MARMARA UNIVERSITY
FACULTY OF ENGINEERING

OPTIMAL CONTROLLER DESIGN FOR A CAR ACTIVE SUSPENSION SYSTEM

MUHAMMED FURKAN KAHYAOĞLU, ENES SOYGÜLLÜCÜ

GRADUATION PROJECT REPORT
Department of Mechanical Engineering

Supervisor
Prof. Dr. Mustafa ÖZDEMİR
ISTANBUL, 2025

MARMARA UNIVERSITY
FACULTY OF ENGINEERING

OPTIMAL CONTROLLER DESIGN FOR A CAR ACTIVE SUSPENSION SYSTEM

by

Muhammed Furkan Kahyaoğlu, Enes Soygüllücü

June 30, 2025, İstanbul

SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE

OF

BACHELOR OF SCIENCE

AT

MARMARA UNIVERSITY

Signature of Author(s) ..... Muhammed Furkan KAHYAOĞLU, Enes SOYGÜLLÜCÜ

Department of Mechanical Engineering

Certified By ..... Prof. Dr. Mustafa ÖZDEMİR

Project Supervisor, Department of Mechanical Engineering

Accepted By .....Prof. Dr. Bülent EKİCİ

Head of the Department of Mechanical Engineering

ACKNOWLEDGEMENT

First of all, we would like to thank our supervisor Prof. Dr. Mustafa Özdemir, for the valuable guidance and advice on preparing this thesis and giving us moral and material support.

June, 2025

Muhammed Furkan KAHYAOĞLU
Enes SOYGÜLLÜCÜ

CONTENTS

ABSTRACT

Optimal Controller Design For a Car Active Suspension System

In this study, an optimal controller design for an active suspension system of a quarter-car model is presented using the Linear-Quadratic Regulator (LQR) approach. The primary goal is to enhance ride comfort and vehicle stability by minimizing body vibrations and limiting suspension travel and tire deflection under various road disturbances. The active suspension model is developed in MATLAB and SIMULINK environments, and its performance is evaluated against a passive suspension system. To increase realism and user interactivity, a MATLAB App Designer-based tool is also developed, allowing for the customization of input profiles, simulation parameters, and controller weighting matrices. Comparative case studies involving step, bump, pothole, and sinusoidal road inputs demonstrate that the active suspension system significantly outperforms the passive one, especially in terms of faster settling time and reduced acceleration levels. The results confirm that the LQR controller is effective and practical for active suspension design. This study also provides a foundation for further development, including advanced control strategies such as LQI and real-time implementation.

SYMBOLS

$\boldsymbol{x_s}$: sprung mass vertical displacement

$\boldsymbol{x_u}$: unsprung mass vertical displacement

$\boldsymbol{\dot{x}_s}$: sprung mass vertical velocity

$\boldsymbol{\dot{x}_u}$: unsprung mass vertical velocity

$\boldsymbol{\ddot{x}_s}$: sprung mass vertical acceleration

$\boldsymbol{\ddot{x}_u}$: unsprung mass vertical acceleration

$\boldsymbol{m_s}$: mass of sprung mass

$\boldsymbol{m_u}$: mass of unsprung mass

$\boldsymbol{k_s}$: suspension spring stiffness

$\boldsymbol{k_t}$: tire stiffness

$\boldsymbol{c}$: suspension damping coefficient

$\boldsymbol{x_r}$: road vertical displacement

$\mathbf{U(t)}$: control force applied by the actuator

$\mathbf{A}$: state matrix

$\mathbf{B}$: input matrix

$\mathbf{C}$: output matrix

$\mathbf{D}$: feed-forward matrix

$\mathbf{Q}$: state cost matrix

$\mathbf{R}$: control cost matrix

$\mathbf{K}$: gain matrix

ABBREVIATIONS

LQR **:** Linear-Quadratic Regulator

LQG **:** Linear-Quadratic Gaussian

LQI **:** Linear-Quadratic Integral

FLC **:** Fuzzy Logic Control

PID **:** Proportional-Integral-Derivative

H_∞ **:** H-Infinity

LIST OF FIGURES

PAGE

LIST OF TABLES

PAGE

1 INTRODUCTION

In today's world, vehicles are inevitable needs of our life. Regardless of having possession of a private automobile, every person appreciates the benefits of the vehicles due to the clear fact that, all humanitarian activities; from health to education, construction, and transportation; are carried out by means of vehicles. So, as vehicles become ever more integral to our daily routines and essential services, technological advancements increasingly focus on enhancing this field. From the steering mechanism to the wheels, every component has been analyzing and improving to make the vehicles better. Among these components, the suspension system plays a crucial role. The first purpose of the suspension system is to reduce the vertical acceleration transmitted from the road to the vehicle body in order to improve the ride comfort. It also enhances the road handling by providing more friction between road and the tire [1].

The history of suspension systems goes back to 1860s. The first example is the use of leaf springs as a suspension element in the velocipede invented by Pierre Michaux and Louis Guillaume Perreaux in 1868 [2]. From that day to this, the thriving technological improvements have made car suspension systems much more better comparing the older versions. In older vehicles, a solid axle is used to transmit the power from the motor to the wheels. Solid axle connects the wheels in two sides. Such mechanisms are called dependent suspension mechanism. In this kind of suspension, disturbances encountered by one of the wheels affect the other wheel's motion [3]. In order to avoid transmission of these effects between wheels, the independent suspension system was developed. In independent suspension systems, the connection arms between wheel and chassis are separate from each other. So, there is no effect transmitted from one wheel to another. Double A-arm and McPherson mechanisms are two independent suspension mechanism examples [3].

The main structure of a suspension system consists of three elements: Spring, damper (shock absorber) and connection arm. Spring is responsible for storing energy caused by the road disturbances while shock absorber, as the name implies, absorbs the energy stored in spring by transforming the energy from vibrations into heat, then releases it into the surrounding environment [4]. The connection arm keeps the wheels aligned, allowing for controlled movement and better handling.

For different purposes, different kinds of elements are used in suspensions. For example, a firm damper is useful for excellent road handling (e.g. in race cars) whereas a soft damper satisfies the ride comfort requirements [5]. On the other hand, leaf springs are practical in heavy-duty vehicles and so on and so forth (*Figure 1*).

