this work with simulated results. Edwards [51] used a double lane change (DLC) in CarSim simulations. The validation maneuver used here was a DLC on an ISO 3888 standard course with a measured vehicle width of $45cm$. A depiction of this course and the path of the vehicle are shown in Figure 3.6. The steer angle and velocity over a typical validation maneuver are shown in Figure 3.7. In total, the model was validated using 30 individual DLC maneuvers in addition to typical driving to position

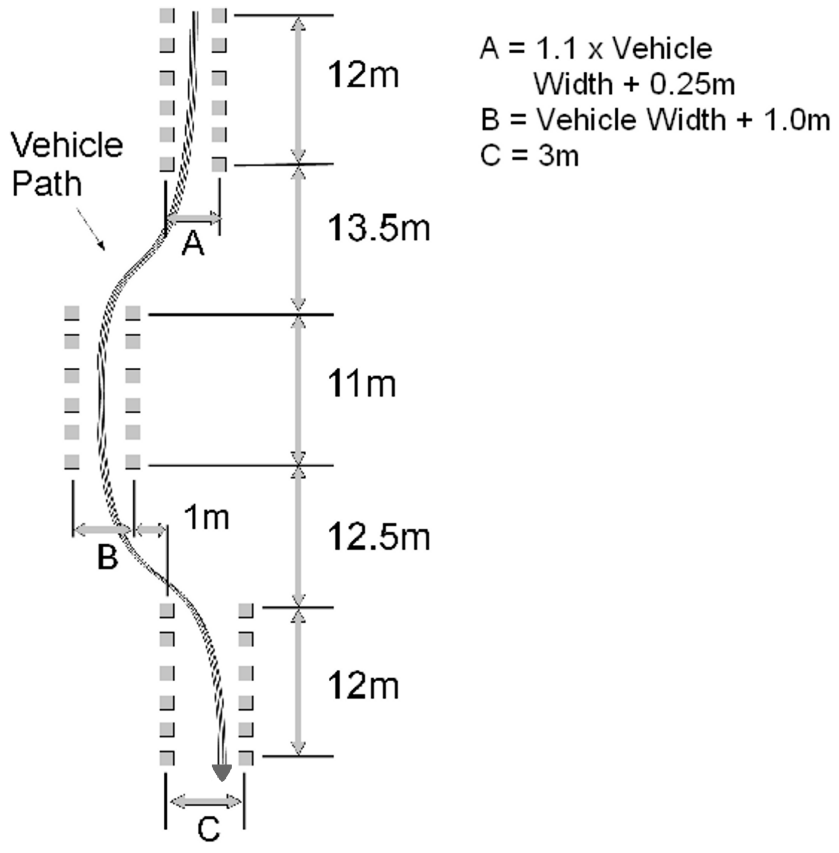


Figure 3.6: ISO 3888-2 double lane change course.

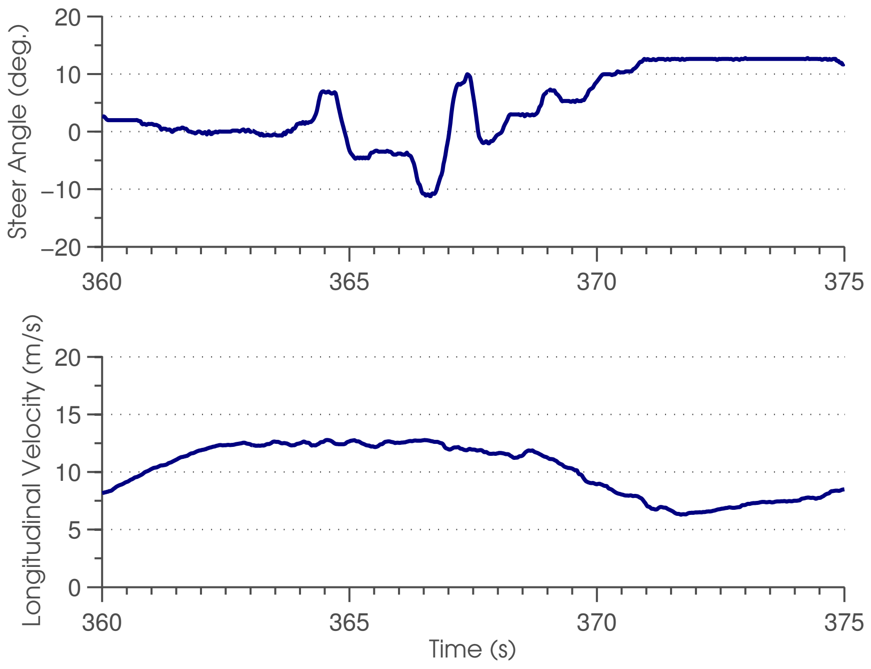


Figure 3.7: Steer angle, δ, and velocity during a single DLC validation maneuver.

the vehicle at the start of the DLC course. The input and velocity during the entire training dataset is shown in Figure 3.8.

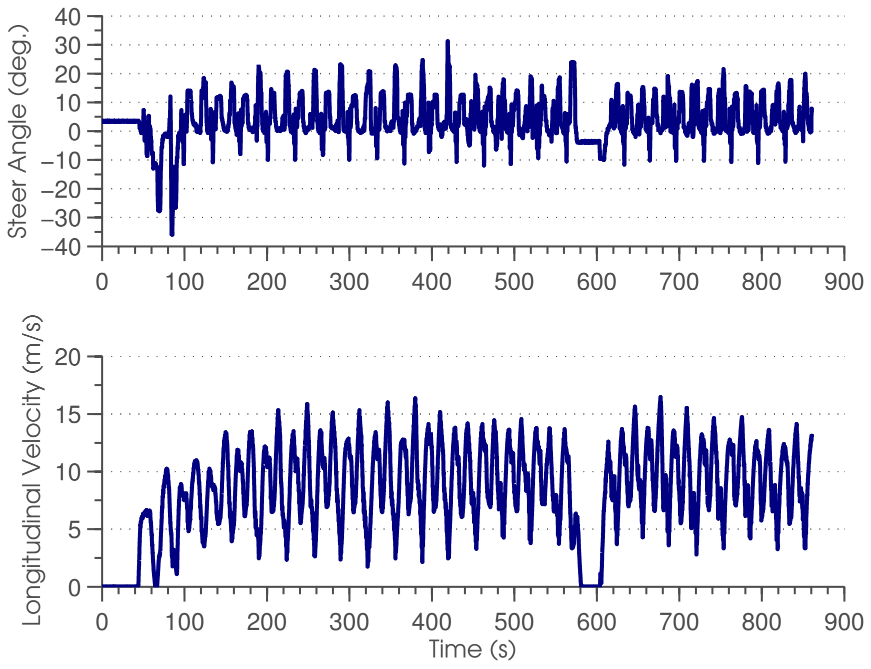


Figure 3.8: Steer angle, δ, and velocity during all validation maneuvers.

Comparing the frequency content during training from Figure 3.5 with that of the frequency content during validation in Figure 3.9 reveals that while the bandwidth of the training data was limited it encompasses that of the validation data.

3.4 Prediction Accuracy

Gaussian process regression was performed four times, once for each state, using data collected during the training maneuvers described in Section 3.3. The functions estimated by these four GPs comprise the model of the system to be used in a model predictive control. The predictive capabilities of the GP model were evaluated by comparing the predictions with measurements gathered during the validation maneuvers. The posterior variance was also calculated for the GP predictions to offer a method of evaluating their quality.

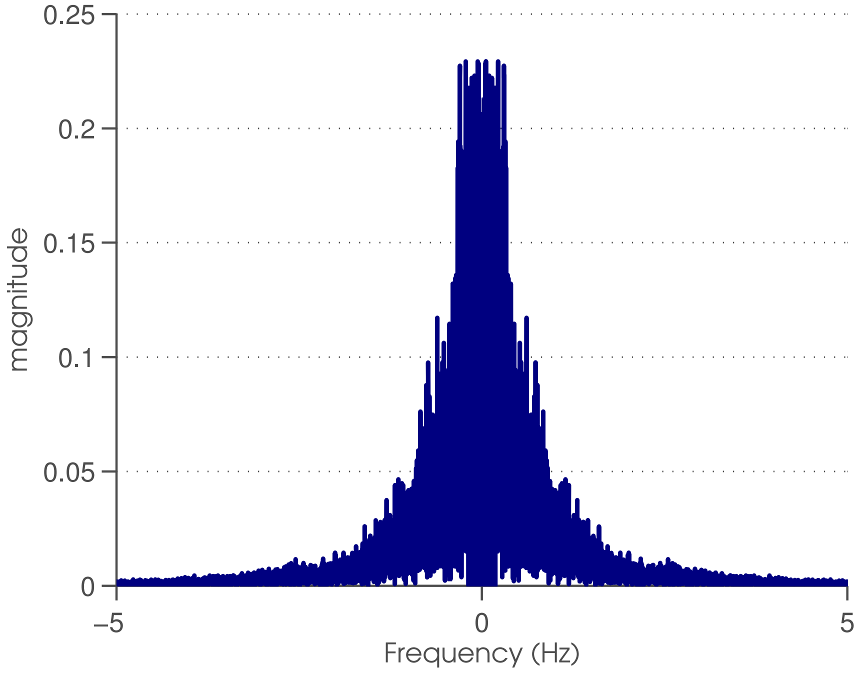


Figure 3.9: Frequency content of steer angle, δ, during all validation maneuvers.

3.4.1 Single-step Prediction Accuracy

Single step predictions during a single DLC are shown in Figures 3.10 through 3.13 for sideslip, yaw rate, roll angle, and roll rate respectively. Two-sigma bounds were also plotted to indicate the posterior standard deviation for those estimates. The standard deviation, σ, for each estimate is interpreted as a measure of confidence in the estimate and the measured value used as truth should lie within the 2σ bound 95.4% of the time. The measured truth lies within the bounds in most cases however the most notable exception is seen in Figure 3.13 near $t = 407.5s$ or more clearly on the phase plane of the roll states in Figure 3.14. Two approaches may be adopted to correct this inaccuracy. First, more data should be collected to further excite the roll dynamics. Second, as constraints are consider in for MPC a sufficient margin should be provided to prevent inaccuracies such as these from allowing the system to operate in an unstable state.

Examination of the phase plane for the lateral states in Figure 3.15 shows the estimated trajectory matches the measured trajectory more closely than the roll dynamics trajectory. The previous figures highlight a typical DLC used for validation of the estimates but the validation dataset is

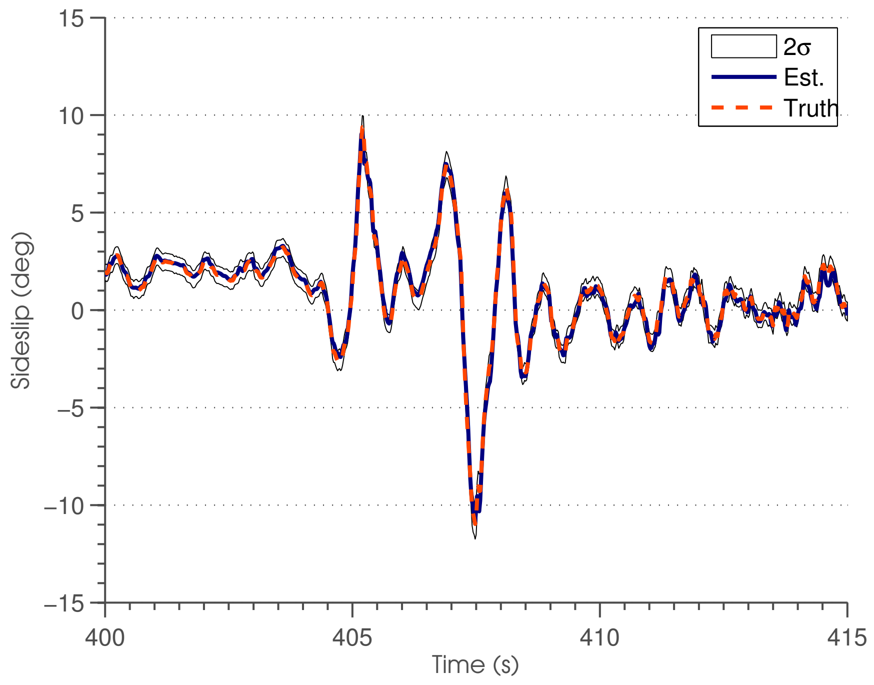


Figure 3.10: Sideslip angle during typical DLC validation maneuver.

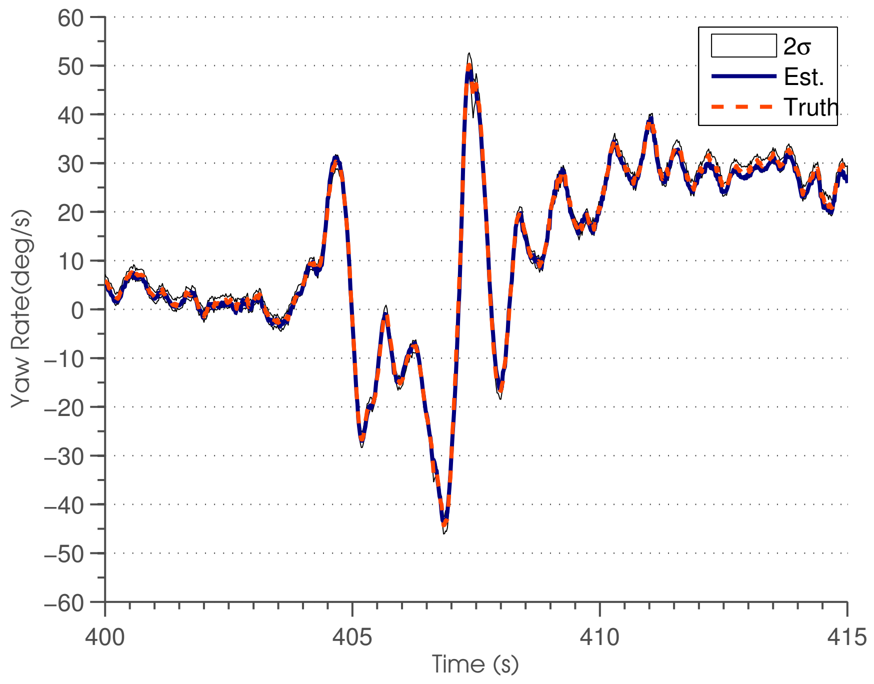


Figure 3.11: Yaw rate during typical DLC validation maneuver.

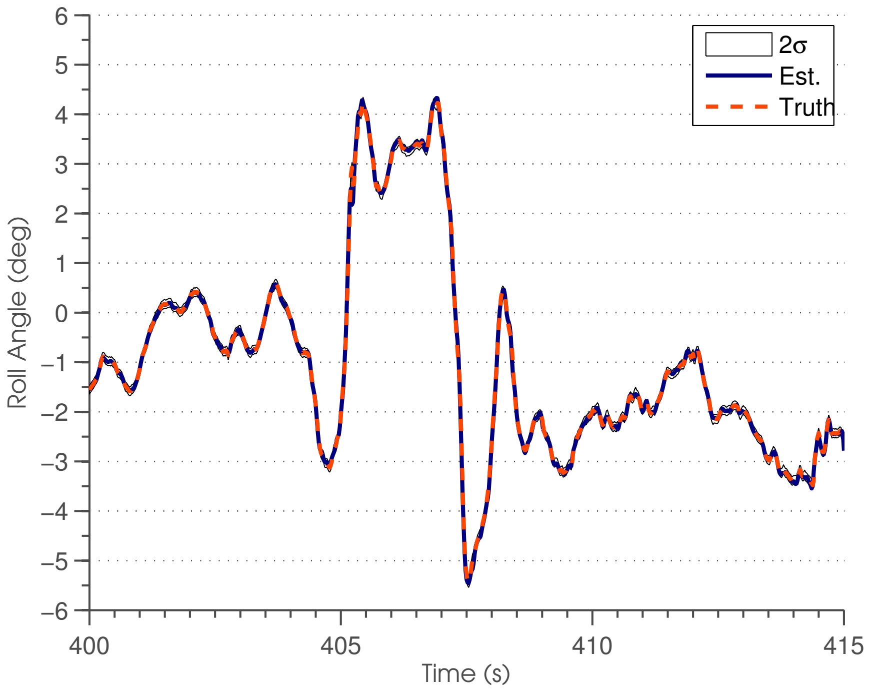


Figure 3.12: Roll angle during typical DLC validation maneuver.

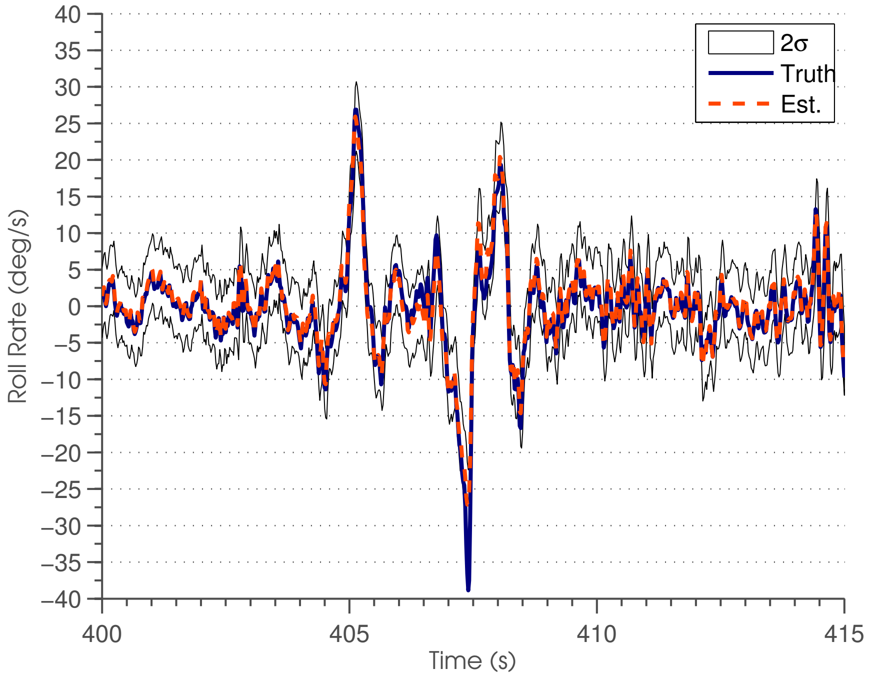


Figure 3.13: Roll rate during typical DLC validation maneuver.

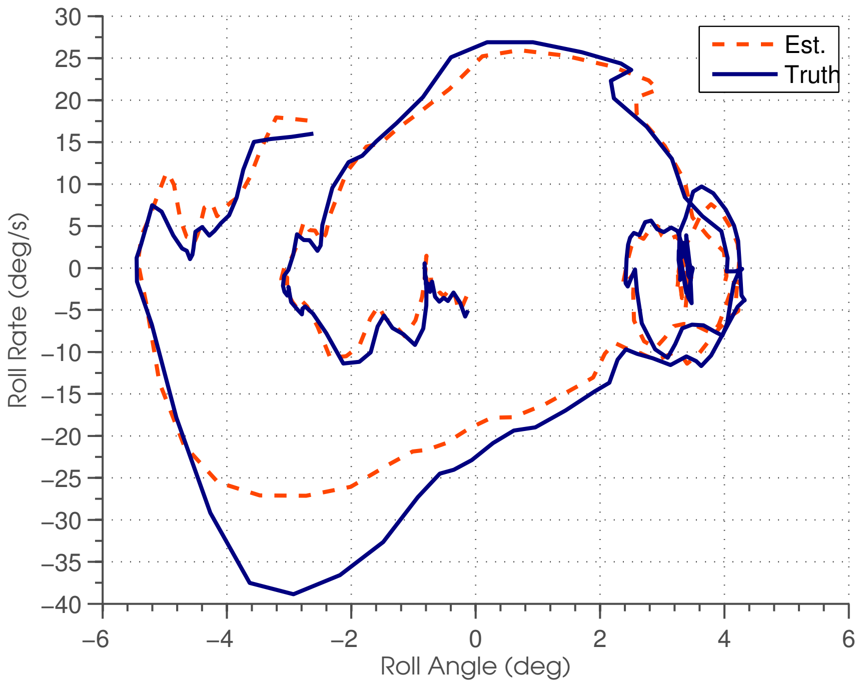


Figure 3.14: Roll dynamics phase plane during typical DLC validation maneuver.

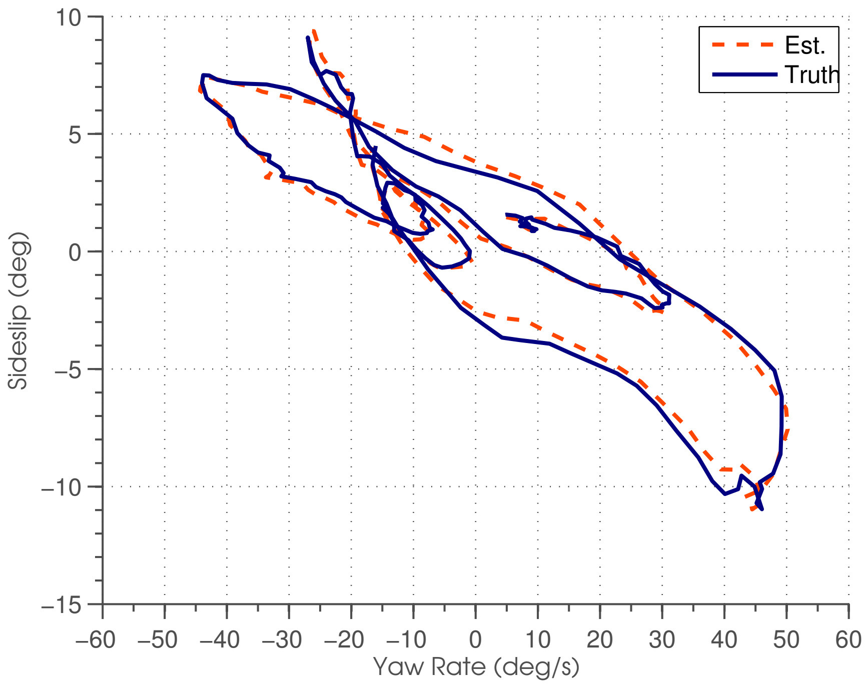


Figure 3.15: Lateral dynamics phase plane during typical DLC validation maneuver.

comprised of many of these maneuvers. The root-mean-square (RMS) errors between the single-step estimates and measurements for the entire validation dataset are listed in the first column of Table 3.1.

Table 3.1: Summary of single-step estimate quality.

