University of Thessaly
Department of Mechanical Engineering

Active vehicle suspension control using non-linear model predictive control and computational intelligence methods

Diploma Thesis of
Papadimitrakis Myron

Thesis Advisor: **Dr. Alex Alexandridis**

Submitted as part of the requirements for a Mechanical Engineering Diploma from University of Thessaly

Samos, June 2018

Ευχαριστίες

Για την εκπόνηση της διπλωματικής εργασίας μου θα ήθελα να ευχαριστήσω αρχικά τον επιβλέποντα καθηγητή μου κο. Αλέξανδρο Αλεξανδρίδη για την πολύτιμη καθοδήγηση του καθώς και για την άψογη επικοινωνία που είχαμε κατά τη διάρκεια της συνεργασίας μας.

Τέλος θα ήθελα να ευχαριστήσω τους γονείς μου και τους φίλους μου που με στήριξαν στις σπουδές μου όλα αυτά τα χρόνια.

Περίληψη:
Τα ενεργά συστήματα αναρτήσεων εφαρμόζονται σε οχήματα δρόμου με σκοπό να μειώσουν τις κατακόρυφες επιταχύνσεις του σασί με μεγαλύτερη αποτελεσματικότητα σε σχέση με τις παθητικές αναρτήσεις, διατηρώντας ταυτόχρονα ή ακόμα και αυξάνοντας την πρόσφυση των ελαστικών. Σε αυτή τη διπλωματική εργασία, εφαρμόζεται ένας ελεγκτής βασισμένος σε Μοντέλα Πρόβλεψης (Model Predictive Control – MPC) με μοντέλα ακτινικών συναρτήσεων βάσης (Radial Basis Function) ως λύση για το πρόβλημα του αυτόματου ελέγχου μίας ενεργής ανάρτησης. Η μεθοδολογία MPC επιλέχθηκε διότι μπορεί να ενσωματώσει λειτουργία προεπισκόπησης δρόμου καθώς και διάφορους περιορισμούς κατά τη λειτουργία του ελεγκτή, ενώ η τεχνολογία RBFN μπορεί να διαχειριστεί μη-γραμμικότητες του συστήματος σε συνδυασμό με γρήγορη διαδικασία εκπαίδευσης μέσω του αλγορίθμου συσταδοποίησης Fuzzy Means. Ο ελεγκτής MPC ενσωματώνει ένα γραμμικό μοντέλο της πλήρους ανάρτησης, με RBFN μοντέλα για τις μεταβλητές κατάστασης που παρουσιάζουν σημαντικές μη-γραμμικότητες. Η προτεινόμενη μέθοδος συγκρίνεται με έναν ελεγκτή MPC με πλήρως γραμμικό μοντέλο, καθώς και μία μορφοποίηση κλασικού PID ελεγκτή. Ο στόχος της σύγκρισης είναι η ελαχιστοποίηση των κατακόρυφων επιταχύνσεων του σασί επί παλμικού και τυχαίου προφίλ δρόμου. Και στις δύο περιπτώσεις, αποδεικνύεται το προτεινόμενο σχήμα υπερτερεί σημαντικά έναντι της παθητικής ανάρτησης, αλλά και έναντι των άλλων μεθόδων για τον έλεγχο ενεργής ανάρτησης, ενώ ταυτόχρονα διατηρεί χαμηλή κατανάλωση ενέργειας για το ηλεκτροϋδραυλικό σύστημα.

Λέξεις Κλειδιά:
Αλγόριθμος Ασαφών Μέσων, Ενεργή Ανάρτηση, Έλεγχος Βασισμένος σε Μοντέλα Πρόβλεψης, Δίκτυα Ακτινικής Συνάρτησης Βάσης, Κατακόρυφη Δυναμική Οχήματος, Πλήρης Ανάρτηση.

Abstract:
Active suspension systems in road vehicles are applied in order to mitigate the road-induced chassis vertical accelerations more effectively than passive suspension systems, while at the same time retaining or even increasing their road holding capabilities. In this thesis, a Model Predictive Controller (MPC) with Radial Basis Function Network (RBFN) models is presented as a control scheme for the full car active suspension system. The MPC methodology is chosen because it can incorporate road preview information as well as various constraints, while the RBFN architecture can handle nonlinearities and can be trained relatively fast and efficiently through the usage of a Fuzzy Means Clustering Algorithm. The MPC controller is based on a Linear model of the Full Car, with RBFN models for the highly nonlinear states. It is compared to an MPC Controller with a fully linear model of the plant, as well as a traditional PID formulation. The objective of the comparison is the minimization of the vertical acceleration of the chassis over a pulse road and a random road. In both instances, it is shown that the proposed scheme significantly outperforms a passive suspension, as well as alternative controllers for active suspension, while at the same time maintaining a low energy expenditure for the electrohydraulic system.

Keywords:
Fuzzy Means Algorithm, Active Suspension, Model Predictive Control, Radial Basis Function Networks, Vertical Vehicle Dynamics, Full Car Model.

Contents

1 Introduction

1.1 Types of Suspension Systems

A suspension system of a conventional road vehicle serves to keep the wheels in a relative position with the chassis while traveling. The two main design objectives are ride comfort for passengers, which can often be expressed as the value of the vertical acceleration of the vehicle's chassis, and road holding capabilities, which can often be expressed as a load variation on the vehicle's tire [1].

The most widely applicable type of suspension system in modern road vehicles is the **passive suspension system**, comprised of a spring and damper of fixed rates (fig. 1-1). Passive systems can only act when the distance between wheel and body, as well as its time derivative, are changed. Obviously, they do not exert forces based on the chassis vertical acceleration (ride comfort), neither based on tire load variation (road holding). The desirable design objectives can only be achieved indirectly, which is by changing the spring and damper rates of the suspension. This approach sometimes leads to conflicting results, since designing for road holding will compromise ride comfort and vice-versa. Moreover, the pursuit of these objectives is highly dependent on the driving conditions that the vehicle is specialized in [2], and once the spring and damper rates are set, they cannot be changed to adapt to different ones.

Figure 1-1 Passive Suspension System, (source: wikipedia.org)

Semi-active suspension systems are a more advanced approach; they are comprised of dampers (and sometimes springs) with controller adjustable rates, such as fig.1-2. Even though they exhibit higher versatility than the passive suspension, they still do not allow for the direct pursuit of the design objectives. [3]

Figure 1-2 Ohlins damper w/ Controller Adjustable Rates, (source: Ohlins Product Catalog)

Active suspension systems allow for the direct pursuit of the objectives of ride comfort and vehicle handling [4, 5], They are usually implemented in vehicles through a hydraulic system that powers a hydraulic piston (fig.1-3) placed in parallel to a conventional spring and damper. This piston is controlled so as to exert desired forces on the wheel in the vertical direction. This system can achieve the objectives in varying road conditions, as well as varying suspension parameters (since they change throughout the life cycle of the vehicle) [6-8].

Figure 1-3 Mercedes-Benz C Class Active Body Control Actuators,[9]

1.2 The importance of controlling the vertical dynamics of the vehicle

The term “vertical dynamics of a road vehicle” refers to the motion of the chassis and wheels in the z-axis (commonly up and down). It is of great importance for the quality of mobility for road vehicles, and can be examined through the following disciplines:

Vehicle Dynamics and Ride Safety

As mentioned before, the road holding characteristics of a vehicle can be enhanced by maintaining small dynamic wheel load fluctuations. These correspond to a good road-tire contact, among others. Moreover, the chassis relative vertical movement affects the kinematics of the wheel, such as the camber and toe angles, which in turn greatly influence lateral and longitudinal dynamics of the vehicle, with obvious results for ride safety[1, 10].

Comfort for human passengers

The primary function of road vehicles has been to transport humans effortlessly and effectively. In modern road vehicles the parameter of comfort has become of growing importance, since cars are as much of an item of luxury as a mode of transport. The comfort of human passengers in regards to ride quality can be quantified through many different metrics, such as the Steffens comfort criteria, the HRF, or the chassis accelerations [11, 12].

Sensitive Cargo and Technology

Controlling the vertical dynamics of the car does not concern only human passengers; sensitive cargo that cannot exceed certain acceleration thresholds may need to be transported. From a technology perspective, the example of a hydrogen fuel cell system applied in a road vehicle would benefit greatly from controlled vertical dynamics [13].

1.3 Developing a control strategy for road vehicle vertical dynamics

The challenge of controlling the vertical dynamics of a vehicle is that the full car plant, as well as the hydraulic actuators, behave in a way that is far from ideal. Notably, hydraulic and geometrical non-linearities, as well as the springs and dampers containing cubic terms of displacement and velocity are the main sources of this behavior [7]; ignoring these characteristics can lead to sub-par results or even system instability[4]. In this respect, nonlinear control strategies need to be considered so that the above requirements are addressed in order to achieve acceptable performance.

Several methods to develop such strategies have been proposed, and have been a topic of research since 1970. [14] presented a detailed analysis of classical control methods in application to the active suspension problem. [2] successfully applied a backstepping control methodology for the control of all chassis modes of a full car active suspension, however they omitted the actuator dynamics by replacing them with their resultant force, thus reducing the nonlinearity of the plant. [7] tackles the system nonlinearities through the use of Linear Parameter-Varying control, where the parameters are rendered nonlinear and the states linear.

An important practical consideration in the creation of active suspension controllers is the variation of the inertial parameters of the chassis - the sprung mass and the center of mass, as well as the moments of inertia, can change with varying degrees of loading of the vehicle (passengers, fuel, cargo) [6]. Thus, a focus has been placed in creating appropriate controllers: [15] implemented an Interval Fuzzy Controller which is able to handle nonlinearity and uncertainty in the system parameters, whereas [16] developed a parameter dependent control strategy that was able to handle the chassis inertial variation, as well as the actuator time delay.

1.4 Proposing an alternative control strategy

An alternative control strategy for the active suspension problem is Model Predictive Control (MPC). MPC relies on the principle that if a discrete-time model of the plant that correlates manipulated with controlled variables exists, then the inversion of this model would provide the sequence of optimal control moves in regards to the setpoint of the process.

The interest in MPC originates from the fact that preview information can be used, as well as the ability to accommodate for various physical constraints, such as suspension and actuator pressure constraints [5, 9]. In addition, multiobjective control can be pursued [17]. In their work, [11] compare the MPC active suspension with the more conventional approaches of Skyhook damping and Linear Quadratic Regulator, and they conclude in the superiority of the first in a quarter car application. [5] applied an MPC controller with preview to a full car plant in order to minimize all vehicle body accelerations under constraints of the manipulated variable. They also showed that dynamic wheel load fluctuations can be reduced indirectly, thus addressing a second design objective.

A prerequisite for an effective MPC Controller is the creation of an appropriate model which can handle the nonlinearities of the plant; in this respect, computational intelligence methods seem to be an attractive choice due to their relevant ability. Such methods have been applied in the creation of full car models: Notably, [18] applied computational intelligence methods to address the vehicles' parameter variation; [8] applied RBFN models for the control of the lateral and longitudinal dynamics of a 3 degree of freedom vehicle. In particular, Radial Basis Function Networks (RBFN) are considered because of their simple architecture as well as their faster training procedure, especially if coupled with a Fuzzy Means Algorithm [19].

To conclude, an MPC Controller with Preview coupled with a non-linear computational intelligence plant model was chosen for this work. In particular, a linear model of the full car is created and tested in the MPC Scheme. Then, Radial Basis Function Network (RBFN) models are created for the highly nonlinear states, and the new full car model is applied. The two MPC Controllers are compared to a traditional PID one for reference. For the sake of comparison, only the vertical acceleration of the chassis is controlled, and the only constraints that are applied are on the manipulated variable. The three formulations are tested on a pulse test and a random road test.

2 Plant Models

In order to begin developing a control strategy, an overall system description is in order. Firstly, the electrohydraulic system is presented since it is applicable in both quarter car and full car models. Then, the quarter car model is presented and finally the full car model as an expansion of the former.

2.1 Electrohydraulic System

System Description

The electrohydraulic piston – valve system is the powerhouse of the Active Suspension system. As shown in fig. 2-1, the main components are the oil sump, axial pump and accumulator, a two-stage power servovalve, a two-stage secondary bypass valve and a hydraulic piston, which connects the chassis (sprung mass) with the wheel (unsprung mass).

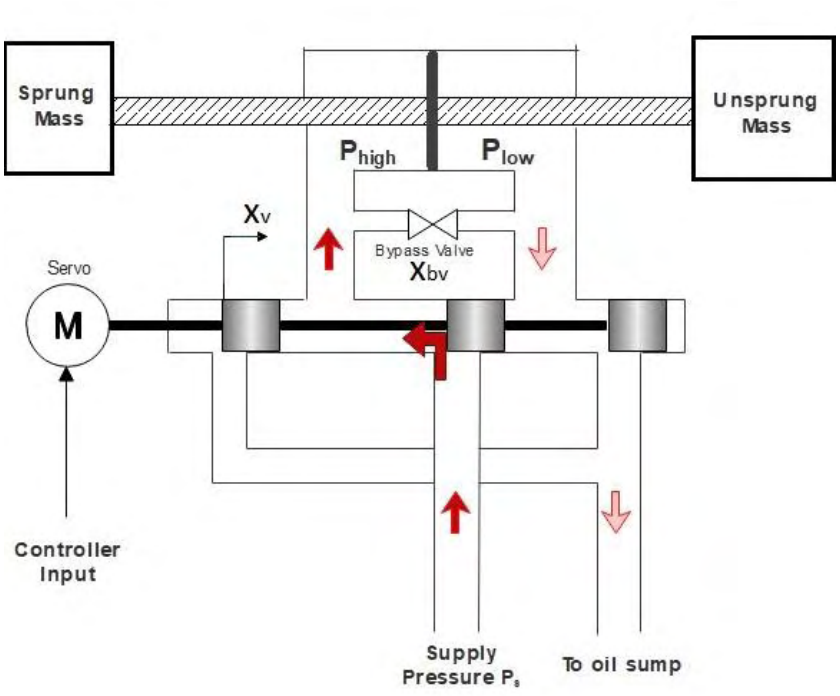


Figure 2-1 Schematic of the electrohydraulic piston-valve system

The operation can be outlined as follows: Hydraulic fluid is pumped from the oil sump by the axial pump and raised to the Supply Pressure P_s. When the power servovalve is open, high pressure fluid flows to either one of the actuator chambers while low pressure fluid flows from the other one back to the sump. Between the two actuator chambers a pressure difference is created; the active force of the actuator originates from this pressure difference.

As far as the bypass valve is concerned, its function is to reduce the total energy consumed by the system [11]. When it actuates, it works to equalize the pressure between the high and low pressure chamber, thus reducing the amount of energy the axial pump has to spend in order to maintain the same active force. Moreover, there are is one more practical aspect that is addressed; the non-linearities increase with ever increasing pressure due to the nature of the fluid flow through the servovalve, thus making it harder to model the system [20]. In our application, the bypass valve acts as a pressure relief valve which actuates at a specific pressure.

Fig. 2-2 depicts a typical operation curve graph for a valve/piston system. The x-axis is the actuator Pressure Difference to Supply Pressure ratio, while the y axis is the Hydraulic Load Flow. As shown, with increasing actuator pressure, the nonlinear behavior of the system increases, since we move towards either end of the operating range. It is therefore of interest, to keep the actuator pressure P_L in the linear range.

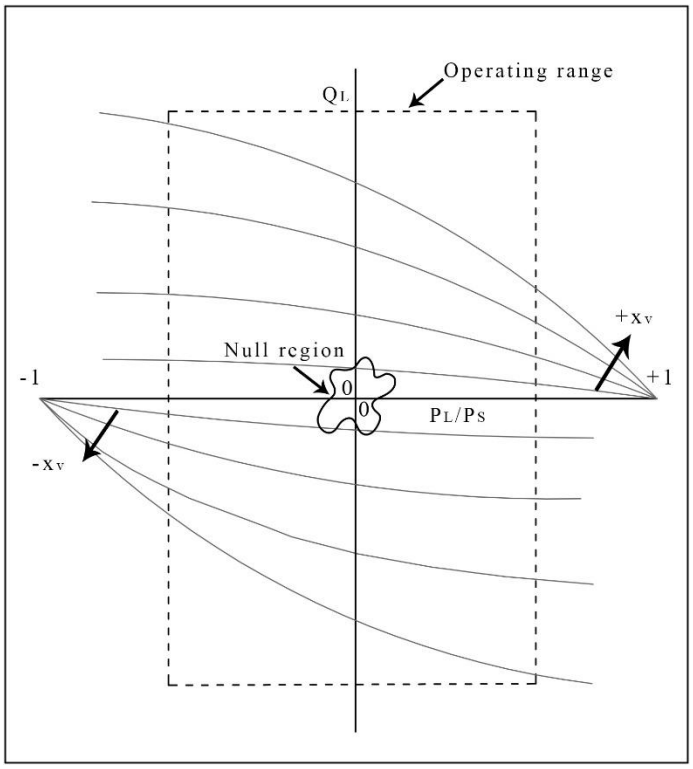


Figure 2-2 Operating graph of a valve/piston system. [20]

System State Equations

The force of the valve-piston system is

$$F = A_p P_L \tag{2.1}$$

Where A_P is the piston area and P_L is the pressure difference between high and low pressure chambers. The derivative of P_L is given as:

$$\frac{dP_L}{dt} = a\left[Q_{x_v} - Q_{x_{bv}} - C_{tm}\,\mathrm{P_L} - A_p\left(x_3 - x_4\right)\right] \tag{2.2}$$

$$Q_{x_v} = C_{dP} S_{xv} x_v \sqrt{\frac{P_s - sign\left(x_v\right) P_L}{\rho}} \tag{2.3}$$

$$Q_{x_{bv}} = C_{dB} x_{bv} sign\left(\mathrm{P_L}\right) \sqrt{\frac{2\,|\,\mathrm{P_L}\,|}{\rho}} \tag{2.4}$$

The first term Q_{xv} is the hydraulic fluid flow through the power valve, while the second one Q_{xbv} is the fluid flow through the bypass valve. The third term corresponds to the leakage through the piston seals and is proportional to the pressure difference P_L, while the fourth one is the contribution of the relative velocity of the wheel to the chassis.

The power valve position is controlled by the input to the servo u:

$$\dot{x}_v = \frac{1}{\tau}\left(-x_v + u\right) \tag{2.5}$$

2.2 Quarter Car Model

System Description

The quarter car suspension model is a useful tool in the study of the vertical dynamics of a vehicle [10, 21]. Essentially it represents one corner of a four wheeled vehicle. Even though it is simplistic, its' results can be used for more advanced vehicle simulations [1, 10], as they provide qualitative information on the effect of each parameter. Additionally, some quantitative information can be extracted to be used directly in real life applications, such as the tuning of suspension damper and stiffness parameters.

The model of the quarter-car suspension consists of the vehicle corner body mass M_s (sprung mass), which is connected to the wheel mass M_u (unsprung mass) through a non-linear spring K_c, a non-linear damper b_c and a hydraulic actuator F (fig.2-3). The wheel mass is in turn connected to the road profile through a non-linear spring K_t which represents the tire stiffness. The displacements are positive as indicated.