Figure 1 - An Example of a Leaf Spring in a Truck

The system is composed of these elements (spring, shock absorber) called passive suspension systems (*Figure 2-a*). Passive suspension system is the simplest and cheapest suspension system however since there is no variable or adaptive parameter or coefficient according to the disturbances caused by the road, it is not appropriate solution for varying road conditions [6]. Another available system is semi-active suspension system (*Figure 2-b*). Unlike passive one, the damper with variable coefficient (the coefficient can be changed by the change in hole diameter of the piston) is used instead of fixed coefficient [6]. Even though semi-active suspensions can perform well compared to the passive ones in various conditions, the features like ride comfort and road handling do not meet the requirements of the desired level. As a consequence, the active suspension systems were developed (*Figure 2-c*). In addition to the spring and shock absorber, active suspension systems use a controllable actuator positioned between the sprung (vehicle body) and unsprung (wheel) masses, allowing it to both supply energy within the system [4]. Even though the active suspension system is expensive, as expected, it is the most capable one in terms of performance [6].

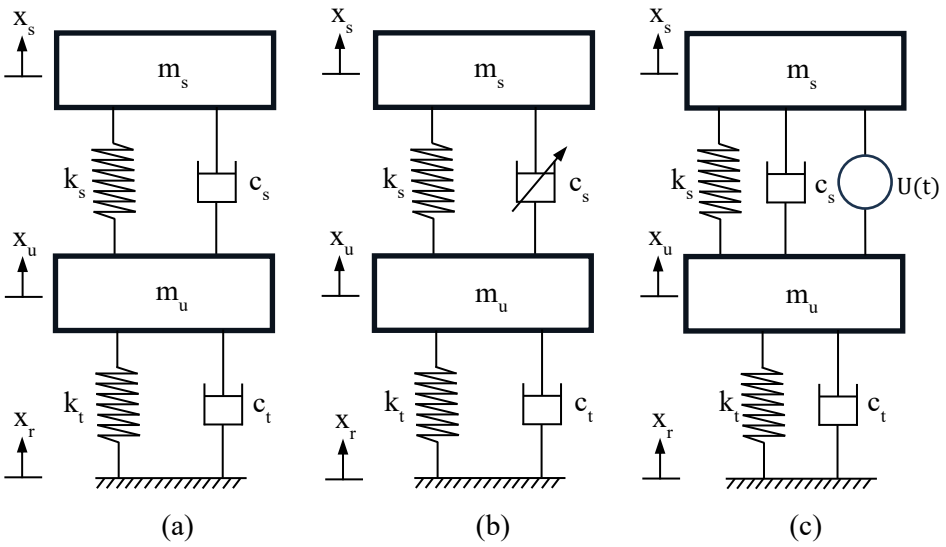


Figure 2 - Schematic of the three types of suspension systems on quarter-car model. (a) Passive, (b) semi-active, and (b) active suspension systems. [9]

In general, there are two main ways to develop an active vibration control system – the feedforward method and the feedback method [7]. In feedforward control, a signal related to the disturbance is sent to the controller, which then generates a signal to move an actuator to cancel out the disturbance. In contrast, feedback control uses signals from the system's response to the disturbance to drive the actuator for reducing the vibrations. In theory, feedforward control can perform better than feedback control. However, feedforward control has a key limitation: it requires a signal that closely matches the disturbance input, which is often difficult to obtain. Therefore, feedback controllers are useful in a wider range of situations. Linear-Quadratic Regulator Control (LQR), Linear-Quadratic Gaussian Control (LQG) and Sliding Mode Control are some methods of feedback control used in designing active suspension systems [8].

The starting point of developing a suspension system is selecting an appropriate model. There are three types of models: Full (*Figure 3-b*), half (*Figure 3-a*), and quarter-car (*Figure 2*) models. In full car model; body bounce, body pitch, body roll and independent wheel movements of the vehicle can be modelled whereas half car model is used for examining rolling (front view) or pitching motion (side view). As the simplest and the basic one, quarter car model focuses solely on the vertical dynamics (bounce and suspension travel) of one wheel and its associated suspension system, so it does not capture any rotational effects.

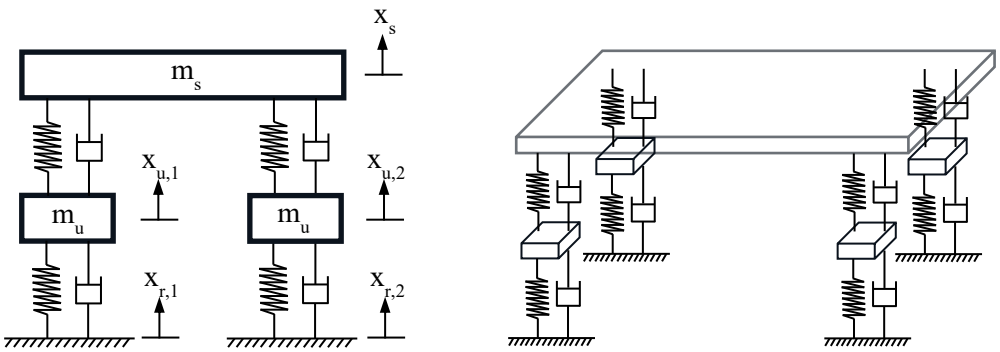


Figure 3 - Schematic of Half-car (a) and Full-car (b) Model.

Throughout this study, among feedback methods and models, the LQR and quarter car model are used, respectively, on MATLAB / SIMULINK Environment to design a controller for an active suspension system.

2 MODELING

The quarter-car model of an active suspension is shown in *Figure 4*. In this figure, x_s, x_u, and x_r represent the displacement of the sprung mass (vehicle body), unsprung mass (tire), and road surface, respectively. The stiffnesses of the spring and the tire are denoted as k_s and k_t whereas the damping coefficient of damper is expressed as c. $U(t)$ corresponds to the actuator force. Lastly, m_s and m_u refer to mass of the sprung and the unsprung masses, respectively.