State	1-step RMS
Sideslip (β)	$0.3190°$
Yaw Rate (r)	$1.108 {}^{deg}/_{s}$
Roll Angle (ϕ)	$0.0474°$
Roll Rate ($\dot{\phi}$)	$2.423 {}^{deg}/_{s}$

3.4.2 Multi-step Prediction Accuracy

Examination of single-step prediction accuracy indicates the success of training but does not speak to the ability of the model to iteratively predict the vehicle state over the distant horizon. Feedback is introduced as shown in Figure 3.16 to predict the vehicle state over the receding horizon necessary for MPC. The initial state provided to the model, $\mathbf{z}_k$, is measured at the current time, k. Ten steps are used in a number of works including [22] and that length horizon was adopted for this work. The state estimate of interest is $\hat{\mathbf{z}}_{k+10}$. $\hat{\mathbf{z}}_{k+1}$ through $\hat{\mathbf{z}}_{k+9}$ are fed back in an iterative fashion to arrive at a value for $\hat{\mathbf{z}}_{k+10}$. V_x is the current longitudinal velocity as measured from the vehicle and is assumed to be constant. This assumption ignores any coupling between the longitudinal, lateral, and roll dynamics of the vehicle but does not adversely affect the estimate quality as demonstrated here. The steer angle, δ_{SW}, was measured at the steering wheel and related to the steer angle at the road wheels with

$$\delta = -0.16592 \cdot (\delta_{SW} - 604) \tag{3.6}$$

where 604 was the measured A2D value at a zero steer angle and -0.16592 was the resolution in degrees per count. For training and validation A2D value representing the steer angle, δ_{SW}, was provided as input to estimate the states through $\hat{\mathbf{z}}_{k+10}$. The relationship in (3.6) was used only to

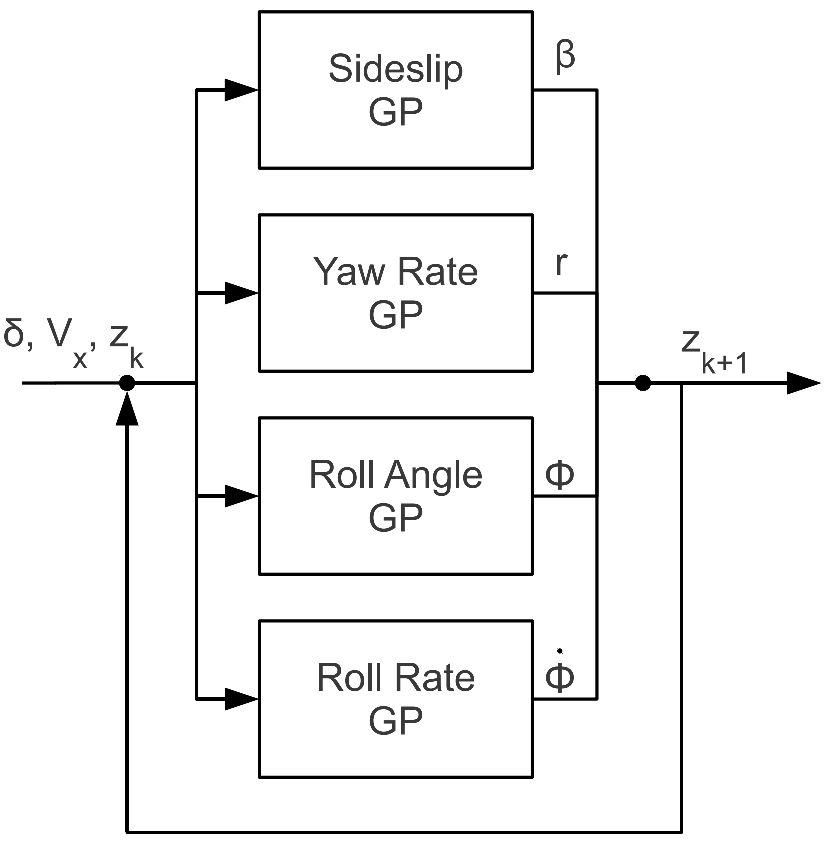


Figure 3.16: Predictive model based on regression of four separate Gaussian processes.

plot the values. This relationship between the A2D count and steer angle was incorporated into the model and learned through the training process.

The state $\mathbf{z}_k$ as measured at the current time was assumed to be exact. The posterior variance as calculated for the single-step predictions with (2.10) is then propagated through the next iteration with (3.3) as described in Section 3.2.1. Each successive estimate carries with it a portion of the uncertainty of the input for that step. The uncertainty will typically grow with each pass through the GP model effectively limiting the usefulness of the model for very distant horizons. Figures 3.17 through 3.20 show a single ten-step prediction for each state predicted by the model. Each

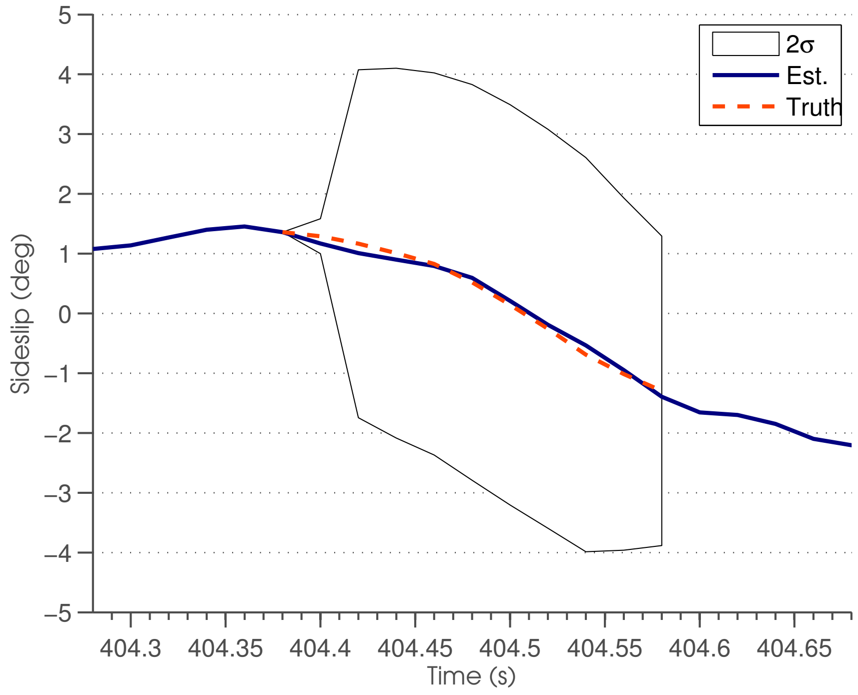


Figure 3.17: Single ten step prediction of Sideslip angle during typical DLC validation maneuver.

of these figures show the 2σ bound growing with each successive estimate starting at a common initial time. The initial value shown depicts the measurement of the current state which is taken to be certain accounting for the large jump in certainty at the beginning of the iterative process. The individual bounds grow at unique rates that depend on how well each GP was trained and the location the posterior of that GP was sampled. In each case the truth measurements lie within the bounds indicated and are often very close to the truth measurement.

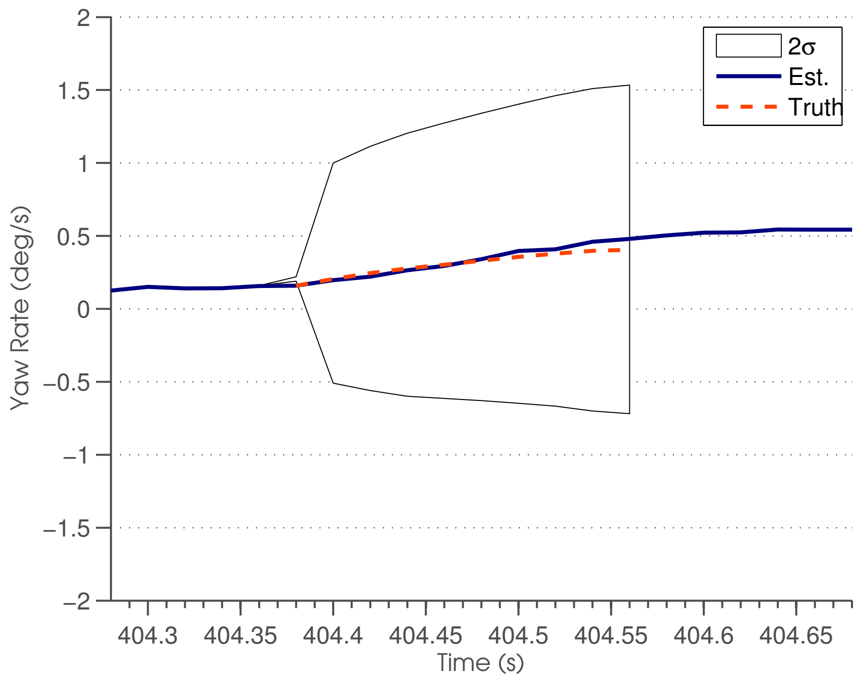


Figure 3.18: Single ten step prediction of yaw rate during typical DLC validation maneuver.

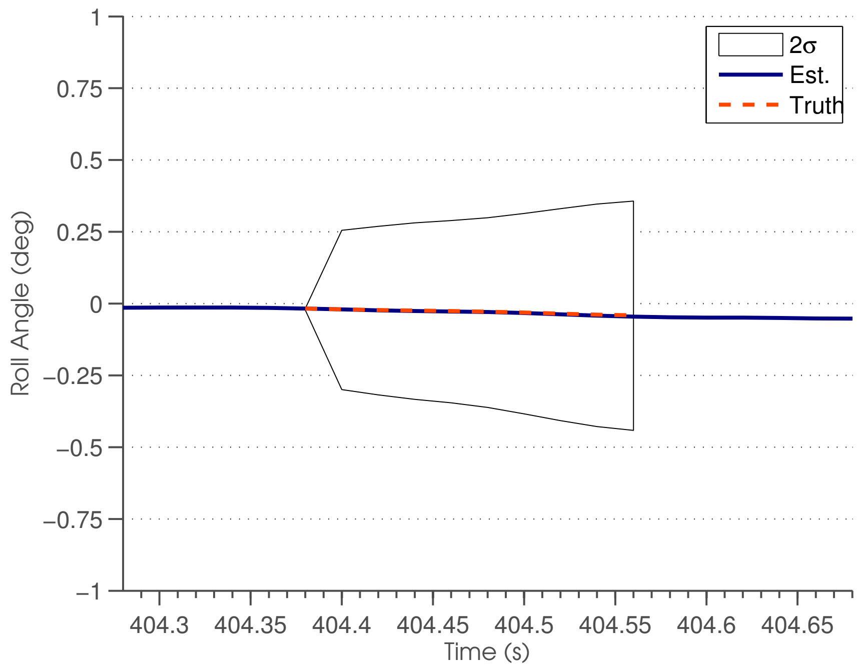


Figure 3.19: Single ten step prediction of Roll angle during typical DLC validation maneuver.

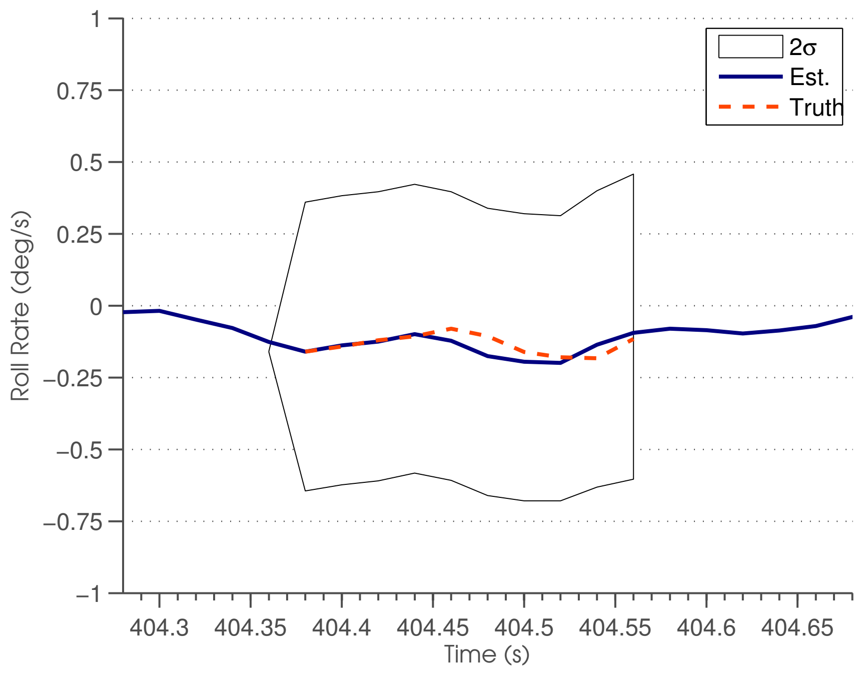


Figure 3.20: Single ten step prediction of roll rate during typical DLC validation maneuver.

The calculation of the states $\hat{\mathbf{z}}_{k+1}$ through $\hat{\mathbf{z}}_{k+10}$ given the current state $\mathbf{z}_k$, longitudinal velocity V_x, and the steer angles δ_k through δ_{k+9} illustrates how predictions are made over the finite horizon during one sample period. The ten-step predictions over a single DLC validation maneuver are shown in Figures 3.21 through 3.24. At each time shown on the plots the ten-step estimate is shown using $\mathbf{z}_{k-10}$ as the initial condition. Each plot shows that the ten-step estimates match the truth measurement closely though to a lesser accuracy than the single-step estimates. This degradation in accuracy is expected as the uncertainty in each individual estimate accumulates with each iteration. The ten-step estimates are less accurate but still lie within the 2σ bounds and match the truth to greater accuracy than the bounds reveal. Examination of the lateral and roll state phase planes in Figures 3.25 and 3.26 show that the ten-step estimates behave in a similar manner to the single step estimates though with reduced accuracy.

Again, these figures only depict ten-step estimate accuracy over a single validation maneuver. To characterize the ten-step estimate quality over the entire validation dataset the RMS error of the estimates are shown in Table 3.2. The single-step estimates are included in that table to offer easier

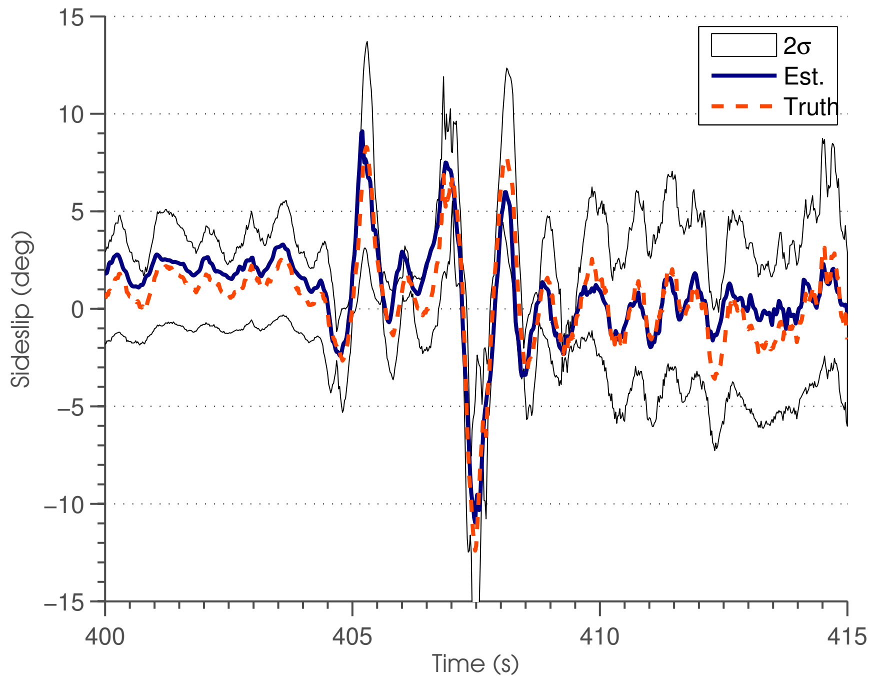


Figure 3.21: Ten step predictions of sideslip angle during typical DLC validation maneuver.

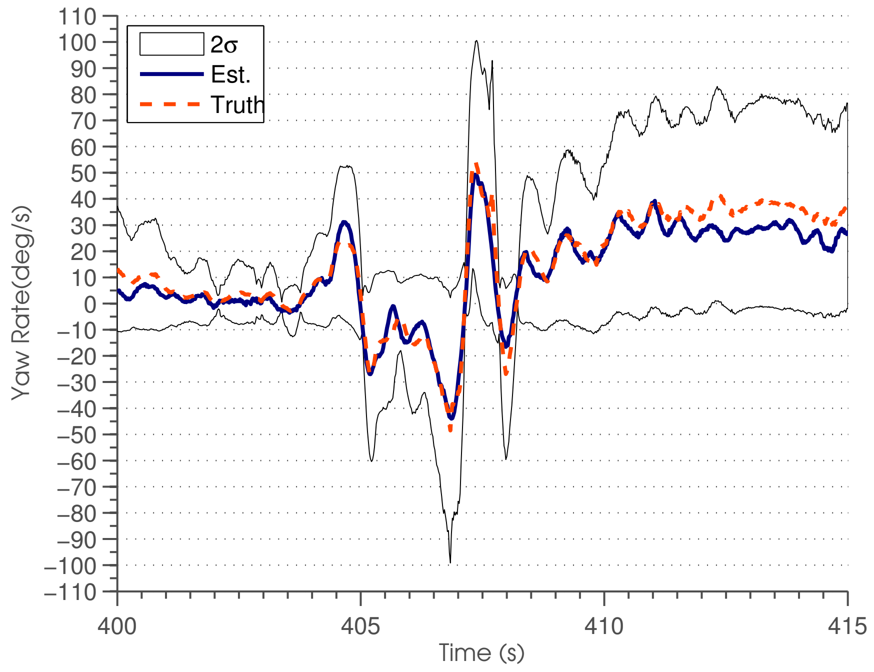


Figure 3.22: Ten step predictions of yaw rate during typical DLC validation maneuver.

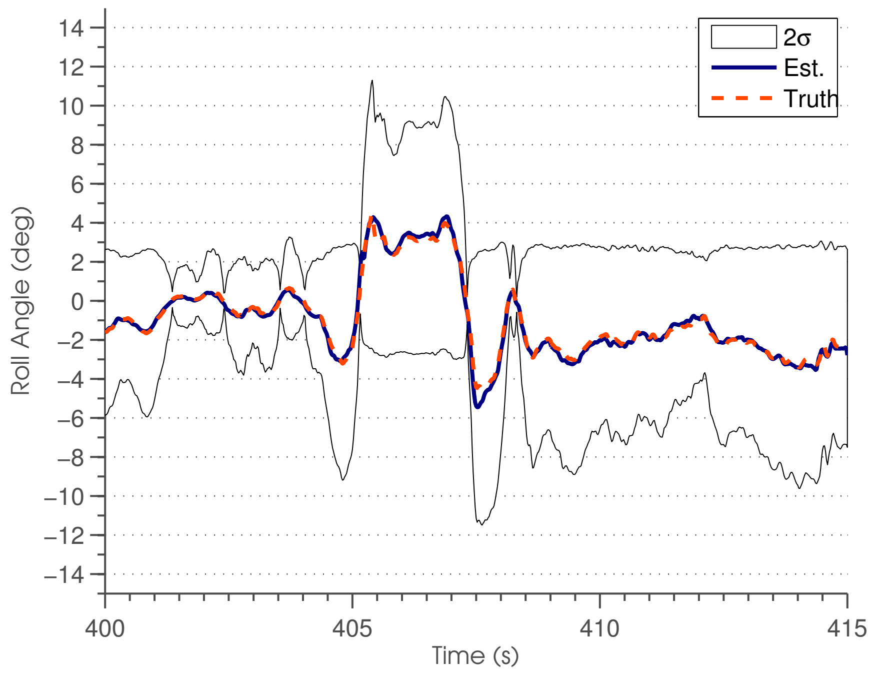


Figure 3.23: Ten step predictions of roll angle during typical DLC validation maneuver.

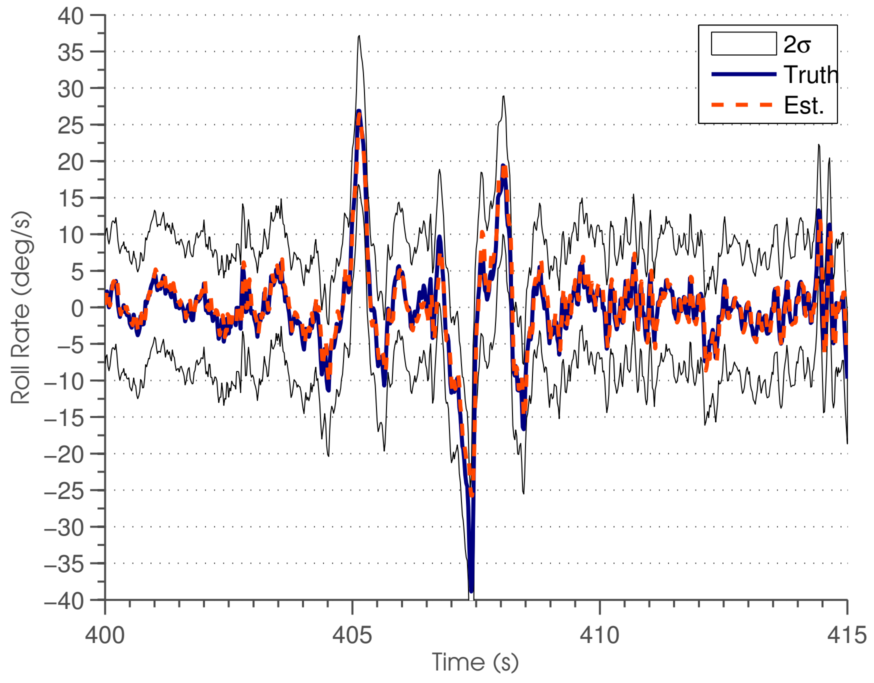


Figure 3.24: Ten step predictions of roll rate during typical DLC validation maneuver.

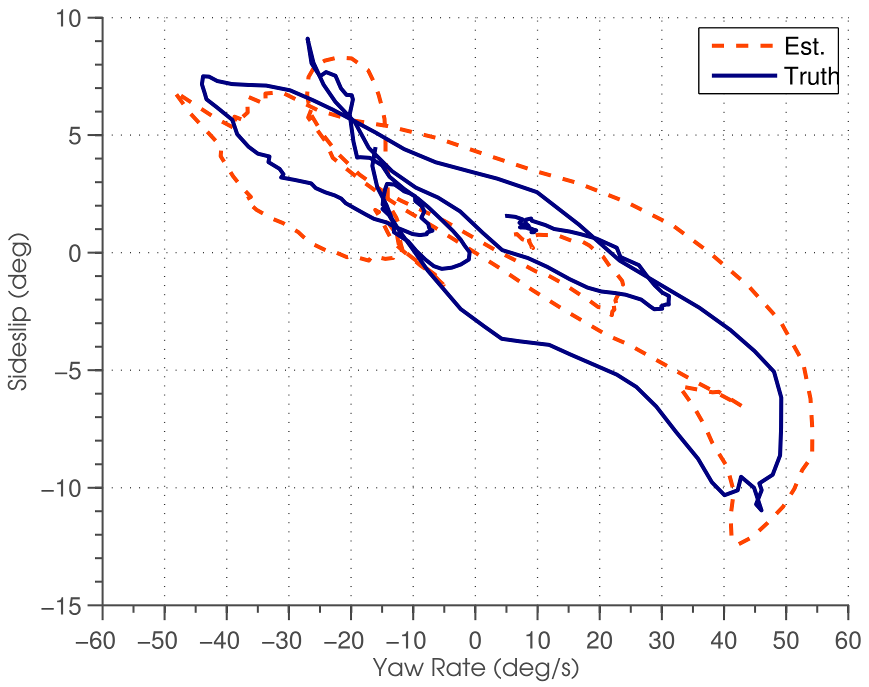


Figure 3.25: Ten step predictions of lateral dynamics on a phase plane during typical DLC validation maneuver.

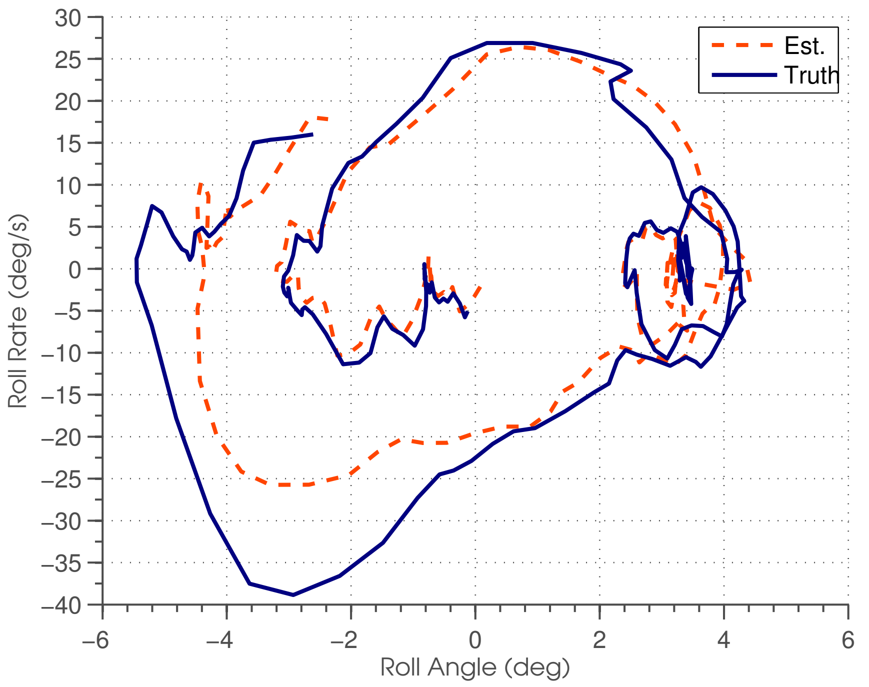


Figure 3.26: Ten step predictions of roll dynamics on a phase plane during typical DLC validation maneuver.

Table 3.2: Summary of single-step and 10-step estimate quality.

State	1-step RMS	10-step RMS
Sideslip (β)	$0.3190°$	$1.629°$
Yaw Rate (r)	$1.108°/s$	$6.046\,deg/s$
Roll Angle (ϕ)	$0.0474°$	$0.2398°$
Roll Rate ($\dot{\phi}$)	$2.423°/s$	$3.964\,deg/s$

comparison and evaluation of how the iterative process degrades the accuracy. In both cases the GP-based model shows acceptable accuracy to make predictions over the finite horizon.