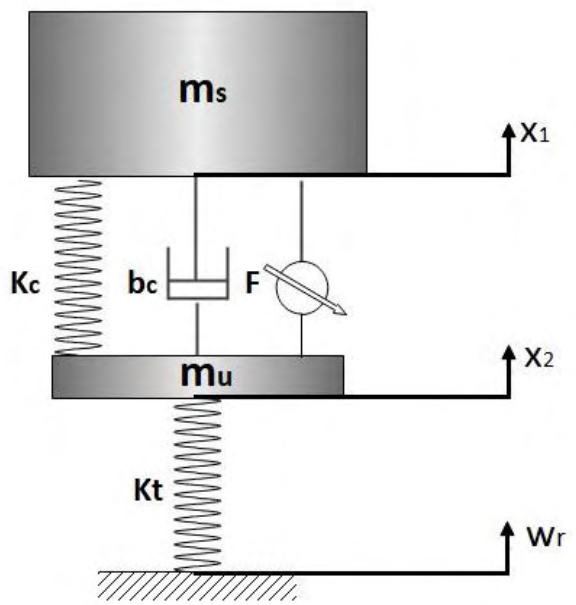


Figure 2-3 Schematic of the Quarter Car Model

Table 2-1 Degrees of Freedom for the Quarter Car Model

1	**Sprung Mass heave**	Up and down movement of the Sprung mass
2	**Unsprung Mass heave**	Up and down movement of the unsprung mass

System state equations

By assuming that all displacements are absolute and along the vertical axis, the quarter car active suspension may be described by the following system of states:

$$\dot{x}_1 = x_3 \tag{2.6}$$

$$\dot{x}_2 = x_4 \tag{2.7}$$

$$\dot{x}_3 = \frac{1}{m_{spr}}\left(F_{ks} + F_{bs} - A_p P_L\right) \tag{2.8}$$

$$\dot{x}_4 = \frac{1}{m_{unsp}}\left(-F_{ks}(x_1, x_2) - F_{bs}(x_3, x_4) - F_{kt}(w_r, x_2) + A_p P_L\right) \tag{2.9}$$

$$\dot{P}_L = a\left[Q_{x_v}\left(x_v, P_L\right) - Q_{x_{bv}}\left(x_{bv}, P_L\right) - C_{tm}\,\mathrm{P_L} - A_p\left(x_3 - x_4\right)\right] \tag{2.10}$$

$$\dot{x}_v = \frac{1}{\tau}\left(-x_v + u\right) \tag{2.11}$$

Where:

$$F_{b_s}(x_3, x_4) = b_s^{lin}\left(x_4 - x_3\right) - b_s^{sym} \mid x_4 - x_3 \mid + \mathrm{b}_s^{nonlin}\sqrt{\mid x_4 - x_3 \mid}\,\mathrm{sign}\left(x_4 - x_3\right) \tag{2.12}$$

$$F_{k_s}(x_1, x_2) = k_s^{lin}\left(x_2 - x_1\right) + k_s^{nonlin}\left(x_2 - x_1\right)^3 \tag{2.13}$$

$$F_{k_t}(w_r, x_2) = k_t\left(x_2 - w_r\right) \tag{2.14}$$

Table 2-2 States of the Quarter Car Model

1	x_1	Absolute displacement of sprung mass
2	x_2	Absolute displacement of unsprung mass
3	x_3	Absolute velocity of sprung mass
4	x_4	Absolute velocity of unsprung mass
5	$\mathrm{P_L}$	Pressure difference in piston chambers
6	x_v	Displacement of Power Valve

Table 2-3 Inputs to the Quarter Car Model

1	$\mathbf{W_{road}}$	Road input to the tire
2	$\mathbf{U_{valve}}$	Controller Input to the servovalve

2.3 Full Car Model

System Description

The vertical full car model represents the vehicle suspension as we know it; four corners, each having its' own spring, damper and actuator connecting the wheel to the sprung mass. The sprung mass now has three degrees of freedom, which are heave, roll, and pitch. Therefore, the whole model has 7 degrees of freedom, three for the chassis and one for each wheel.

Table 2-4 Degrees of freedom for the Full Car

1	**Chassis Heave**	Up and down movement of the chassis
2	**Chassis Roll**	Left to Right tilt of the chassis
3	**Chassis Pitch**	Rear to Front tilt of the chassis
4	**FL Wheel disp.**	Up and down movement of FL wheel
5	**FR Wheel disp.**	Up and down movement of FR wheel
6	**RL Wheel disp.**	Up and down movement of RL wheel
7	**RR Wheel disp.**	Up and down movement of RR wheel

The full car suspension can be modeled as an expansion of the quarter car suspension which is described in the previous chapter. The chassis can essentially be modeled as a rectangle, with moments of inertia along the x and y axis and a mass of 4 M_{sprung}. The combined forces of spring, damper and actuator act on each strut of the vehicle. As far as the wheel and actuator dynamics are concerned, equations 5, 6 and 7 remain valid, while 3 and 4 which correspond to the motion of the strut are altered since the quarter car sprung mass M_{sprung} has been replaced by the chassis' corner. The resulting system is depicted in fig.2-4.

The chassis can heave vertically (zb), and also roll (θ) and pitch (φ). Each wheel can be displaced in the vertical direction ($x1_{ii}$), while the road acts on them (wr_{ii}).

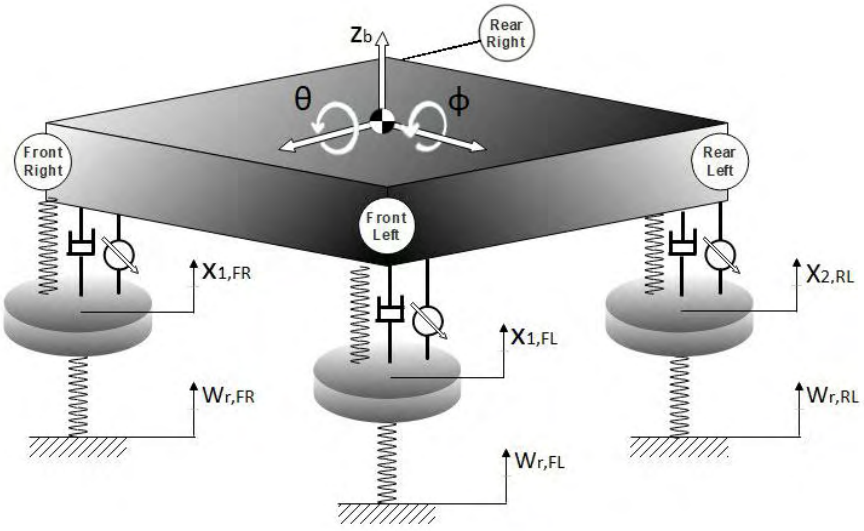


Figure 2-4 Schematic of the Full Car Model

System State Equations

For simplicity reasons, numbered notation is required for each variable. The index i ∈ $\{1,2,3,4\}$ corresponds to the specific strut of the full car, as {*Front Left, Front Right, Rear Left, Rear Right*}. The index j ∈ $\{1,2,3,4,5,6\}$ corresponds to the specific state derivative equation as described in the previous section that presented the quarter car model. For example, the subscript *(1,5)* indicates the pressure state of the Front Left strut.

By assuming that all displacements are absolute and along the vertical axis, the full car active suspension may be described by the following system of states:

Chassis State derivative equations:

$$
\begin{aligned}
&(A)\ \dot{x}_{5,1} = \frac{b_{track}}{2Ixx}\left(F_{strut}(2)+F_{strut}(4)-F_{strut}(1)-F_{strut}(3)\right)\\
&(B)\ \dot{x}_{5,2} = \frac{L_{base}}{2Iyy}\left(F_{strut}(1)+F_{strut}(2)-F_{strut}(3)-F_{strut}(4)\right)\\
&(C)\ \dot{x}_{5,3} = \frac{1}{4M_{spr}}\left(F_{strut}(1)+F_{strut}(2)+F_{strut}(3)+F_{strut}(4)\right)\\
&(D)\ \dot{x}_{5,4} = x_{5,1}\\
&(E)\ \dot{x}_{5,5} = x_{5,2}\\
&(F)\ \dot{x}_{5,6} = x_{5,3}
\end{aligned}
\qquad (2.15)
$$

Front Left Strut State derivative equations:

$$
\begin{aligned}
&(A)\ \dot{x}_{1,1} = -x_{5,1}\frac{b_{track}}{2} + x_{5,2}\frac{L_{base}}{2} + x_{5,3}\\
&(B)\ \dot{x}_{1,2} = x_{1,4}\\
&(C)\ \dot{x}_{1,3} = -\dot{x}_{5,1}\frac{b_{track}}{2} + \dot{x}_{5,2}\frac{L_{base}}{2} + \dot{x}_{5,3}\\
&(D)\ \dot{x}_{1,4} = \frac{1}{m_{unsp}}\left(-F_{ks}\left(x_{1,1},x_{1,2}\right)-F_{bs}\left(x_{1,3},x_{1,4}\right)-F_{kt}\left(\mathrm{w}_1,x_{1,2}\right)+A_p x_{1,5}\right)\\
&(E)\ \dot{x}_{1,5} = a\left[Q_{x_v}\left(x_{1,6},x_{1,5}\right)-Q_{x_{bv}}\left(x_{1,bv},x_{1,5}\right)-C_{tm}x_{1,5}-A_p\left(x_{1,3}-x_{1,4}\right)\right]\\
&(F)\ \dot{x}_{1,6} = \frac{1}{\tau}\left(-x_{1,6}+u_1\right)
\end{aligned}
\qquad (2.16)
$$

Front Right Strut State Derivative Equations:

$$
\begin{aligned}
(A)\ \dot{x}_{2,1} &= x_{5,1}\frac{b_{track}}{2} + x_{5,2}\frac{L_{base}}{2} + x_{5,3} \\
(B)\ \dot{x}_{2,2} &= x_{2,4} \\
(C)\ \dot{x}_{2,3} &= \dot{x}_{5,1}\frac{b_{track}}{2} + \dot{x}_{5,2}\frac{L_{base}}{2} + \dot{x}_{5,3} \\
(D)\ \dot{x}_{2,4} &= \frac{1}{m_{unsp}}\left(-F_{ks}\left(x_{2,1}, x_{2,2}\right) - F_{bs}\left(x_{2,3}, x_{2,4}\right) - F_{kt}\left(\mathrm{w}_2, x_{2,2}\right) + A_p x_{2,5}\right) \\
(E)\ \dot{x}_{2,5} &= a\left[Q_{x_v}\left(x_{2,6}, x_{2,5}\right) - Q_{x_{bv}}\left(x_{2,bv}, x_{2,5}\right) - C_{tm}x_{2,5} - A_p\left(x_{2,3} - x_{2,4}\right)\right] \\
(F)\ \dot{x}_{2,6} &= \frac{1}{\tau}\left(-x_{2,6} + u_2\right)
\end{aligned}
\tag{2.17}
$$

Rear Left Strut State Derivative Equations:

$$
\begin{aligned}
(A)\ \dot{x}_{3,1} &= -x_{5,1}\frac{b_{track}}{2} - x_{5,2}\frac{L_{base}}{2} + x_{5,3} \\
(B)\ \dot{x}_{3,2} &= x_{3,4} \\
(C)\ \dot{x}_{3,3} &= -\dot{x}_{5,1}\frac{b_{track}}{2} - \dot{x}_{5,2}\frac{L_{base}}{2} + \dot{x}_{5,3} \\
(D)\ \dot{x}_{3,4} &= \frac{1}{m_{unsp}}\left(-F_{ks}\left(x_{3,1}, x_{3,2}\right) - F_{bs}\left(x_{3,3}, x_{3,4}\right) - F_{kt}\left(\mathrm{w}_3, x_{3,2}\right) + A_p x_{3,5}\right) \\
(E)\ \dot{x}_{3,5} &= a\left[Q_{x_v}\left(x_{3,6}, x_{3,5}\right) - Q_{x_{bv}}\left(x_{3,bv}, x_{3,5}\right) - C_{tm}x_{3,5} - A_p\left(x_{3,3} - x_{3,4}\right)\right] \\
(F)\ \dot{x}_{3,6} &= \frac{1}{\tau}\left(-x_{3,6} + u_3\right)
\end{aligned}
\tag{2.18}
$$

Rear Right Strut state derivative equations:

$$
\begin{aligned}
(A)\ \dot{x}_{4,1} &= x_{5,1}\frac{b_{track}}{2} - x_{5,2}\frac{L_{base}}{2} + x_{5,3} \\
(B)\ \dot{x}_{4,2} &= x_{4,4} \\
(C)\ \dot{x}_{4,3} &= \dot{x}_{5,1}\frac{b_{track}}{2} - \dot{x}_{5,2}\frac{L_{base}}{2} + \dot{x}_{5,3} \\
(D)\ \dot{x}_{4,4} &= \frac{1}{m_{unsp}}\left(-F_{ks}\left(x_{4,1}, x_{4,2}\right) - F_{bs}\left(x_{4,3}, x_{4,4}\right) - F_{kt}\left(\mathrm{w}_4, x_{4,2}\right) + A_p x_{4,5}\right) \\
(E)\ \dot{x}_{4,5} &= a\left[Q_{x_v}\left(x_{4,6}, x_{4,5}\right) - Q_{x_{bv}}\left(x_{4,bv}, x_{4,5}\right) - C_{tm}x_{4,5} - A_p\left(x_{4,3} - x_{4,4}\right)\right] \\
(F)\ \dot{x}_{4,6} &= \frac{1}{\tau}\left(-x_{4,6} + u_4\right)
\end{aligned}
\tag{2.19}
$$

Functions:

$$
\begin{aligned}
(A)\ & Q_{x_v}\left(x_{\mathrm{i},6}, x_{\mathrm{i},5}\right) = C_{dP}S_{xv}x_{\mathrm{i},6}\sqrt{\frac{P_s - sign\left(x_{\mathrm{i},6}\right)x_{\mathrm{i},5}}{\rho}} \\
(B)\ & Q_{x_{bv}}\left(x_{i,bv}, x_{\mathrm{i},5}\right) = C_{dB}x_{i,bv}sign\left(x_{\mathrm{i},5}\right)\sqrt{\frac{2\,|\,x_{\mathrm{i},5}\,|}{\rho}} \\
(C)\ & F_{b_s}\left(x_{i,3}, x_{i,4}\right) = b_s^{lin}\left(x_{i,4} - x_{i,3}\right) - b_s^{sym}\,|\,x_{i,4} - x_{i,3}\,| + \mathrm{b}_s^{nonlin}\sqrt{|\,x_{i,4} - x_{i,3}\,|}\,\mathrm{sign}\left(x_{i,4} - x_{i,3}\right) \\
(D)\ & F_{k_s}\left(x_{i,1}, x_{i,2}\right) = k_s^{lin}\left(x_{i,2} - x_{i,1}\right) + k_s^{nonlin}\left(x_{i,2} - x_{i,1}\right)^3 \\
(E)\ & F_{k_t}\left(x_{i,2}, w_i\right) = k_t\left(x_{i,2} - w_i\right) \\
(F)\ & F_{strut}(i) = F_{k_s}\left(x_{i,1}, x_{i,2}\right) + F_{b_s}\left(x_{i,3}, x_{i,4}\right) - A_P\,x_{i,5}
\end{aligned}
\tag{2.20}
$$

Table 2-5 States of the Full Car Model

i,1	$x_{i,1}$	Absolute displacement of i strut
i,2	$x_{i,2}$	Absolute displacement of i wheel
i,3	$x_{i,3}$	Absolute velocity of i strut
i,4	$x_{i,4}$	Absolute velocity of i wheel
i,5	$x_{i,5}$	Pressure difference in i piston chambers
i,6	$x_{i,6}$	Displacement of i power valve
5,1	$\dot{x}_{5,1}$	Angular Roll Velocity of Chassis
5,2	$\dot{x}_{5,2}$	Angular Pitch Velocity of Chassis
5,3	$\dot{x}_{5,3}$	Absolute Velocity of Chassis
5,4	$x_{5,1}$	Roll angle of Chassis
5,5	$x_{5,2}$	Pitch Angle of Chassis
5,6	$x_{5,3}$	Heave Displacement of Chassis

Table 2-6 Inputs of the Full Car Model

1	w_1	Road input to Front Left Wheel
2	w_2	Road input to Front Right Wheel
3	w_3	Road input to Rear Left Wheel
4	w_4	Road input to Rear Right Wheel
5	u_1	Valve input to Front Left Actuator
6	u_2	Valve input to Front Right Actuator
7	u_3	Valve input to Rear Left Actuator
8	u_4	Valve input to Rear Right Actuator

Total States: 30

Total Inputs: 8

Total Outputs: 30 (all states)

3 Model Predictive Control

3.1 Introduction

Model Predictive Control (MPC) is one of the most widespread control methodologies currently in use in academic or industrial applications, second only to traditional PID controllers [22]. It relies on the principle that if a discrete-time model of the plant that correlates manipulated with controlled variables exist, then the inversion of this model would provide the sequence of optimal control moves in regards to the setpoint of the process. Therefore, an optimization problem can be formulated that needs to be solved in real time.

It is obvious from the above description that the success of the MPC methodology rests upon the approximation capabilities of the used discrete-time model. Most applications in the industry make use of linear models, with the advantages being easiness of formulation and real-time solving. For example, the Dynamic Matrix Control (DMC)[2] methodology has been applied in industrial systems through a linear finite-step response model whose parameters can be easily determined. At the same time, its' objective function is quadratic, so its minimization is rather trivial [22].

The downside is that linear models exhibit lacking performance when used to approximate nonlinear plants or processes. Linear models can nevertheless be used to approximate their near-linear regions, but these can be very narrow in some cases. Therefore, it is valid to consider nonlinear models that can offer better approximation capabilities throughout the whole region of interest, with the drawback of increased optimization complexity.

3.2 The MPC Framework

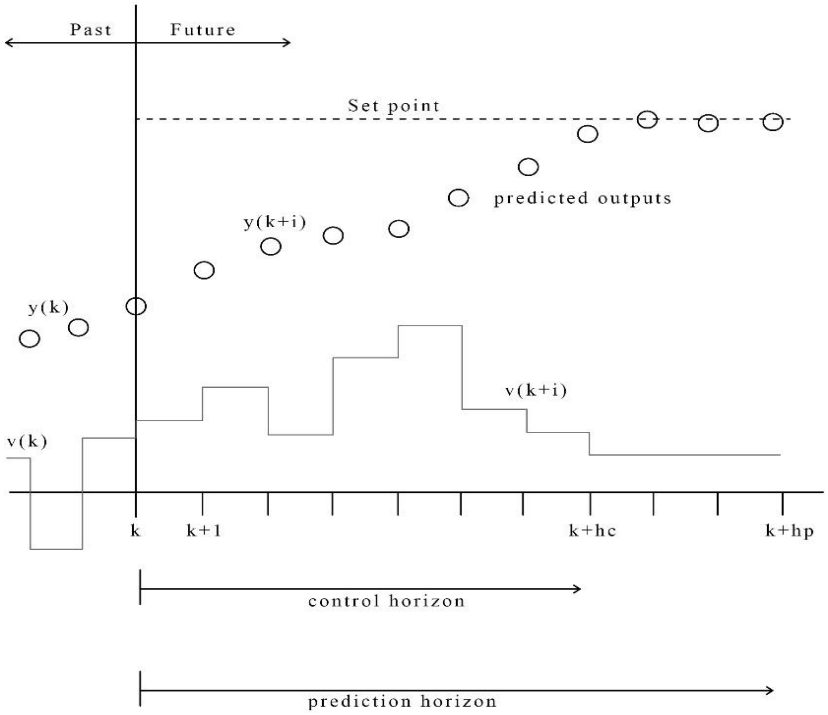


Figure 3-1 Schematic representation of the MPC Framework [22]

As described in the previous chapter, the idea is to yield the optimal control moves through the real time optimization of an objective function. These are the best sequence of values for the manipulated variables u, that will yield the optimum value of the objective function for given timesteps ahead of the current one. The number of these timesteps is called the prediction horizon hp, while the length of the sequence of optimal manipulated variables is called the control horizon hc. Fig. 3-1 depicts a schematic representation of the MPC characteristics.

The goal of the MPC algorithm then, is to find the optimal u values for every timestep up until the control horizon hc, that would yield the optimization of the objective function.

Model:

$$\hat{\mathbf{Y}}(k+1) = f\left(\hat{\mathbf{Y}}(k), \mathbf{u}(k)\right) \quad k = 1, \ldots, hp \tag{3.1}$$

Objective Function:

$$\min_{u(k),u(k+1),\ldots,u(k+hc)} \sum_{j=1}^{hp} \left[\left(\mathbf{Y}(k+j) - \mathbf{Y}_{sp}(k+j) \right) \Theta(j) \right]^2 \tag{3.2}$$

Subject to

$$u_{\min} \le u(k+j) \le u_{\max}, \; 1 \le j \le hc \tag{3.3}$$

$$\Delta u_{\min} \le \Delta u(k+j) \le \Delta u_{\max} \; 1 \le j \le hc \tag{3.4}$$

Y(k) is the current true vector of the controlled variables, which is sampled by an appropriate sensor set on the vehicle. The vector Y is the prediction of the controlled variables for j timesteps ahead. It is the heart of the MPC framework, since the effectiveness of the resulting control moves rely on the accuracy of the Y vectors produced by the discrete-time plant model. The Y vector is element-by-element multiplied by the theta vector, which weights each Y prediction in the objective function. The theta vector serves as a tuning tool. If the objective function is multiobjective, then an additional weight vector is used. As far as equations (3.3, 3.4) are concerned, they represent constraints of physical nature for the controlled variables, and in our case describe servovalve saturation and gradient constraint.

The algorithm retains and applies to the plant the optimal control move u(k) for the current timestep. It then reruns the optimization for the next timestep which is formulated with the new sampled initial conditions. In this application, the previous vector of optimal control moves found is stored and used as a starting point for the next optimization (fig.3-2).