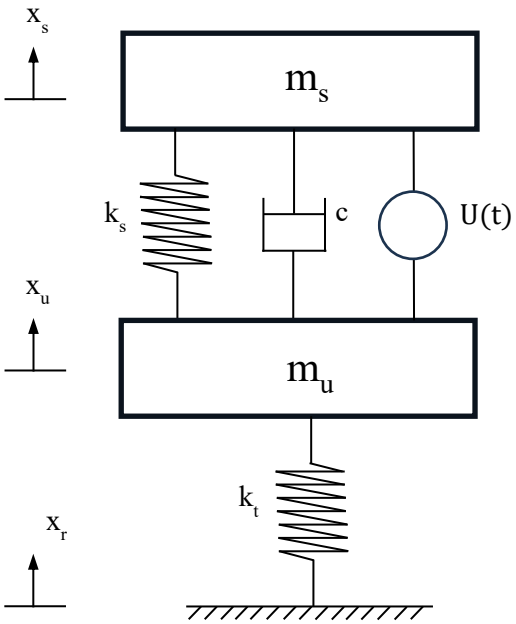


Figure 4 - Quarter-car Model of an Active Car Suspension System

To start mathematical modeling, the free body diagram of the two masses is to be drawn assuming: $x_s > x_u > x_r$.

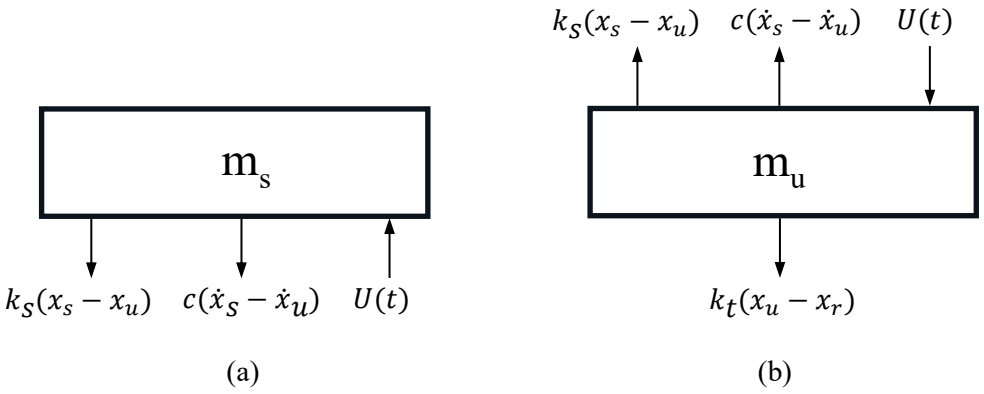


Figure 5 - Free Body Diagram of the Sprung (a) and Unsprung (b) Masses

According to *Figure 5-a*, the equation of motion of the sprung mass can be written as:

$$m_s\ddot{x}_s = -k_s(x_s - x_u) - c(\dot{x}_s - \dot{x}_u) + U(t) \quad (1)$$

If $U(t)$ is left alone:

$$m_s\ddot{x}_s + c(\dot{x}_s - \dot{x}_u) + k_s(x_s - x_u) = U(t) \quad (2)$$

For the unsprung mass (*Figure 5-b*), the equation of motion is:

$$m_u\ddot{x}_u = k_s(x_s - x_u) + c(\dot{x}_s - \dot{x}_u) - k_t(x_u - x_r) - U(t) \quad (3)$$

Some arrangements yield:

$$m_u\ddot{x}_u + c\dot{x}_u - c\dot{x}_s + (k_t + k_s)x_u - k_s x_s = -U(t) + k_t x_r \quad (4)$$

Writing Eq. 2 and 4 in matrix form gives:

$$\begin{bmatrix} m_s & 0 \\ 0 & m_u \end{bmatrix}\begin{bmatrix} \ddot{x}_s \\ \ddot{x}_u \end{bmatrix} + \begin{bmatrix} c & -c \\ -c & c \end{bmatrix}\begin{bmatrix} \dot{x}_s \\ \dot{x}_u \end{bmatrix} + \begin{bmatrix} k_s & -k_s \\ -k_s & k_t + k_s \end{bmatrix}\begin{bmatrix} x_s \\ x_u \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}U(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix}k_t x_r \quad (5)$$

In order to assign proper values for the parameters existing in equations, the resources were scanned. Using findings, the below table was generated.

Table 1 – Parameter Values, Inputs, and Control Methods Obtained From Different Sources

Reference No	m_s [kg]	m_u [kg]	k_s [kN/m]	k_t [kN/m]	c [kN.s/m]	Input	Method
[10]	338.5	59	15	190	0.6	Disruptive Road	PID
[11]	300	50	18	190	2	Pothole – Bump – High Frequency Random	PID – FLC – Passive
[5]	250	50	16.812	190	1	Step	PID – FLC – LQR
[12]	350	50	35	100	3	Bump - Step	PID – LQR – Passive
[1]	214.5	41.5	6	140	0.3 (Wheel Damping: 1.5)	Step	LQR
[13]	338.5	59	15	190	0.45	Impulse	Passive
[14]	290±87	59	16.182	190±76	1	Bump	H_∞, LQR, Fuzzy
[15]	350	50	8	190	1	Random Road	LQR

In the table above, besides the values of the parameters to be used in the equations of motion, the input and controller design methods used in the studies in various models are also

included. In some of these studies, random input was selected in order to be suitable for reality, while in others, bumps of standard sizes frequently encountered in daily driving were taken as input.

The used control methods are also mentioned in the table above. It is seen that in some studies only a method is used to design a controller for active suspension system whereas in others, several methods are used and compared with each other. For example, in [12], LQR and PID were used as control methods. In this study, it is seen that LQR method is effective in reducing the amplitude of vertical displacement of the vehicle body and PID method is effective in reducing the vertical acceleration of the vehicle body. As another example, in [5], it is observed that the LQR method is more effective in terms of the speed of reaching stability and the PID method is more effective than the other methods used in the study in terms of reducing the magnitude of the vertical displacement of the vehicle body. The difference between the results of these two studies can be caused by the values of the parameters, the input types etc.