3.5 Optimization Method and Computation Speed

Solving the nonlinear constrained optimization at the heart of MPC as described in Section 2.1 remains the primary impediment to broader adoption for nonlinear systems with a faster sample rate. The computation of the optimal control effort must be carefully carried out in a manner that allows for rapid optimization without sacrificing predictive accuracy. Much of the prior work focuses on linearizing the model in a manner that facilitates this trade-off. Here, a counter-intuitive method to perform nonlinear optimization with a GP model will be discussed.

The calculation of posterior means as seen in (2.9) is a linear operation. Given that the system is assumed to be time-invariant, a portion of the calculation can be performed a priori such that

$$\bar{y}_* = K_* \underbrace{K^{-1}\mathbf{y}}_{\substack{\text{calculated}\\ \text{a priori}}} \tag{3.7}$$

is reduced to a single vector-vector operation to produce a single estimate where K_* is a $1 \times n$ vector. K_* can be augmented with additional rows in the case where multiple estimates are desired. Each row represents a point in feature-space where an estimate is to be calculated. As K_* is now an $m \times n$ matrix, where m is the number of estimates to be calculated in a single operation, the calculation of estimates is a linear matrix-vector operation.

Modern processors are centered around this type of calculation. Linear algebra operations are implemented and optimized by processor manufacturers and a standard, portable interface is provided through the Basic Linear Algebra Subprograms (BLAS). Further study and optimization of the BLAS routines has been performed in a number of works such as [52]. Further improvement in calculation speed can be gained through the use of graphical processing units (GPUs) as described in [53].

Calculation speed can be made shorter through the use of BLAS, or similar routines and often outperforms a naive implementation of matrix-vector multiplication. The complexity of matrix-vector multiplication is described as $\mathcal{O}(2mn)$ with respect to the size of the two operands described above. However, as each implementation of the matrix-vector operation is tailored to the platform on which it will be performed, the advantages of each architecture, including parallelization, can be leveraged to speed calculation. The improvement in speed through the use of optimized code is depicted in Figure 3.27.

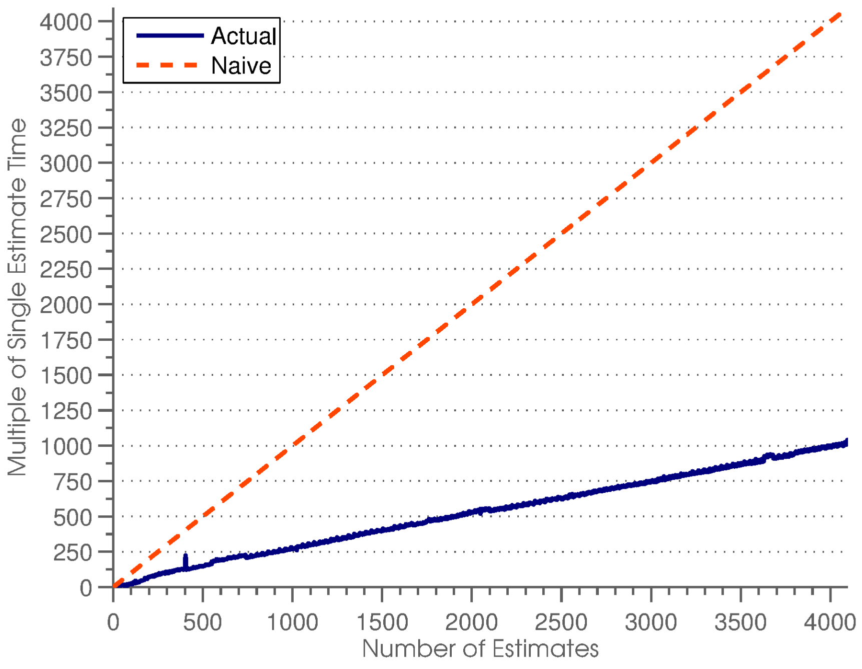


Figure 3.27: Matrix-vector computation time vs. naive complexity.

Estimates were calculated on a desktop computer with four Intel processors. The number of estimates calculated by a single matrix-vector operation was increased and the average time over 100 calculations was recorded and normalized with the single estimate calculation time. Parallelization becomes more effective as the number of estimates becomes large. Figure 3.27 clearly shows that as the number of estimates calculated increases, the advantage of exhaustive search with the GP model over exhaustive search with a naive nonlinear-model grows.

How an algorithm's complexity scales is an important characteristic but ultimately the calculation must be completed within the sample period of the system. Given the vehicle is controlled and sampled at $50Hz$ in this chapter the calculation of control effort must be completed within $20ms$. Generating estimates of the four states in (3.1) for 200 discrete steer angles over the system input-space for a ten step horizon was completed in only $54\mu s$. Plenty of time remains to calculate the cost and constraints necessary for the GP model and select the optimal steer angle while respecting the constraints. These calculations were performed on a contemporary desktop PC that is commercially available thus lowering the hardware requirements to implement a nonlinear MPC for systems with "fast" dynamics. The requirements are low enough to make such an approach feasible with commonly available hardware rather than requiring expensive groups of machines. The choice of 200 steer angles was chosen to be sufficiently dense within the steer actuator constraints. Those two hundred predictions correspond to an accuracy of one quarter degree of steer angle at the drive wheel. While that choice was arbitrary here, it may be possible to develop a method to define this value more deliberately using the characteristic length scale for the steer angle dimension identified during training.

This approach to search for the optimal steer angle in the linear feature-space does not completely circumvent costly nonlinear calculations. The covariances detailed in (2.7) call for a large number of exponentials to be calculated prior to estimating the next state. Repetitive calculation of the exponential is fundamental to many algorithms including neural networks. Schraudolph studied the exponential to increase speed of neural computation in [54]. As GP and neural models are closely related that work is directly applicable to this effort. An approximation of the natural exponential function is formed by taking advantage of the binary exponential form used by most processors to store floating point values.

The IEEE-754 standard representation of a floating point value calls for a sign bit followed by an eleven bit exponent that includes an offset of 1023. A fifty-two bit mantissa completes the value that occupies a total of 8 bytes in memory. The first four bytes (comprised of the sign, exponent, and the most significant 20 bits of the mantissa) are designated as an integer value i_{exp}. The other four bytes (the least significant 32 bits of the mantissa) are designated as j_{exp} and are initialized to zero. An approximation of e^x can be made using

$$i_{\text{exp}} = ay + (b - c) \tag{3.8}$$

where $a = 2^{20}/ln(2)$ and $b = 1023 \times 2^{20}$ are calculated and stored once. Quantization error is corrected by the value of c. The values of a, b, and c presented in this section are distinct from those values as presented elsewhere in this work. The minimum RMS error as compared to the original function is obtained for $c = 60,801$. i_{exp} and j_{exp} are stored in adjacent memory locations and the result of (3.8) is allowed to overflow into j_{exp}. The resulting approximation is equivalent to that obtained from a lookup table with 2^{11} entries for integer valued exponents and linear interpolation for real-valued exponents. The quality of the estimates is identical but the computation time is reduced. This method is over 3 times faster than the typical algorithm that is implemented in math function libraries and 1.6 times faster than the equivalent lookup table with linear interpolation.

The reduced relative speed is beneficial but it must translate into a real-time calculation for the application at-hand. $32,768,000$ exponential calculations are required during each sample period to estimate 4 states over 200 possible steer angles over a 10 step horizon with a training set of 4096 data points. On the same desktop discussed above these operations are completed in $15.463ms$ averaged over one thousand samples. Not surprisingly, the nonlinear portion of the method remains the most time consuming. However, through the use of optimized methods of computation an exhaustive search for the optimal control effort becomes feasible.

3.6 Conclusion

A suitable method of modeling lateral and roll dynamics of a passenger vehicle for model predictive control was presented. The model presented here was formed using Gaussian processes to learn, and later predict, the states of a passenger vehicle relevant to constraining the roll dynamics. Additionally, a method of performing nonlinear constrained optimization with the model was presented and shown to be computationally simple enough to be performed within the sampling period typical of stability control systems. The accuracy and speed of this model will be put to use in the next chapter.

First, training and validation of the model was described and evaluated for accuracy. Single-step prediction for a series of double lane changes demonstrated that training was successful. Examination of the posterior variance associated with these estimates provided a measure of confidence in the estimates. Multi-step prediction was performed over the same validation dataset and the uncertainty in the estimates was propagated to the prediction horizon. The measured values of the states consistently fell with in the 2σ bounds of the estimates. This comparison showed the predictions were often more accurate than evident by examining the bounds taken from the posterior variance.

Second, it was shown that an exhaustive search for the optimal steer angle was made reasonable through the nature of the Gaussian process regression calculations. The nonlinear projection of the input to each Gaussian process onto the feature-space is time consuming. However, the uniformity of the projections allows for optimization of the implementation and straight-forward parallelization. After projection onto the feature-space, the calculation of multiple estimates is reduced to a single matrix-vector operation. This type of linear operation is the foundation of modern processing and is efficiently implemented by a number of currently available devices.

Accurate and timely predictions were demonstrated in this chapter by using a Gaussian process to learn the nonlinear state-space model. Where other models suffer from degraded accuracy due to linearization [16, 18, 19, 20, 21, 22] this approach offers a path forward in implementing MPC using a nonlinear model. The computations required were performed on a modest desktop

PC that is commercially available. An expensive and intricate cluster of machines was not necessary as would be the case with a more conventional approach to computation. With the ability to make accurate predictions quickly, the performance of the model must be demonstrated within a model predictive controller and additional computational savings should also be investigated.

Chapter 4
Large Dataset Approximation

4.1 Introduction

Gaussian process regression offers accuracy in the presence of measurement noise and the ability to describe nonlinear functions without prior knowledge of the functions. These attractive features of the technique come at a price, namely a large computational burden due to the large datasets required. GP regression is being used in a growing number of applications, however this burden provides an outstanding issue to be addressed. Methods to incorporate the information contained in these large datasets will need to account for the the limitations of contemporary computing systems. It is also paramount that any method to reduce computation should limit its effects on model accuracy to preserve the benefits of GP regression that encourage its use.

The work in this chapter aims to reduce the computation required to make accurate estimates with GP regression through the sparsification of the covariance matrix, K. Rather than selecting a subset of points and discarding the remaining samples of the underlying function, the information gained through extensive sampling is incorporated into a reduced-sized set of points that are characteristic of the sampled data as well as the underlying function. A reduced-sized covariance matrix allows faster calculation of estimates as required for distant-horizon prediction of dynamic systems. Without a method of reducing computation, the use of GP in predictive control and distant-horizon prediction will continue to be limited to systems with "slow" dynamics.

The remainder of this chapter begins with a review of previously developed techniques. A new method of reducing computation is then detailed with the use of a simple problem before being expanded for use in higher dimensional problems and dynamics systems. The selection of algorithm parameters is discussed and a suggested method of selection is presented.

4.2 Previous Work

Already, a number of methods to reduce computation in Gaussian process regression and classification have been presented in Chapter 1. Examining the literature regarding support vector machines (SVM), a closely related method, reveals a number of works where clustering of the data was employed to reduce the size of the covariance matrix prior to inversion. These works use SVMs in the role of a classifier rather than a regressor, but the similarities in the underlying algorithm remain the same. Zhang used the average value of training data within each class to train a SVM [55]. Two separate works by Koggalage [56] and Wang [57] clustered training data using the k-means algorithm prior to SVM training. Such an approach requires the choice of how many clusters will be formed prior to processing data. This issue of selecting algorithm parameters exists in the work by Qi [58] that employs fuzzy C-means clustering in addition to another parameter that accounts for soft boundaries between clusters. However, these works leave parameter selection as an open issue.

Wang uses k-means clustering for classification [59] and suggests a heuristic method of selecting the number of clusters. Boley uses Principal Direction Divisive Partition clustering in [60], a method dependent on selection of initial clusters but allowing for variation in this number during processing of the dataset.

An alternative clustering method is used by Shaohua in [61] that requires selection of a radius to be used as a threshold. The algorithm used in that work is similar to that presented here though that work focuses on classification rather than regression and is limited to the SVM framework. Choice of the cluster radius is not detailed in that work as it will be here.

Reduction of computation in GP regression is treated with a clustering method in [30]. That work shows that clusters of data points can be described by averaging both inputs and outputs and using the cluster center points to train the GP. The computation versus accuracy compromise can then be managed directly through the adjustment of the clustering threshold. The work demonstrated the technique by modeling the lateral dynamics of a vehicle. While sensitivity to the cluster threshold is discussed no guidelines for its selections were offered. Here, a method for algorithm parameter selection is suggested. Additionally, that work relied on a single cluster threshold for all dimensions of the input-space. This is not an acceptable limitation as the function being regressed may be more sensitive along some dimensions in comparison to others. The extension of the clustering method to allow for varied thresholds is also addressed here.

4.3 Clustering

While clustering has proven useful in classification problems using the related SVM method, the path to using clustering for regression is less clear. Clustering in classification problems is performed by treating all data belonging to a single class as a cluster. In the case of classification all data belonging to a single cluster is easily identified. This is not the case in regression where all data is associated with a location in the input-space. It follows that data located in the same neighborhood in the input-space may be considered a cluster but the definition of what constitutes a neighborhood remains an open issue.

As previously described, a Gaussian process can be thought of as a Gaussian distribution in a higher dimensional space. Given this similarity, a method of averaging multiple samples measured at the same point within the input-space is considered. The presence of noise on the input and output complicate this by causing measurements that would otherwise be considered to have the same location to appear as if they were gathered from unique, but similar, locations in the input-space. Rather than averaging measurements from the exact same locations, measurements that are similar enough are considered to be from the same neighborhood and therefore the associated outputs are averaged. Choosing the definition of similarity is often an important step in finding clusters within a dataset. Here, similar points in the datasets will be within a small Euclidean distance in the input-space given the assumption that a smooth function underlies the noisy data. The task of grouping similar points is often performed using a method of finding clusters.

There are many methods of finding and clustering points within a dataset. A few clustering methods are reviewed below to explore the attributes necessary for simplification of GP regression. Each of these methods require its own set of parameters prior to locating clusters in a dataset. Some works have given casual mention to the use of clustering to reduce computation [26] but have not prescribed a method of clustering nor a method of determining the parameters for the clustering algorithm. The review in the following subsections explores the parameters necessary for some of the most popular clustering methods in order to properly choose the method appropriate for this work.

4.3.1 k-Means Clustering

Among the most commonly used algorithms is k-means clustering. The k-means algorithm, as described in [62], requires the number of clusters to be selected a priori and proceeds to use Euclidean distance to determine similarity. The resulting clusters are spherical in shape and of varying sizes. These attributes of the clusters can prove troublesome given that their purpose is to negate the measurement noise on the inputs. The output may vary more significantly with respect to some inputs than others requiring ovate clusters as opposed to the spherical clusters provide by the k-means algorithm. The size of the clusters should be dependent on the underlying function rather than how much data was sampled in a particular neighborhood of the input-space. As k-means does not limit the size of the cluster, an individual cluster can grow to incorporate measurements from locations that should be considered distinct and, in doing so, degrade estimates of the output. The k-means approach to clustering also results in every point being assigned to an individual cluster. The number of clusters is fixed and must be selected prior to processing the data leaving outliers to be assigned to the nearest cluster. The forced inclusion of outliers within a cluster serves to skew the distribution of points with that cluster and adversely affecting the value assigned to represent its average output.

4.3.2 Hierarchical Clustering

Hierarchical clustering offers an alternative approach and was presented in [63]. The algorithm results in a tree structure that describes the proximity of different measurements. Depending on how proximity is determined, spherical clusters may result. This is not an insurmountable problem with the algorithm but remains undesirable is a single characteristic length scale is assumed as it is in Gaussian process regression. While hierarchical clustering requires no a priori parameter selection, the tree structure needs to be split to describe individual clusters after execution. The choice of how and where to split the tree is an open issue and can directly affect the size of a cluster relative to other clusters. All points will be incorporated into a cluster in a similar manner to the k-means algorithm. This results in sensitivity to noise and outliers as described earlier.

4.3.3 Wilamowska's Algorithm

Hierarchical and k-Means clustering both form clusters based only on how similar the points are in the input-space without regard for the nature of the underlying function. This lack of consideration for the application at hand can result in points being clustered that are not appropriately similar. A method that limits the cluster size will allow for clusters that counter the presence of noise on the inputs but differentiate points that can be explained by a significant change in the underlying function. Wilamowska and Manic describe such a method in [64].

Identifying the clusters within the input-space begins by assuming the first data point of a training set is the location of the first cluster. Each data point is then evaluated in turn. If it is within a maximum Euclidean distance from a preexisting cluster that cluster is adjusted according to

$$\bar{\mathbf{x}}_i^{(j_i+1)} = \frac{j_i \bar{\mathbf{x}}_i^{(j_i)} + \mathbf{x}}{j_i + 1} \tag{4.1}$$

where $\bar{\mathbf{x}}_i$ is the cluster location, $\mathbf{x}$ is the current measurement location, and j_i is the number of measurements currently included in that one cluster. The output values within each cluster are averaged in the same manner. If no cluster is within the maximum distance, a new cluster is formed. A list of cluster center points and average output values for each of the clusters is the output of the algorithm. The index of these clusters, i, ranges from 1 to m where m is the quantity of clusters.

In summary the following steps were taken to form clusters of the input-output pairs:

Algorithm 1

```
(∗ Clustering method described by Wilamowska ∗)
1.  assume first input-output pair to be initial cluster
2.  for each input-output pair in dataset
3.      evaluate Euclidean distance to previous clusters
4.      if distance to any previous cluster is below threshold distance
5.          add input-output pair to closest cluster
6.      else
7.          form new cluster centered on input-output pair
```

This clustering algorithm requires that a threshold distance be determined prior to processing the input-output pairs. A method of determining a suitable value for this parameter is detailed below. The ability to grow the set of clusters as necessary allows the algorithm to handle outliers more gracefully than the k-means and hierarchical methods. The outlier will form its own cluster and not affect the others as is the case with the other algorithms. For this application an outlier indicates a location within the input-space that was not sampled as throughly as others. It may be the only data from that neighborhood and should not be discarded but it is improper to force it into a cluster with an otherwise dissimilar set of input-output pairs. The resulting spherical clusters still prohibit accounting for varying sensitivities to individual inputs. However, a simple transformation of the input-space based on the relative sensitivities to each input can rectify this deficiency.

4.4 Single Dimension Example of Clustering

A one dimensional function is used to demonstrate the results of the method in a manner that is easily visualized. A sinc function,

$$y(x) = \frac{sin\ \pi x}{\pi x} \tag{4.2}$$

provides a nonlinear function for the GP to learn. Measurement noise will be added as necessary for demonstration. Figure 4.1 shows the sinc function along with the uniformly distributed random sample locations used for training.

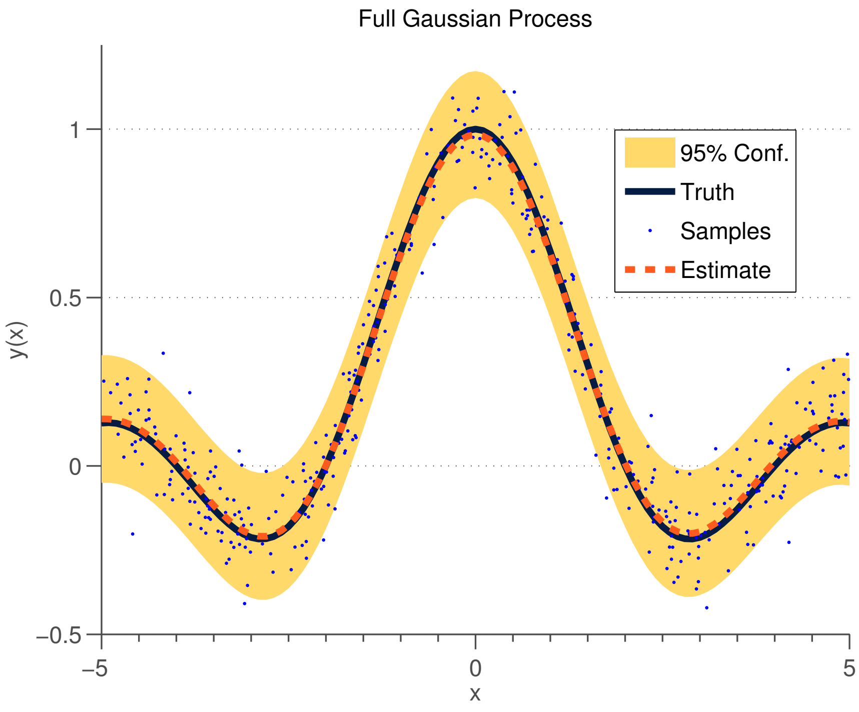


Figure 4.1: A GP trained to learn the sinc function without clustering.

Figures 4.2 and 4.3 demonstrate the effectiveness of clustering on the one dimensional sinc function. The cluster threshold was determined by increasing its value from zero until the entire dataset was incorporated into a single cluster. The RMS error of the estimates and size of the covariance matrix, K, are plotted as functions of the clustering threshold in Figure 4.4. If the threshold is chosen too small the matrix size will remain large and the method will fail to reduce computation. If the threshold is chosen too large dissimilar data points will be averaged and the accuracy of the estimate will be degraded. In the extreme case of selecting an over-sized threshold, all points will be treated as a single cluster and the estimator will only provide the mean value of the training data albeit very quickly. This degraded accuracy is shown in Figure 4.5 and 4.6. Therefore, properly selecting the clustering threshold is critical to the success of the method.

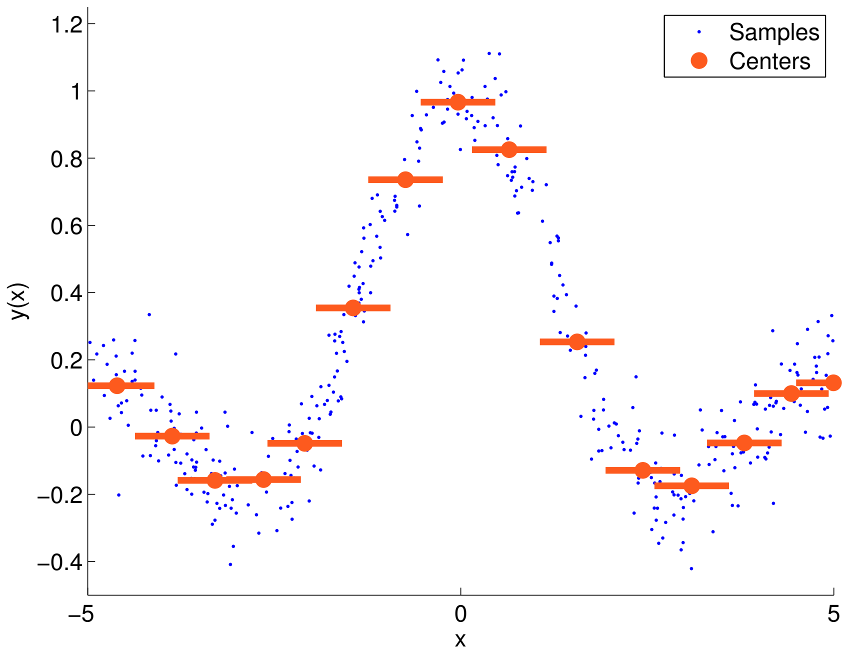


Figure 4.2: Cluster centers found with appropriately sized threshold found through exhaustive calculation.