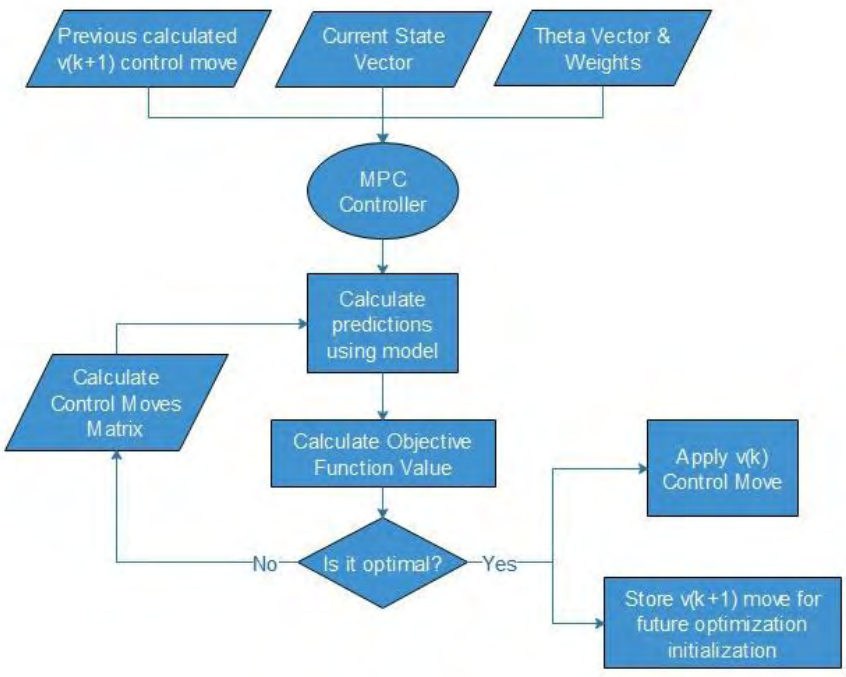


Figure 3-2 Schematic of the MPC Framework

Moreover, in real life applications, the MPC controller does not have infinitesimal computation times, meaning that there is a minimum time period between two subsequent controller moves. This time period is referred to as “controller sample time” and varies with the complexity of the model as well as the hardware specifications of the controller. In [11] they factored in for 30ms of controller sample time. In this work, for the sake applying more complex nonlinear models, we have opted for a 10Hz Controller Bandwidth. In other words, this means that the controller is able to execute a control move every 0.1s.

3.3 The MPC-Preview Framework

In the previous section, the "traditional" MPC Controller was introduced. However, an alternative formulation to it exists, which can relay upcoming road information to the controller.

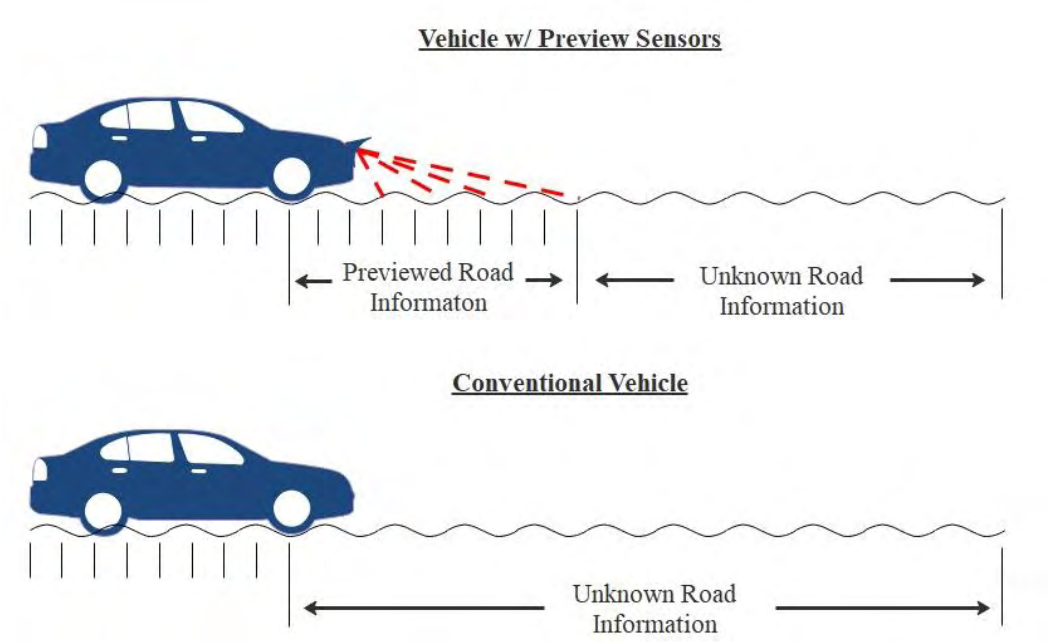


Figure 3-3 Schematic of the MPC Preview application in a road vehicle

This alternative MPC formulation is called Model Predictive Control with Preview Information (MPC-P), and can be realized with appropriate optical sensors on the vehicle [9] which scan the road profile ahead (fig. 3-3). This information is then input to the prediction model of the controller so that more accurate state estimation can occur. This will in turn result in better control moves.

In a practical application, the optical sensors have a fixed range of road length that they can scan, or alternatively, a time range. This time range is dependent on vehicle speed, and it is worth noting that it is usually smaller than the prediction horizon of the controller. As a result, in an MPC-P Controller evaluation, road preview information will be available only for a fraction of the prediction horizon time, which is called the preview horizon h_{prev}. Beyond that, the road preview information vector is zero. It is noteworthy than in this work, ideal preview performance is assumed; this means that no preview error exists over the sampled road information.

The MPC-P Controller can be described in mathematical notation:

Model:

$$\hat{\mathbf{Y}}(k+1) = f\left(\hat{\mathbf{Y}}(k), \mathbf{u}(k), \mathbf{w}(k)\right) \quad k = 1, \ldots, hp$$

$$where\ \mathbf{w}(k) = \begin{cases} \mathbf{w}_{prev}(k) & 1 \le k \le h_{prev} \\ 0 & k > h_{prev} \end{cases} \tag{3.5}$$

Objective Function:

$$\min_{u(k),u(k+1),\ldots,u(k+hc)} \sum_{j=1}^{hp} \left[\left(\mathbf{Y}(k+j) - \mathbf{Y}_{sp}(k+j) \right) \Theta(j) \right]^2 \tag{3.6}$$

Subject to

$$u_{\min} \le u(k+j) \le u_{\max}, \quad 1 \le j \le hc \tag{3.7}$$

$$\Delta u_{\min} \le \Delta u(k+j) \le \Delta u_{\max}, \quad 1 \le j \le hc \tag{3.8}$$

In the MPC-P Formulation, the prediction model accepts the road input vector **w**(k) along with the valve and state inputs, **u**(k) and **Y**(k) respectively. An important consideration regarding the road input vector **w**(k) needs to be expressed: its' sample rate does not necessarily coincide with the controllers', in fact it is usually much higher. This means that the road input vector contains sampled road points that are between the controllers' timesteps k. A schematic representation is in order (fig. 3-4):

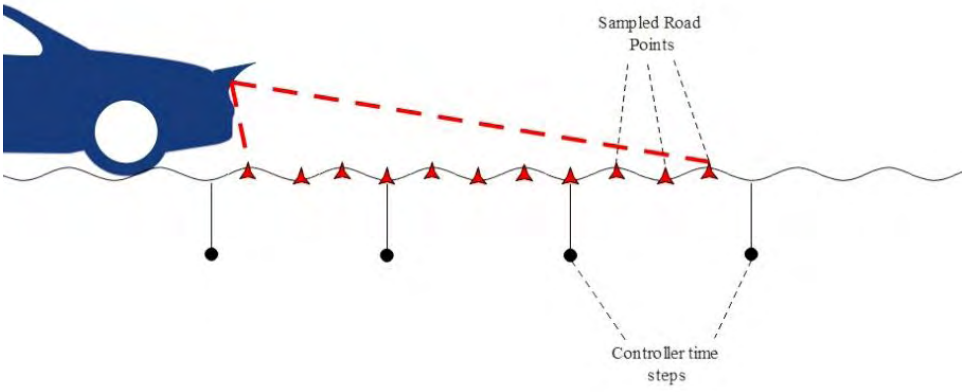


Figure 3-4 Schematic of the preview sampling and the controller sample points

This difficulty can be overcome by creating a prediction model that will incorporate the road information in between the controller timesteps, and is something that will be described in the following chapters.

3.4 Tuning the MPC Controller

The MPC Controller has four items of tuning:

- Control horizon
- Prediction horizon
- Theta vector
- Objective function weights.

The effect of the first two are straightforward on the Controller. Reducing the prediction horizon means that the Controller has less available time to guide the controlled variables to the setpoint. Reducing the control horizon means that the Controller has less available moves to control the controlled variables, therefore more aggressive actions have to be applied. The relevant difference between the prediction and control horizon is also of interest; it needs to be more than the settling time of the system, otherwise the last control move will not realize all of its potential effect on the controlled variables by the end of the prediction horizon.

Thus, a method for calculating the initial values of the control and prediction horizon can be applied. First, the settling time of the controlled variable after a valve input is calculated, with a 10% margin.

If the settling time is Δt seconds for the chassis vertical acceleration, the prediction horizon will be Δt / 0.1 discrete time steps, since the controller sample time is 0.1. Thus, the prediction horizon is

$$h_p = h_c + \frac{\Delta t}{0.1} \qquad (3.9)$$

The control horizon is initially chosen to be 3 – it needs to remain lower than 4 since the computational cost of optimizing in real time an objective function with 16 optimized variables (4 control moves x 4 valve inputs) is deemed too high. The optimal values of hc, theta and objective function weights are calculated iteratively as follows (fig. 3-5):

- The iteration initializes with the creation of a linearly spaced theta vector from 0 to 1. The objective function weight vector initializes with values of inversely proportional magnitude to the corresponding weighted states - the idea is to bring all weighted states to the same order of magnitude in the objective function.
- We then need to find an acceptable weight vector to begin the main iteration. A useful tool is to plot the values of each weighted state in the objective function throughout the benchmark test; This way we can establish which weighted state dominates the final value of the objective function in every stage of the benchmark test.
- Once an acceptable weight vector has been found, the main iteration can begin for a given hc value: An interior-point optimization procedure is applied to find the optimum theta vector, with a variable range of (0,1). Then, the weight vector is calculated again, and the inner iteration restarts. When an acceptable result is reached, the hc is changed and the main iteration restarts.
- In the end, the optimum tunings for each hc are compared and the best one is applied to the controller.

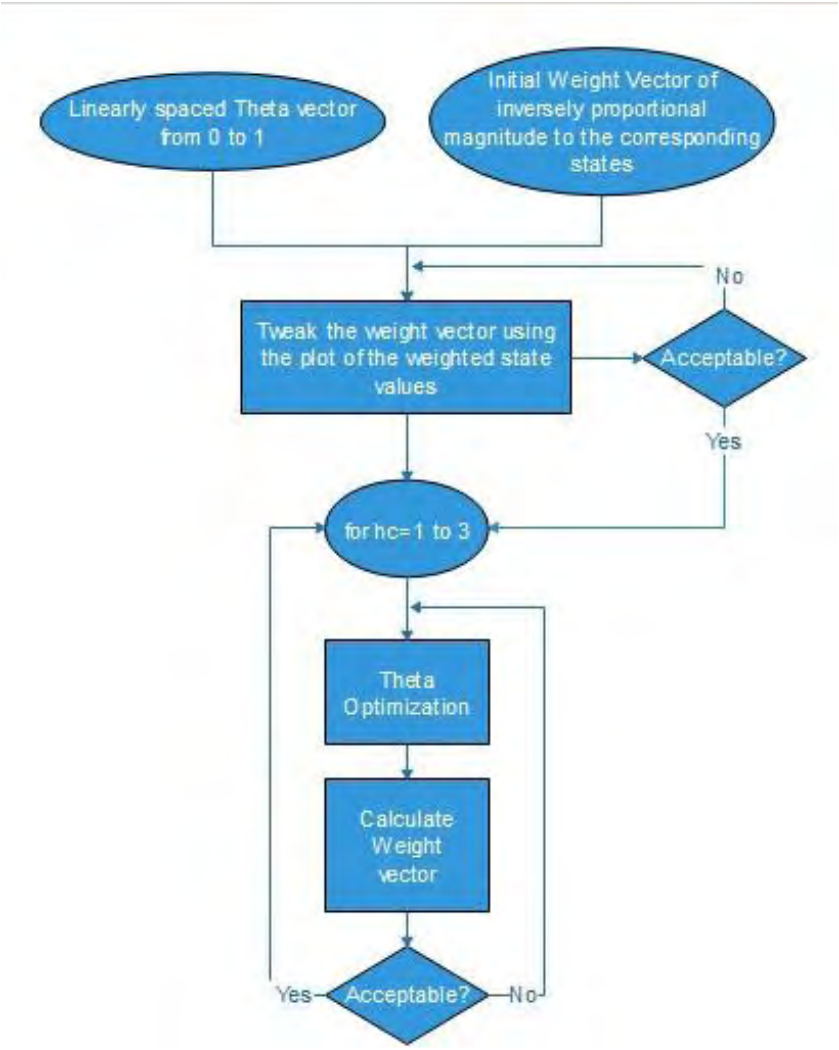


Figure 3-5 Schematic of the algorithm for the tuning of the MPC Controller

The benchmark test that these parameters are evaluated on will determine the application of the MPC Controller; for example, if it is to be used on a road sedan or executive car (which are supposed to travel on roads of A quality with the occasional road bump), then the appropriate benchmark test is a pulse road overlaid with a random road of A quality. If, on the other hand, the controller is to be used on an offroad vehicle, then a random road of E quality should be used. [23]

In this work, the two MPC Controllers that are created were tuned in both benchmark tests, which has resulted in two sets of tuning parameters for each. The basis for this approach was two-fold: Firstly, we would like to avoid limiting the range of the applications of the MPC Controller. Secondly, both sets of tuning parameters can be incorporated in one controller for the purpose of an all-round vehicle, such as a Sport Utility Vehicle. In such an application, the driver or even the Controller itself can switch between the 'on-road' or 'off-road' tuning set to adapt to the changing road conditions.

Therefore, in the following chapters where the MPC implementation is analyzed in depth, two sets of tuning parameters are created and evaluated.

4 Radial Basis Function Networks

4.1 Introduction

In the case of the fullcar active suspension, there are many sources of nonlinearities that would urge us to choose a more advanced model to approximate the plant output. For example, the fluid flow through the servo valve and the pressure in the system are highly non-linear phenomena containing root and denominator terms (eq. 2.2-2.4). At the same time, the spring and damper rates for the suspension contain third order terms as well as non-symmetric terms (eq. 2.12-2.14). It is of interest then, to develop a non-linear model of the plant intended for usage in the MPC Framework.

Radial Basis Function (RBF) networks are a popular Neural Network architecture with application in nonlinear systems, both in industry and academia. They are comprised of an input layer and a single hidden layer with linear attachments to the output layer of the network. As a result, their training algorithms are faster and more efficient than the more complicated multilayer perceptron (MLP) counterparts. [19]

In RBF networks, the training aims to find the multidimensional surface that best approximates the training data, which is constituted as a sum of simpler surfaces exhibiting radial basis symmetry around centers specifically placed in the input space. Therefore, training an RBF network with constant widths corresponds to finding the following parameters:

- Number of RBFs
- Coordinates of RBF centers in the input space
- Synaptic weights for the hidden layer to output layer connections

Great focus is placed on the determination of RBF center locations in the input space. An initial approach [24, 25] is to place the center of each RBF on top of each datapoint, which would result in a very large number of hidden nodes, thus compromising computational efficiency, especially in large datasets such as ours. A workaround is to cluster the input data into a number of regions far smaller than the number of data. Then, RBFs are placed at the center of each data cluster.

So, the focus shifts towards finding an appropriate clustering algorithm; one approach is k-means [26] which requires a trial-and-error procedure. An alternative solution is the fuzzy means algorithm [27], which generates the RBF centers and their locations with good computational efficiency by segmenting the input space through fuzzy clustering.

4.2 RBF Neural Networks Framework

As described in the previous section, RBF networks are inherently simple, containing an input layer and a single hidden layer connected with weighted attachments to the output layer. Fig. 4-1 shows the typical structure of an RBF network:

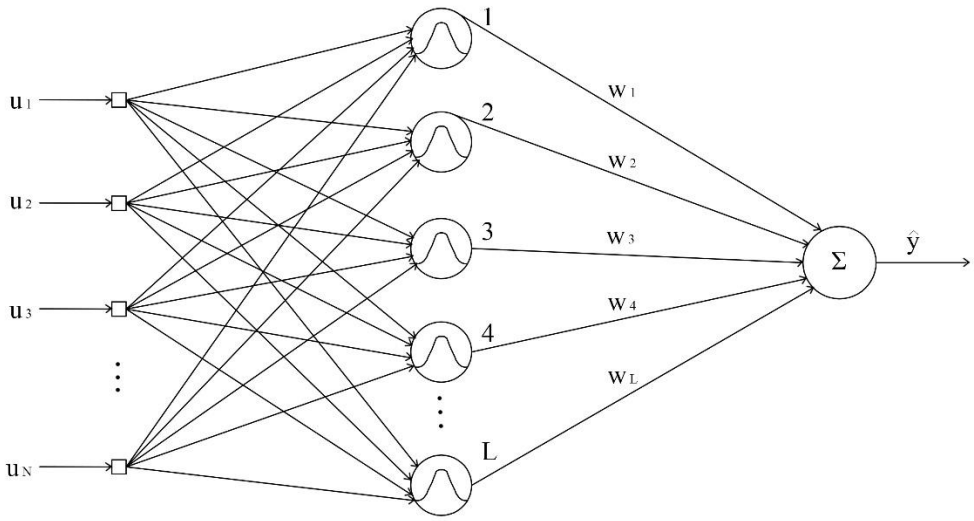


Figure 4-1 Typical RBF Structure with N inputs and L hidden nodes, [19]

The input layer distributes N input variables to L nodes of the hidden layer. Each node in the hidden layer is comprised of a center with N dimensions, which essentially constitutes a nonlinear transformation mapping the input space on a new, higher dimensional space.

After inputting the u vector, the first step in computing the output is to compute the activity μ for every node l. For RBF networks, it is the Euclidean norm of the difference between the kth input vector $\mathbf{u}(k)$ and the lth node center $\hat{\mathbf{u}}_l$

$$\mu_l\left(\mathbf{u}(k)\right) = \left\|\mathbf{u}(k) - \hat{\mathbf{u}}_l\right\| = \sqrt{\sum_{i=1}^{\mathrm{N}}\left(\mathbf{u}_i(k) - \mathbf{u}_{i,l}\right)^2}, \quad k = 1,\ldots,\mathrm{K} \qquad (4.1)$$

In the above equation, K is the number of datapoints, $\mathbf{u}(k)$ is the input vector and $\hat{\mathbf{u}}_l$ is the RBF center of node l. Using the activity value, the activation function of the node can be computed. In this work, a thin plate spline function is used; note that the activity needs to be shifted +1 to the right so that the activation function can be defined even if the activity is equal to zero:

$$g(\mu_l) = \mu_l^{\,2}\log(\mu_l + 1) \qquad (4.2)$$

For each datapoint, an activation function value is computed for a specific node l. The hidden node responses for all datapoints K are written as such:

$$\mathbf{z}(k) = \left[g\left(\mu_1\left(\mathbf{u}(k)\right)\right), g\left(\mu_2\left(\mathbf{u}(k)\right)\right), \ldots, g\left(\mu_L\left(\mathbf{u}(k)\right)\right)\right] \qquad (4.3)$$

The final output $\hat{y}$ of the RBF network is calculated as a linear combination of the hidden node responses as

$$\hat{y}(\mathrm{k}) = \mathbf{z}(k) \cdot \mathbf{w} \qquad (4.4)$$

where w is a vector containing synaptic weights:

$$\mathbf{w}^T = [w_1, w_2, \ldots, w_L] \qquad (4.5)$$

The training procedure of the RBF Network essentially consists of finding the node centers and then computing the synaptic weights via linear least squares between the real output and the predicted output for all training data:

$$\mathbf{w}^T = \mathbf{Y}^T \cdot \mathbf{Z} \cdot \left(\mathbf{Z}^T \cdot \mathbf{Z}\right)^{-1} \qquad (4.6)$$

4.3 The Fuzzy Means Algorithm

The Fuzzy Means (FM) Algorithm [27] has been successfully applied to the automatic control of industrial processes [28] and intelligent control [29]. There exist numerous variations of the algorithm, however in our application the symmetric version is used.

Consider a system with N normalized input variables u_i. Fuzzy partitioning of the input space would correspond to the partition of the domain of each input variable u_i into s_i number of (1-D) triangular fuzzy sets, which are equal to $\delta\alpha$. In mathematical notation that is:

$$A_{i,j} = \{a_{i,j}, \delta a\}, \quad i = 1, \ldots, N \quad j = 1, \ldots, s_i \tag{4.7}$$

$\alpha_{i,j}$ is the center of fuzzy set $A_{i,j,}$ $\delta\boldsymbol{\alpha}$ is half the width, which is the same for each input direction. The result of the partition is a total of S fuzzy subspaces $\mathbf{A}^l$:

$$S = \prod_{i=1}^{N} s_i \tag{4.8}$$

Each subspace A is created by the combination of N fuzzy sets for each input direction. It is possible to define the fuzzy subspaces through the center vector $\boldsymbol{\alpha}^l$ containing the centers and the side vector $\delta\boldsymbol{\alpha}$ containing the width halves.

$$A^l = \{\boldsymbol{\alpha}^l, \delta\boldsymbol{\alpha}\} = \left\{\left[\boldsymbol{\alpha}^l_{1,j1}, \boldsymbol{\alpha}^l_{2,j2}, \ldots, \boldsymbol{\alpha}^l_{N,jN}\right], \left[\delta\boldsymbol{\alpha}_1, \delta\boldsymbol{\alpha}_2, \ldots, \delta\boldsymbol{\alpha}_N\right]\right\} \quad l = 1, \ldots, S \tag{4.9}$$

The resulting subspaces form a grid in the N-dimensional input space, where each node of the grid can become an RBF center, and it is the objective of the algorithm to determine which ones will be finally selected. Here a tradeoff exists; a small subset of subspaces should be selected in order to create a computationally efficient model, but at the same time they should be able to describe the training datas' space sufficiently.