The parameters of our quarter vehicle active suspension system were designated to be compatible with the data in *Table 1* as:

Table 2 – Parameter Values to be Used in This Study

Parameters	Symbol	Value	Unit
Sprung Mass	m_s	300	kg
Unsprung Mass	m_u	50	kg
Suspension Stiffness	k_s	16000	N/m
Suspension Damping Coefficient	c	1000	N.s/m
Tire Stiffness	k_t	190000	N/m

After determining the values of the parameters, as a preliminary study, a passive suspension system was modelled in computer environment (MATLAB / SIMULINK) as shown in the figure below, using the values in *Table 2*.

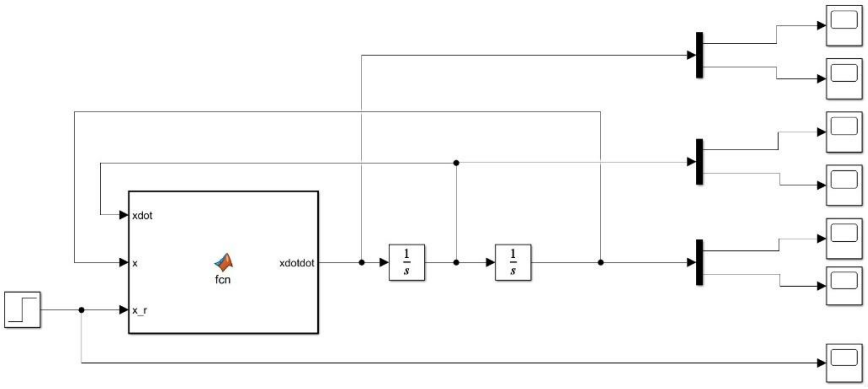


Figure 6 - SIMULINK Model of the Preliminary Study

In this model, the input was taken as step input (*Figure 7*). The necessary calculations, as in Eq. 5, were computed in the MATLAB Function block (in ***8.1***). After generating the model, it was run and the results in (*Figure 8*) were obtained.

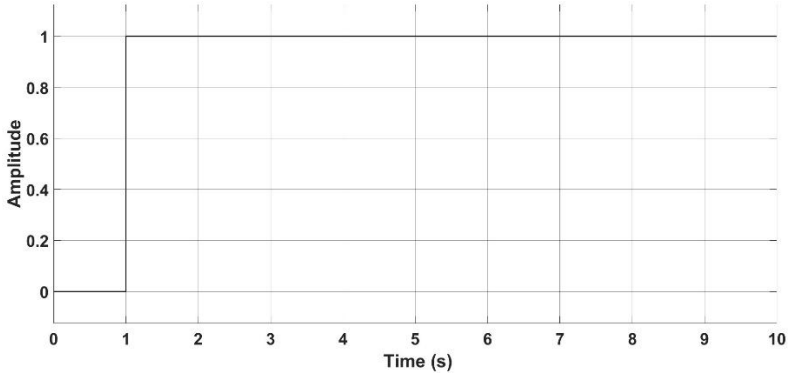
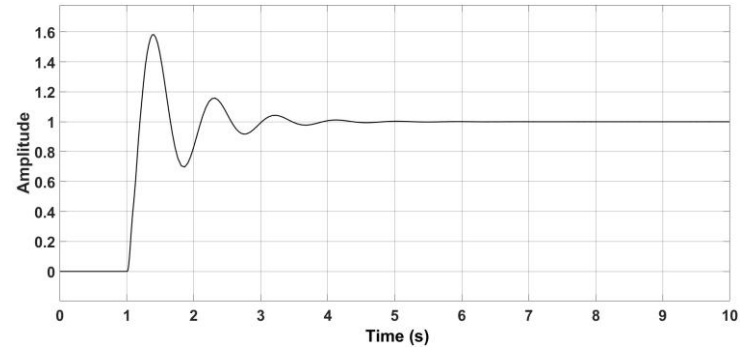
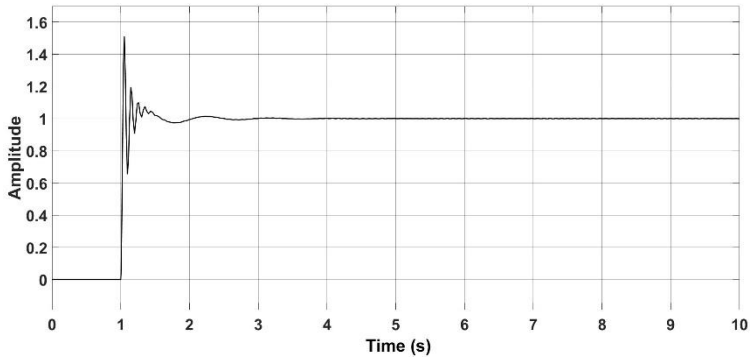


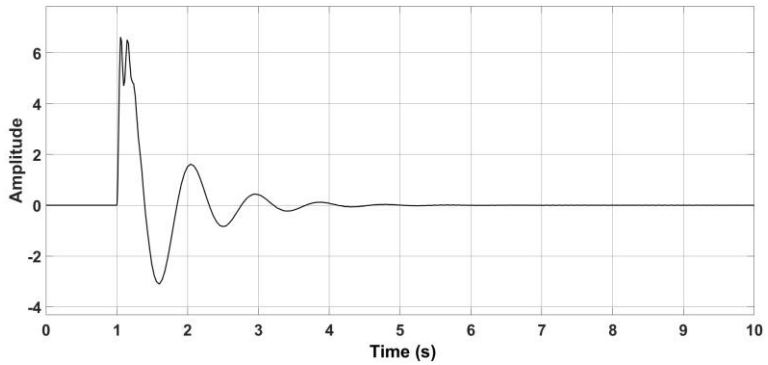
Figure 7 - Step Input in m



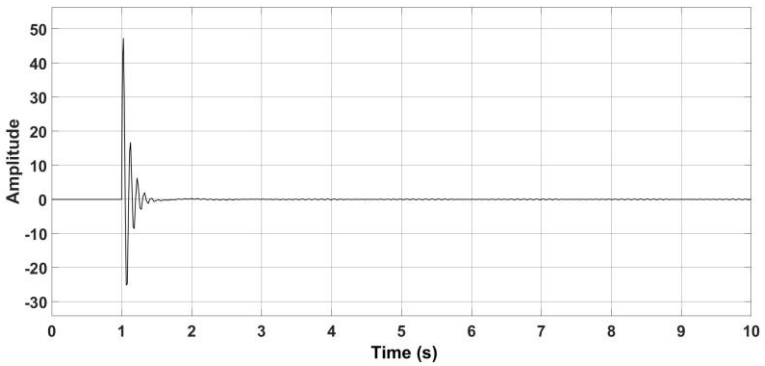
(a)