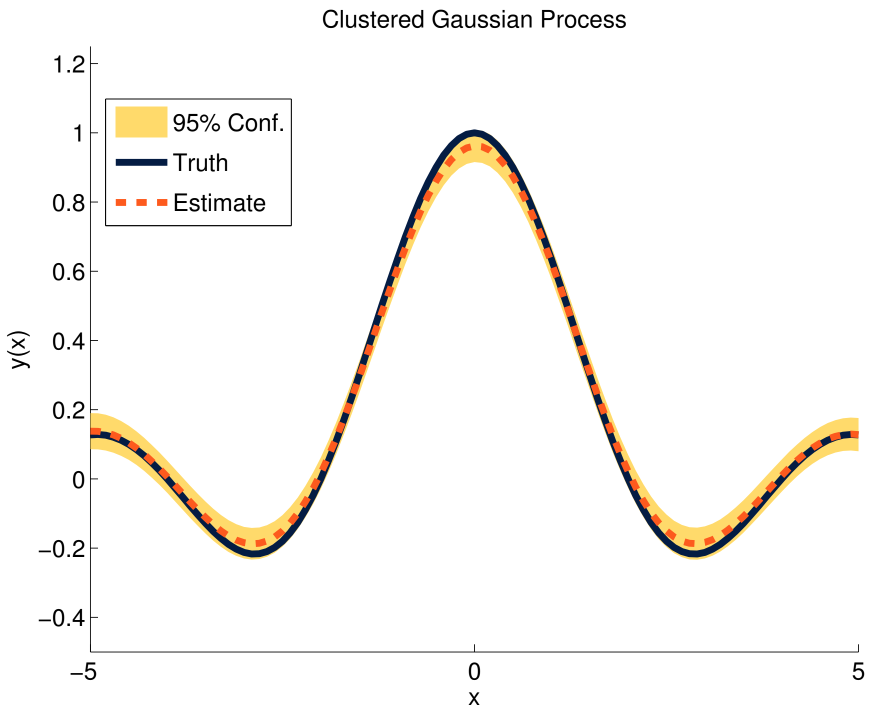


Figure 4.3: GP regression from cluster centers with appropriately sized threshold found through exhaustive calculation.

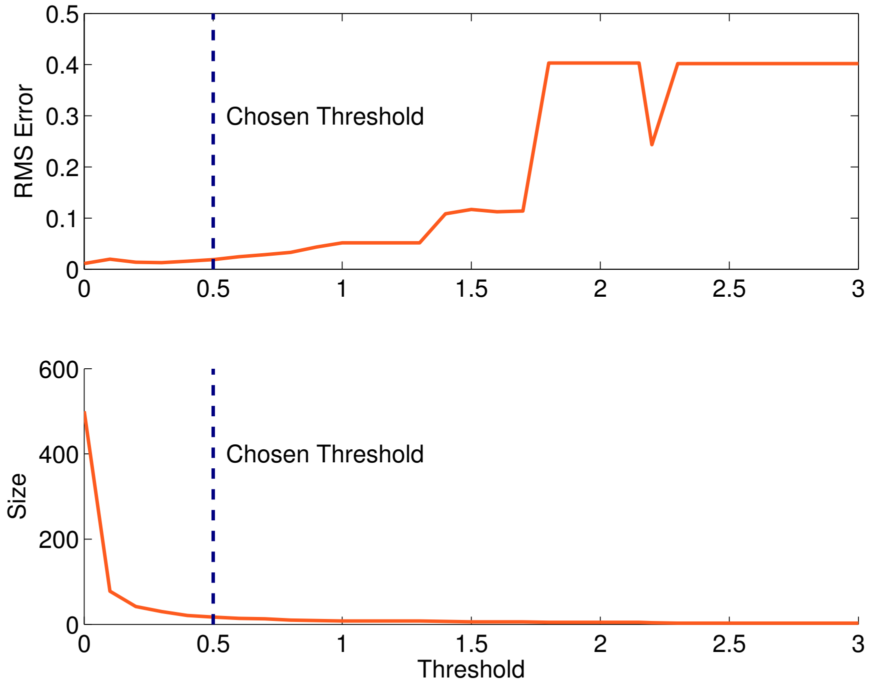


Figure 4.4: Effect of threshold on mean squared error and computation.

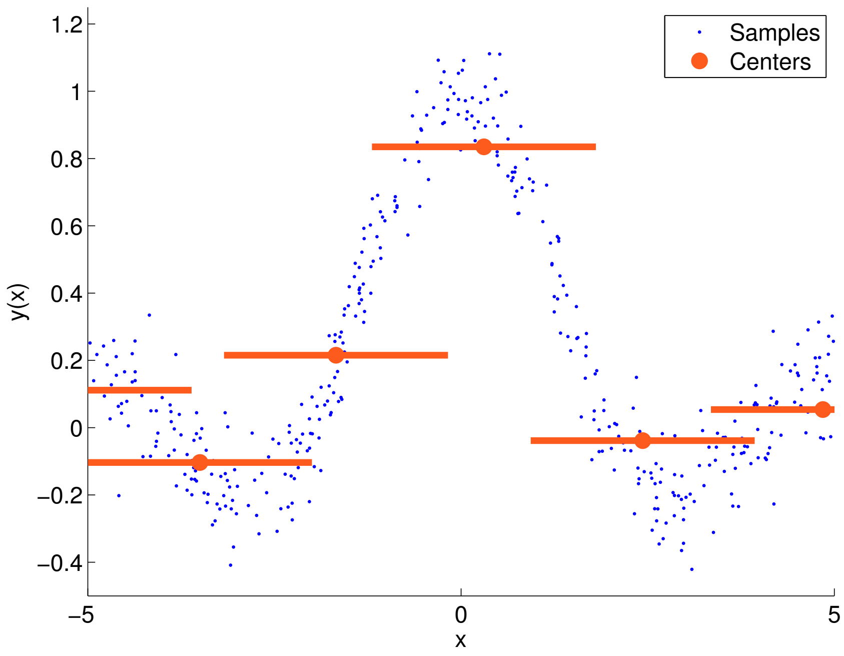


Figure 4.5: Cluster centers found with inappropriately sized threshold.

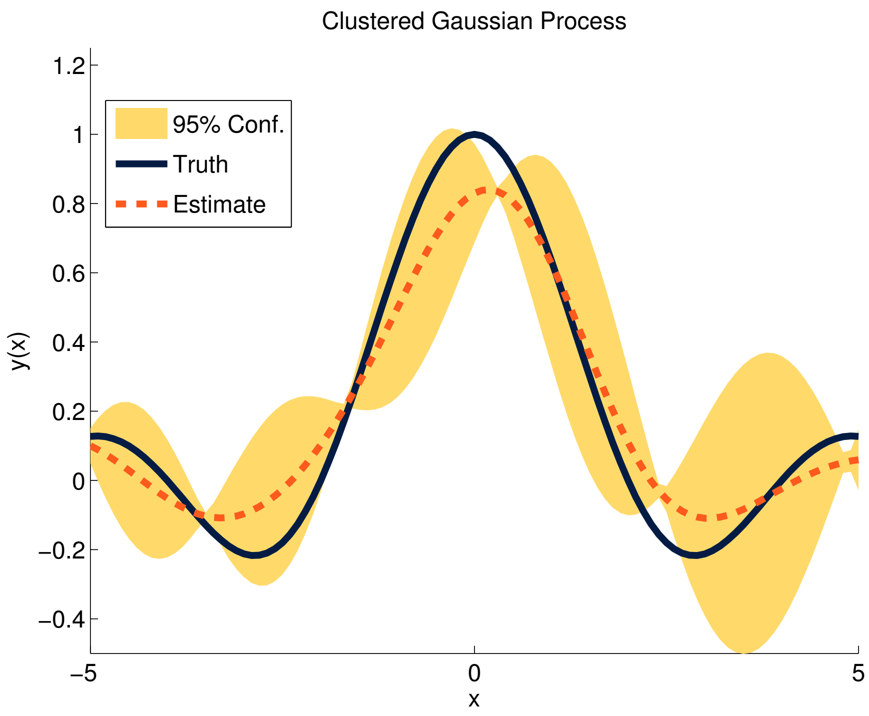


Figure 4.6: GP regression from cluster centers with inappropriately sized threshold.

4.5 Determination of Clustering Threshold

In selecting a clustering method, a meaningful and effective way of determining its parameters must also be provided. The clustered data in this work is intended to group together points within the input-space that are close enough to be considered similar. The proposed clustering algorithm forms clusters with a given radius. However, it is necessary to provide the clustering algorithm with this radius prior to execution. A suggested value for this threshold can be formed based on knowledge of the function being estimated by GP regression.

Among the hyperparameters discussed as part of GP regression were the length scales, l_i, of each dimension of the input-space. The concept of a length scale is covered throughly in [65] though [26] speaks of it informally as "the distance you have to move in input-space before the function value can change significantly". Taking Rasmussen's [26] definition it is clear that the length scale of a function may form a basis for defining similar points within the input-space. This chapter asserts only that in using the length scale as a basis to chose the clustering threshold the training dataset can be reduced without significantly affecting the accuracy of the estimates provided. The length scales of each dimension of the input-space are identified as part of training in GP regression. However, any method that identifies the length scales would be appropriate to determine the clustering threshold parameter.

Two similar points within a one dimensional neighborhood with a diameter equal to the length scale may be clustered together appropriately. The clustering threshold defines a radius around a cluster center within which similar points should lie. Adding an additional factor of two leads to the selection of a clustering threshold that is one fourth the length scale. Applying this rule to (4.2) results in cluster centers as shown in Figure 4.7 and the corresponding estimates in Figure 4.8.

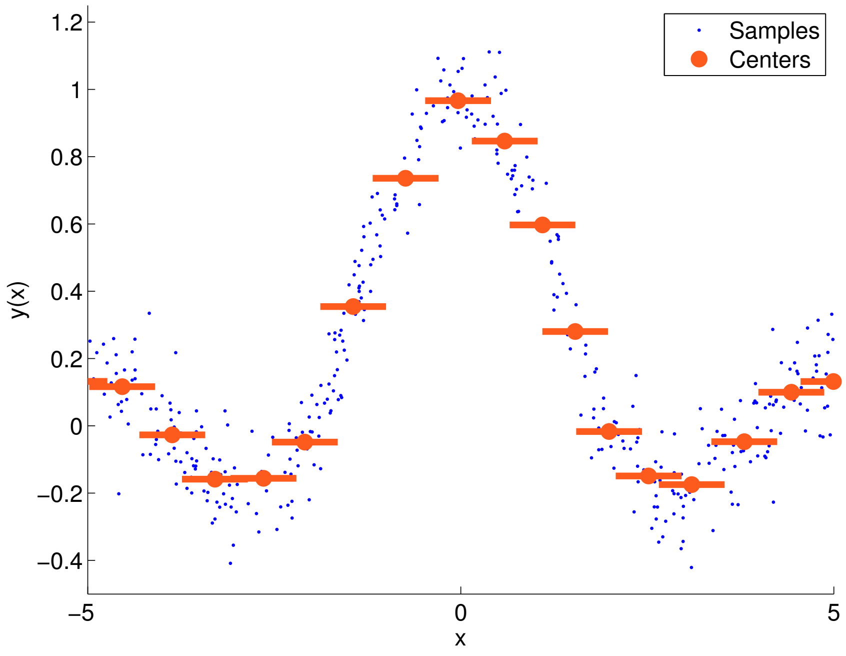


Figure 4.7: Cluster centers with clustering threshold from length scale estimate.

The frequency of the sinc function was varied to demonstrate the effectiveness of this criteria for a variety of length scales. The function

$$y(x) = \frac{\cdot \sin \omega\pi x}{\omega\pi x} \tag{4.3}$$

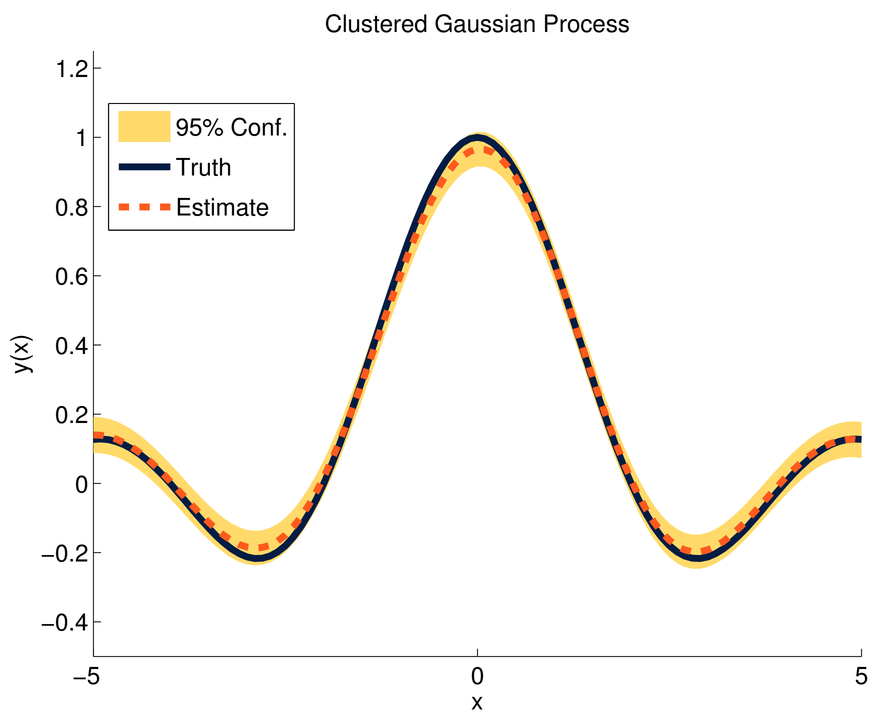


Figure 4.8: Estimates of sinc function with clustering threshold from length scale estimate.

was sampled 500 times on the interval $[-5,5]$ while varying values of ω from 1 to 10. Figure 4.9 shows the RMS error of the GP estimates from unclustered and clustered datasets over the same interval.

It is clear that the values match closely, although the clustered RMS is typically worse to a small degree. The trending as ω is increased can be explained by the frequency of sampling in comparison to the frequency of the sinc being sampled. This is not a concern as the trending is seen in both the unclustered estimates and the clustered estimates.

The size of the covariance matrix is also shown in Figure 4.9. Each unclustered GP was the size of the sampled data, or 500 by 500. After clustering, considerable reduction in size could be seen for all values of ω shown. As the frequency of the sinc function increased, the length scale found while training the unclustered GP became smaller (calling for smaller clusters). The size of the covariance matrix grew with the decreased size of the clusters as expected in order to preserve the accuracy of the estimates provided. This method of clustering and selecting the clustering

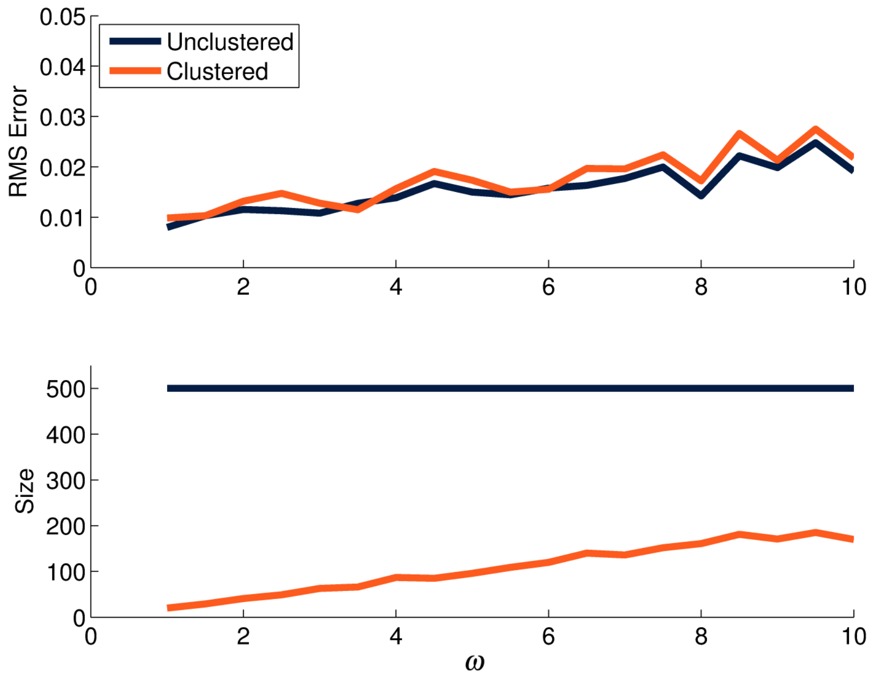


Figure 4.9: RMS estimate error and matrix size before and after clustering for functions with decreasing length scale.

threshold best suits applications where estimate accuracy is the focus and computation time is less critical but still requires reduction.

4.6 Application to Higher-Dimensional Input Space

Clustering is demonstrated on a two dimensional function to extended the example from the previous section. The two dimensional sinc function

$$z(x) = \frac{sin\ \pi\sqrt{x^2+4y^2}}{\pi\sqrt{x^2+4y^2}} \tag{4.4}$$

was sampled one thousand times within the intervals $x = [-5,5]$ and $y = [-5,5]$ to form the training dataset. The full training dataset was then used to learn the function and form a baseline for prediction accuracy prior to clustering. The estimates generated using GP regression and the full training dataset are shown in Figure 4.10.

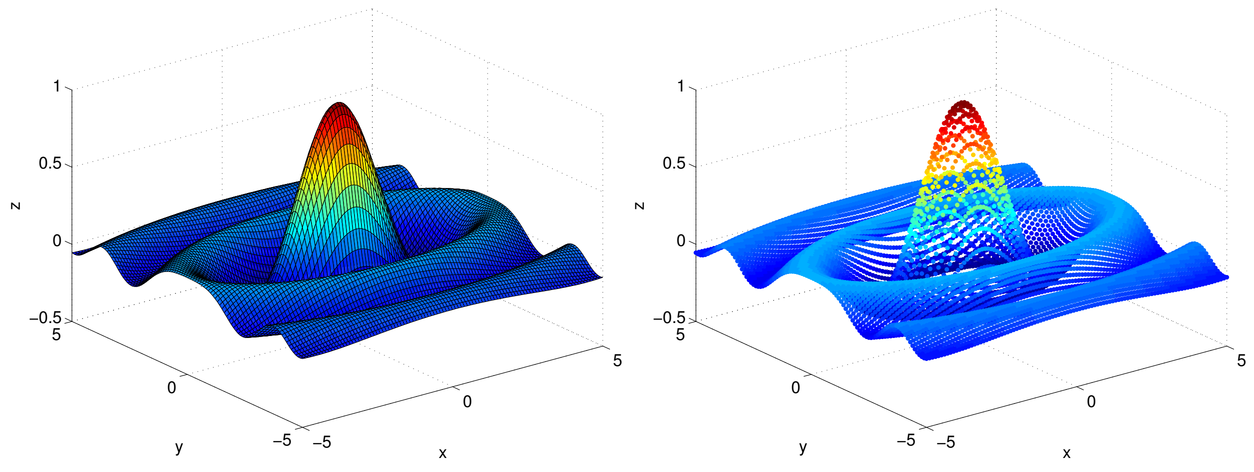


Figure 4.10: 2D sinc function with varied length scales (left). Estimates from full GP of 2D sinc function (right).

Training with the full dataset yielded length scales of $l_x = 2.420$ and $l_y = 1.201$. Each point in the input-space was transformed to accommodate the differing values of the length scales using the transformation

$$\begin{bmatrix} x_T \\ y_T \end{bmatrix} = \text{diag}\left(\begin{bmatrix} x \\ y \end{bmatrix} \cdot \begin{bmatrix} {}^{4}\!/_{l_x} & {}^{4}\!/_{l_y} \end{bmatrix} \right) \tag{4.5}$$

as suggested previously in Section 4.5. Clustering of the training dataset was then performed with a threshold of one using the transformed inputs x_T and y_T. The resulting cluster centers were then transformed back into the original input-space. Figure 4.11 shows the location and shape of the clusters. The ovate shape of the clusters is introduced by the process of transforming the input-space and accounts for the functions varied sensitivities to the inputs x and y in this example. The more sensitive dimension, y, requires the clusters to be more dense while the x dimension allows for a more sparse clustering. Ignoring the varied sensitivities would require a compromise to be made. In choosing an isotropic clustering threshold according to the length scale of the y dimension would yield accurate estimates but at the expense of computation time. Choosing an isotropic clustering threshold according to the length scale of the x dimension will cause dissimilar samples to be clustered together at the cost of estimate accuracy.

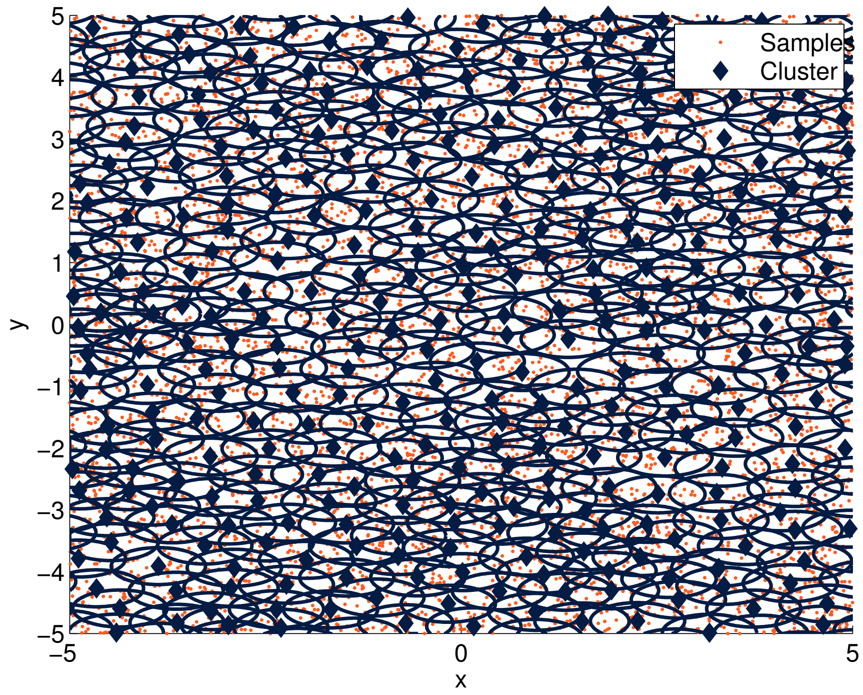


Figure 4.11: Clusters in two dimensions found using length scales and transformed input-space.

The cluster centers were used to train a GP and estimates over the same interval were made. The estimated surface is shown in Figure 4.12.

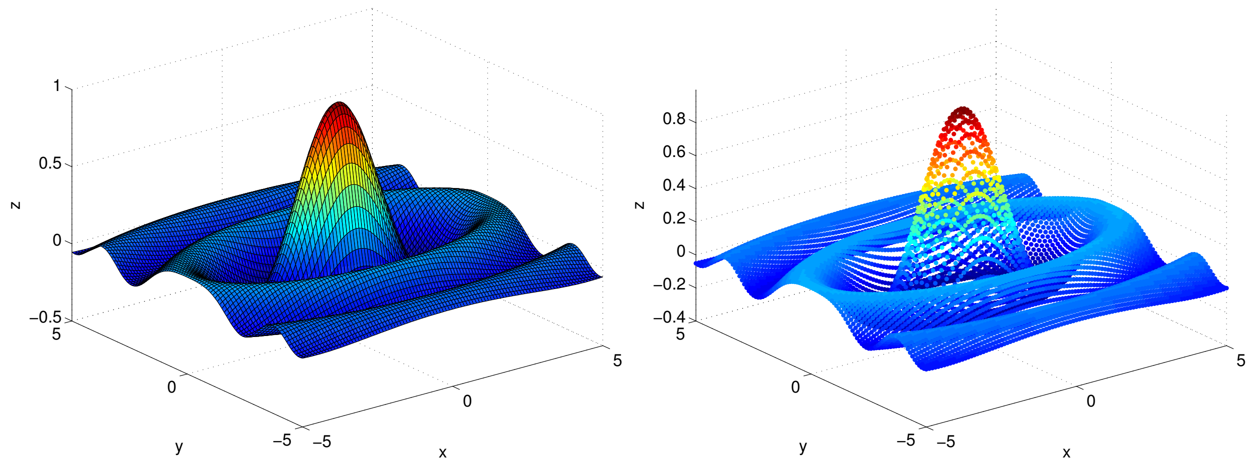


Figure 4.12: 2D sinc function with varied length scales (left). Estimates from clustered GP of 2D sinc function with transformed input-space to allow for ovate clusters (right).

The RMS error of the estimates prior to clustering was 0.0004407. After applying clustering with the threshold chosen according to the suggestion made above the RMS error was .006834.

Table 4.1: GP input length scales for each state estimator from recorded data

Estimator	l_β	l_r	l_ϕ	$l_{\dot{\phi}}$	l_{V_x}	l_δ
$\hat{\beta}_{k+1}$	10.7770	1.569	0.212643	1.7382	15.7994	160.6614

While some accuracy was sacrificed it was not a significant degradation. However, the size of the covariance at the heart of GP regression was reduced significantly. Using the full dataset required a 1000 by 1000 covariance matrix. After clustering was applied the covariance matrix was reduced to 490 by 490 thus reducing the time required to generate estimates and train the new GP.