The FM Algorithm makes this selection through the use of the membership function which will indicate whether the subspace $\mathbf{A}^l$ will be a selected subset.

$$\mu_{\mathbf{A}^l}(\mathbf{u}(k)) = \begin{cases} 1 - d_r^l(\mathbf{u}(k)) & , \; if \;\; d_r^l(\mathbf{u}(k)) \le 1 \\ 0, & \quad if \;\; otherwise \end{cases} \tag{4.10}$$

In our application a symmetric fuzzy partition is applied – thus the distance equation d_r^l can be formed as

$$d_r^l(\mathbf{u}(k)) = \sqrt{\frac{\sum_{i=1}^{N}\left(a^l_{i,ji} - u_i(k)\right)^2}{\sqrt{N}\;\delta\alpha}} \tag{4.11}$$

where $\mathbf{u}(k)$ is the kth input vector, $a^{l}_{i,ji}$ is the center of the fuzzy subspace $\mathbf{A}^{l}$, and N is the dimensionality of the input space. The above equation defines a surface in the input space that bounds the input vectors that will be included in the fuzzy subspace $\mathbf{A}^{l}$, or in other words, that will receive nonzero membership degree in the membership function.

It is apparent that the resulting RBF centers depend solely on the number of fuzzy sets s, which is the same for all input variables. This is the basis of the easiness of tuning of the RBFN model through the FM Algorithm, since the optimum number of fuzzy sets can be found heuristically in a discrete region.

5 Application and Results

5.1 Introduction and Design Objective

In the previous chapters the mathematical and theoretical framework, as well as some of the practical aspects of the system in question, have been presented. As stated, the goal of this work is to propose a new control scheme for the Full Car vertical dynamics control problem. In this respect, three different controllers are assessed:

- Traditional PID controllers
- MPC strategy with a linear model of the plant
- MPC strategy with an RBFN model of the plant

Even though the MPC implementation allows for multiobjective control, for the sake of comparison with the PID strategy it was chosen that only one variable should be controlled. This is because PID controllers are strictly SISO, whereas MPC can be MIMO. In this work, the vertical acceleration has been chosen as a controlled variable, because it is a physically tangible metric for comfort in road vehicles.

This chapter is structured as follows: First, the three implementations are presented, together with the methodologies for tuning them according to the design objective. Then, they are applied to two benchmark tests, one being a pulse test, and one being a random road test. Finally, they are compared to each other using specific metrics.

5.2 Implementation of PID controller

The implementation of the PID control strategy begins with the creation of a Simulink ™ quarter car model. Simulink offers a schematic programming environment where it is easy to realize many controllers, including a PID one. Moreover, it contains a powerful programming block called S-Function, which can accommodate discrete models such as ours. A Simulink program containing the S-function quarter car model as described in equations (2.6-2.11) is shown in fig. 5-1:

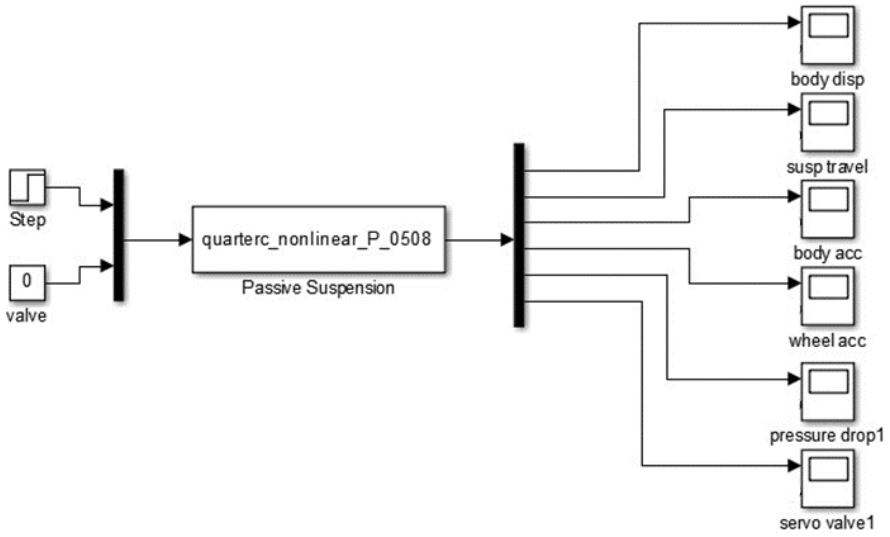


Figure 5-1 A Simulink representation of a Passive Quarter Car Suspension

The input vector to the S-Function contains two variables: The road input and the valve input. The S-function then calculates the state derivatives and returns them back to Simulink where they are integrated. The output vector contains the body displacement, suspension travel, body acceleration, wheel acceleration, pressure difference, and servovalve position, which are plotted throughout the simulation timespan.

A Simulink program describing an active suspension quarter car is shown in fig. 5-2:

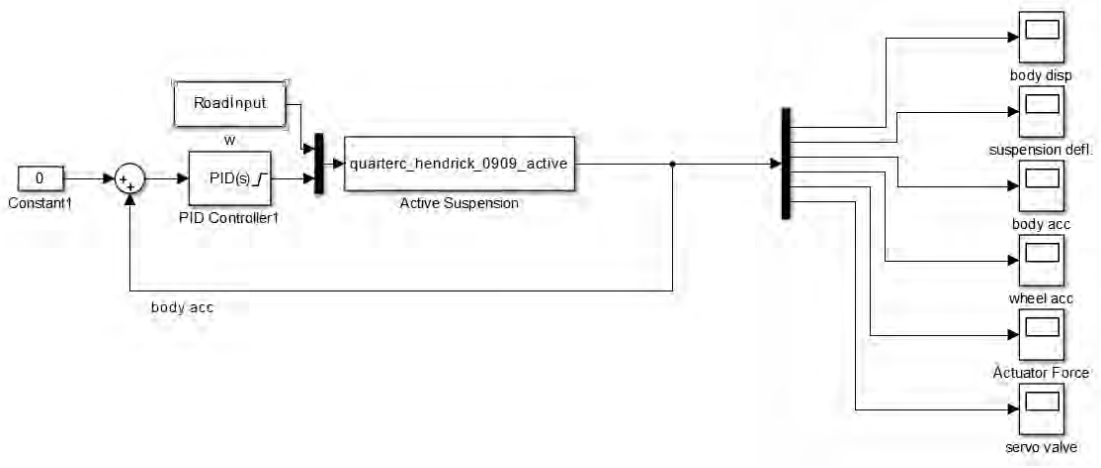


Figure 5-2 A Simulink representation of an Active Quarter Car Suspension

Here, the valve input is no longer zero, and is dictated by the feedback loop, setpoint and PID tuning. The setpoint of the controlled variable is of course zero, and the PID can be tuned by the built-in autotuner script in Simulink. This script performs plant identification based on test input and measured output and uses the current operating point to create a linearized plant model. Afterwards, it computes the P, I, D parameters. The results are:

Table 5-1 PID tuning parameters

	Parameter	Value
1	Proportional (P)	1.40094e-12
2	Integral (I)	3.70705e-13
3	Derivative (D)	9.87406e-13
4	Filter Coefficient (N)	3.91276

Whereas the PID Controller transfer function is:

$$G_{PID}(s) = P + I\frac{1}{s} + D\frac{N}{1+N\frac{1}{s}} \qquad (5.1)$$

After creating the quarter car Simulink model and successfully tuning it, the Full Car Simulink Model could be implemented using the eqs. 2.15-2.20. The input vector contains eight variables, four valve inputs and four road inputs (one for each wheel), whereas the output vector contains the heave displacement, theta angle, phi angle, and heave acceleration.

However, the PID implementation here differs since the system is no longer SISO. In order to overcome this difficulty, 4 PID controllers are implemented, one intended for each corner where they are fed back the respective measured vertical acceleration (fig.5-3). The result is that the system achieves the desired operation.

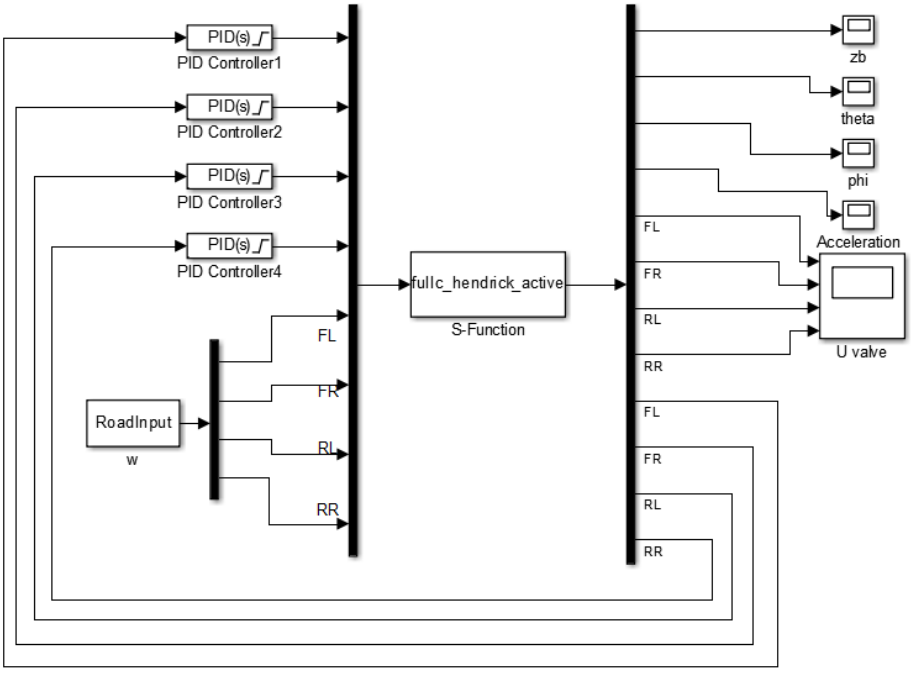


Figure 5-3 A Simulink representation of an Active Full Car Suspension

5.3 Implementation of MPC Controller with Linear Model

In contrast to the PID controller, the Simulink approach had to be abolished, since the implementation of an MPC Controller in *MATLAB* requires more advanced programming tools. Therefore, the plant model had to be realized in *MATLAB* code, where now the state derivatives are numerically integrated using the *ode45* solver. The optimization problem within the MPC controller is solved using the *fmincon* optimization function (see Appendix I).

Creating the Linear Least Squares Model

In order to create a linear model of the system, two types of datasets need to be created: One containing all input to the system, and one containing all output. Then, simple linear least squares can be performed so that we can extract the linear coefficients that correlate the input variables with the output of the model. Since we aim to use it in an MPC scheme, the model ought to be discrete-time:

$$x_{k+1} = A \cdot x_k + B \cdot u_k + C \cdot w_k \qquad (5.2)$$

Eq. (5.2) is a linearized discrete-time model of the original state equations. The next state vector x_{k+1} is calculated using the current state vector x_k, valve input vector u_k, and road input vector w_k. The matrix A incorporates the contribution of the current state vector to the next state vector. the Matrices B and C incorporate the contribution of the valve and road model inputs respectively, to the next state vector.

As mentioned in the MPC Chapter, the controller sample time has been chosen to be 100ms, so one discrete step of the model should correspond to a 0.1s time step. This should be reflected in the training data, meaning that the datapoints should be sampled with a 10Hz sampling frequency.

In addition, the data should be adequate in number and in quality in order to produce a good model: This means that the data should describe sufficiently the system in question, which is achieved by utilizing enough datapoints from all the input variable region. That way, the resulting *Xreg* regression matrix will not be rank deficient [30]. Another numerical consideration is that the linear least squares input data should all be within the same order of magnitude; otherwise matrix ill-conditioning problems will occur. If this does happen, a common way to amend it is by scaling the problematic variables (see Appendix I).

One important consideration is that input data should not be omitted from the matrices that will be used to create the input model. In other words, all the "information" that was input to the full car plant should be made available to the model through the regression matrices *Xreg* and *Yreg*. Sampling from the simulation data for the *Xreg* matrix with a sample time of 0.1s is going to yield incorrect results, because the road values that change every 0.0025s (simulations' timestep) will not be included. In order to accommodate for this consideration, the discrete model should be changed accordingly.

The creation of the Linear model follows the following procedure (fig. 5-4):

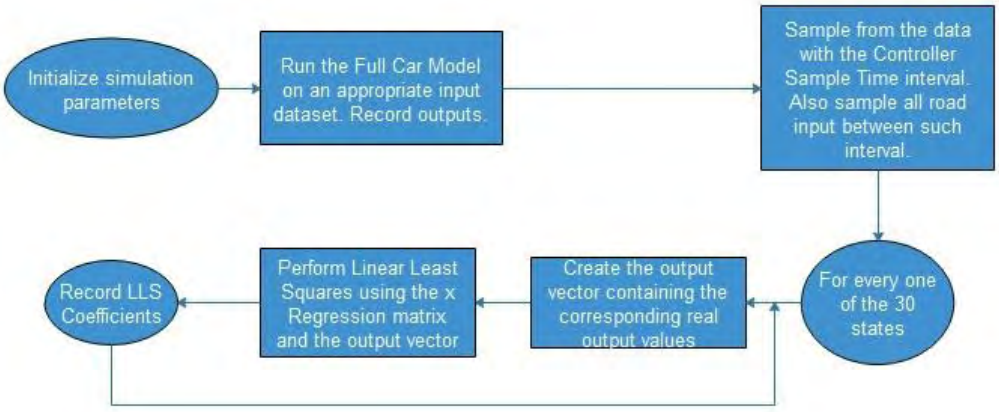


Figure 5-4 Schematic of the algorithm for creating a Linear car model

- **Simulation Parameters:** The simulation parameters of the Full Car model evaluation are a simulation time step of 0.0025s, a controller sample time of 0.1s, and a vehicle speed of 10m/s.

- **Creating the Input to the Full Car:** As described previously, the input variable data is the most important part in the creation of a model of a system. The inputs to the model are the *u* valve inputs and the *w* road inputs. As far as the valve inputs are concerned, they are uniform-randomly picked from the range of the servovalve and are changed every 0.1s of simulation time (or every 40 simulation timesteps), so that it corresponds to the actual MPC Controller operation (fig. 5-5).
  Secondly, the random road inputs are formulated as per the ISO 8606, simulating a road of bad condition [31]. In order to reproduce the real-life characteristics of a full car suspension, the road inputs differ between the left and right side (fig. 5-5).

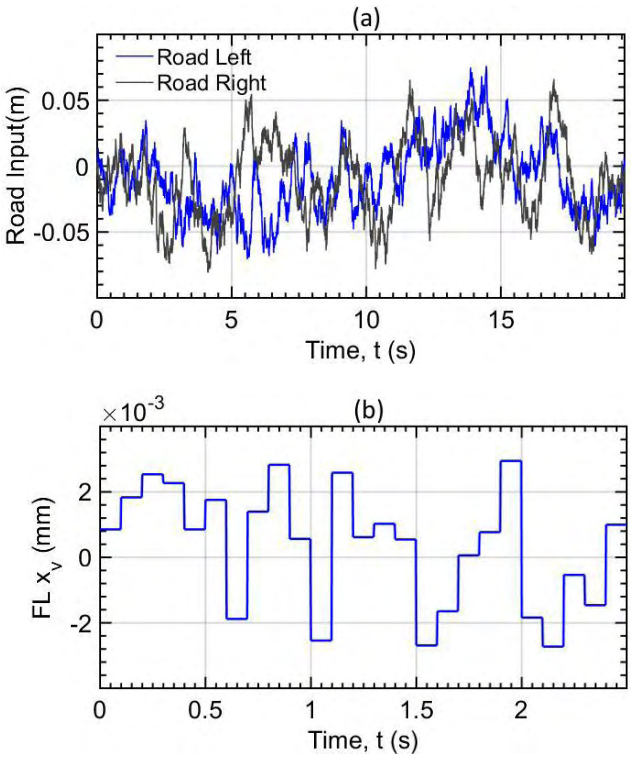


Figure 5-5 Road (a) and Valve (b) inputs to the Full Car Plant

Moreover, the rear wheels receive the same road input as the front ones with a time delay that is related to the speed of the vehicle. The result is an input dataset *w* of size 3.199.364 x 8 (Datapoints x Inputs).

- **Evaluation of the Full Car plant:** Given the input matrix *w* containing the road and valve inputs throughout the simulation timespan, the *ode45* solver calculates and returns the state matrix *x* of the plant. Note that this timespan has a timestep of 0.0025s. The state matrix *x* is of size 3.199.364x30.

- **Creation of the regression matrices *Xreg* and *Yreg*:** Since we are interested in creating a model of the plant with a timestep of 0.1s, we need to formulate the regression matrix *Xreg* in such a way that no information that was used by the actual plants' simulation is omitted from the regression of the linear model. In our application, this means that the road profile information that lies between two discrete controller steps (k → k+1) should be included as a model input. The resulting *Xreg* matrix contains 62946 datapoints for 194 model input variables, whereas the *Yreg* matrix contains the 30 real state outputs, which are shifted one datapoint up in order to create the $y(k+1) = f(x(k))$ form.

  A desirable addition to the Linear Least Squares model is to add a y-intercept for every state. This is done by appending the *Xreg* matrix by a column of ones. Then, the y-intercepts are computed as parameters in the Linear Regression between the *Xreg* and *Yreg* matrices.

- **Performing the linear regression procedure:** The linear regression is carried through using the \ MATLAB operator. It is applied iteratively for the total of 30 states of the model. The result of every iteration is a vector of 195 coefficients that correlate the 195 inputs with the specific state. These are stored and used to create the coefficient matrices A, B and C, which contain the coefficients correlating state, valve, and road inputs respectively with the output states.

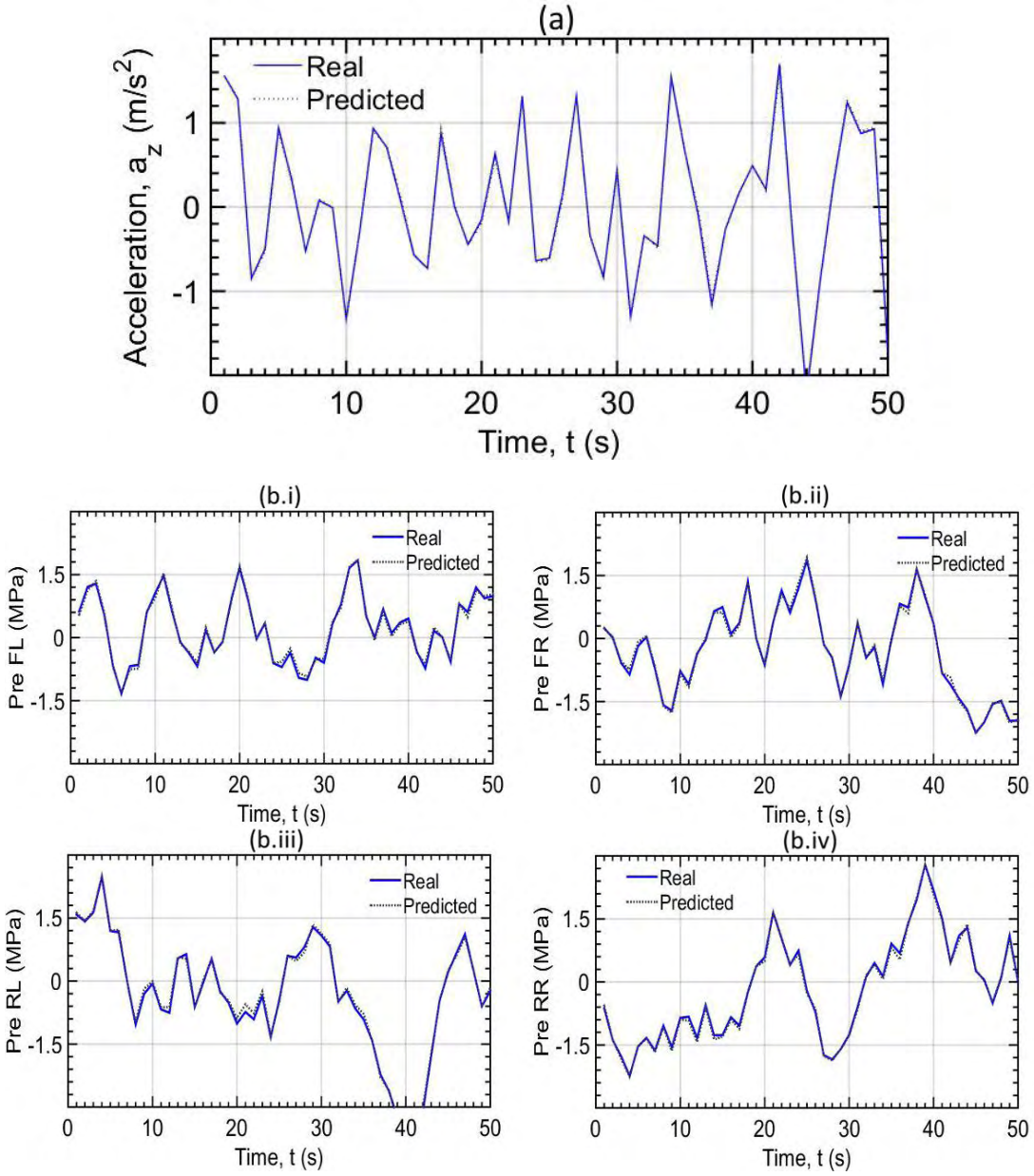
Validating the Linear Least Squares Model

Every model that is fitted to data needs to be validated using the R^2 and RMSE metrics. The first step to evaluating a fit is applying those metrics to the predictions of the model when the input data are its' training data. The second step is to apply the metrics to predictions of the model when the input data are different from the ones it has been trained on. The usual approach to model creation is that the total data available for the creation of the model are split using a ratio of 70-30 into training and validation data. The results of the training and validation of the LLS model are presented in table (5-2), for all states.