(b)



(c)



(d)

Figure 8 - Simulation Results: Vertical Displacement of the Vehicle Body (a) and the Wheel (b) in m, and the Vertical Velocity of the Vehicle Body (c) and the Wheel (d) in m/s

3 SIMULATION

Throughout this project, controller for the active car suspension problem is going to be designed using LQR approach.

3.1 LQR

The Linear-Quadratic Regulator (LQR) is one of the popular methods in control algorithms selecting the best gain value which is multiplied by input to minimize the error between the input and feedback signal. The LQR method is applied on systems that possess linear dynamic behavior. Q represents the quadratic cost function which needs to be minimized. Therefore, this approach, in short, regulates the behavior of linear dynamic systems to minimize a specific quadratic cost function. Unfortunately, there is no way to obtain both performance and effort maximized at the same time. Trade-off presents between the system's state deviations and the control effort required [20]. The main idea is tuning the parameters in order to acquire the optimal adjustments for both performance and effort.

3.1.1 Theoretical Foundation of LQR

Continuous-time linear system described by the state-space equations can be expressed as:

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{6}$$

where:

- $x(t)$ is the state vector,
- $u(t)$is the control input,
- A is the system matrix,
- B is the input matrix.

The objective is to determine the control input $u(t)$ that minimizes the quadratic cost function [8]:

$$J = \int_0^\infty \left(x^T(t)Qx(t) + u^T(t)Ru(t)\right) dt \tag{7}$$

where:

- Q is a symmetric positive semi-definite state weighting matrix, $(Q \geq 0)$
- R is a symmetric positive definite control input weighting matrix [20]. $(R > 0)$

3.1.2 Role of the Riccati Equation

The optimal control law that minimizes the cost function can be given by a state feedback mechanism by:

$$u(t) = -Kx(t) \tag{8}$$

where, K is the gain matrix obtained by manipulating the cost function (Eq. 7) as:

$$K = R^{-1}B^T P \tag{9}$$

Here, P is a symmetric positive definite matrix that satisfies the Continuous-Time Algebraic Riccati Equation (CARE) [21]:

$$A^T P + PA - PBR^{-1}B^T P + Q = 0 \tag{10}$$

Solving the CARE has crucial importance, as the matrix P embodies the trade-offs between state deviations and control efforts, thereby determining the optimal feedback gains. Efficient numerical methods are available for solving the CARE, facilitating the practical implementation of LQR controllers [22].

3.2 lqr Function in MATLAB

LQR controllers are widely used in various engineering applications due to their optimality and robustness. Software tools like MATLAB offer functions such as "lqr" to compute the optimal gain matrix K by solving the Riccati equation, regulate the design process in an easy way. The way in which this function works, and its arguments are given in the following.

$$K = \text{lqr}(A, B, Q, R) \tag{11}$$

Here,

- A, B: System matrices from state-space model
- Q: Weighing matrix for states,
- R: Weighing matrix for control inputs,
- K: Optimal gain matrix such that $u(t) = -Kx(t)$.

3.3 State-Space Model

In order to be compatible with the lqr function context in MATLAB, state space representation should be defined.

The states are chosen as:

$$x_1 = x_s \tag{12}$$

$$x_2 = \dot{x}_s \tag{13}$$

$$x_3 = x_u \tag{14}$$

$$x_4 = \dot{x}_u \tag{15}$$

Differentiating both sides of the equations and using Eq. 2 and Eq. 4, following expressions are obtained.

$$\dot{x}_1 = x_2 = \dot{x}_s \tag{16}$$

$$\dot{x}_2 = \ddot{x}_s = \frac{U(t) - k_s x_1 - c x_2 + k_s x_3 + c x_4}{m_s} \tag{17}$$

$$\dot{x}_3 = x_4 = \dot{x}_u \tag{18}$$

$$\dot{x}_4 = \ddot{x}_u = \frac{-U(t) + k_t x_r + k_s x_1 + c x_2 - (k_t + k_s) x_3 - c x_4}{m_u} \tag{19}$$

The general form of a state-space model is represented as:

$$\dot{x} = Ax + Bu \tag{20}$$

$$y = Cx + Du \tag{21}$$

where; A is state matrix, B is input matrix, C is output matrix, and D is feed-forward matrix. The model is completed by substituting Eq. 16, Eq. 17, Eq. 18, and Eq. 19 into Eq. 20 and Eq. 21.

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{k_s}{m_s} & -\frac{c}{m_s} & \frac{k_s}{m_s} & \frac{c}{m_s} \\ 0 & 0 & 0 & 1 \\ \frac{k_s}{m_u} & \frac{c}{m_u} & -\frac{(k_t + k_s)}{m_u} & -\frac{c}{m_u} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & \frac{1}{m_s} \\ 0 & 0 \\ \frac{k_t}{m_u} & -\frac{1}{m_u} \end{bmatrix} \begin{bmatrix} x_r \\ U(t) \end{bmatrix} \tag{22}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ -\frac{k_s}{m_s} & -\frac{c}{m_s} & \frac{k_s}{m_s} & \frac{c}{m_s} \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & \frac{1}{m_s} \\ -1 & 0 \end{bmatrix} \begin{bmatrix} x_r \\ U(t) \end{bmatrix} \tag{23}$$

where:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{k_s}{m_s} & -\frac{c}{m_s} & \frac{k_s}{m_s} & \frac{c}{m_s} \\ 0 & 0 & 0 & 1 \\ \frac{k_s}{m_u} & \frac{c}{m_u} & -\frac{(k_t + k_s)}{m_u} & -\frac{c}{m_u} \end{bmatrix} \tag{24}$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & \frac{1}{m_s} \\ 0 & 0 \\ \frac{k_t}{m_u} & -\frac{1}{m_u} \end{bmatrix} \tag{25}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ -\frac{k_s}{m_s} & -\frac{c}{m_s} & \frac{k_s}{m_s} & \frac{c}{m_s} \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{26}$$

$$D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & \frac{1}{m_s} \\ -1 & 0 \end{bmatrix} \tag{27}$$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \tag{28}$$

$$\dot{x} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} \tag{29}$$

$$y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \tag{30}$$

The four outputs that can be obtained by constructing the C matrix as described above are as follows:

1. **Sprung Mass Displacement (y_1):** The vertical displacement of the car body (sprung mass) relative to its equilibrium position.
    - Represents how much the body moves in response to road irregularities.
    - Displacement is one of the key indicators of ride comfort and stability—excessive displacement means that passengers will strongly feel road disturbances, leading to reduced comfort.
2. **Suspension travel (y_2):** The relative vertical displacement between the sprung mass and the unsprung mass (wheel assembly).
    - Indicates whether the suspension is operating within its mechanical limits.
    - If suspension travel exceeds bounds, it may cause bottoming out or top-out, which damages the system or leads to unsafe behavior.
    - Also connected to handling and vehicle stability [19].
3. **Sprung mass Acceleration (y_3):** The vertical acceleration of the car body (sprung mass).
    - Directly related to ride comfort—especially human perception [19].
    - High acceleration values mean jerkiness or harshness felt by passengers.
    - ISO 2631 standards even relate acceleration levels to human comfort thresholds [18].
4. **Tire Deflection (y_4):** The relative vertical displacement between the unsprung mass and the road.
    - Road-holding ability is a key factor for vehicle safety and depends on maintaining adequate tire-road contact forces [19].
    - This ability is typically quantified by tire deflection, which must remain within acceptable limits to ensure safe performance [19].

Additionally, although the control force is not an output defined in Eq. 23, it has considerable importance on overall design of active suspension system.

- **Control Force**: The force applied by the actuator in the active suspension system.
  - Represents the effort or energy required by the control system.
  - Important for practicality: too much force may require a larger actuator, more power, or lead to faster wear.

These features cannot take any arbitrary value, there always be some limitations or range that operate without facing any issue in terms of physical limitations, comfort requirements for both driver and passengers, and mechanics. In literature, there are numerous research take place on this topic representing restriction values in other words limitations.

Table 3 – Ranges and Limitations for the Attributes

Attribute	Range	Limitation	Reference No
Suspension Travel	$\leq \lvert 127\ \text{mm} \rvert$	Exceeding mechanical limits causes bottoming out or top-out	[19]
Sprung Mass Acceleration	$\leq \lvert 10.3\ \text{m/s}^2 \rvert$	High acceleration causes discomfort and reduces ride quality by introducing harshness and vibrations felt by passengers	[19]
Tire Deflection	$\leq \lvert 50.8\ \text{mm} \rvert$	Too much deflection reduces handling and increases tire wear; too little causes harsh ride	[19]
Control Force	0-3000 N (vehicle-dependent)	Exceeding limits increases power use, actuator wear	[16,17]

3.4 SIMULINK Model

In the preliminary study (the study for the passive system), the model was generated using 'MATLAB function' block. In this study, the active suspension system was modeled in SIMULINK using the 'State-Space' block, which was chosen to reduce the complexity of the model. The resulting model is shown in the figure below.

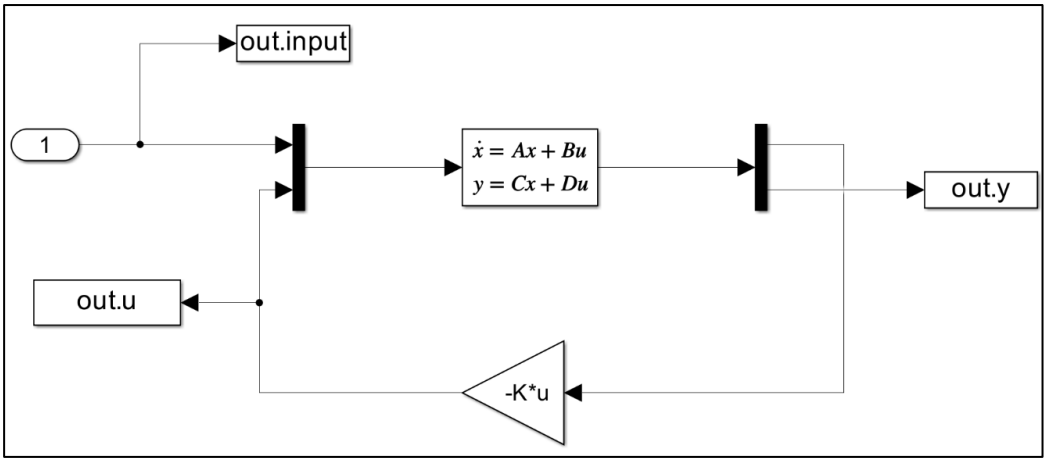


Figure 9 - SIMULINK Model for the Active Suspension System

In this model, the block which has the annotation 1 is 'Inport' block. It imports the input specified in the MATLAB workspace to the SIMULINK environment. The blocks 'out.u' and 'out.y' export the control force and outputs (generated by running the model), respectively, to the MATLAB workspace for post-processing (plotting etc.). The input is also exported back to MATLAB workspace (by out.input block) in order to make the input and outputs consistent during plotting the results.