4.7 Application to Dynamic Systems

The example above demonstrates the clustering algorithm on a two dimensional problem but scales nicely to the higher dimensions required by the dynamic vehicle models in this dissertation. The clustering algorithm was applied to the sideslip estimator developed in Chapter 3. The length scales found during training with the full training dataset are listed in Table 4.1. The input-space of the training dataset was transformed according to (4.5) and clustering was performed. Figure 4.13 shows the single step predictions for both the unclustered and clustered training data. Visually the two estimators match each other and the truth measurements well. Prior to clustering the RMS error was 0.3190 using a covariance matrix of size 4048 by 4048. After clustering the RMS error was 0.3635 using a covariance matrix of size 818 by 818. Again, a small sacrifice in accuracy (13.95%) yields a training dataset that is less than a quarter the size of the unclustered dataset. Given the $\mathcal{O}(n^2)$ scaling for a naive calculation of estimates and the $\mathcal{O}(n^3)$ scaling for matrix inversion during training this reduction in size reduces computation time effectively. Furthermore, the results for this problem are consistent with those of the illustrative examples presented above indicating that the method is applicable to more practical problems. In the Chapter 5, it will be shown that the sacrificed accuracy does not prevent control and constraint of the vehicle dynamics.

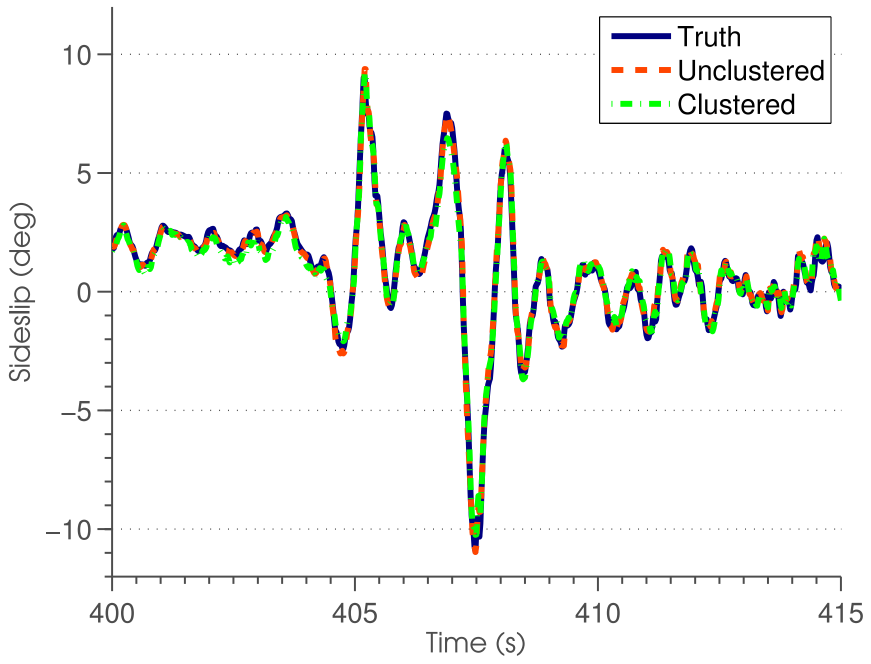


Figure 4.13: Sideslip estimate using clustered training data.

4.8 Conclusion

A clustering method of reducing the computation required for Gaussian process regression was presented. Clustering had been applied to classification problems using the related support vector machine method of learning where similarity between points was easily distinguished. The issue of finding similar points for clustering in GP regression was addressed here using a clustering method that parameterized the size of clusters rather than the quantity. The size of the clusters was dictated by a clustering threshold that corresponds to the maximum euclidean distance between points and the cluster center.

A suggested value for this clustering threshold was made and tested on a variety of problems These problems included illustrative examples as well as data recorded for a dynamic model of an ATV. The initial example showed the easily visualized single dimension problem and tested the efficacy of the suggested value for a function with varying frequency. Higher dimensional functions with varying sensitivities (length scales) to individual input dimensions was addressed using a transformation to the input-space. Clustering was applied within the transformed input-space.

A two dimensional function was developed to illustrate the method of clustering and parameter selection. A comparison was made with a GP trained using the unclustered data and a GP trained using the cluster centers. The results of that comparison was presented and evaluated.

Finally, the clustering method was applied to data recorded from an instrumented all-terrain vehicle and used to predict the sideslip of the vehicle. This problem demonstrated the successful application of the technique to data taken from a practical problem and required clustering be performed in a six dimensional space to reduce the computation required for training and prediction of the vehicle dynamics.

Chapter 5

Gaussian Process Models in Model Predictive Control

5.1 Introduction

In Chapter 3 a model of the lateral and roll dynamics of a passenger vehicle was developed using Gaussian process regression to learn the dynamics of the vehicle. The speed of computation was examined when making both single and multiple estimates with this model over the distant horizon called for by model predictive control. Thorough validation of the model showed that the nonlinear nature of the vehicle behavior was accurately captured for both single-step and multi-step predictions. Additional computational savings were demonstrated in Chapter 4 through the clustering of training data prior to regression. In that chapter it was shown that significant computational savings could be realized with a small but acceptable degradation in prediction accuracy.

This chapter combines and demonstrates those two works to effectively constrain roll dynamics in a model predictive controller. The use of simulation in this chapter allows for additional maneuvers to be used in validation where safety concerns prohibited their use in previous experiments with an all-terrain vehicle (ATV). The component steps of the model predictive controller are described and detailed to make clear each component's role in realizing an effective approach to control. The constraints placed on the roll states are described before the controller is demonstrated for two maneuvers that pose the risk of rolling the vehicle.

5.2 Model Predictive Controller (MPC)

In MPC, a dynamic model is used to predict the system's open-loop response to control inputs over a finite horizon. The control input that optimizes the cost function,

$$J = \sum_{k=1}^{N} \mathbf{w}_{\mathbf{z}_k} (\mathbf{r}_k - \mathbf{z}_k)^2 + \sum_{k=1}^{N} \mathbf{w}_{u_k} \delta u_k^2 \tag{5.1}$$

with respect to a reference trajectory, $\mathbf{r}_k$, and the predicted state, $\mathbf{z}_k$, over the finite horizon, N. Weights $\mathbf{w}_{z_k}$ and $\mathbf{w}_{u_k}$ are left to tune the controller. Constraints can be placed on the state and inputs explicitly defined by the inequality

$$C(\mathbf{z}_k, u_k) < 0 \tag{5.2}$$

and can be either linear on nonlinear

The procedure implemented by the MPC at each time step can be stated as:

1. Generate reference trajectory, $\mathbf{r}_k$, over a finite horizon of N steps.

2. Find the optimal set of inputs to the system that minimizes the cost function in (5.1) as a function of the predicted state, $\mathbf{z}_k$, error and control effort. This optimization is subject to the constraints described in (5.2).

3. Apply the first control effort in the set of inputs to the system.

While only open-loop predictions are computed, a closed-loop control is formed by recomputing the optimal input to the system at each time step. MPC is often referred to as receding horizon control given that the distance to the horizon is fixed and the location is relative to the current time step.

Here the controller will be described as three separate components: the reference generator, the predictive model, and the constrained optimizer. These components are shown in Figure 5.1 and are described in the following sections.

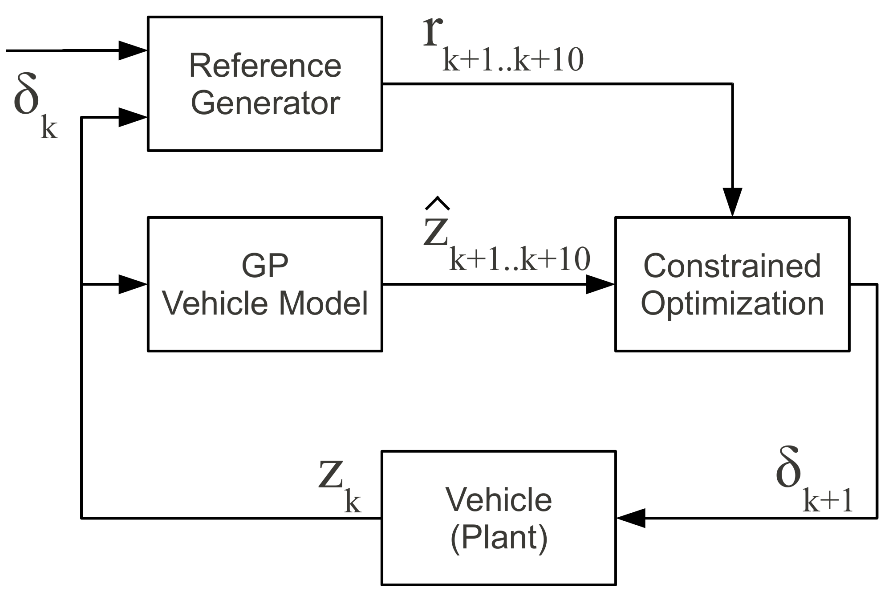


Figure 5.1: GP-based model predictive controller block diagram.

5.2.1 Reference Generator

The reference generator is used to take input from the driver and produce a desired state trajectory over the finite horizon. The controller described here controls the lateral dynamics while constraining the roll dynamics. The reference generator should provide a trajectory the vehicle is capable of following and the driver is accustom to. As in [22] and many other works, the bicycle model is suitable for this task. A diagram of the bicycle model is shown in Figure 5.2 and the dynamics are described by

$$\dot{x}_{rg} = \begin{bmatrix} \frac{-(C_{\alpha f}+C_{\alpha r})}{mV_x} & \frac{-(aC_{\alpha f}-bC_{\alpha r})}{mV_x^2}-1 \\ \frac{-(aC_{\alpha f}-bC_{\alpha r})}{I_{zz}} & \frac{-(a^2C_{\alpha f}+b^2C_{\alpha r})}{I_{zz}V_x} \end{bmatrix} x_{rg} + \begin{bmatrix} \frac{C_{\alpha f}}{mV_x} \\ \frac{aC_{\alpha f}}{I_{zz}} \end{bmatrix} \delta \tag{5.3}$$

The roll dynamics of the vehicle are important only for the purposes of constraining the behavior of the vehicle. Therefore, the reference trajectory is only defined by the lateral states for sample times $k+1$ through $k+10$. The longitudinal dynamics are not controlled so a reference trajectory is not generated for V_x though it is measured from the vehicle and provided as an input to the reference generator. The remaining parameters of the bicycle model are listed in Table 5.1.

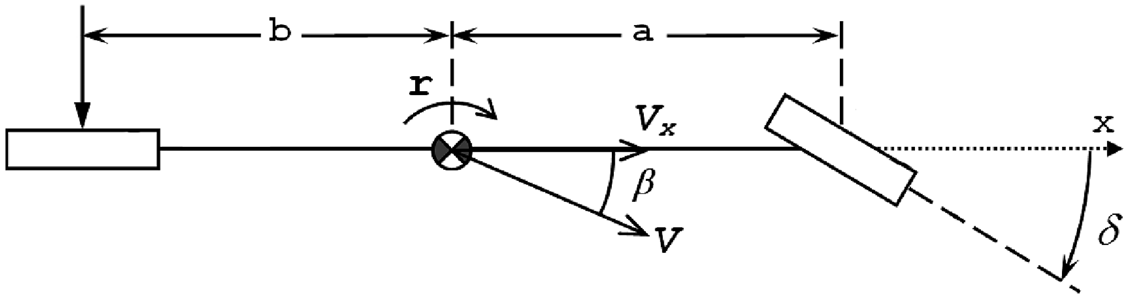


Figure 5.2: Bicycle model used as reference generator in model predictive controller.

Table 5.1: Parameters used in reference generator from CarSim Small SUV model.

Parameter	Description	Value	Unit
$C_{\alpha f}$	Front cornering stiffness	57,295.5	N/deg
$C_{\alpha r}$	Rear cornering stiffness	57,295.5	N/deg
a	Distance from cg to front axle	0.88	m
b	Distance from cg to rear axle	1.32	m
m	Total mass of vehicle	1142	kg
I_{zz}	Yaw inertia	1296	$kg \cdot m^2$

5.2.2 Gaussian Process Predictive Model

The modeling technique based on Gaussian process regression developed in the previous chapters was again applied to simulated data. the software package CarSim provides high-fidelity models of a number of passenger vehicles. Using CarSim allows for an additional validation maneuver to be performed, namely, the NHTSA fishhook (FH) that had been excluded to prevent destruction of the instrumented vehicle previously presented. The simulations described here use the "small SUV" model from CarSim as it is a similar vehicle to the Prowler ATV modeled in Chapter 3.

Again, the model takes a nonlinear state-space form

$$\begin{aligned} \mathbf{z}_{k+1} &= G(\mathbf{z}_k, u_k) \\ y_k &= H(\mathbf{z}_k, u_k) \end{aligned} \tag{5.4}$$

leaving the Gaussian process to learn the function $G(\mathbf{z}_k, \delta_k)$ which assumes the state is fully observable. The state vector is defined as

$$\mathbf{z} = \begin{bmatrix} \beta \\ r \\ \phi \\ \dot{\phi} \end{bmatrix} \tag{5.5}$$

as in Chapter 3 previously. The vehicle was simulated performing a number of maneuvers to provide training data for use in GP regression. The training dataset was comprised of the data from a series of sinusoids with increasing amplitude. One such maneuver is depicted in Figure 5.3. The training dataset consisted of similar sinusoids with frequencies ranging from $0.5^{rad}/_{s}$ to $15^{rad}/_{s}$ in $0.5^{rad}/_{s}$ increments. Each of these increments provides data between which inference is performed to generate estimates.

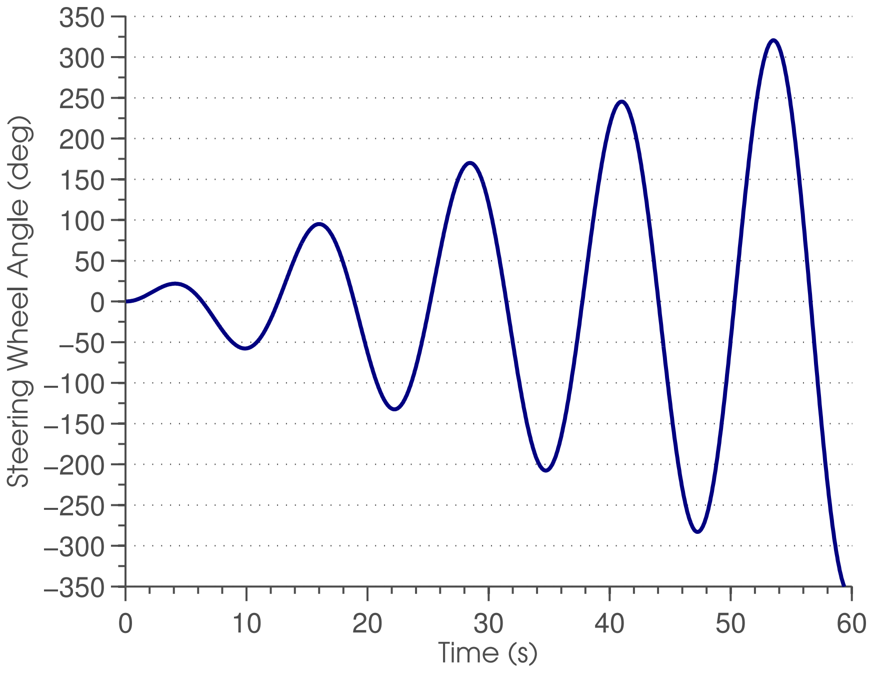


Figure 5.3: Typical training maneuver used in simulated maneuvers.

The clustering method of reducing computation time detailed in Chapter 4 was applied to the training data prior to validation. After initial training, the length scales listed in Table 5.2 were used to transform the input-space of the GPs according to (4.5) to allow circular clusters to be formed.

Table 5.2: GP input length scales for each state estimator used in simulation

Estimator	l_β	l_r	l_ϕ	$l_{\dot{\phi}}$	l_{V_x}	l_δ
$\hat{\beta}_{k+1}$	3.8756	37.220	5.9185	18.993	24.398	182.93
$\hat{r}_{k+1}$	3.5887	41.494	3.9185	43.434	2.3543	268.64
$\hat{\phi}_{k+1}$	4.3066	47.747	3.7480	17.191	16.028	471.44
$\hat{\dot{\phi}}_{k+1}$	2.1504	16.774	1.9302	9.5511	13.151	126.88

Two maneuvers were used to validate the GP model after training and clustering. The first mimicked the validation maneuver performed in Chapter 3. This double lane change (DLC) maneuver is a typical evasive maneuver performed within two road lanes. The input to the vehicle, δ_{SW}, the steer angle at the steering wheel is shown in Figure 5.4.

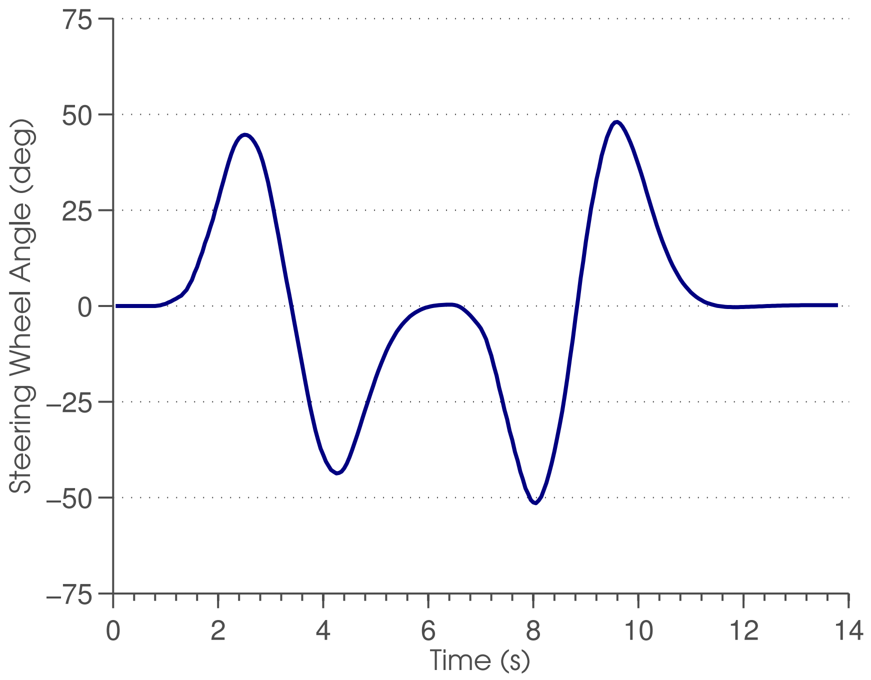


Figure 5.4: Steering wheel input during double lane change maneuver used for validation.

The second validation maneuver was developed by NHTSA to examine a vehicle's proclivity to rolling during extreme maneuvers. The fishhook test is intended to test the vehicle's stability under conditions similar to those found when a driver leaves the roadway and over corrects in an attempt to return to the roadway. The input for this maneuver is shown in Figure 5.5. After a period of driving straight the steering wheel is turned to the right to $300deg$ at $720^{deg}/_{s}$. Following this, the wheel is turned to the left to $-300deg$ at $-720^{deg}/_{s}$.

The success of training can be evaluated by examining the GP model's performance on both the DLC and FH validation maneuvers. Examining the GP models' performance on maneuvers that are dissimilar from the training maneuver evaluates the ability of the model to describe the vehicle dynamic in general and ensure that the model was not overfit to the training data. The single step predictions of the lateral states are shown in Figures 5.6 and 5.7 for the DLC and FH maneuvers respectively. The single step predictions of the roll states are shown in Figures 5.8 and 5.9 for the DLC and FH maneuvers respectively. All four figures show the measured values falling within the 2σ bounds as expected. This initial examination reveals that the training was

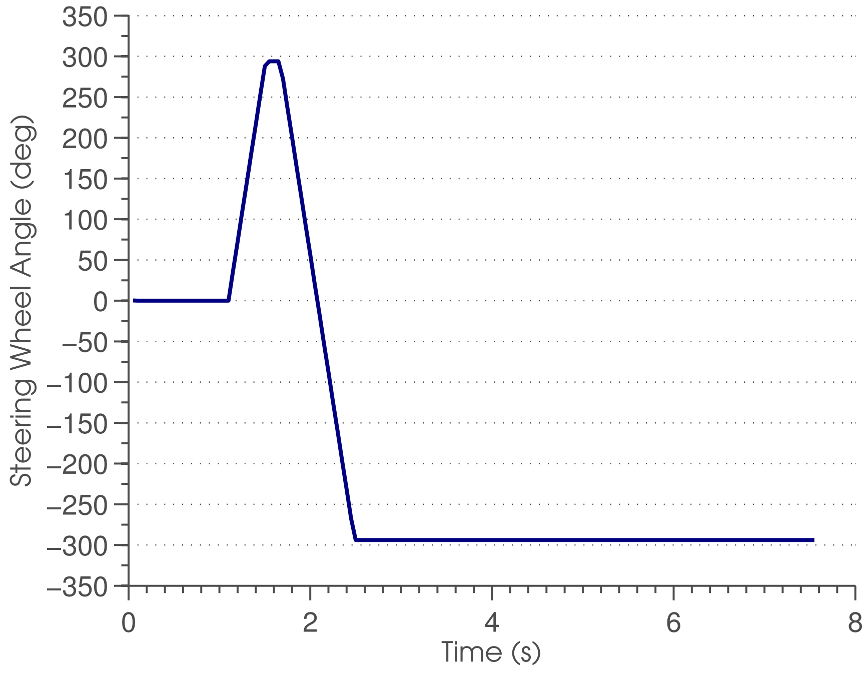


Figure 5.5: Steering wheel input during fishhook maneuver used for validation.

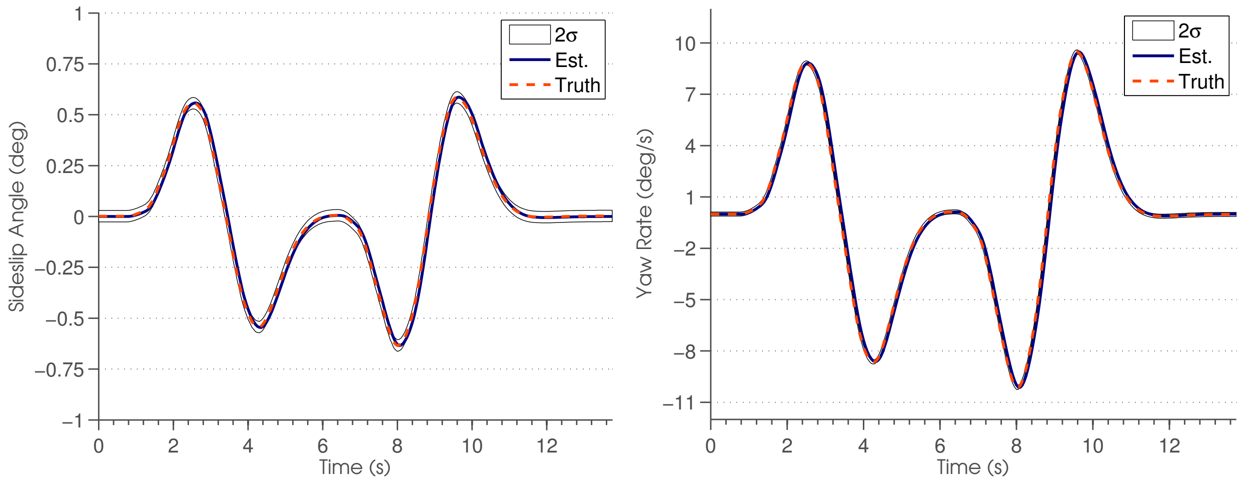
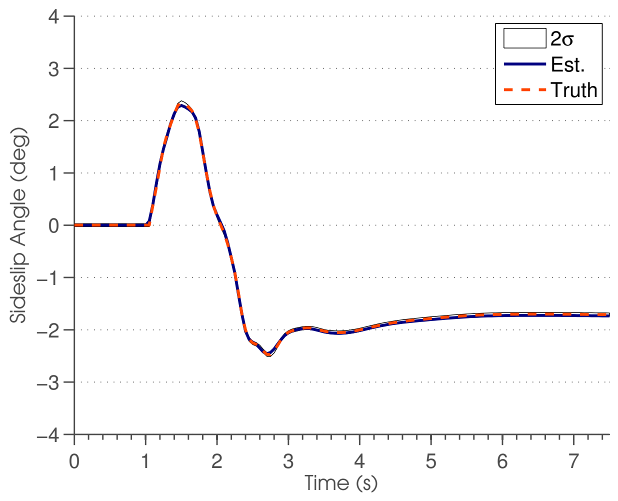


Figure 5.6: Single step double lane change validation of lateral estimates.