Table 5-2 The R^2 and RMSE Metrics for every state of the Linear car model. The pressure states are in blue.

State	R^2 Training	R^2 Validation	RMSE Training	RMSE Validation
1	1,0000	1,0000	0,00015	0,00015
2	1,0000	1,0000	0,00010	0,00011
3	0,9995	0,9995	0,00346	0,00359
4	0,9995	0,9995	0,00338	0,00334
5	0,9896	0,9889	0,03420	0,03639
6	1,0000	1,0000	0,00006	0,00006
7	1,0000	1,0000	0,00015	0,00016
8	1,0000	1,0000	0,00010	0,00010
9	0,9995	0,9995	0,00353	0,00361
10	0,9995	0,9995	0,00335	0,00339
11	0,9890	0,9889	0,03519	0,03500
12	1,0000	1,0000	0,00006	0,00007
13	1,0000	1,0000	0,00015	0,00015
14	1,0000	1,0000	0,00011	0,00011
15	0,9996	0,9995	0,00350	0,00361
16	0,9995	0,9995	0,00332	0,00332
17	0,9887	0,9877	0,03557	0,03637
18	1,0000	1,0000	0,00006	0,00006
19	1,0000	1,0000	0,00015	0,00015
20	1,0000	1,0000	0,00010	0,00010
21	0,9996	0,9996	0,00343	0,00349
22	0,9995	0,9995	0,00333	0,00329
23	0,9893	0,9900	0,03446	0,03302
24	1,0000	1,0000	0,00006	0,00006
25	0,9993	0,9993	0,03037	0,03094
26	0,9945	0,9946	0,02597	0,02579
27	0,9990	0,9990	0,01782	0,01753
28	1,0000	1,0000	0,00007	0,00007
29	0,9999	0,9999	0,00006	0,00007
30	1,0000	1,0000	0,00004	0,00004

All the above values have been rounded to the fourth significant digit. The x(i,5) pressure states have the worst performance in terms of goodness of fit, as indicated by their R^2 value. The states of x(i,3) and x(i,4), which correspond to strut and wheel velocity respectively, also exhibit average goodness of fit. These results are to be expected because the respective equations are nonlinear (eq. 2.15-2.18 C,D). On the other hand, the rest of the states are almost perfectly predicted by the LLS model, by exhibiting R^2 of almost 1.



Figures 5-6 Acceleration (a) and Pressure (b) Predictions of the Linear car model in a validation random road & input test

As shown in fig. 5-6, in a random road and valve validation test the LLS model can predict the vertical acceleration perfectly, as well as the pressure states, with some minor incongruities. Note that the input data in this test have the same characteristics as the ones that the model has been trained on.

One additional validation test is due, as the models are intended to be used in the MPC context: Since the MPC Controller that is being applied accepts road preview information (meaning that the upcoming few meters of road profile are available to the model making the predictions), the only input to the model from the real world is the initial condition state vector and a road vector containing the road profile for some of the prediction steps. It is through this information that the model makes the predictions, and upon this that the MPC Controller calculates the control moves. It can be concluded then, that the response of the model in a relevant test is of high importance in the MPC Scheme.

Such a test is an initial condition test, where all valve and road inputs are zero.

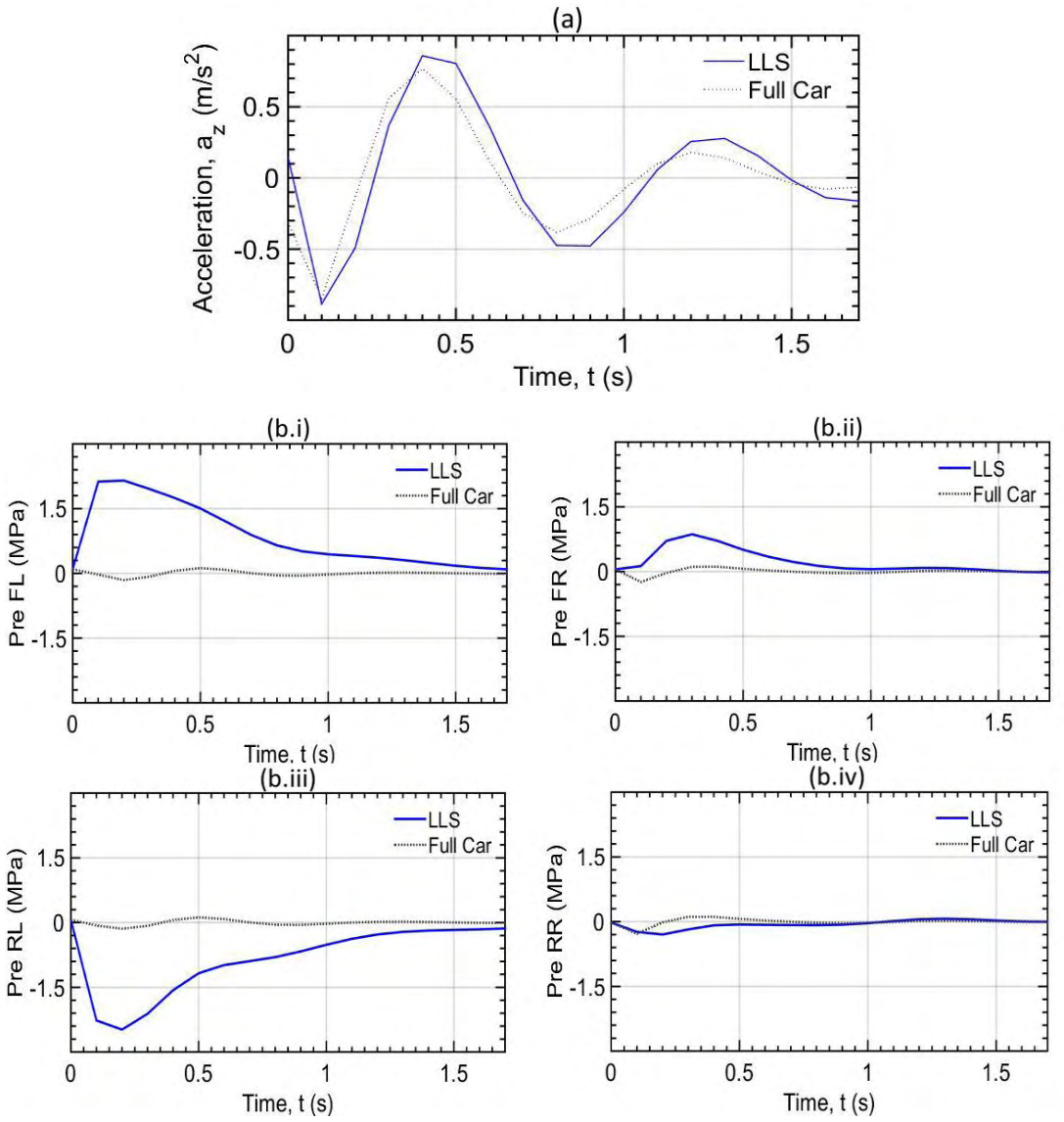


Figure 5-7 Acceleration (a) and Pressure (b) predictions of the Linear car model in an initial condition validation test

The LLS model succeeds in predicting the state of chassis acceleration, however it is completely unable to predict the pressure states (fig. 5-7). This indicates that in the MPC context the objective function values will reflect the reality with satisfying accuracy, however the controller will not be able to assess the effect of the control moves, due to the significant deviation of the pressure states.

Creating the Linear Model MPC Controller

As mentioned briefly in the previous subchapter, the linear model accepts the plant states of the current timestep as an initial condition vector, as well as the road preview and the control move matrix which contains the control moves up until the control horizon. For every discrete step up until the prediction horizon, the state vector is estimated. Then, the values of the controlled variable(s) throughout the prediction horizon are inserted in the objective function, where they are multiplied by the theta vector. Each variable in the objective function is also weighted.

In our application, only the variables of heave acceleration and valve input are controlled. The first is of course the design objective, while the second serves as a mathematical incentive to the controller to minimize control input to the system and thus save energy in a real-life application [11, 32]. More design objectives could be added, such as roll and pitch acceleration, but this would compromise the comparison with the PID Controller and the Passive system. In mathematical form, the Linear Model MPC Controller is:

Model:

$$\hat{\mathbf{Y}}(k+1) = A\cdot\hat{\mathbf{Y}}(k) + B\cdot\mathbf{u}(k) + C_1\cdot\mathbf{w}(k,1) + C_2\cdot\mathbf{w}(k,2) + \ldots + C_R\cdot\mathbf{w}(k,\mathrm{R})$$

$$where\ \ \mathbf{w}(k,r) = \begin{cases} \mathbf{w}_{prev}(k,r) & 1 \le k \le h_{prev} \\ 0 & k > h_{prev} \end{cases} \qquad (5.3)$$

$$R = \frac{ST_{controller}}{ST_{road}}\ , \qquad \mathrm{r} = 1,\ldots,R, \qquad k = 1,\ldots,h_p$$

Objective Function:

$$\min_{u(k),u(k+1),\ldots,u(k+M)} \sum_{j=1}^{hp}\left[\left(\hat{\mathbf{Y}}_{acc}(k+j)\right)\Theta(j)\right]^2 + \sum_{j=1}^{hp}\left[\left(\hat{\mathbf{Y}}_{x\,valve}(k+j)\right)\Theta(j)\right]^2 \qquad (5.4)$$

Subject to

$$-0.004 \le \mathbf{u}(k+j) \le 0.004\ ,\ 1 \le j \le hc \qquad (5.5)$$

$$-0.0015 \le \Delta\mathbf{u}(k+j) \le 0.0015\ ,\ 1 \le j \le hc \qquad (5.6)$$

As mentioned briefly in the subsection that concerns the creation of the LLS Full Car model, there was a difficulty to overcome regarding the ability of the model to handle the road datapoints that were internal to its' discrete time steps. The solution, as shown in eq. (5.3), was to create R C_r matrices which correlate each road datapoint **w**(k,r) with the states of the model. The R is the number of internal datapoints between two discrete time points, $ST_{controller}$ is the sample time of the controller and ST_{road} is the preview sample time of the road.

The objective function consists of the sum of squares of the Prediction-Theta multiplication for the two controlled variables. The objective function is subject to two constraints. The first is a saturation constraint and a physical requirement for the system, since the power valve bottoms out at 4mm displacement. The second is a gradient constraint and ensures that the absolute difference between two consecutive control moves will remain below 1.5mm.

In pseudocode form:

```
function MPC LinearController(theta, weights, ABC Matrices, current_state_vector, road preview info)

W_Input = road_preview_info
U_Input  = matrix of optimal control moves
InternalPoints = controller sample time / timestep = 40
PredictedStates(1) = current_state_vector

for i = 1 to prediction horizon
        for j = 1 to 40
                index = (i-1)*InternalPoints + j;
                roadContribution(j) = Cj*W_Input(index)
        end
        totalRoadContribution = sum(roadContribution)
        PredictedStates(i+1) = A*PredictedStates(i) + B*U_input(i) + totalRoadContribution
end

sumAcceleration = SumOfSquaredValues (PredictedStates_Acceleration *.theta)
sumInput= SumOfSquaredValues (PredictedStates_Input *.theta)

ObjectiveFunctionValue = sumAcceleration * weight(1) + sumInput * weight(2)

end function
```

Tuning the Linear model MPC Controller

As described in the MPC Chapter, there are four items of tuning: prediction horizon hp, control horizon hc, theta vector and objective function weights. These are tuned on a benchmark test of a pulse test for the ‘on-road’ mode, and on a random road test for the ‘off-road’ mode.

For the ‘on-road’ mode, in order to compute the hc and hp parameters, the settling time of the controlled variable after a valve step input is calculated, with a 10% margin (fig.5-8).

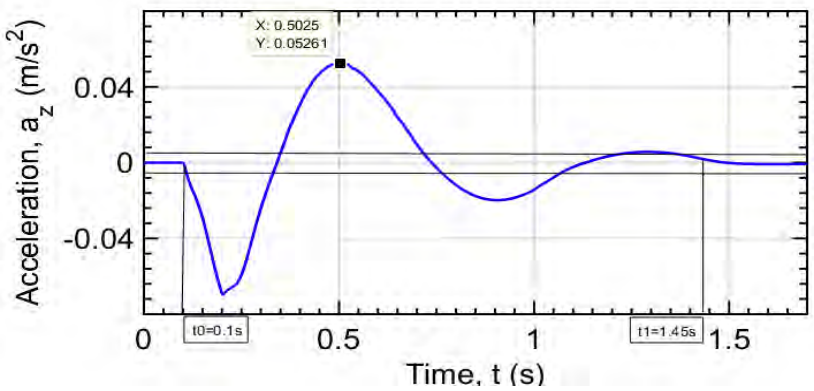


Figure 5-8 Valve step input test of the full car plant

As shown, the settling time is 1.35s for the chassis vertical acceleration. Since the controller sample time is 0.1s, this equals to 14 discrete steps. Thus, the prediction horizon is

$$h_p = h_c + 14 \qquad (5.7)$$

The process of tuning the MPC Controller is described in fig. 3-4. The results of the tuning process are shown in table 5-3:

Table 5-3 Tuning Parameters for the MPC Linear Controller

'On-Road' mode Tuning Parameters

Theta Vector																
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0,021	0,034	0,001	0,028	0,021	0,034	0,037	0,045	0,050	0,048	0,017	0,030	0,014	0,125	0,191	0,234	0,319

Prediction Horizon hp	17
Control Horizon hc	3
Acceleration Weight	100
Xvalve Weight	30000
Preview Length (m)	5

'Off-Road' mode Tuning Parameters

Theta Vector				
1	2	3	4	5
0,098	0,923	0,100	0,327	0,405

Prediction Horizon hp	5
Control Horizon hc	1
Acceleration Weight	10
Xvalve Weight	40000
Preview Length (m)	5

Applying the Linear Model MPC Controller

0.05m right Pulse test:

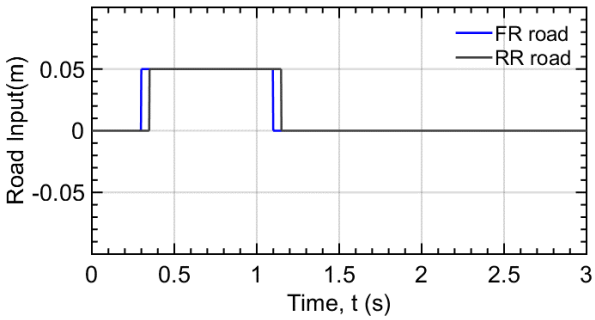
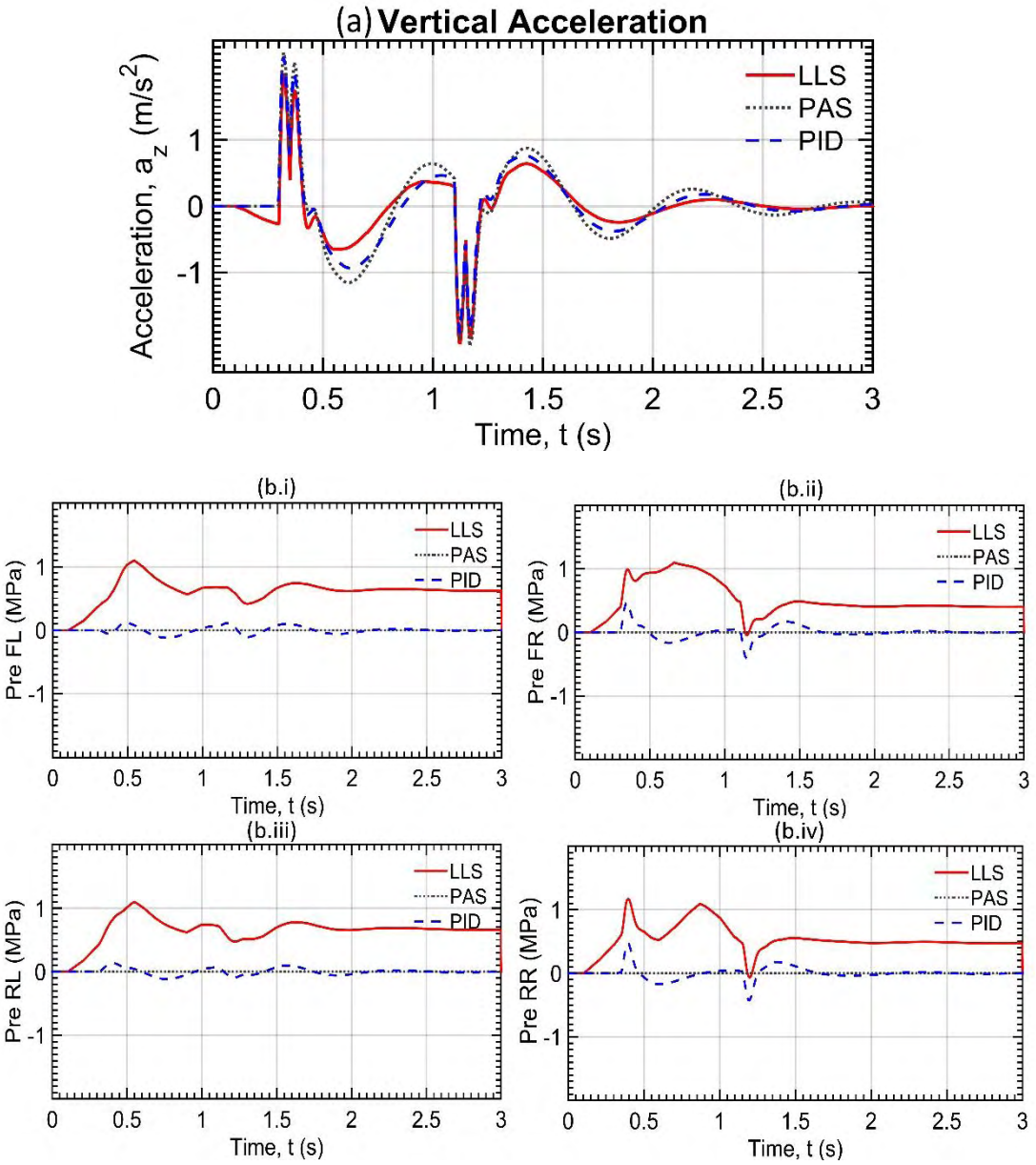