The matrices used in state space model were constructed in *3.3*. However, in 'State-space' block in the SIMULINK model, the C and D matrices are not used directly. Because, if it is the case, since the out port of the 'State-space' block gives only the outputs specified according to the C matrix, the states cannot be fed back to the system. To be able to feed them back, the C and D matrices existing in the 'State-space block' were assigned as:

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ -\frac{k_s}{m_s} & -\frac{c}{m_s} & \frac{k_s}{m_s} & \frac{c}{m_s} \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{31}$$

$$D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & \frac{1}{m_s} \\ -1 & 0 \end{bmatrix} \tag{32}$$

It is seen that the upper half of the C matrix is an 4×4 identity matrix and the first 4 rows of the D matrix consist of zeros. By this way, the out port of the State-space block gives 8 outputs. First four of them are states and the others are desired outputs. These obtained 8 outputs are separated into 2 groups as states and outputs using 'Demux' block. Then, the states are multiplied by the gain K and fed back to the system via 'Mux' block.

3.5 Input

The inputs used in this study were decided as a step and a bump input since these two are often encountered in real life. Inputs were modelled as a function of time. To be consistent with the real applications, it is convenient to determine the height of both inputs as 10 cm. Moreover, to be able to see the effect of suspension clearly, the simulation time was chosen as 10 second while the inputs are applied on the first three seconds at most. The input formulations and the shapes are given below. Step input was generated by a sigmoid function instead of a classical step in order to get rid of the discontinuities at the sharp corners. MATLAB code for the inputs are provided in *8.2* and *8.3*.

3.5.1 Step Input

$$x_r(t) = \frac{0.1}{1 + e^{-25(t-1)}} \tag{33}$$

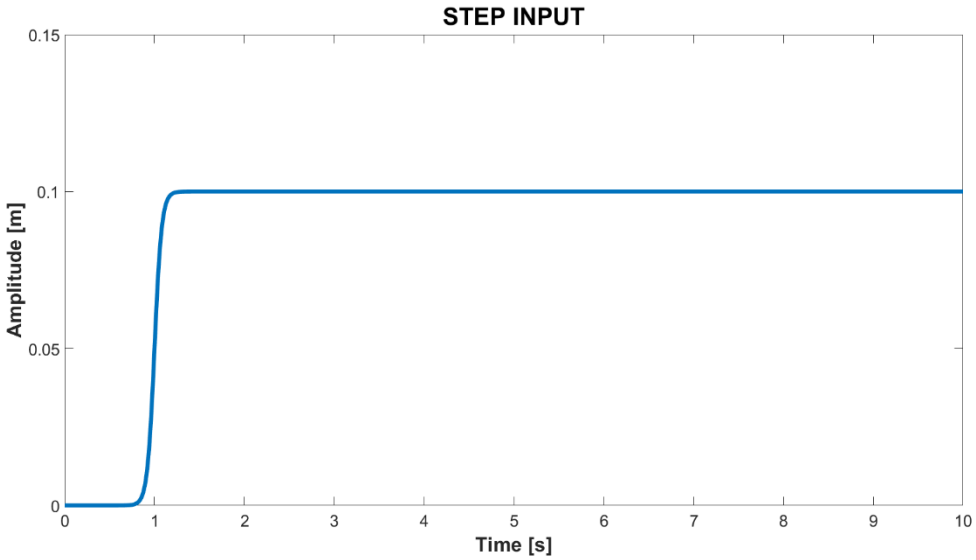


Figure 10 - Step Input

3.5.2 Bump Input

$$x_r(t) = \begin{cases} (0.05)\left(1 - \cos\left(2\pi\left(\frac{t-1}{2}\right)\right)\right), & 1 \leq t \leq 3 \\ 0, & \text{otherwise} \end{cases} \tag{34}$$

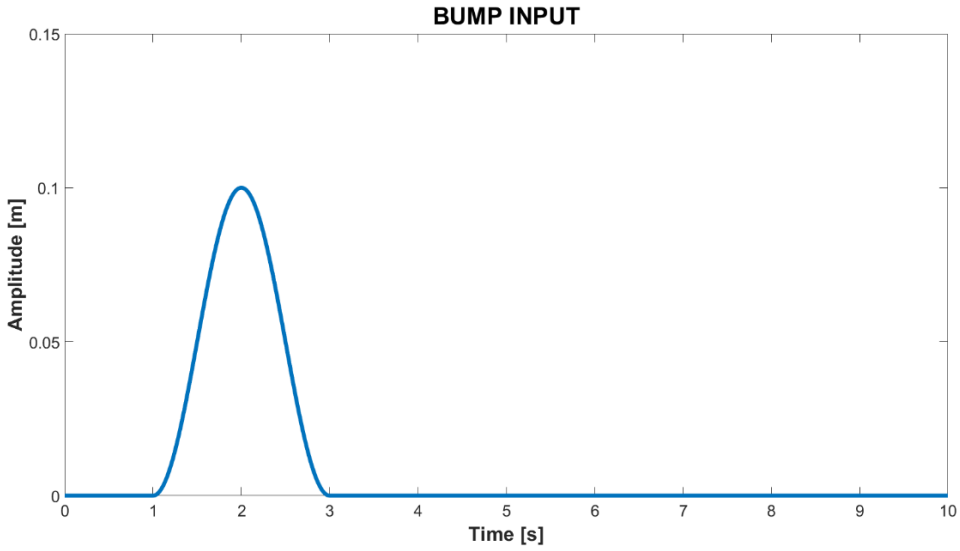


Figure 11 - Bump Input

3.6 Results

After determining the inputs, to simulate the model, the gain K should be evaluated. As mentioned above in *3.2*, the gain is obtained using lqr command in MATLAB. The two indices of the lqr command, Q and R matrices, are to be assigned. To determine these matrices, trial and error was applied until a satisfactory result was obtained. Q and R matrices used in this study are as below:

$$Q = \begin{bmatrix} 10900 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 10000 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix} \tag{35}$$

$$R = 0.0001 \tag{36}$$

Then, when the lqr command is run with these matrices, the gain (K matrix) is evaluated as:

$$K = [3104.97 \quad 818.33 \quad -3090.32 \quad -47.75] \tag{37}$$

The model is simulated by assigning the gain and the results below are obtained.