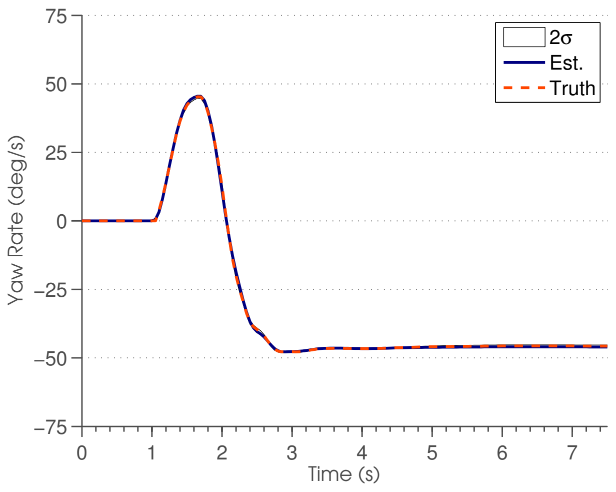
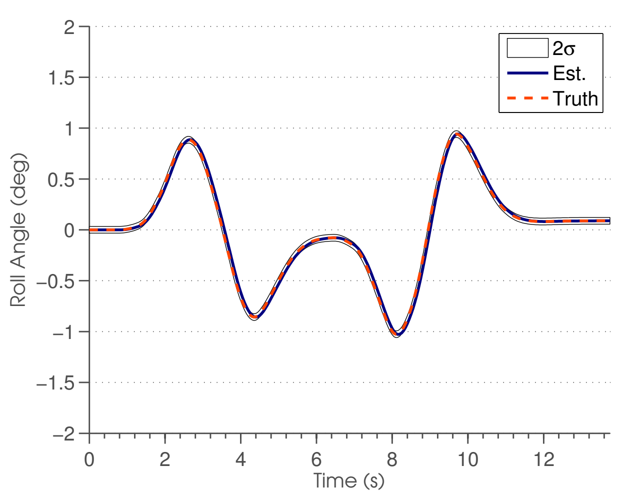


Figure 5.7: Single step fishhook validation of lateral estimates.



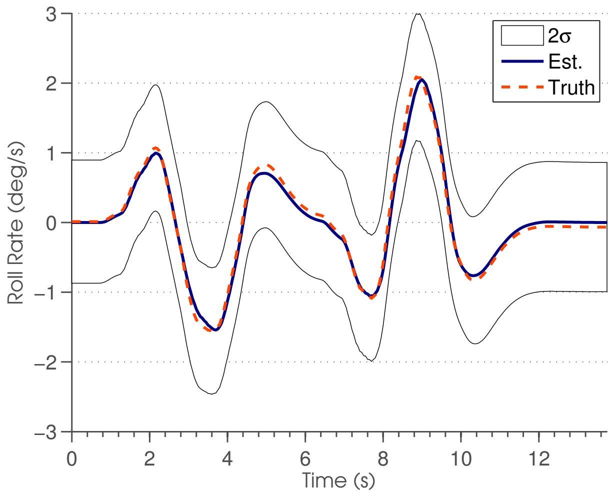


Figure 5.8: Single step double lane change validation of roll estimates.

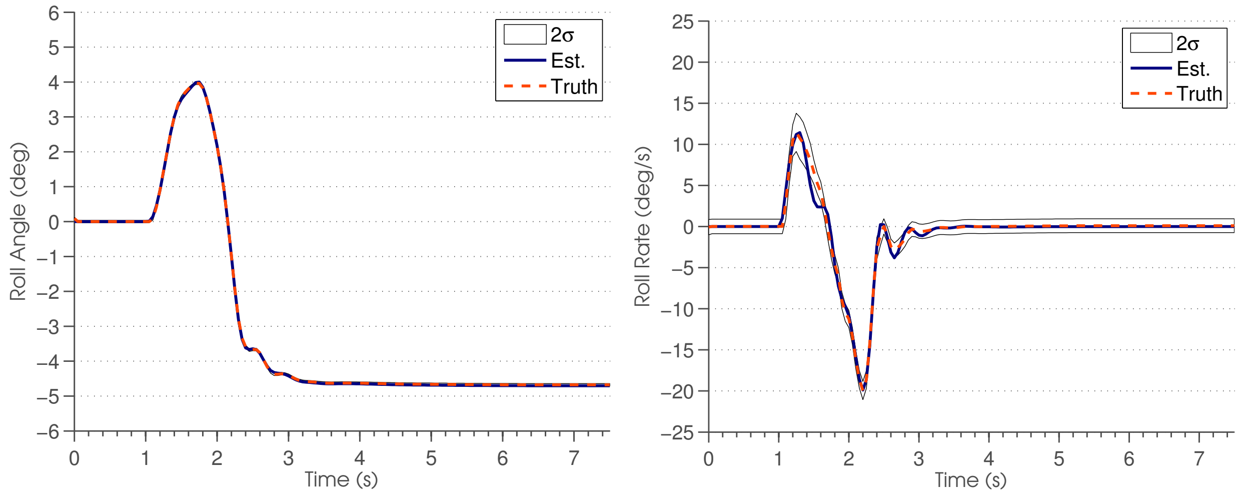


Figure 5.9: Single step fishhook validation of roll estimates.

successful and the GP model can be expected to describe the dynamics of the vehicle in general rather than only for the training maneuvers.

MPC calls for the multi-step prediction over a finite horizon. A ten-step horizon was adopted here as it was a common, though arbitrary, choice in previous works[20, 22]. Feedback was introduced to the single-step GP model to produce the multi-step estimates. A depiction of the four-state multi-step GP model is shown in Figure 5.10. Again, the accuracy of the estimates must be evaluated on maneuvers separate from the training data. The ten-step predictions for the lateral states are shown in Figures 5.11 and 5.12 respectively for the DLC and FH maneuvers. The estimates shown were generated using $\mathbf{z}_{k-10}$ as initial conditions and iterating 10 times to calculate and estimate for $\hat{\mathbf{z}}_k$. The uncertainty for each estimate reflects the accumulated uncertainty over each iteration as prescribed by Girard in [31] and detailed in (3.3). Similar ten-step predictions for the roll states are shown in Figures 5.13 and 5.14 respectively. Again, the measured values of each state agree with the ten-step prediction in that they lie within the 2σ bounds that accompany each estimate.

The results of the computation-reducing clustering method are summarized in Table 5.3. Each individual state estimator is based on a separate GP. All four GP estimators were trained with four thousand points chosen randomly from the entire training dataset. Any reduction in size would be acceptable as the training of each GP scales as $\mathcal{O}(n^3)$ and the calculation of estimates scales as

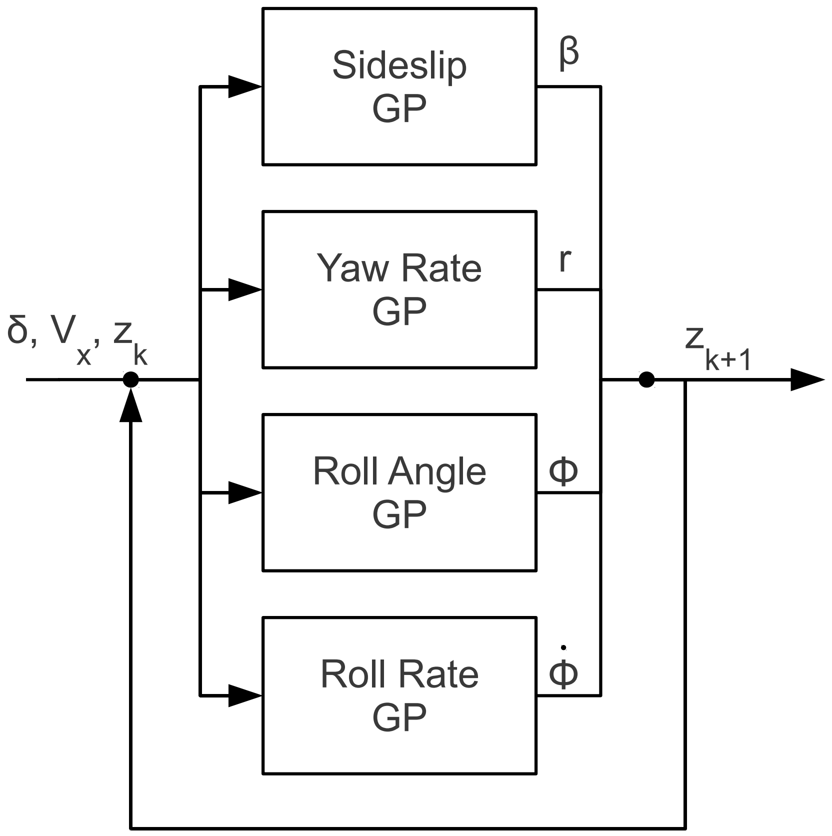


Figure 5.10: Predictive vehicle model comprised of four Gaussian processes corresponding to four vehicle states.

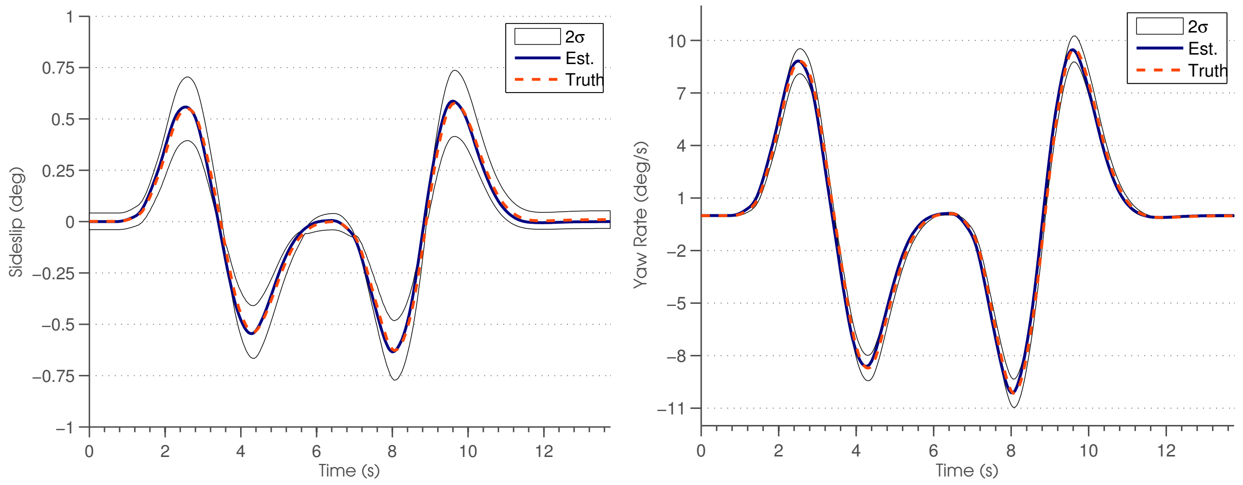


Figure 5.11: Double lane change validation of lateral estimates using ten step prediction.

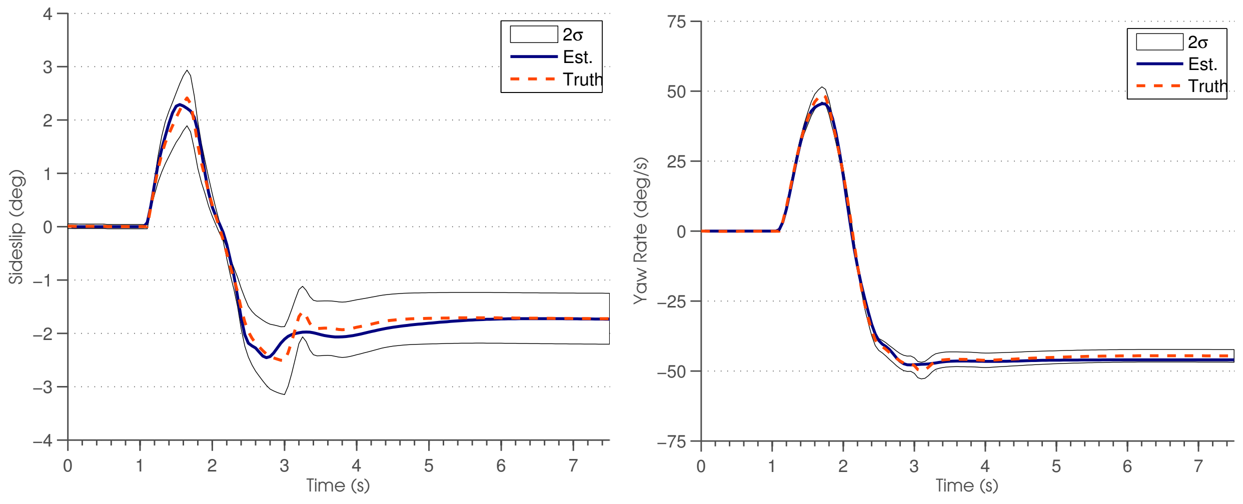


Figure 5.12: Fishhook validation of lateral estimates using ten step prediction.

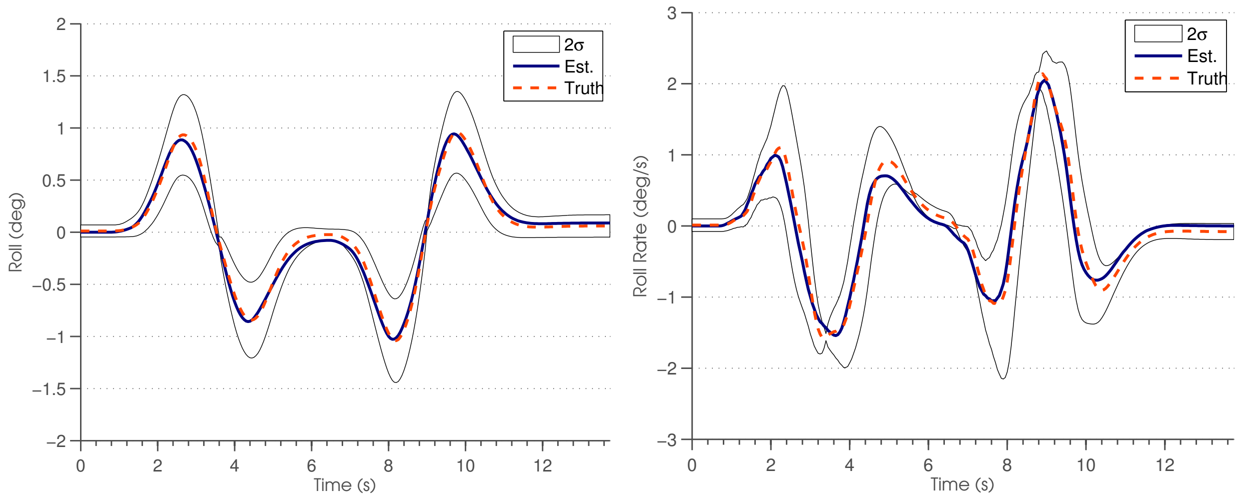


Figure 5.13: Double lane change validation of roll estimates using ten step prediction.

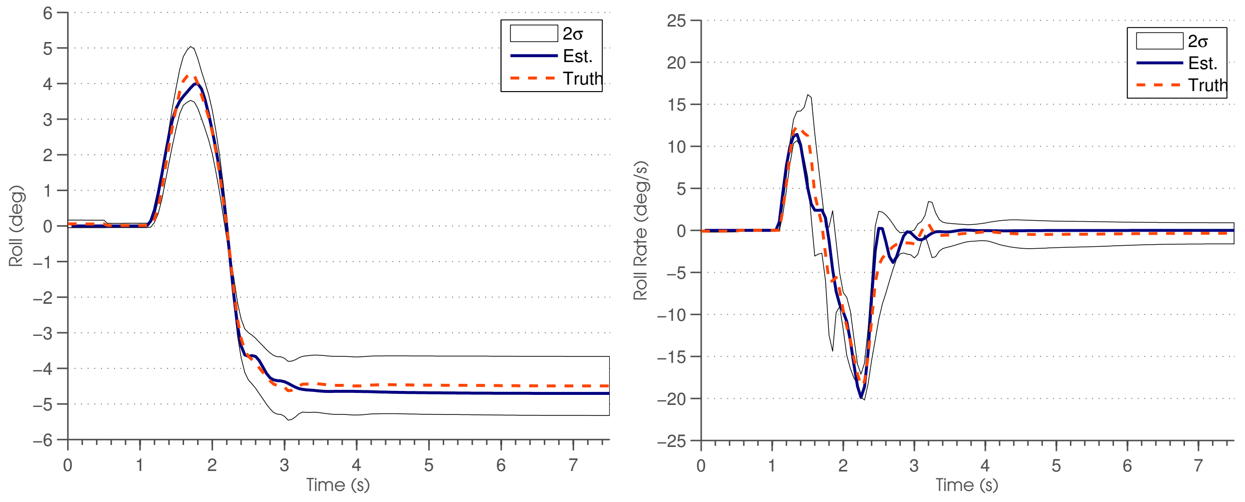


Figure 5.14: Fishhook validation of roll estimates using ten step prediction.

$\mathcal{O}(n^2)$. Examining the second and third columns shows that three out of four of the GP estimators (β, r, ϕ) were reduced to less than a quarter of the original size. The fourth ($\dot{\phi}$) was slightly more than half the original size reflecting the complexity of the function being learned relative to the first three estimators.

The fourth and fifth columns of Table 5.3 reveal the low cost in accuracy paid for this reduction in computation. In all four cases additional error was introduced to the GP estimates, however, in all cases the reduction in accuracy was minimal and well worth the additional computational speed.

Further error was introduced as the uncertainty in each estimate was accumulated over the ten-step predictions need for MPC. The final column of Table 5.3 shows the reduction in accuracy for the multi-step predictions with clustered GPs. Three out of the four RMS errors for the estimates remain within a degree of the measured value with the yaw rate estimate being the only stand out. The yaw rate estimate remains within roughly 2 degrees per second of the measured value, a result that is not surprising given the greater range of values that function covers in validation.

The output from the GP model during one sample period of the controller relies on the speed of calculation detailed earlier. The calculation of multiple estimates can be performed within the sample period of 50*ms* used in the simulations. The current measured state, $\mathbf{z}_k$, was used as initial conditions to begin computation. The predicted trajectories, $\hat{\mathbf{z}}_{k+1..k+10}$, for two hundred and one

Table 5.3: Change in covariance matrix size and RMS error before and after clustering.

Estimator	Size Before	Size After	Single-step RMS Before	Single-Step RMS After	10-Step RMS After
$\hat{\beta}_{k+1}$	4000	796	0.0034	0.0040	0.0969
$\hat{r}_{k+1}$	4000	668	2.1087	2.6693	2.0693
$\hat{\phi}_{k+1}$	4000	674	0.0238	0.0277	0.2387
$\hat{\dot{\phi}}_{k+1}$	4000	2368	0.0543	0.0734	0.8383

possible steer angles were then estimated corresponding to 1/4 degree increments of steer angle, δ, at the road wheels.

5.2.3 Constraints on Rollover Index

A measure of rollover risk was computed for each of the 201 predicted trajectories. If an individual trajectory resulted in a high risk of rollover, it was discarded from the subsequent optimization. The form of this measure of rollover risk is unimportant as long as it accurately gives a quantitative measure of the risk posed by each predicted trajectory. The rollover index used in this worked was detailed in [66].

The rollover index at a single sample time was calculated with

$$RI = \begin{cases} C_1\left(\frac{|\phi(t)|\dot{\phi}_{th}+|\dot{\phi}(t)|\phi_{th}}{\phi_{th}\dot{\phi}_{th}}\right)+C_2\left(\frac{|a_y|}{a_{y,c}}\right)+C_3\left(\frac{|\phi(t)|}{\sqrt{(\phi(t))^2+(\dot{\phi}(t))^2}}\right), & \phi(\dot{\phi}-k_1\phi)>0 \\ 0, & \phi(\dot{\phi}-k_1\phi)\leq 0 \end{cases} \tag{5.6}$$

which can be broken down into three individual terms. The first term places a constraint on the $\phi-\dot{\phi}$ phase plane. The second term constrains the lateral acceleration as excessive lateral acceleration can also contribute to rollover. The third and final term represents the proximity of the roll states to the thresholds ϕ_{th} and $\dot{\phi}_{th}$ on the $\phi-\dot{\phi}$ phase plane. The angle θ as indicated in Figure 5.15 decreases as wheel lift becomes imminent. The method of determining the rollover threshold indicate in this figure is detailed below. The sine of θ approaches zero as the vehicle nears the unstable region. The angle remains in the first quadrant as follows upon examination of (5.6). Therefore, the time to wheel lift will range from zero to one.

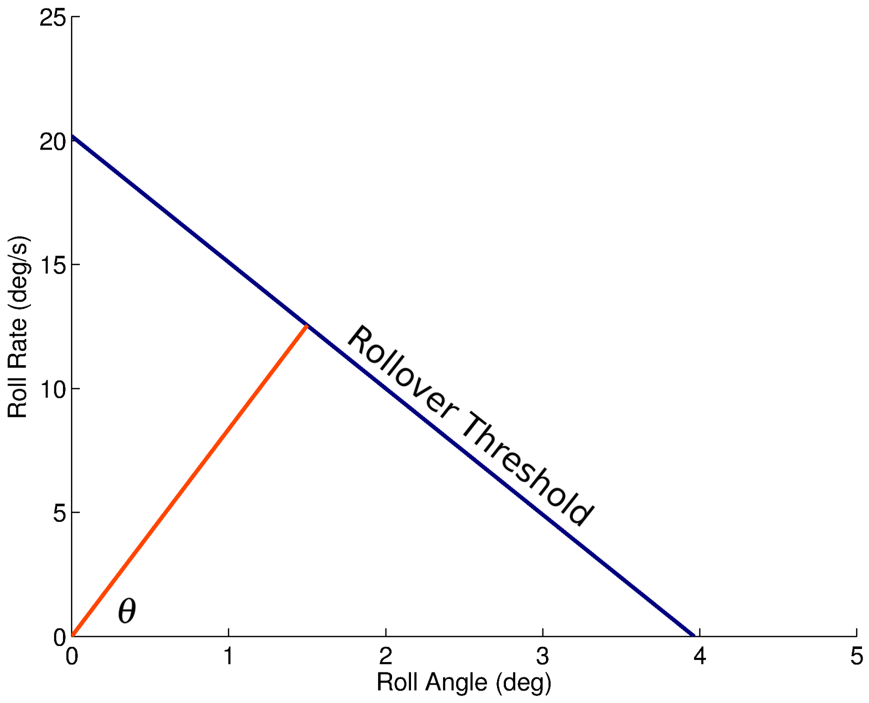


Figure 5.15: Illustration of time to wheel lift for rollover index.

Three parameters must be determined to calculate these, and subsequent terms, ϕ_{th}, $\dot{\phi}_{th}$, and a_{yc}. An illustration depicting how ϕ_{th} and a_{yc} were determined is shown in Figure 5.16. The values for ϕ_{th} and a_{yc} were found by solving the system

$$\begin{aligned} a_{yc} &= K_{ROLL} \cdot \phi \\ a_{yc} &= \frac{-t/2h}{arctan(2h/t)}\phi + \frac{t}{2h} \end{aligned} \tag{5.7}$$

Three vehicle parameters are needed to solve the system. The values of these vehicle parameters are listed in Table 5.4, and corresponded to the diagram shown in Figure 5.4.

Table 5.4: Parameters used in determining constraint parameters taken from CarSim “Small SUV” Model.

Parameter	Description	Value	Unit
t	Track width	1.47	m
h	cg height	0.64	m
K_{ROLL}	Roll stiffness	15	N/mm

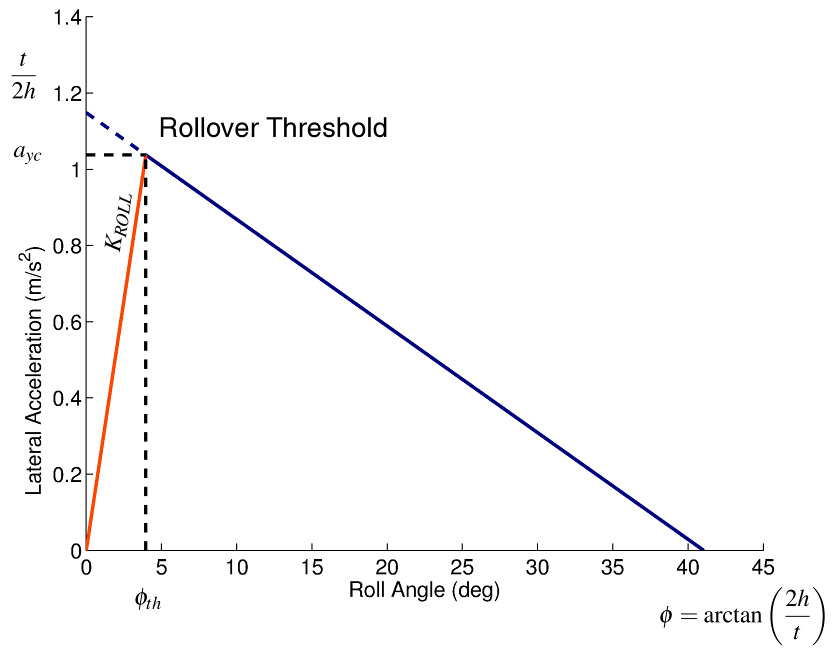


Figure 5.16: Method of determining rollover index thresholds.