Figure 5-9 Right pulse test road input



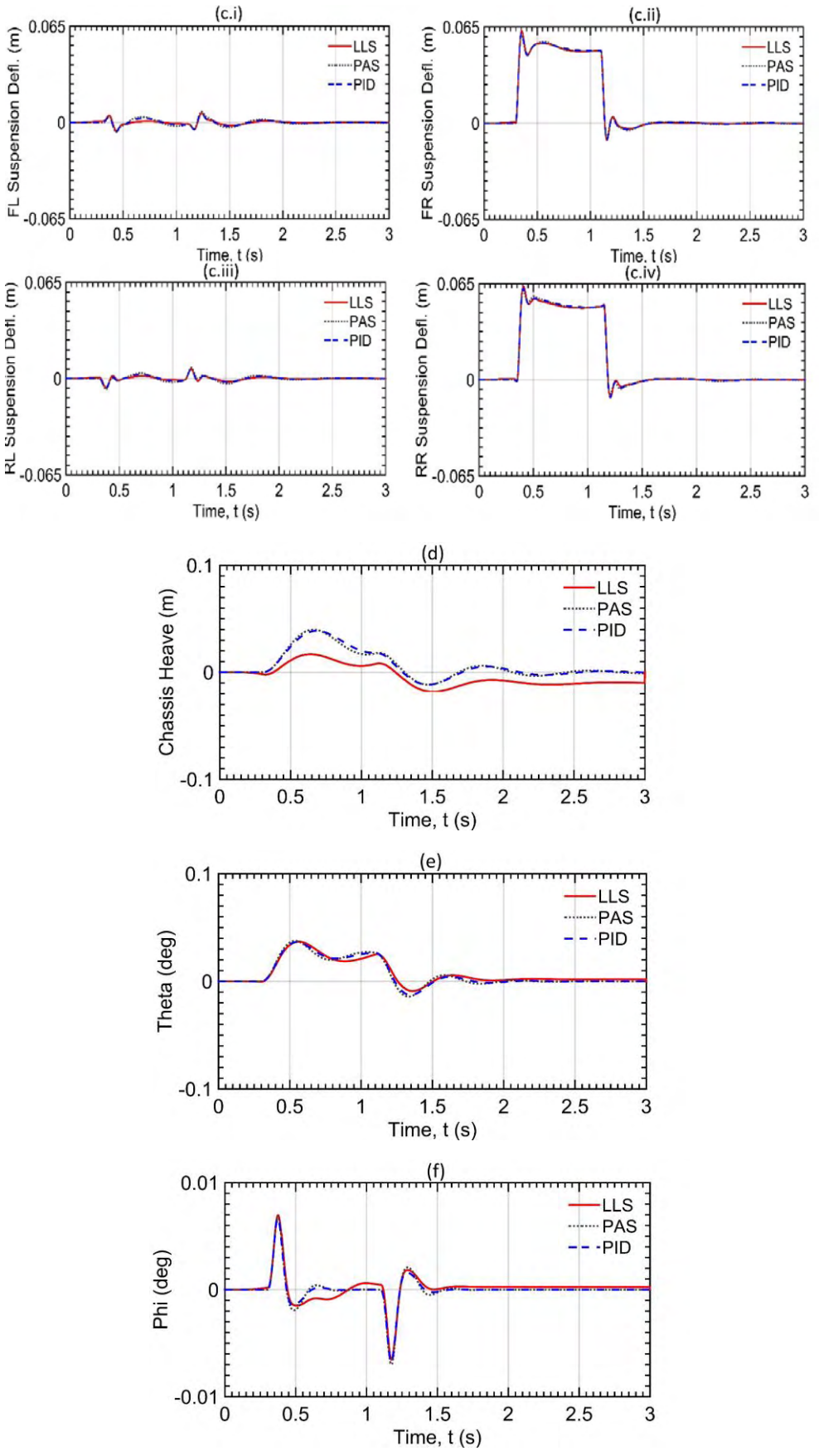
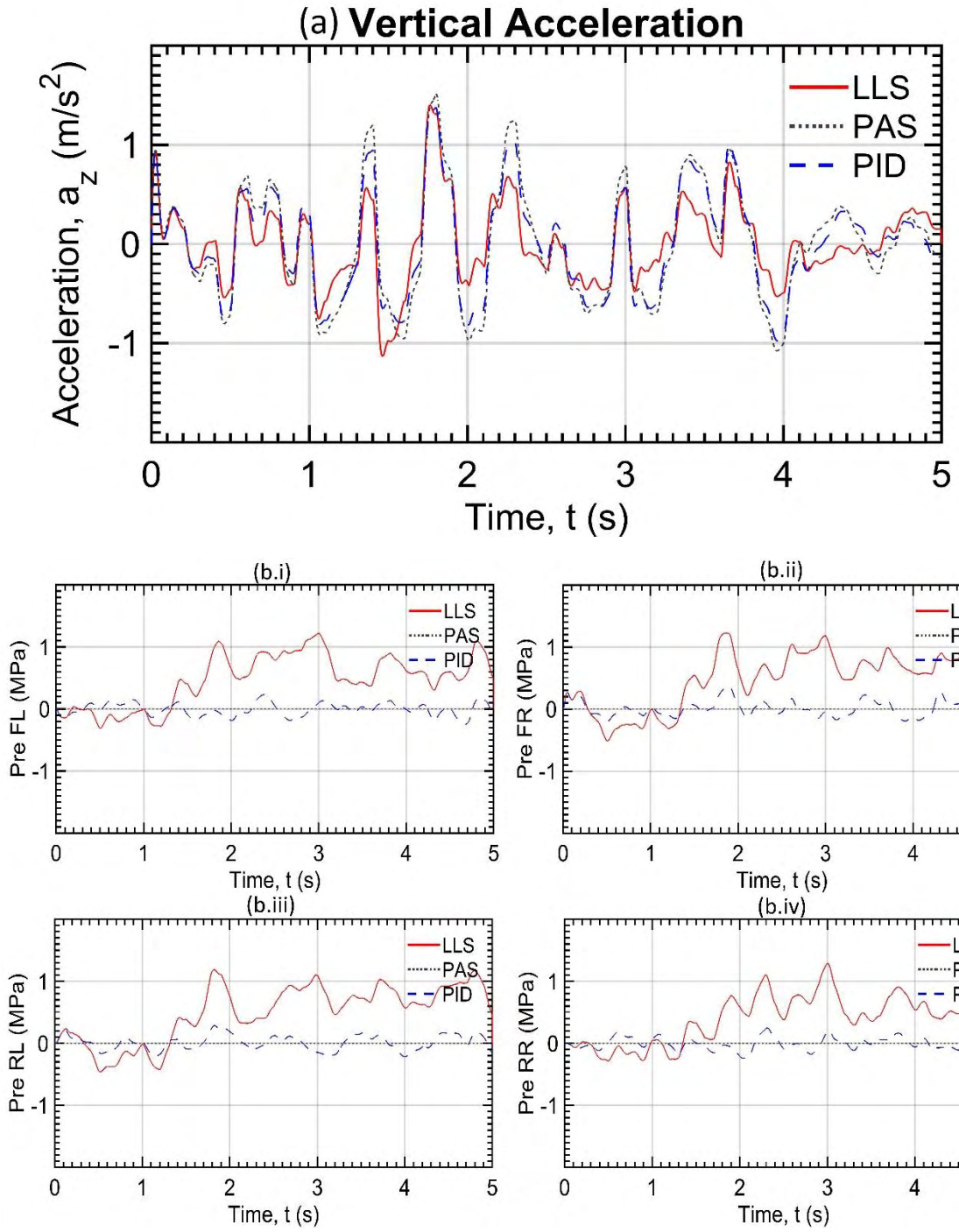


Figure 5-10 Right Pulse Road Results for the MPC Linear Controller, a)Heave acc., b)Pressure, c)Susp. deflection, d)Heave disp., e)Chassis Theta Angle f)Chassis Phi Angle

Random Road Test:



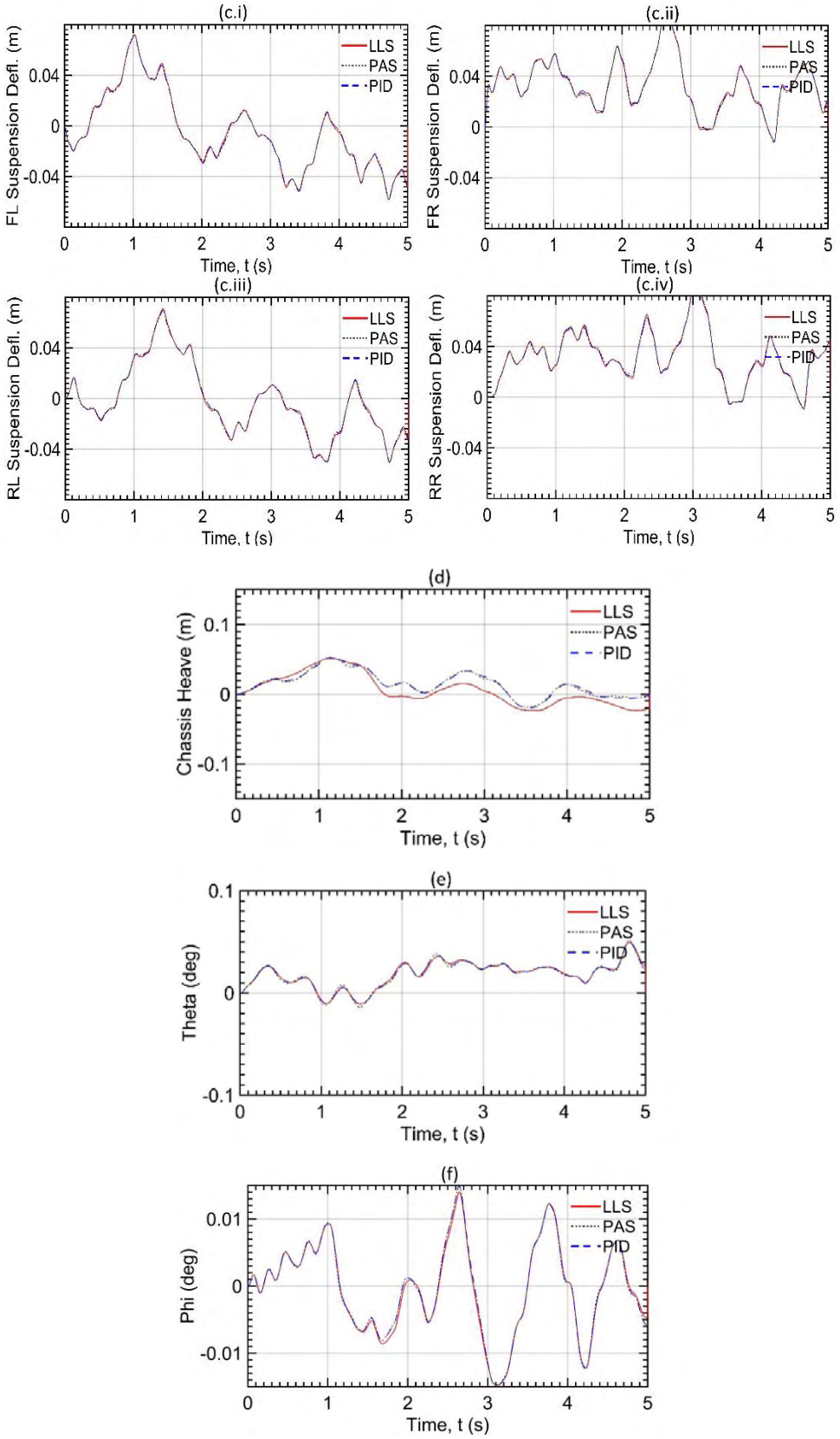


Figure 5-11 Random Road Results for the MPC Linear Controller a)Heave acc., b)Pressure, c)Susp. deflection, d)Heave disp., e)Chassis Theta Angle f)Chassis Phi Angle

5.4 Implementation of MPC Controller with RBFN Model

Before an MPC Controller based on an RBFN Model of the active full car can be implemented, there is a trade-off to be considered; the evaluation of an RBF Neural Network is a computationally intensive process, and it can compromise the ability of the MPC Controller to solve the optimization problem in real time. It makes sense then, to build models only for the highly nonlinear states of the plant, which have been proved to be the states of actuator pressures. These models should have few input variables. Therefore, it was elected that the input data only included the ones contained in the actual actuator pressure equation (see eq. 5.8 - the variables of interest are colored in blue):

$$\dot{x}_{i,5} = a\left[Q_{x_v}\left(x_{i,6}, x_{i,5}\right) - Q_{x_{bv}}\left(x_{i,bv}, x_{i,5}\right) - C_{tm}x_{i,5} - A_p\left(x_{i,3} - x_{i,4}\right)\right] \qquad (5.8)$$

As far as the rest of the states are concerned, they can be predicted by Linear models– in fact the exact same that were described in the previous chapter will be used in order to create a combined LLS-RBFN full car model. From now on in this work, this model will be referred to as the RBFN car model for reasons of brevity.

Creating the RBFN Models for the Actuator Pressure States

The goal is to create four RBFN models, one for each actuator pressure state. In the same way as the creation of a linear model, two types of datasets need to be created: One containing all input to the system, and one containing all output. For comparisons' sake, the same datasets are used for the creation of both linear and RBFN car model formulations. The input data are split into training and validation data in a 70-30 ratio, containing only the states that concern each model.

As described in chapter 4.3, the creation of an RBFN Model corresponds to finding the following parameters:

- Number of RBFs (fuzzy sets)
- Coordinates of RBF centers in the input space
- Synaptic weights for the hidden layer connections.

For a given number of fuzzy sets, the latter two are straightforward: The coordinates of the RBF centers can be calculated after fuzzy partitioning, whereas the synaptic weights can be obtained as a result of linear least squares between the node output and the real output. However, establishing the "optimal" number of fuzzy sets of each RBFN model is a heuristic process. Too little fuzzy sets will fail to capture the data's variation sufficiently, while too many will lead to overfitting over the training data. The idea is to plot the number of fuzzy sets versus the R^2 number, in order to pinpoint the optimal value.

Fig. 5-12 explains the process to find the optimal number of fuzzy sets for the given set of data. The examined metrics are R^2 and RMSE, which are applied both on Training data and on Validation data.

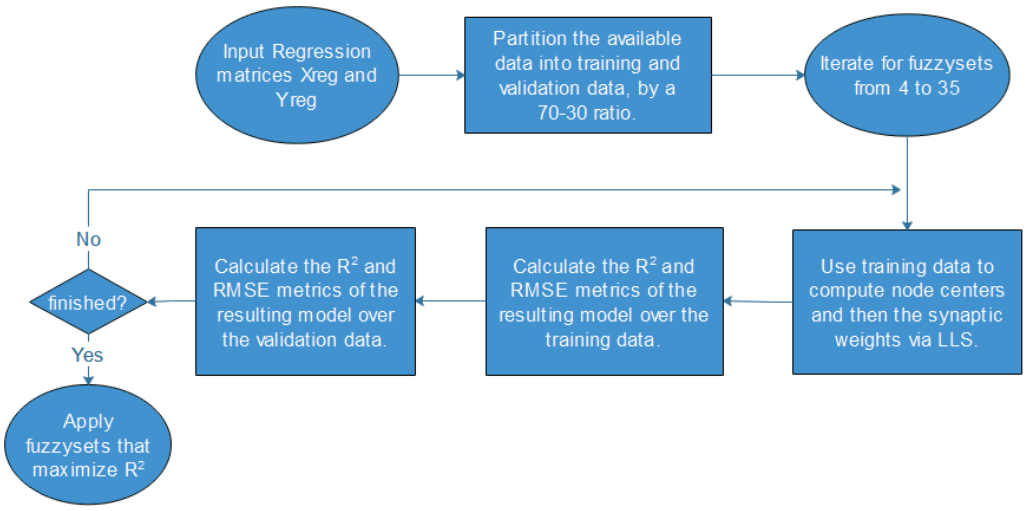


Figure 5-12 Schematic of the algorithm for the computation of the optimum number of fuzzy sets

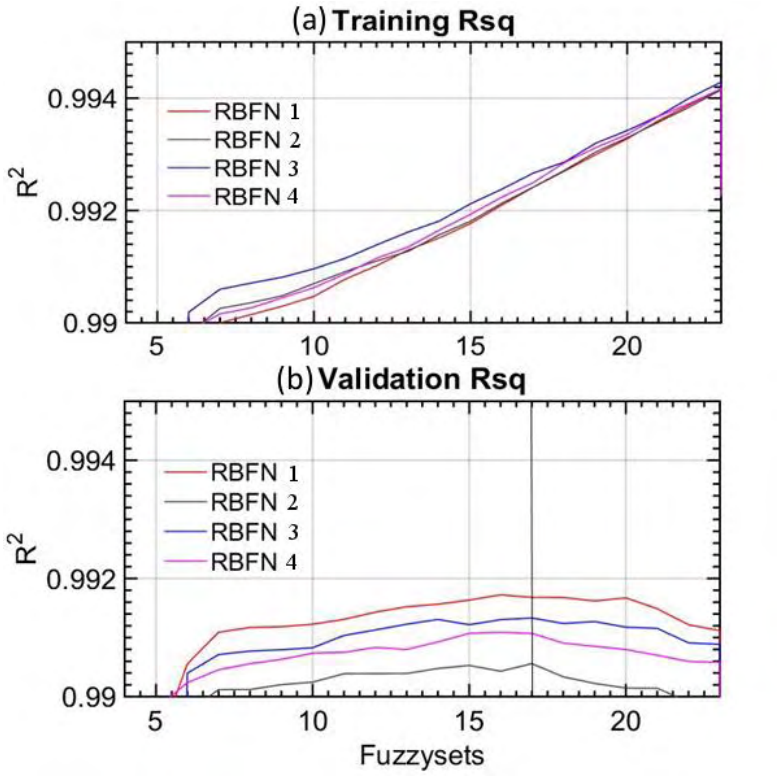


Figure 5-13 Training and Validation R^2 results for the RBFN models

As it is shown in fig. 5-13, the R^2 metric is ever-increasing on a given set of training data for larger numbers of fuzzy sets. However, in the validation data it reaches a plateau of improvement, for reasons that have been mentioned in this chapter. In our application, this plateau occurs at 17 fuzzy sets. Therefore, this is the number of fuzzy sets that will be used for the creation of the models.

Table 5-4 R^2 Comparison table for the RBFN and Linear pressure models

R^2	Pressure FL	Pressure FR	Pressure RL	Pressure RR
RBFN Training	0,9924	0,9924	0,9927	0,9925
RBFN Validation	0,9917	0,9906	0,9913	0,9911
Linear Training	0,9896	0,9890	0,9887	0,9893
Linear Validation	0,9889	0,9889	0,9877	0,9900

A noteworthy consideration when comparing the R^2 values of two models is that they concern a single discrete timestep evaluation. This is why even a small difference can have significant effect on the prediction of the model, since the error propagates through the following timesteps. This is something that can be shown through an initial condition test, which, as mentioned, is an important performance metric for an MPC model. For comparison purposes, both models as well as the full car response are depicted in fig.5-14.

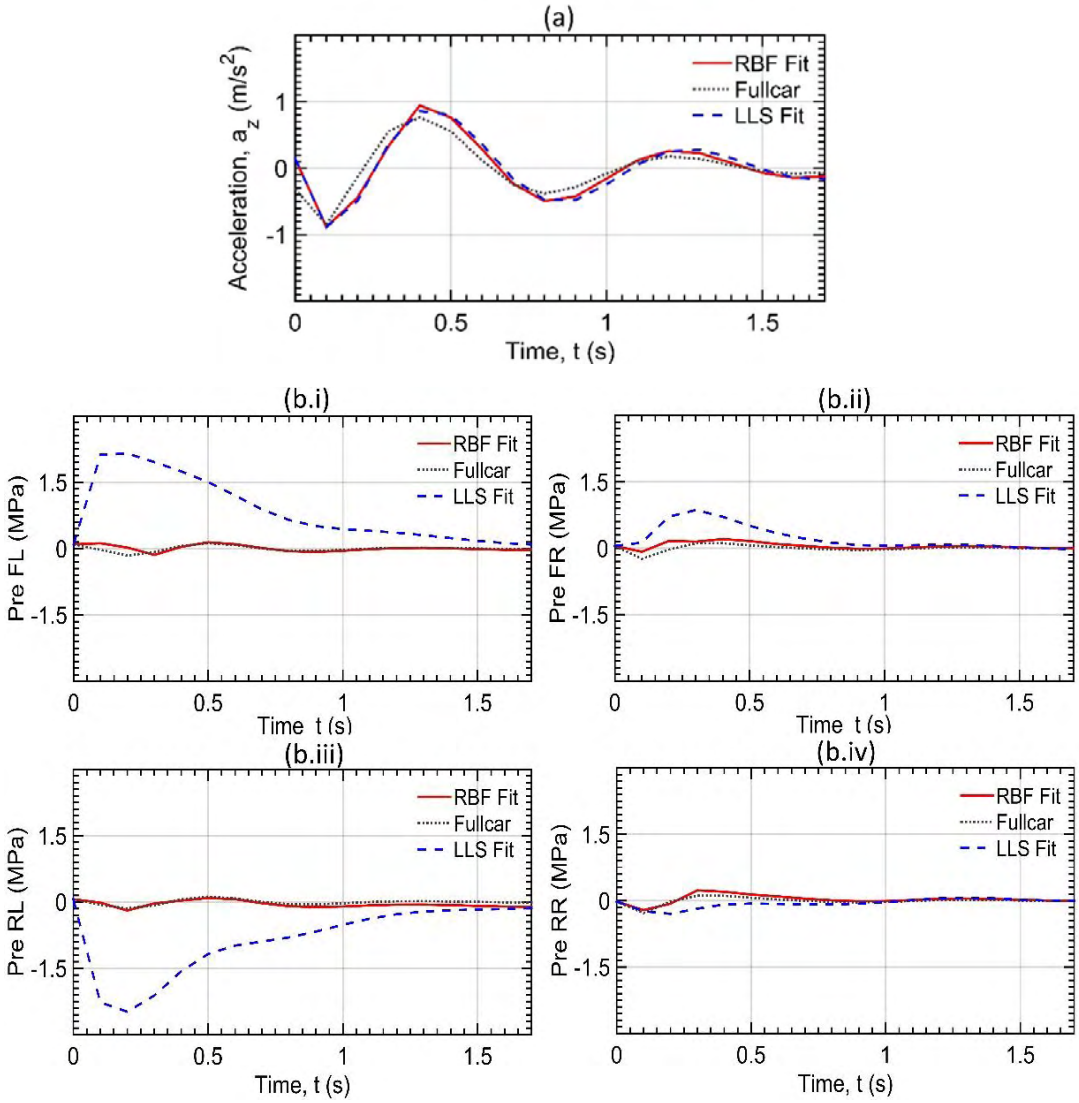


Figure 5-14 Acceleration (a) and Pressure (b) predictions of the RBFN Car model in an initial condition validation test

The RBFN car model successfully predicts the vertical acceleration as well as its' counterpart does - however, there is a significant improvement on the prediction of the pressure states. The result is that the RBFN controller will be able to make more effective control moves than the LLS one.