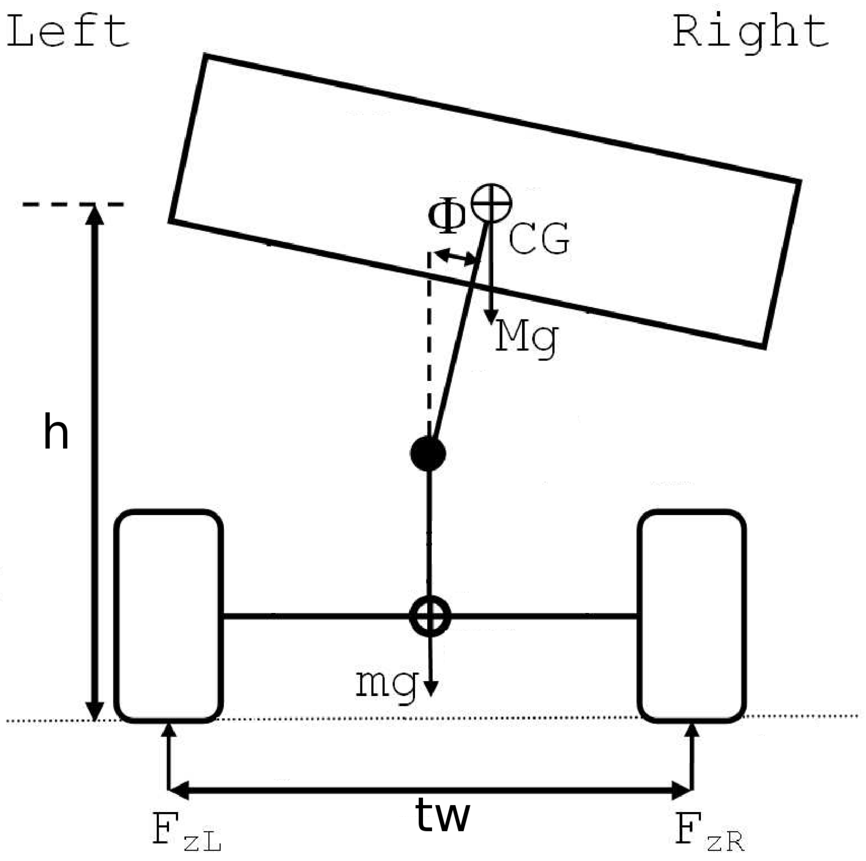


Figure 5.17: Body roll free body diagram.

A roll model was required to determine the threshold $\dot{\phi}_{th}$. Using the model detailed in [12], simulations were run with varying initial conditions. A series of simulations were performed with an initial roll angle of zero. The roll rate was increased until wheel lift was observed during the simulation. Figure 5.18 shows the $\phi - \dot{\phi}$ phase plane during the simulation with the initial conditions set at the threshold. The tire forces are also shown indicating wheel lift has occurred.

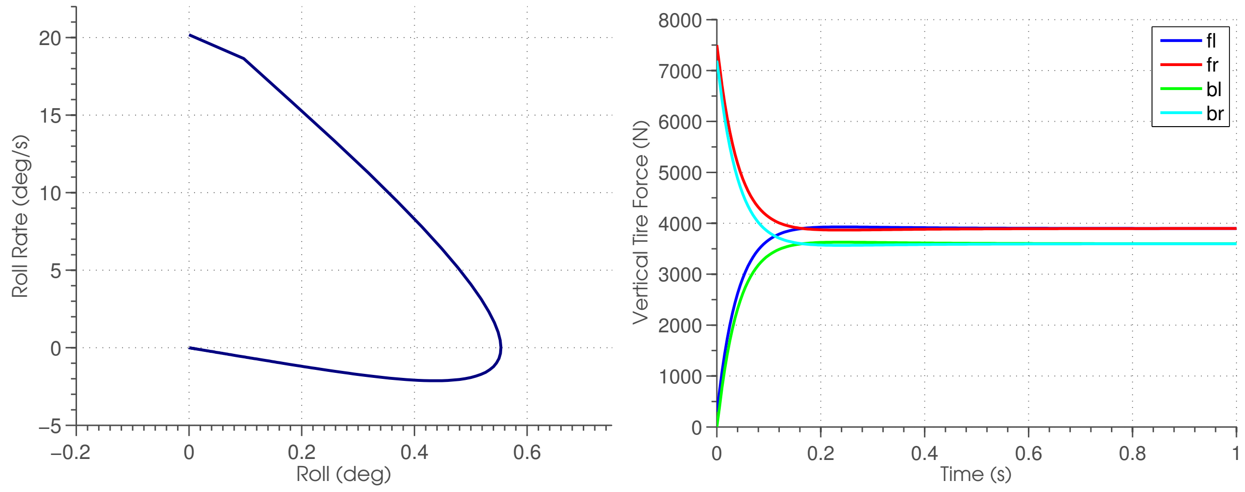


Figure 5.18: Roll phase plane trajectory of vehicle initially equivalent to roll rate threshold. (left) Vertical tire forces showing wheel lift at threshold. (right)

The three terms of rollover index (RI) taken together indicate the relative risk of rollover at each point during rollover. The RI for the DLC and FH maneuvers are shown in Figures 5.19 and 5.20. Both maneuvers consist of two separate turns. In the DLC maneuver the first turn poses a risk of rollover however the second turn poses a greater risk given the higher roll rate at the initiation of this turn. The RI shown in Figure 5.19 agrees with this intuitive description of the maneuvers.

Examining the RI during the FH maneuver in Figure 5.20 reveals a similar result. The FH maneuver emulates a driver avoiding an obstacle and over correcting to return to the roadway. The RI increases during the turn to avoid the obstacle as well as in the overcorrection. The same RI was applied to the predicted trajectories provided by the GP predictive model. If an unfavorable RI was indicated for a predicted trajectory, $\hat{\mathbf{z}}_{k+1..k+10}$, it was excluded from the optimization in the subsequent step of calculating the control effort to prevent an unsafe control effort. It should be noted that calculation of the RI is a function of the roll states and excludes the lateral states.

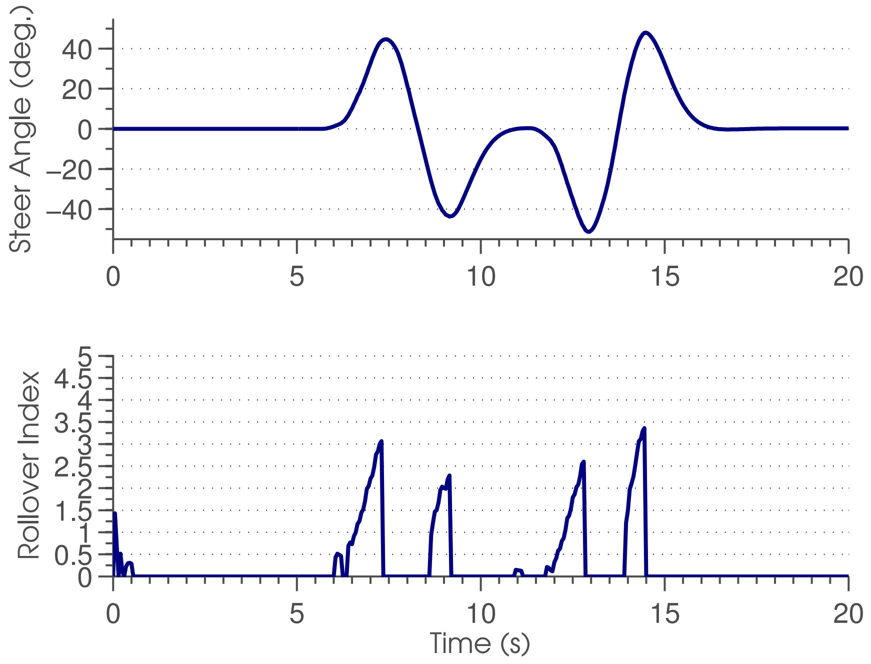


Figure 5.19: Rollover index during double lane change maneuver.

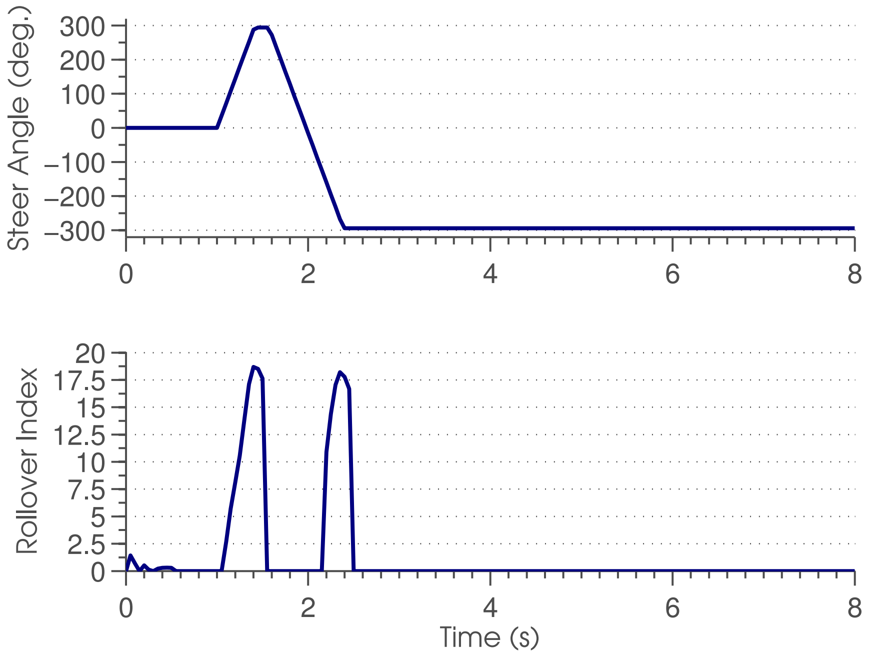


Figure 5.20: Rollover index during fishhook maneuver.

5.2.4 Optimization with Gaussian Process Model

The lateral states must be predicted since the path of the vehicle depends on matching the predicted path of the vehicle with the reference generated with the bicycle model. A cost is calculated for each predicted trajectory using

$$J = \sum_{k=1}^{N} \mathbf{w}_{z_k} (\mathbf{r}_k - \hat{\mathbf{z}}_k)^2 \tag{5.8}$$

where the state vector and predicted state vector are weighted by the vector quantity $\mathbf{w}_{z_k}$. The calculation of the the cost was then performed with element by element operations. Constraints placed on the steer actuator were handled in the generation of predicted trajectories. Each predicted trajectory respects the limited range of steer angle possible with the simulated vehicle. The steer angle associated with the predicted trajectory that satisfied the constraints on RI and had the minimum cost as defined by J was used as the control effort for the next sample period. Tuning of the controller is possible through the weights indicated in (5.8). Adjustment of the $\mathbf{w}_{z_k}$ weights can cause each iteration's predictions to affect the optimization differently. In this work it was found that a naive use of uniform weights over the value of k yielded acceptable results. Additionally, adjustment of the weights $\mathbf{w}_{z_k}$ can affect each state's contribution to the cost function. It may be called for if one state is more critical than others or varying scales artificially emphasize the importance of one state over others. in this chapter $\mathbf{w}_{z_k}$ was the row vector $[5, 1]$ to adjust for the greater range of yaw rate when compared to sideslip for the anticipated operation of the vehicle.

5.3 Simulated Validation

The controller was simulated using CarSim and each validation maneuver was performed with and without the controller to evaluate the ability to adhere to constraints placed on the roll dynamics. The RI was constrained and successful performance of the controller was defined as adhering to this constraint while following the reference trajectory in a reasonable fashion.

Figure 5.21 shows that given the same input used in the DLC earlier the RI has been constrained appropriately. The RI was constrained to 2 for this simulation. Figure 5.22 depicts the closed-loop trajectory with and without constraints. Figure 5.23 shows each state individually.

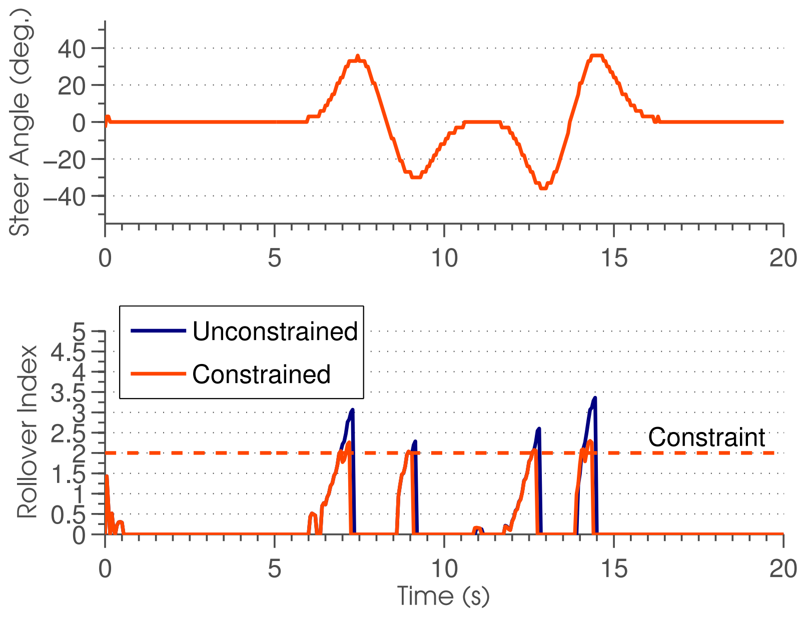


Figure 5.21: Rollover index during double lane change with and without constraints.

Similarly, a favorable result is seen when a FH maneuver was simulated with a constraint on RI of 15. Figure 5.24 shows the RI was constrained appropriately and Figure 5.25 shows the measured closed-loop trajectory matches that of the reference trajectory. Figure 5.26 shows each state individually.

5.4 Conclusion

An architecture for a model predictive controller was presented that leverages the strengths of Gaussian process regression to estimate passenger vehicle dynamics. The Gaussian process model developed in this chapter allows for accurate predictions to be made of the vehicle's future state. The speed of computation allows for an exhaustive search of the system input-space ensuring globally optimal solutions of the nonlinear constrained optimization problem inherent to MPC.

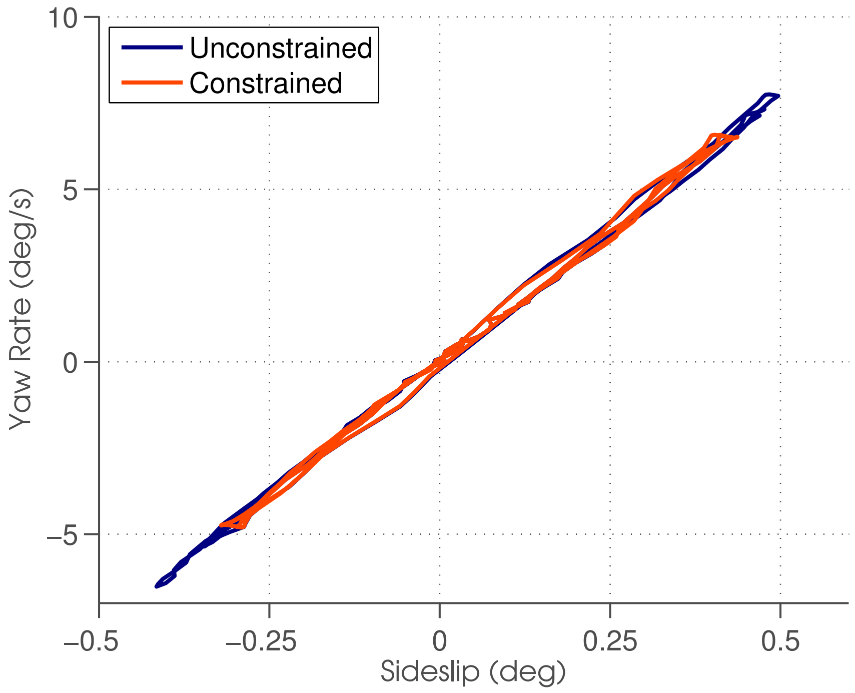


Figure 5.22: Phase plane of lateral dynamics during double lane change with and without constraints.

The presented architecture was demonstrated on the simulated CarSim "small SUV" model. Training and validation of the Gaussian process model was presented. The clustering method presented in Chapter 4 was applied to the training dataset. Simulation of the controller indicated that the clustered training dataset provided estimates that were accurate enough to control the lateral states while constraining the roll states of the vehicle.

A method of quantifying the risk of vehicle rollover was selected from the literature and applied in the model predictive controller. A comparison of the vehicle's response to two validation maneuvers revealed that the presented architecture was effective in constraining the risk of rollover.

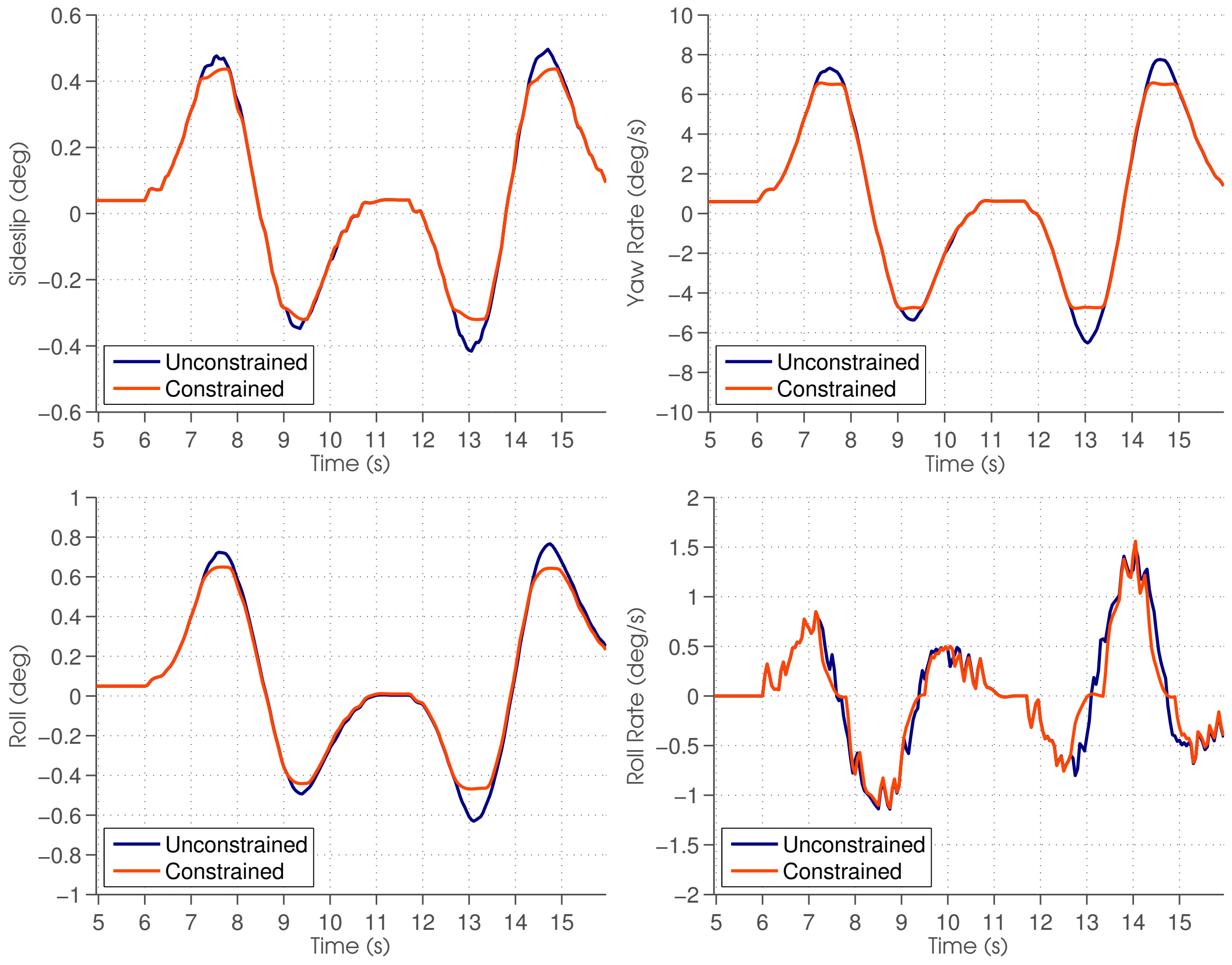


Figure 5.23: Four states during double lane change with and without constraints.

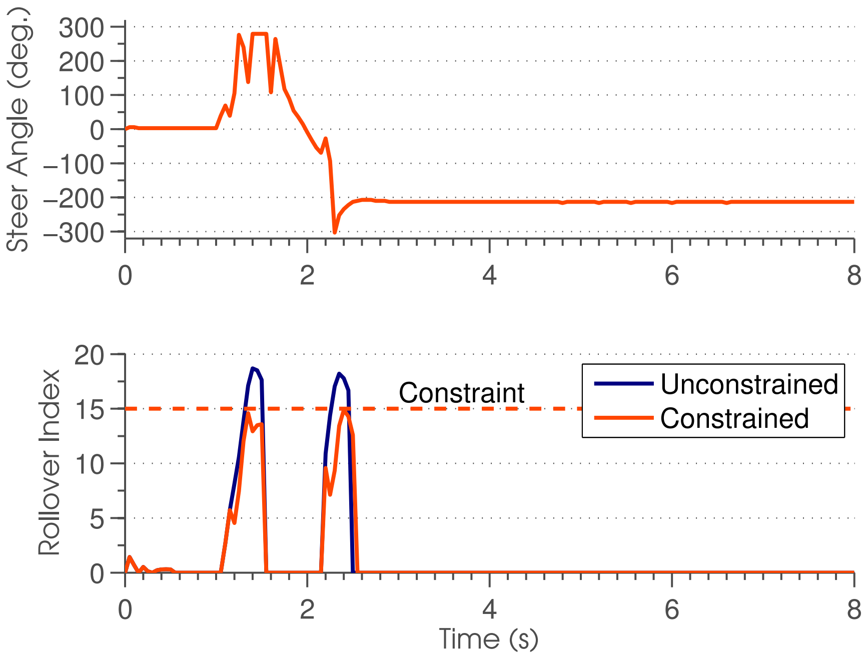


Figure 5.24: Rollover index during fishhook with and without constraints.

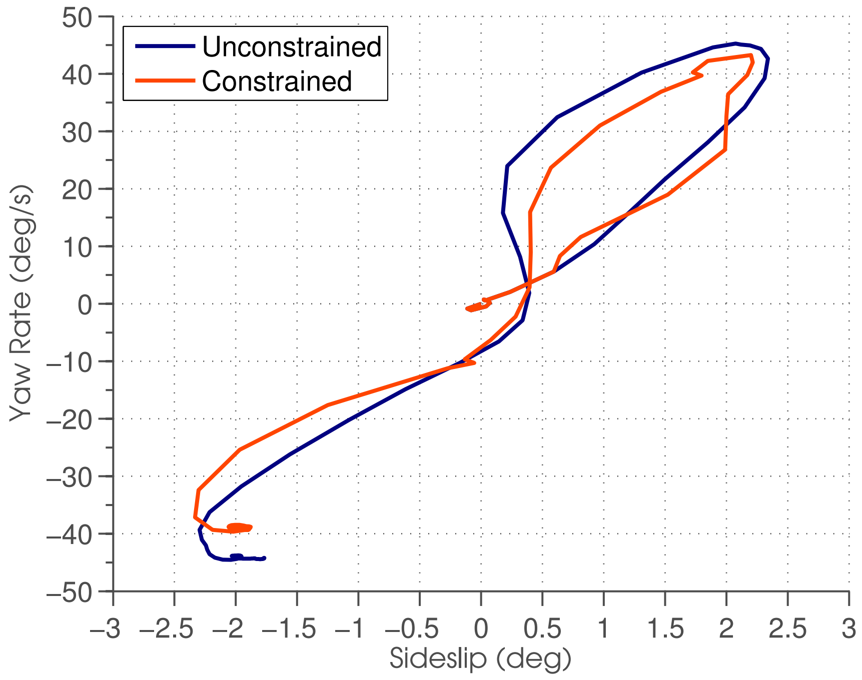


Figure 5.25: Phase plane of lateral dynamics during fishhook with and without constraints.

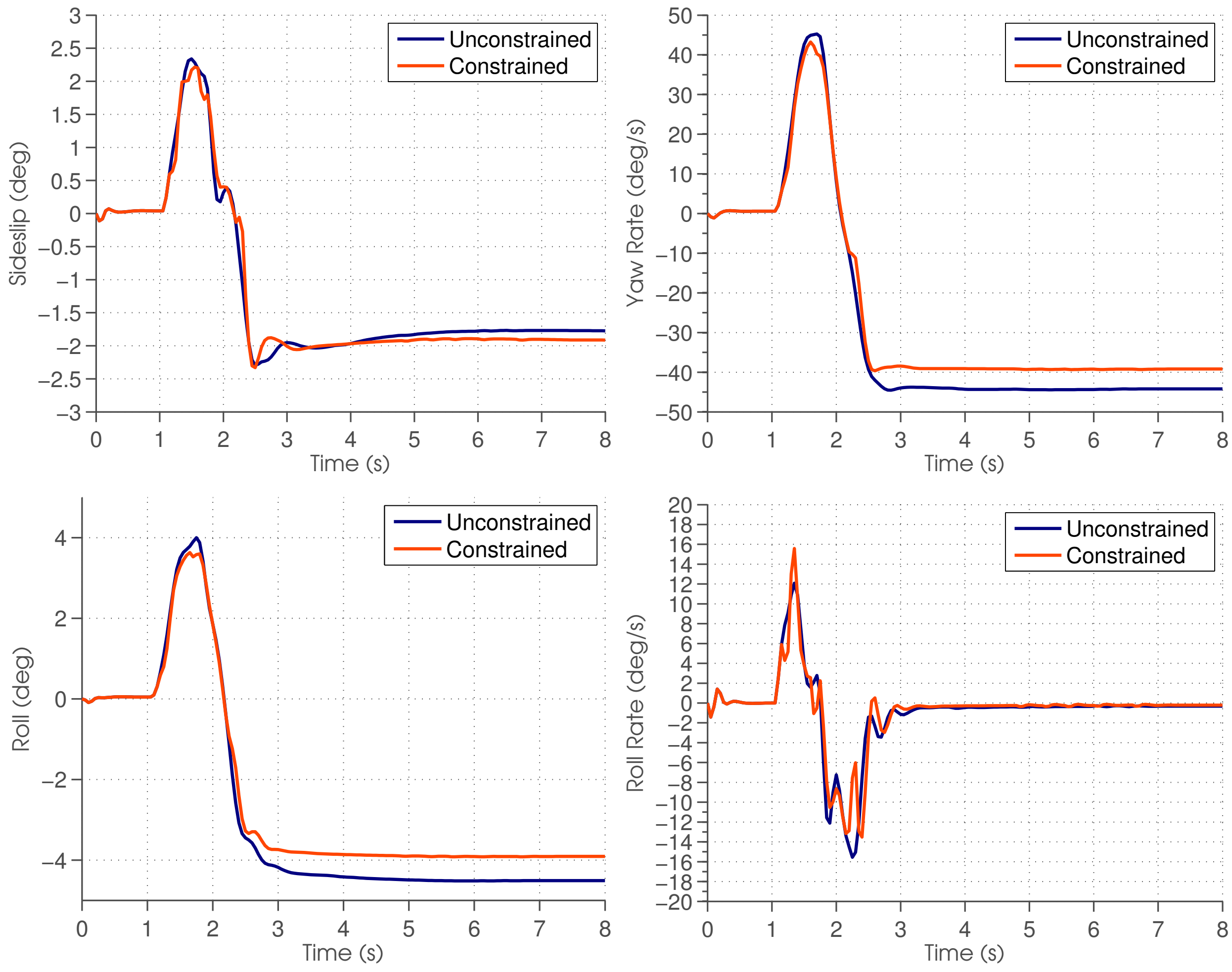


Figure 5.26: Four states during fishhook with and without constraints.

Chapter 6

Conclusion

The research described in this dissertation developed a Gaussian process based model of the lateral and roll dynamics of a vehicle for use in model predictive control. The accuracy of predictions and speed of computation was a large focus of the work. Modeling and validation was performed using an instrumented all-terrain vehicle (ATV). Accuracy and speed of computation was evaluated over single and multi-step predictions as required by model predictive control. A computation-reducing method was described and evaluated for Gaussian process regression to further speed the predictive controller. A method was suggested for selecting the parameters of the computation-reducing method and applied to a variety of datasets, both illustrative and recorded from the ATV. An architecture to incorporate the Gaussian process-based model was described and demonstrated using the CarSim simulation environment and simulation software.

Gaussian process regression provides a distribution as output in the form of a mean and variance for a given estimate. The nonlinear state-space model consisted of four GP functions, one for each of the four states describing the lateral and roll dynamics of the vehicle. Gaussian process regression was used to learn functions describing the nonlinear state-space form of the dynamic vehicle model. The model was trained using a series of sinusoidal maneuvers. Validation was performed using the double lane change and fishhook maneuvers common to vehicle rollover evaluation. The single-step and multi-step predictions of the GP-based model were evaluated against measured data from the ATV. The measured data was found to lie within the distribution/prediction from the GP-based model.

Speed of computation was also a concern given the small sample period necessary to control the vehicle. Gaussian process regression calculates estimates as the sum of many Gaussian regressors. These regressors form a projection of the input-space onto what is termed the feature-space. The nonlinear projection onto feature-space remained the most time consuming piece of generating estimates. However, through the use of an optimized calculation and parallelization the problem becomes tractable for modest hardware that is available today. Once in the feature-space, the computation of many estimates appeared as a matrix-vector operation that is the linchpin of contemporary computing architectures. The computation necessary to generate a large number of estimates to cover the input of the vehicle was shown to be completed within the necessary sample period ($20ms$).

After initial modeling and validation, a method of further reducing computation based on clustering was described and extended. Clustering of data within the input-space in GP regression has been discussed but selecting a method of clustering and the associated selection of parameters has been previously ignored. In this discussion, guidelines for defining similarity were suggested based on characteristics of the underlying function, namely the characteristic length scale. A clustering algorithm that parametrized the size of each cluster rather than the quantity was then used to take advantage of this definition of similarity. Demonstrations of the method were made on one and two dimensional functions to easily visualize the method. A transformation of the input-space was also introduced to account for varying sensitivities to each input of the GP. The transformation extended the clustering method in order to accommodate ovate clusters where it had previously been limited to spherical clusters. The clustering method was then applied to the GP- based model trained with recorded data from the ATV. This demonstrated the clustering method's applicability to higher dimensional problems, six in this case. All demonstrations showed the ability of the clustering method and associated guidelines for parameter selection to reduce the size of the training dataset while sacrificing only a small amount of accuracy.

Given the results described above, that a GP-based model can provide accurate predictions of nonlinear system dynamics, a controller architecture was described to leverage the nature of calculations in GP regression. A GP model's ability to calculate a large number of estimates in a single, fast operation opens up the possibility to search exhaustively for an optimal system input on a contemporary desktop computing system. Additionally, given the search is exhaustive within the constraints of the system input-space, the method avoids the pitfall of settling on locally optimal solutions. Model predictive control allows for constraints to be handled explicitly. A measure of risk of vehicle rollover was selected from the literature and used to place constraints on the dynamics. Simulation of the controller demonstrated the efficacy of the controller architecture in limiting risk of rollover during evasive maneuvers.

6.1 Future Work

This work may be furthered in a number of ways. First, a method of tuning the MPC controller should be detailed to improve performance. The current method of arbitrarily selecting weights for the cost function in the controller resulted in reasonable performance, but the controller could benefit from a more deliberate search for weights. It may be desirable to emphasize one state over the other or earlier predictions over distant predictions. Investigation into incorporating the variance of predictions should be performed to determine the advantages it offers. Steering the vehicle to maintain operation in the well-known region of the dynamics may offer a path to developing a robust application of GP-based model predictive control. Beyond using the prediction variance in the cost function, it should be incorporated into the calculation of constraints to further improve the robustness of the method. Prediction variance was used in this work as a means of validating the trained model though not used in the controller. It was therefore excluded from the cost of computation and not optimized. Including the variance calculations in the cost function and constraints would require a method of computation be proposed to maintain the ability of the controller to operate at a high sample frequency.

Second, real-time implementation should be performed to strengthen the argument that a GP-model predictive control can effectively constrain vehicle roll. Where other works required extensive use of groups of computers to make predictions efficiently this work makes clear that a single desktop PC is capable of carrying out the required calculation. Additional constraints may be required to accommodate actuator limitations beyond the simple limits placed on steer angle above. Particularly it is desirable to maintain a smooth input to the steering actuator and should be ensured by the controller whether as part of the cost function or constraints.

Third, while the clustering method showed an ability to reduce computation without adversely affecting estimate quality, it still required the training of the GP with the full dataset to acquire the characteristic length scales. After acquiring the length scales training was performed again more rapidly. Calculating estimates was also more rapid however, if a method of identifying the length scales was introduced the initial training would not be necessary. While this work was primarily interested in basing the clustering threshold on the characteristic length scales of each input dimension, the ability to find the length scales quickly would further improve the methods utility.

Finally, the Gaussian process-based model was demonstrated to provide accurate and fast computation of dynamic equations. It did so by projecting the input-space into the higher dimensional feature-space. The relationship between feature-space and the output was then linear and could leverage the speed of computation available for linear operations. This model type already offers some intriguing features to be used in nonlinear dynamic problems. To gain further favor within the controls community the more traditional issues of control should be investigated including stability, controllability, and observability. Investigation should be made into any advantages offered by the projection into the linear feature-space and its ability to answer these common questions in studying dynamic systems.

Bibliography

[1] “Traffic safety facts 2005,” National Highway Traffic Safety Administration, National Center for Statistics and Analysis, 200 New Jersey Avenue SE., Washington, DC 20590, Tech. Rep., 2005. [Online]. Available: http://www-nrd.nhtsa.dot.gov/pubs/tsf2005.pdf

[2] “Traffic safety facts: 2008,” National Highway Traffic Safety Administration, National Center for Statistics and Analysis, 200 New Jersey Avenue SE., Washington, DC 20590, Tech. Rep., 2008. [Online]. Available: http://www-nrd.nhtsa.dot.gov/CATS/index.aspx

[3] IIHS, “Electronic stability control could prevent nearly one-third of all fatal crashes and reduce rollover risk by as much as 80%,” Insurance Institute For Highway Safety, Tech. Rep., June 13 2006.

[4] L. Blincoe, A. Seay, E. Zaloshnja, T..Miller, E. Romano, and R. S.Luchter, “The economic impact of motor vehicle crashes 2000,” National Highway Traffic Safety Administration, 200 New Jersey Avenue SE., Washington, DC 20590, Tech. Rep., 2002. [Online]. Available: http://www-nrd.nhtsa.dot.gov/CATS/index.aspx

[5] “Motor vehicle accident costs,” U.S. Department of Transportation Federal Highway Administration, Tech. Rep., 1994. [Online]. Available: http://safety.fhwa.dot.gov/facts_stats/t75702.cfm

[6] T. Pilutti, G. Ulsoy, and D. Hrovat, “Vehicle steering intervention through differential braking,” in *Proceedings of the American Control Conference*, vol. 3, jun. 1995, pp. 1667 –1671 vol.3.

[7] J. Lu, D. Messih, and A. Salib, “Roll rate based stability control-the roll stability control? system,” in *Proceedings of the 20th Enhanced Safety of Vehicles Conference*, no. 07–0136, 2007.

[8] K. T. Feng, H. S. Tan, and M. Tomizuka, “Automatic steering control of vehicle lateral motion with the effect of roll dynamics,” in *Proceedings of American Control Conference 1998*, 1998.

[9] T. Xinpeng and K. Yoshimoto, “Research of driver assistance system for recovering vehicle stability from unstable states,” *SAE 2001 World Congress*, March 2001.

[10] C. G. Bobier, S. Joe, and J. C. Gerdes, “Sliding surface envelope control: Keeping the vehicle within a safe state-space boundary,” in *Proceedings of the ASME 2010 Dynamic Systems and Control Conference*, Cambridge, Massachusetts, USA, September 2010.

[11] J. H. Plumlee, "Multi-input ground vehicle control using quadratic programming based control allocation techniques," Master's thesis, Auburn University, 2004.

[12] R. J. Whitehead, "A study of the properties that influence vehicle rollover propensity," Master's thesis, Auburn University, December 2005.

[13] K. D. Lambert, "A study of vehicle properties that influence rollover and their effect on electronic stability controllers," Master's thesis, Auburn University, 2007.

[14] H.-S. Tan, Fanping, Bu, and J. Huang, "Charactering driving skill based on entropy analysis of steering frequency response," in *Proceedings of the ASME 2010 Dynamic Systems and Control Conference*, Cambridge, Massachusetts, USA, September 2010.

[15] C. Carlson and J. Gerdes, "Optimal rollover prevention with steer by wire and differential braking," in *Proceedings of IMECE*, 2003, pp. 16–21.

[16] C. E. Beal and J. C. Gerdes, "Enhancing vehicle stability through model predictive control," in *Proceedings of the ASME 2009 Dynamic Systems and Control Conference*, Cambridge, Massachusetts, USA, September 2009.

[17] C. G. Bobier and J. C. Gerdes, "Envelope control: Stabilizing within the limits of handling using a sliding surface," in *In Proceedings of the 2010 IFAC Symposium on Advances in Automotive Control*, Munich, Germany, 2010.

[18] P. Falcone, F. Borrelli, H. Tseng, J. Asgari, and D. Hrovat, "Linear time-varying model predictive control and its application to active steering systems: Stability analysis and experimental validation," *International Journal of Robust and Nonlinear Control*, vol. 18, no. 8, pp. 862–875, 2008.

[19] C. Beal and J. Gerdes, "Experimental validation of a linear model predictive envelope controller in the presence of vehicle nonlinearities," in *Proceedings of the 2010 IFAC Symposium on Advances in Automotive Control*, Munich, 2010.

[20] G. Adireddy, T. Shim, D. Rhode, and J. Asgari, "Model predictive control (mpc) based combined wheel torque and steering control using a simplified tire model," in *Proceedings of the ASME 2010 Dynamic Systems and Control Conference*, Cambridge, Massachusetts, USA, September 2010.

[21] G. Adireddy, T. Shim, and D. Rhode, "Wheel torque controller development using a simplified tire for vehicle handling enhancement," in *Proceedings of the ASME 2009 Dynamic Systems and Control Conference*, 2009.

[22] C. E. Beal and J. C. Gerdes, "A method for incorporating nonlinear tire behavior into model predictive control for vehicle stability," in *Proceedings of the ASME 2010 Dynamic Systems and Control Conference*, Cambridge, Massachusetts, USA, September 2010.

[23] C. Williams and C. Rasmussen, "Gaussian processes for regression," *Advances in Neural Information Processing Systems*, vol. 8, pp. 514–520, 1996.

[24] G. Cybenko, “Approximation by superpositions of a sigmoidal function,” *Mathematics of Control, Signals and Systems*, vol. 2, pp. 303–314, 1989.

[25] K. Hornik, “Approximation capabilities of multilayer feedforward networks,” *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991.

[26] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*, T. Dietterich, Ed. The MIT Press, 2006.

[27] N. Lawrence, M. Seeger, and R. Herbrich, “Fast sparse gaussian process methods: The informative vector machine,” *Advances in neural information processing systems*, pp. 625–632, 2003.

[28] L. Csato, “Gaussian processes - iterative sparse approximations,” Ph.D. dissertation, Aston University, 2002.

[29] G. Wahba, *Spline models for observational data*. Society for Industrial Mathematics, 1990. [Online]. Available: http://books.google.com/books?id=OAgB_odUE7sC&lpg=PR11&ots=e8Wo9ktO8u&dq=wahba&lr&pg=PA4#v=onepage&q&f=false

[30] D. J. Broderick, D. M. Bevly, and J. Y. Hung, “Modeling vehicle lateral dynamics by gaussian processes,” in *Proceedings of the ASME 2010 Dynamic Systems and Control Conference*, Cambridge, Massachusetts, September 2010.

[31] A. Girard, C. Rasmussen, J. Quinonero-Candela, R. Murray-Smith, J. Quiñonero-Candela, O. Winther, J. Quiñonero-Candela, A. Girard, J. Larsen, C. Rasmussen *et al.*, “Multiple-step ahead prediction for non linear dynamic systems–a gaussian process treatment with propagation of the uncertainty,” *Advances in Neural Information Processing Systems*, vol. 15, 2002.

[32] A. Girard, “Approximate methods for propagation of uncertainty with gaussian process models,” Ph.D. dissertation, University of Glasgow, 2004.

[33] J. Kocijan, R. Murray-Smith, C. Rasmussen, and A. Girard, “Gaussian process model based predictive control,” in *American Control Conference, 2004. Proceedings of the 2004*, vol. 3, June-2 July 2004, pp. 2214–2219 vol.3.

[34] K. Ažman and J. Kocijan, “Non-linear model predictive control for models with local information and uncertainties,” *Transactions of the Institute of Measurement and Control*, vol. 30, no. 5, p. 371, 2008.

[35] J. Bernardo, M. Bayarri, J. Berger, A. Dawid, D. Heckerman, A. Smith, M. West *et al.*, “Gaussian processes to speed up hybrid monte carlo for expensive bayesian integrals,” *Bayesian Statistics*, vol. 7, pp. 651–659, 2008.

[36] R. Kalman, “Contributions to the theory of optimal control,” *Bol. Soc. Mat. Mexicana*, vol. 5, no. 2, pp. 102–119, 1960.

[37] J. Richalet, A. Rault, J. Testud, and J. Papon, “Model predictive heuristic control: Applications to industrial processes,” *Automatica*, vol. 14, no. 5, pp. 413 – 428, 1978. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0005109878900018

[38] C. Cutler and B. Ramaker, “Dynamic matrix control - a computer control algorithm,” in *Poceedings of the Joint Automatic Control Conference*, no. 4,349,869, 1980.

[39] C. E. Garcia, “Quadratic programming solution of dynamic matrix control (qdmc),” *Chemical Engineering Communications*, vol. 46, pp. 73–87, 1986. [Online]. Available: http://ci.nii.ac.jp/naid/80003227083/en/

[40] P. B. Sistu and B. W. Bequette, “Nonlinear model-predictive control: Closed-loop stability analysis,” *AIChE Journal*, vol. 42, no. 12, pp. 3388–3402, 1996. [Online]. Available: http://dx.doi.org/10.1002/aic.690421210

[41] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, “Constrained model predictive control: Stability and optimality,” *Automatica*, vol. 36, pp. 789–814, 2000.

[42] J. Kocijan, A. Girard, B. Banko, and R. Murray-Smith, “Dynamic systems identification with gaussian processes,” *Mathematical and Computer Modelling of Dynamical Systems*, vol. 11, no. 4, pp. 411–424, 2005.

[43] J. Wang, D. Fleet, and A. Hertzmann, “Gaussian process dynamical models,” *Advances in neural information processing systems*, vol. 18, p. 1441, 2006.

[44] J. Ko, D. Klein, D. Fox, and D. Haehnel, “Gaussian processes and reinforcement learning for identification and control of an autonomous blimp,” in *Robotics and Automation, 2007 IEEE International Conference on*, April 2007, pp. 742–747.

[45] C. Wilson and T. Roppel, “Low-cost wireless mobile ad-hoc network robotic testbed,” in *Testbeds and Research Infrastructures for the Development of Networks Communities and Workshops, 2009. TridentCom 2009. 5th International Conference on*, april 2009, pp. 1 –6.

[46] S. K. Gan, K. Yang, and S. Sukkarieh, “3d path planning for a rotary wing uav using a gaussian process occupancy map,” in *Australasian Conference on Robotics and Automation (ACRA)*, 2009.

[47] C. Williams and D. Barber, “Bayesian classification with gaussian processes,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, no. 12, pp. 1342 –1351, dec 1998.

[48] J. Britt, D. J. Broderick, D. M. Bevly, and J. Y. Hung, “Lidar attitude estimation for vehicle safety systems,” in *The Position Location and Navigation System (PLANS) Conference*, 2010.

[49] D. J. Broderick, J. Britt, D. M. Bevly, and J. Y. Hung, “Simple calibration for vehicle pose estimation using gaussian processes,” in *Proceedings of Institute of Navigation (ION) 2011 International Technical Meeting (ITM)*, 2011.

[50] J. Crassidis and J. Junkins, *Optimal estimation of dynamic systems*. Chapman & Hall, 2011, vol. 24.

[51] D. Edwards, “Parameter estimation techniques for determining safe vehicle speeds in ugvs,” Master’s thesis, Auburn University, 2008.

[52] R. C. Whaley and J. J. Dongarra, "Automatically tuned linear algebra software," in *Proceedings of the 1998 ACM/IEEE conference on Supercomputing (CDROM)*, ser. Supercomputing '98. Washington, DC, USA: IEEE Computer Society, 1998, pp. 1–27. [Online]. Available: http://dl.acm.org/citation.cfm?id=509058.509096

[53] J. Krüger and R. Westermann, "Linear algebra operators for gpu implementation of numerical algorithms," in *ACM SIGGRAPH 2005 Courses*, ser. SIGGRAPH '05. New York, NY, USA: ACM, 2005. [Online]. Available: http://doi.acm.org/10.1145/1198555.1198795

[54] N. Schraudolph, "A fast, compact approximation of the exponential function," *Neural Computation*, vol. 11, no. 4, pp. 853–862, 1999.

[55] X. Zhang, "Using class-center vectors to build support vector machines," in *Neural Networks for Signal Processing IX, 1999. Proceedings of the 1999 IEEE Signal Processing Society Workshop*. IEEE, 1999, pp. 3–11.

[56] R. Koggalage and S. Halgamuge, "Reducing the number of training samples for fast support vector machine classification," *Neural Information Processing-Letters and Reviews*, vol. 2, no. 3, pp. 57–65, 2004.

[57] Y. Wang and D. Casasent, "New weighted support vector k-means clustering for hierarchical multi-class classification," in *Neural Networks, 2007. IJCNN 2007. International Joint Conference on*. IEEE, 2007, pp. 471–476.

[58] Y. Qi, W. He, and H. Shu, "An optimized approach on reduced kernel matrix to clustersvm," in *Intelligent Information Hiding and Multimedia Signal Processing, 2008. IIHMSP'08 International Conference on*. IEEE, 2008, pp. 1446–1449.

[59] J. Wang, X. Wu, and C. Zhang, "Support vector machines based on k-means clustering for real-time business intelligence systems," *International Journal of Business Intelligence and Data Mining*, vol. 1, no. 1, pp. 54–64, 2005.

[60] D. Boley and D. Cao, "Training support vector machine using adaptive clustering," in *Proceedings of the Fourth SIAM International Conference on Data Mining, MW Berry, U. Dayal, C. Kamath, and D. Skillicorn, eds., SIAM Press, Philadelpha*. Citeseer, 2004, pp. 126–137.

[61] T. Shaohua and Z. Qingfang, "Fast svm incremental learning based on clustering algorithm," in *Intelligent Computing and Intelligent Systems, 2009. ICIS 2009. IEEE International Conference on*, vol. 1. IEEE, 2009, pp. 13–17.

[62] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 281-297. California, USA, 1967, p. 14.

[63] J. W. Jr., "Hierarchical grouping to optimize an objective function," *Journal of the American statistical association*, vol. 58, no. 301, pp. 236–244, 1963.

[64] K. Wilamowska and M. Manic, “Unsupervised pattern clustering for data mining,” in *Industrial Electronics Society, 2001. IECON ’01. The 27th Annual Conference of the IEEE*, vol. 3, 2001, pp. 1862–1867 vol.3.

[65] R. Adler, *The geometry of random fields*. Siam, 1981, vol. 62.

[66] J. Yoon, D. Kim, and K. Yi, “Design of a rollover index-based vehicle stability control scheme,” *Vehicle system dynamics*, vol. 45, no. 5, pp. 459–475, 2007.