Creating the RBFN MPC Controller

The controller can be described in mathematical form as:

Model

$$\hat{\mathbf{Y}}(k+1) = A\cdot\hat{\mathbf{Y}}(k) + B\cdot\mathbf{u}(k) + C_1\cdot\mathbf{w}(k,1) + C_2\cdot\mathbf{w}(k,2)\ldots + C_R\cdot\mathbf{w}(k,\mathrm{R})$$

$$\hat{\mathbf{Y}}_{i,5}(k+1) = RBFN\left(\hat{\mathbf{Y}}_{i,3}(k), \hat{\mathbf{Y}}_{i,4}(k), \hat{\mathbf{Y}}_{i,5}(k), \hat{\mathbf{Y}}_{i,6}(k)\right) \tag{5.9}$$

$$where\ \ \mathbf{w}(k,r) = \begin{cases} \mathbf{w}_{prev}(k,r) & 1 \le k \le h_{prev} \\ 0 & k > h_{prev} \end{cases}$$

$$R = \frac{ST_{controller}}{ST_{road}}\ , \qquad \mathrm{r} = 1,\ldots,R\,, \qquad k = 1,\ldots,h_p$$

Objective Function

$$\min_{u(k),u(k+1),\ldots,u(k+M)} \sum_{j=1}^{hp}\left[\left(\hat{\mathbf{Y}}_{acc}(k+j)\right)\Theta(j)\right]^2 + \sum_{j=1}^{hp}\left[\left(\hat{\mathbf{Y}}_{x\,valve}(k+j)\right)\Theta(j)\right]^2 \tag{5.10}$$

Subject to

$$-0.004 \le \mathbf{u}(k+j) \le 0.004\ ,\ 1 \le j \le hc \tag{5.11}$$

$$-0.0015 \le \Delta\mathbf{u}(k+j) \le 0.0015\ ,\ 1 \le j \le hc \tag{5.12}$$

The difference from the Linear model is that now the predictions for the $x_{i,5}$ state (the actuator pressure for every strut) come from the RBFN models. Note that the RBFN models accept only the states that they have been trained on, which are the absolute velocities of strut and wheel, the actuator pressure and the valve position. Of course, the same constraints as the LLS formulation apply here.

In pseudocode form:

```
function MPC RBFN-LLS Controller(theta, objFunweights, ABC Matrices, RBFNcenters, synaptic weights, current state vector, road preview info)

W_Input = road preview info
U_Input  = matrix of optimal control moves
InternalPoints = controller sample time / timestep = 40
PredictedStates(1) = current_state_vector

for i = 1 to prediction horizon
        for j = 1 to 40
                index = (i-1)*InternalPoints + j;
                roadContribution(j) = Cj*W_Input(index)
        end
        totalRoadContribution = sum(roadContribution)
        PredictedStates(i+1) = A*PredictedStates(i) + B*U_input(i) + totalRoadContribution
        for k = 1 to 4
                PredictedStates_k,5_(i+1) = RBFN(PredictedStates(i),RBFNcenters,SynapticWeights)
        end

end

sumAcceleration = SumOfSquaredValues (PredictedStates_Acceleration *.theta)
sumInput= SumOfSquaredValues (PredictedStates_Input *.theta)
ObjectiveFunctionValue = sumAcceleration * objFunweights (1) + sumInput * objFunweights (2)

end function
```

Tuning the RBFN Model MPC Controller

Exactly the same procedure has been followed as outlined in fig. 3-4 for the tuning of the RBFN car model. The final tuning parameters for the RBFN MPC Controller are shown in table 5-4:

'On-Road' mode Tuning Parameters

Theta Vector																
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0,021	0,034	0,001	0,028	0,021	0,034	0,037	0,045	0,050	0,048	0,017	0,030	0,014	0,125	0,191	0,234	0,319

Prediction Horizon hp	17
Control Horizon hc	3
Acceleration Weight	250
Xvalve Weight	30000
Preview Length (m)	5

'Off-Road' mode Tuning Parameters

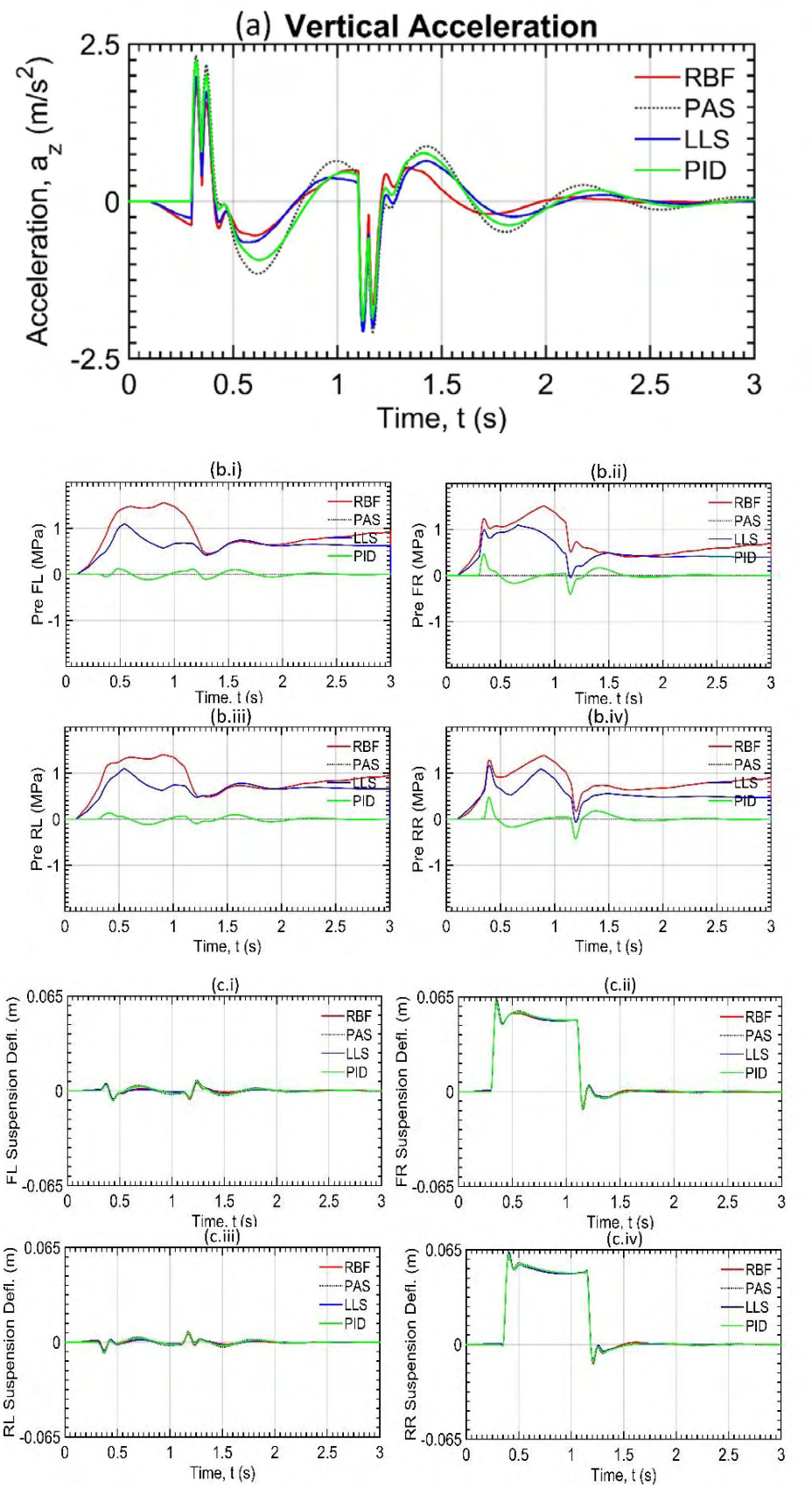
Theta Vector				
1	2	3	4	5
0,110	0,939	0,360	0,425	0,885

Prediction Horizon hp	5
Control Horizon hc	1
Acceleration Weight	10
Xvalve Weight	40000
Preview Length (m)	5

Table 5-5 Tuning parameters for the RBFN car model

Applying the RBFN Model MPC Controller

0.05m right Pulse test:



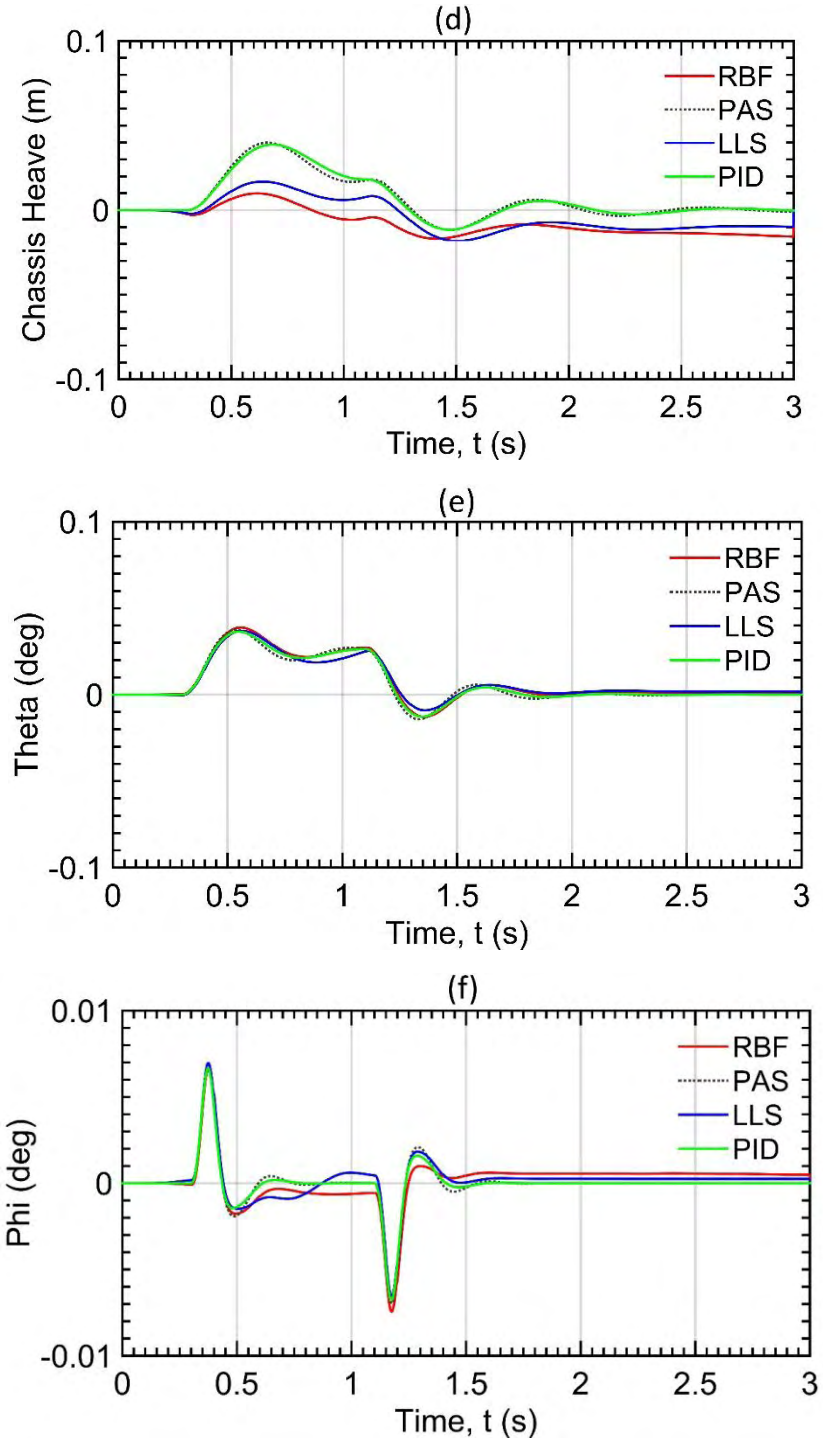
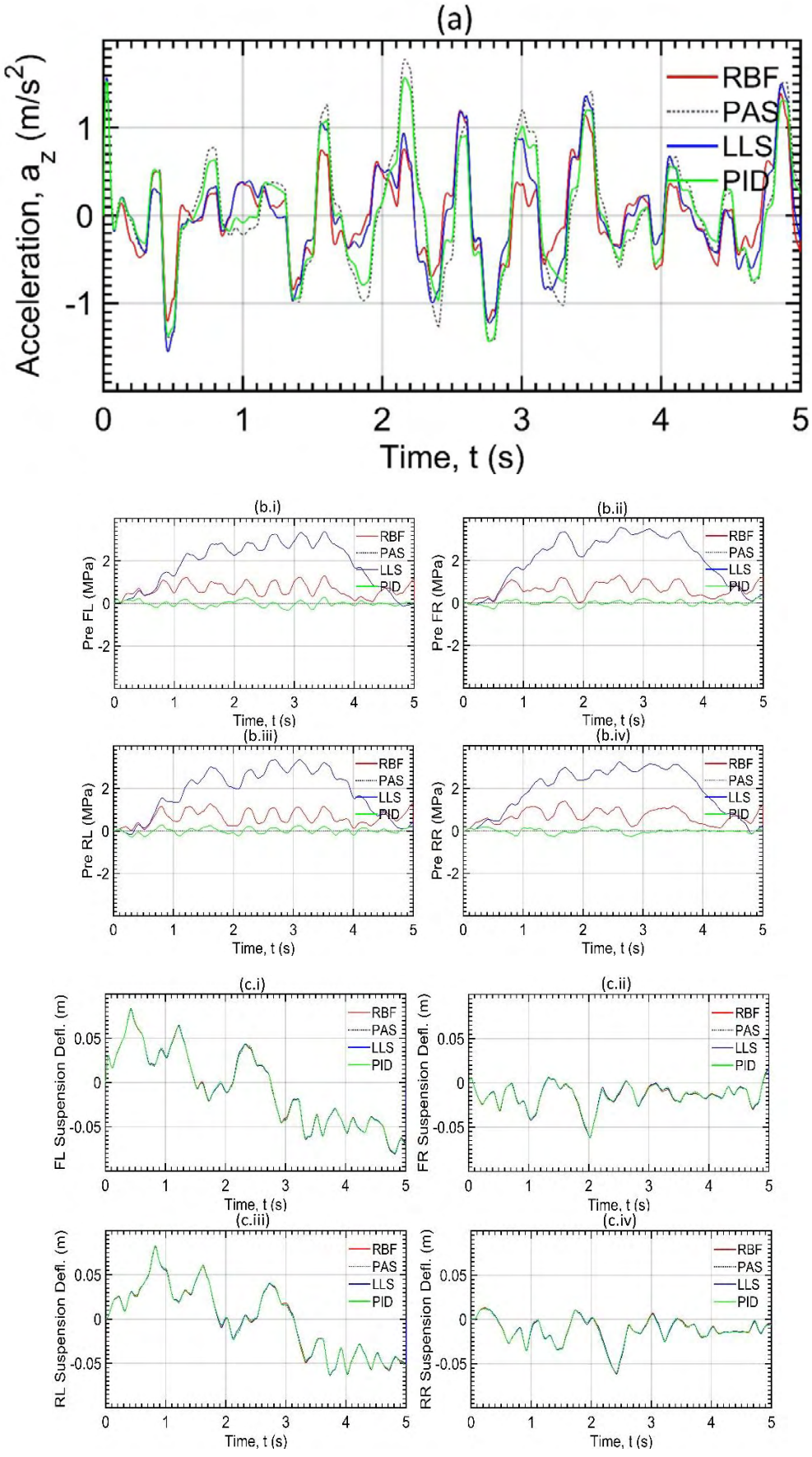


Figure 5-15 Right Pulse Road Results for the MPC RBF Controller, a)Heave acc., b)Pressure, c)Susp. deflection, d)Heave disp., e)Chassis Theta Angle f)Chassis Phi Angle

Random Road Test:



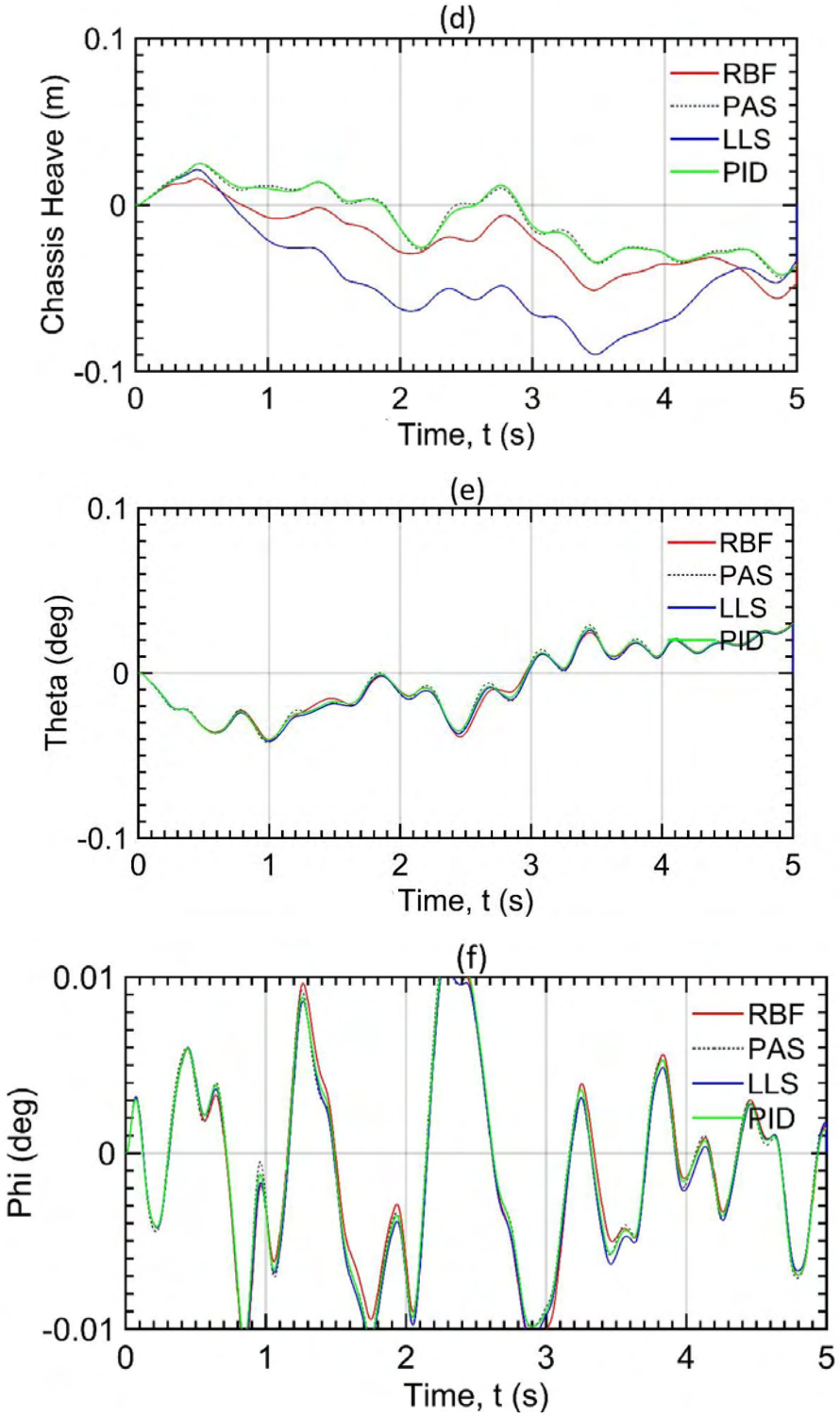


Figure 5-16 Random Road Results for the MPC RBF Controller a)Heave acc., b)Pressure, c)Susp. deflection, d)Heave disp., e)Chassis Theta Angle f)Chassis Phi Angle

A useful plot in the evaluation of the Random Road benchmark test is the FFT of the vertical acceleration of the Chassis (fig. 5-17). Thus, the performance of the controller in mitigating vibrations of different frequencies can be evaluated.

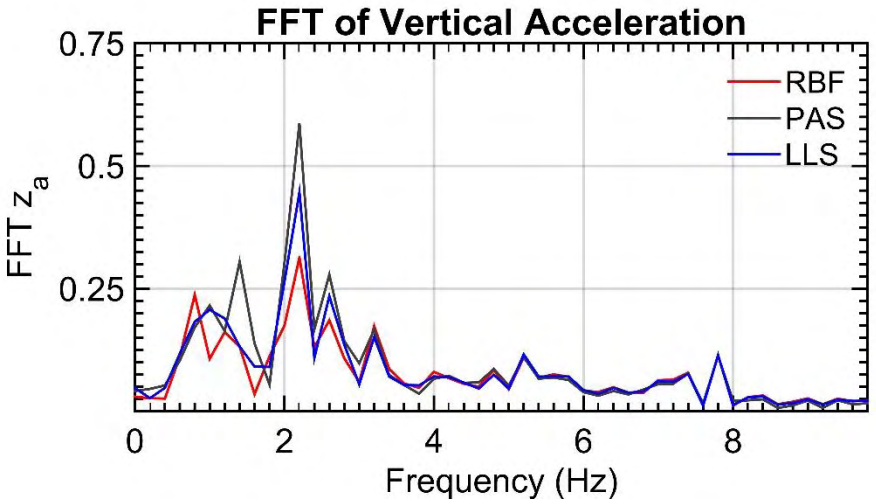


Figure 5-17 FFT of Heave acc. over a random road

The MPC-RBFN Controller achieves a significant reduction of the FFT value of the acceleration at the 2.2 Hz eigenfrequency of the chassis. It is also reduced for almost every frequency.

Table 5-6 Summary table for the Pulse test

	Pulse Test			
	Peak (m/s^2)	**5% settling time (s)**	**SAE**	**RMS**
Passive Full Car	2.316	2.61	464.02	0.5961
PID Controller	2.247	2.34	390.31	0.5204
LLS-MPC Controller	1.991	2.01	325.86	0.4593
RBFN-MPC Controller	1.850	1.88	278.42	0.3972

Table 5-7 Summary table for the Random Road test

	Random Road Test		
	FFT Peak	**SAE**	**RMS**
Passive Full Car	0.5868	1090.01	0.6869
PID Controller	0.5004	942.38	0.6043
LLS-MPC Controller	0.4462	885.39	0.5755
RBFN-MPC Controller	0.3134	774.67	0.4944

5.5 Discussion

Design Objective

Considering the design objective of the reduction of the vertical acceleration of the chassis, from tables 5-5 and 5-6 as well as fig.5-10,11,15,16 it is apparent that the MPC-P Formulation, whether it is implemented with a Linear model Controller or an RBFN model Controller, is better than the PID Controller. The results of the two benchmark tests are discussed below:

1. **In the case of a pulse road** of 0.05m the LLS Controller achieves a 30% reduction of the SAE value of the controlled variable. It reduces significantly its' maximum value by 14% and its' settling time by 22%.
   In comparison, the RBF Controller achieves a 40% reduction of the SAE value of the controlled variable. Moreover, it succeeds in reducing the maximum value by 20%, and secures a shorter settling time by 28%.

2. **In the case of a random road** the MPC-P Controller outperforms the PID Controller in general. In particular, the LLS Controller achieves a 23% reduction of the FFT Peak Value of the controlled variable, whereas the RBF ones does so with a 46% reduction. The RMS Value over the 5s evaluation of the random road is reduced by 16% and 28% by the LLS and RBF Controller respectively.

The reason that the MPC-P scheme is better than the traditional PID one in general, is because it can execute more effective, prediction-based control moves. On the other hand, the PID controller in our application acts on an approximate basis, since each actuator aims to reduce the respective vertical acceleration which is measured at each corner of the vehicle. Another consideration is that the MPC-P scheme accepts road preview information, which is a clear advantage over the PID one.

As far as the LLS and RBF controllers are concerned, the superior performance of the latter is based on the better approximation capabilities of its respective model. This in turn is owed to the implementation of RBFN models for the nonlinear pressure states. As a result, the RBF controller can execute more effective control moves and thus achieve better performance.

Other Considerations

As far as the other chassis accelerations are concerned, no differences are observed between the different formulations. This is because they are not present in the objective function of the MPC Controller. They could very well be incorporated in future work, as well as other vehicle states such as the actuator pressures or suspension deflections. The last two could be used in the formulation of soft constraints for the vehicle, as it is done in [11].

One more metric that should be evaluated in an active suspension is its' energy expenditure. The more control moves are applied (in number and in "aggressiveness"), the more power is required by the hydraulic pump in order to maintain the supply pressure. The required hydraulic power can be computed by evaluating $\dot{W} = \Delta P \cdot Q$, which is the actuator pressure multiplied by the hydraulic flow per second [32]. As shown in fig 5-18, the RBF Controller is much more energy efficient than the LLS Controller, and this can be attributed to the better actuator pressure estimation of the RBFN models.

A final consideration for the MPC Controller is that an off-road application would benefit from smaller controller sample times, since it can react quicker to the inputs from the road. A preview formulation with a longer preview horizon would also improve the effectiveness of the control moves.

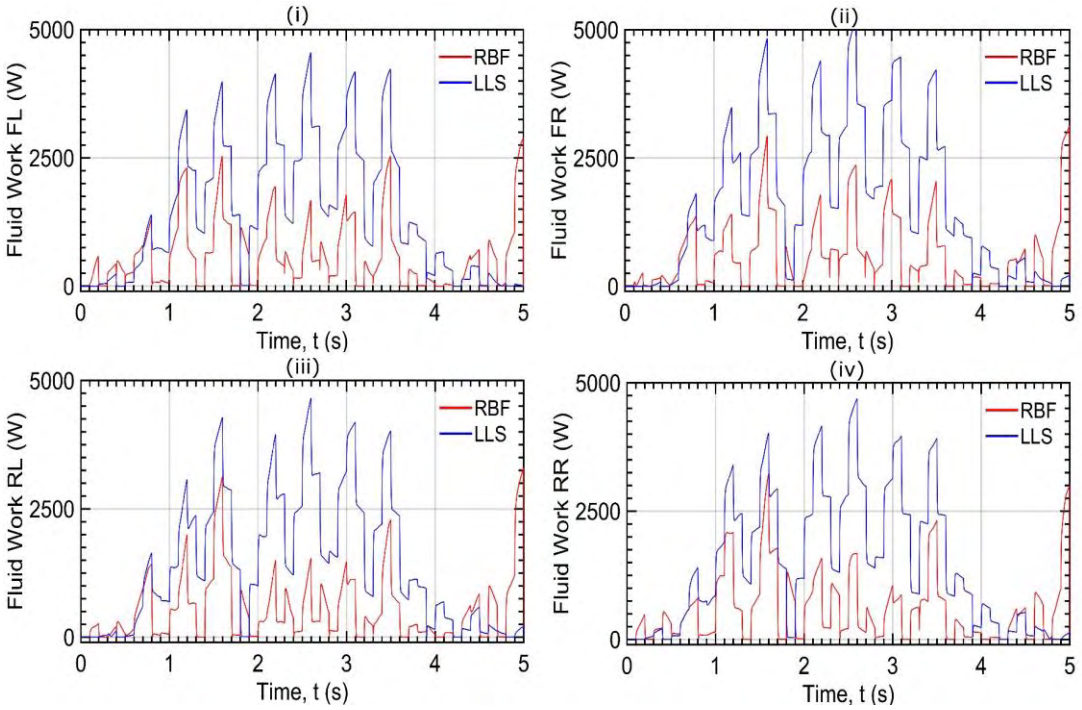


Figure 5-18 Hydraulic Fluid Power over a random road test for each actuator

6 Conclusions and Future Work

Conclusions

In this work, a new method for the control of the full car active suspension was developed. An MPC Road Preview strategy was applied, and a Linear full car model with RBFN models for the highly non-linear actuator pressure states was used within. The results were very promising, achieving significant reduction of the chassis' vertical vibrations, both in 'on-road' and 'off-road' modes (tables 5-5, 5-6).

This new strategy was compared with a PID controller and an MPC-P Linear controller. The MPC-RBFN Controller was able to outperform them, because of its' successful prediction of the nonlinear states through the RBFN models. In addition, it did so with a lower energy expenditure (fig. 5-18).

Future Work

Future work should examine the more practical aspects of the implementation of an MPC-P RBFN Controller in a full car. Notably:

- The determination of the computational requirements of the evaluation of the LLS-RBFN car model by the on-board vehicle computer.
- Applying a more computationally efficient MPC scheme through the integration of inverse neural models.[33]
- Creating a robust Controller that can handle parameter variation in the system [18].
- The implementation of soft constraints in the MPC Controller such as maximum allowable suspension travel and actuator pressure [11].

7 List of Variables and Symbols

Symbol	Meaning	Value	Units
	Valve & Hydraulics		
P_s	Supply Pressure to Valve	20,683 e6	Pa
A_p	Actuator Piston Area	0.0044	m^2
P_L	Pressure Difference in Actuator	Variable	Pa
Q_{xv}	Hydraulic Fluid Flow through valve	Variable	m^3/s
Q_{xbv}	Hydraulic Fluid Flow through bypass valve	Variable	m^3/s
C_{tm}	Leakage constant through piston rings	15 e-12	-
α	Hydraulic Coefficient	2.27e8	N/m^5
ρ	Density of Hydraulic Fluid	850	kg/m^3
τ	Time constant of valve	0.003	s
C_d	Hydraulic Discharge Coefficient	0.7	-
S_{xv}	Valve width	0.008	m
x_v	Valve displacement	Variable	m
x_{bv}	Bypass valve displacement	Variable	m
	Car Model		
$x_{i,1}$	Absolute displacement of i strut	Variable	m
$x_{i,2}$	Absolute displacement of i wheel	Variable	m
$x_{i,3}$	Absolute velocity of i strut	Variable	m/s
$x_{i,4}$	Absolute velocity of i wheel	Variable	m/s
$x_{i,5}$	Pressure Difference in i Actuator	Variable	Pa
$x_{i,6}$	i Valve displacement	Variable	m
$x_{5,1}$	Angular Roll Velocity of the Chassis	Variable	rad/s
$x_{5,2}$	Angular Pitch Velocity of the Chassis	Variable	rad/s
$x_{5,3}$	Absolute Vertical Velocity of the Chassis	Variable	m/s
$x_{5,4}$	Roll angle of the Chassis	Variable	rad
$x_{5,5}$	Pitch angle of the Chassis	Variable	rad
$x_{5,6}$	Vertical Displacement of the Chassis	Variable	m
F_{ks}	Force of suspension spring	Function	N
F_{bs}	Force of suspension damper	Function	N
F_{kt}	Force of tire spring	Function	N
F_{strut}	Sum of suspension forces acting on a strut	Function	N
u_i	Input to i valve	Variable	m
w_i	Road profile of i wheel	Variable	m
b_s^{lin}	Linear suspension damper coefficient	12 e3	Ns/m
b_s^{sym}	Non-symmetric suspension damper coefficient	400	Ns/m
$b_s^{nonlinear}$	Non-Linear suspension damper coefficient	400	Ns/m
k_s^{lin}	Linear suspension spring coefficient	240 e3	N/m
$k_s^{nonlinear}$	Non-Linear suspension spring coefficient	235 e4	N/m
k_t	Tire spring coefficient	100 e4	N/m
b_{track}	Track width (front and rear)	2	m
L_{base}	Wheelbase	4	m
I_{xx}	X moment of inertia	5000	$kg\ m^2$
I_{yy}	Y moment of inertia	8000	$kg\ m^3$
M_{spr}	Quarter sprung mass	2800	kg
M_{unsp}	Unsprung mass	270	kg

8 List of Acronyms

Acronym / Symbol	Meaning
MPC	Model Predictive Control
MPC-P	Model Predictive Control with Preview Information
PID	Proportional - Integral - Derivative Controller
h_c	Control Horizon
h_p	Prediction Horizon
h_{prev}	Preview Horizon
RBFN	Radial Basis Function Network
FM	Fuzzy Means
SISO	Single input - Single output
MIMO	Multiple Input - Multiple output
LLS	Linear Least Squares
Xreg	Matrix containing all inputs for the creation of the LLS model
Yreg	Matrix containing all outputs for the creation of the LLS model
RMSE	Root Mean Square Error
FFT	Fast Fourier Transform
SAE	Sum of Absolute Error

9 Appendix I: Solvers & other Numerical Considerations

ODE45 Solver

The solver that was used for the simulation is the *ODE45 MATLAB* solver, which is based on an algorithm of Dormand and Prince. It applies six steps and provides fourth and fifth order formulas. For more details on the algorithm see [34]. The settings that are applied are:

Table 9-1 ODE45 Settings

Relative Tolerance	1 e-8
Absolute Tolerance	1 e-10
Maximum Step	0.005

Fmincon Optimization Function

The optimization solver that was used in the context of the MPC Optimization was *fmincon*, which contains a list of algorithms to choose from. In this work an interior-point method is used which can handle constrained nonlinear optimization. The problems that the solver is applied to are (5-3,6), which is constrained linear quadratic, and (5-9,12) which is constrained nonlinear quadratic.

The following settings were applied while keeping in mind that the optimization of the above problems should be computationally effective and within practical limits.

Table 9-2 fmincon settings

Maximum Evaluations	100
Function Tolerance	1 e-5
X Tolerance	3 e-5
Minimum Change in variables	5 e-4

Linear Least Squares ‘ \ ’ operator

The linear Least Squares computation was performed using the \ operator on the regression matrices *Xreg* and *Yreg*. The condition number of the *Xreg* matrix was aimed to be kept within reasonable levels. In this respect, the scaling of the variables was due, because large order differences between variables can lead to numerical problems in inversion [30]. The difference was because of the units of the pressure states, which were in Pa; this lead to an order difference of O(1e6) in the *Xreg* matrix. The Pressure states were scaled up to MPa, thus eliminating the order difference and reducing the condition number of the matrix from e22 to e14.

10 Literature

1. Douglas F. Milliken, W.F.M.J., *Race Car Vehicle Dynamics.* SAE International, 1995.
2. Yagiz, N. and Y. Hacioglu, *Backstepping control of a vehicle with active suspensions.* Control Engineering Practice, 2008. 16(12): p. 1457-1467.
3. Zapateiro, M., et al., *Semiactive Control Methodologies for Suspension Control With Magnetorheological Dampers.* IEEE/ASME Transactions on Mechatronics, 2012. 17(2): p. 370-380.
4. R. Ben Mrad, J.A.L., S.D. Fassois, *Non-linear Dynamic Modeling of an Automobile Hydraulic Active Suspension System.* 1993.
5. Christoph Gohrle, A.W., Andreas Schindler, Oliver Sawodny, *Active Suspension Controller using MPC based on a full-car model with preview information.* 2012 American Control Conference, 2012.
6. Pan, H., et al., *Adaptive tracking control for active suspension systems with non-ideal actuators.* Journal of Sound and Vibration, 2017. 399: p. 2-20.
7. P. Gaspar, I.S., J. Bokor, *Active suspension design using linear parameter varying control.* International Journal of Vehicle Autonomous Systems, 2003. Vol. 1(2).
8. Kumarawadu, S. and T.T. Lee, *Neuroadaptive Combined Lateral and Longitudinal Control of Highway Vehicles Using RBF Networks.* IEEE Transactions on Intelligent Transportation Systems, 2006. **7**(4): p. 500-512.
9. Schindler, A., *Neue Konzeption und erstmalige Realisierung eines aktiven Fahrwerks mit Preview-Strategie.* KIT Scientific Publishing, 2009.
10. C., D.J., *Tires, Suspension and Handling.* SAE International, 1996.
11. M. D. Donahue, J.K.H., *Implementation of an Active Suspension, Preview Controller for Improved Ride Comfort.* Nonlinear and Hybrid Systems in Automotive Control, 2003. XVIII: p. 446.
12. Shoukry, Y., M. Watheq El-Kharashi, and S. Hammad, *An embedded implementation of the Generalized Predictive Control algorithm applied to automotive active suspension systems.* Computers & Electrical Engineering, 2013. 39(2): p. 512-529.
13. Deshpande, J., T. Dey, and P.C. Ghosh, *Effect of Vibrations on Performance of Polymer Electrolyte Membrane Fuel Cells.* Energy Procedia, 2014. 54: p. 756-762.
14. Hrovat, D., *Survey of Advanced Suspension Developments and Related Optimal Control Applications.* Automatica, 1997. 33(10): p. 1781-1817.
15. Cao, J., P. Li, and H. Liu, *An Interval Fuzzy Controller for Vehicle Active Suspension Systems.* IEEE Transactions on Intelligent Transportation Systems, 2010. 11(4): p. 885-895.
16. Du, H., N. Zhang, and J. Lam, *Parameter-dependent input-delayed control of uncertain vehicle suspensions.* Journal of Sound and Vibration, 2008. 317(3-5): p. 537-556.
17. Gopalasamy S., O.C., Hedrick K., Rajamani R., *Model Predictive Control for active suspensions - controller design and experimental study.* ASME Dynamic Systems and Control Division, 1997. 61: p. 725-733.
18. A. Aldair, A. and W. Wang, *FPGA Based Adaptive Neuro Fuzzy Inference Controller for Full Vehicle Nonlinear Active Suspension Systems.* International Journal of Artificial Intelligence & Applications, 2010. 1(4): p. 1-15.
19. Alexandridis, A., Chondrodima E. , Sarimveis H., *Radial Basis Function Network Training Using a Nonsymmetric Partition of the Input Space and Particle Swarm Optimization.* IEEEE Transactions on Neural Networks and Learning Systems. Vol. 24.
20. Merritt, H.E., *Hydraulic Control Systems.* John Wiley & Sons, 1967.
21. Verros, G., S. Natsiavas, and C. Papadimitriou, *Design Optimization of Quarter-car Models with Passive and Semi-active Suspensions under Random Road Excitation.* Modal Analysis, 2005. 11(5): p. 581-606.

22. Alexandridis, A., H. Sarimveis, and G. Bafas, *Modeling and Control of Continuous Digesters Using the Pls Methodology.* Chemical Engineering Communications, 2010. 191(10): p. 1271-1284.
23. Feng Tyan, Y.-F.H., Shun-Hsu, Wes S. Jeng, *Generation of random road profiles.*
24. Michelli, C.A., *Interpolation of scattered data: Distance matrices and conditionally positive definite functions.* Constrained Approximation, 1986. 2(11-22).
25. T. Poggio, F.G., *Networks for approximation and learning.* Proceedings IEEE, 1990. **78**(9): p. 1481.
26. C. Darken, J.M., *Fast adaptive K-means clustering: Some empirical results.* Proceedings IEEE International Joint Conference on Neural Networks, 1990: p. 233-238.
27. H. Sarimveis, A.A., G. Tsekouras, G. Bafas, *A fast and efficient algorithm for training radial basis function neural networks based on a fuzzy partition of the input space.* Industrial & Engineering Chemical Research, 2002. 41(4): p. 751-759.
28. A. Alexandridis, H.S., *Nonlinear adaptive model predictive control based on self-correcting neural network models.* AIChE J., 2005. 51(9): p. 2495-2506.
29. K. Ninos, C.G., I. Kompogiannis, I. Stavrakas, A. Alexandridis, *Nonlinear control of a DC-motor based on radial basis function neural networks.* Proc. IEEE Int. Symp. Innov. Intell. Syst. Appl., 2011: p. 611-615.
30. Steven C. Chapra, R.P.C., *Numerical Methods for Engineers.* McGraw-Hill International Edition, 2010. 6th edition.
31. Agostinacchio, M., D. Ciampa, and S. Olita, *The vibrations induced by surface irregularities in road pavements – a Matlab® approach.* European Transport Research Review, 2013. 6(3): p. 267-275.
32. Pilbeam, C. and R.S. Sharp, *Performance Potential and Power Consumption of Slow-Active Suspension Systems with Preview.* Vehicle System Dynamics, 1996. 25(3): p. 169-183.
33. Stogiannos, M., A. Alexandridis, and H. Sarimveis, *Model predictive control for systems with fast dynamics using inverse neural models.* ISA Trans, 2018. 72: p. 161-177.
34. J.R Dormand, P.J.P., *A family of Embedded Runge-Kutta Formulae.* Journal of Computational and Applied Mathematics, 1980. 6(1).