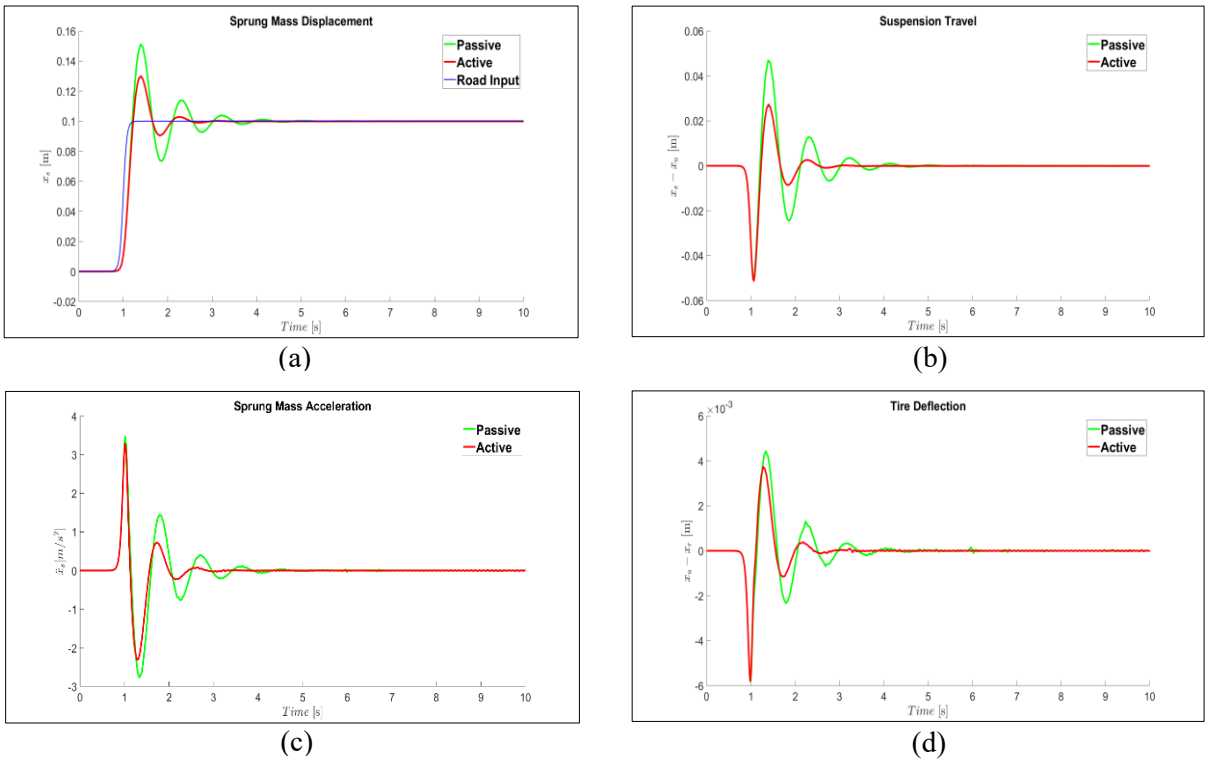


Figure 12 - Comparison Between the Response of the Active and Passive Suspensions to the Step Input on the Chosen Parameters: Sprung Mass Displacement (a), Suspension Travel (b), Sprung Mass Acceleration (c), and Tire Deflection (d)

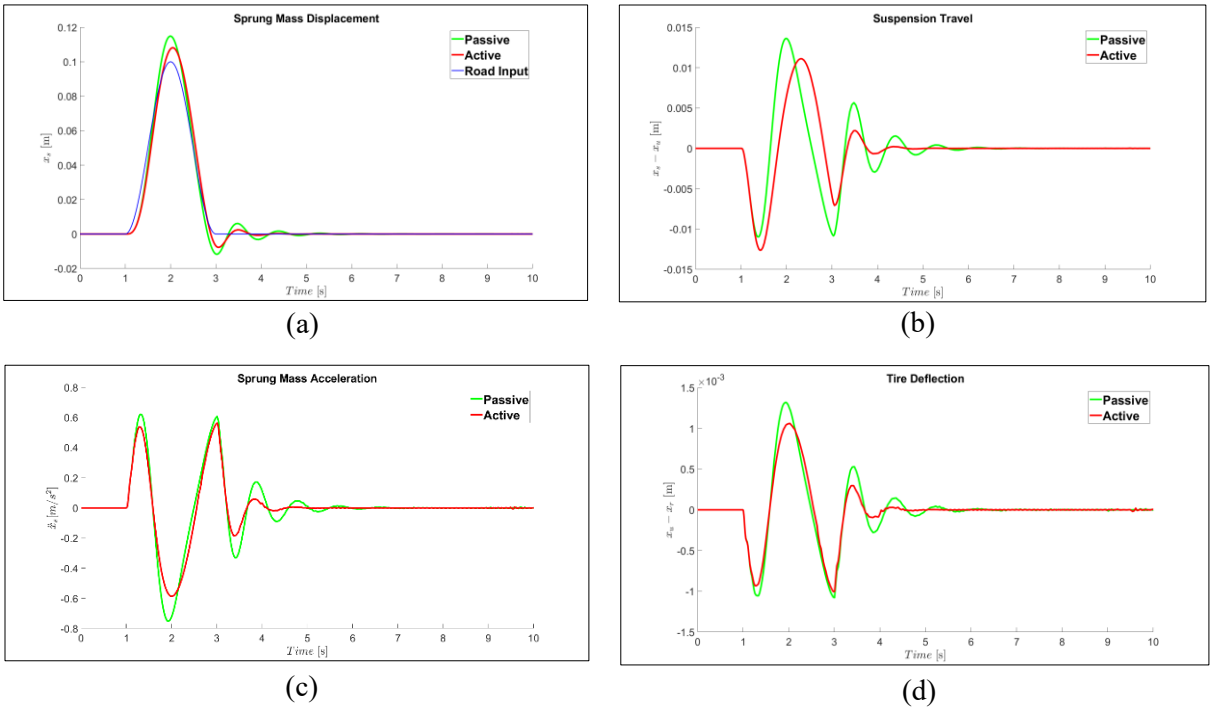


Figure 13 - Comparison Between the Response of the Active and Passive Suspensions to the Bump Input on the Outputs: Sprung Mass Displacement (a), Suspension Travel (b), Sprung Mass Acceleration (c), and Tire Deflection (d)

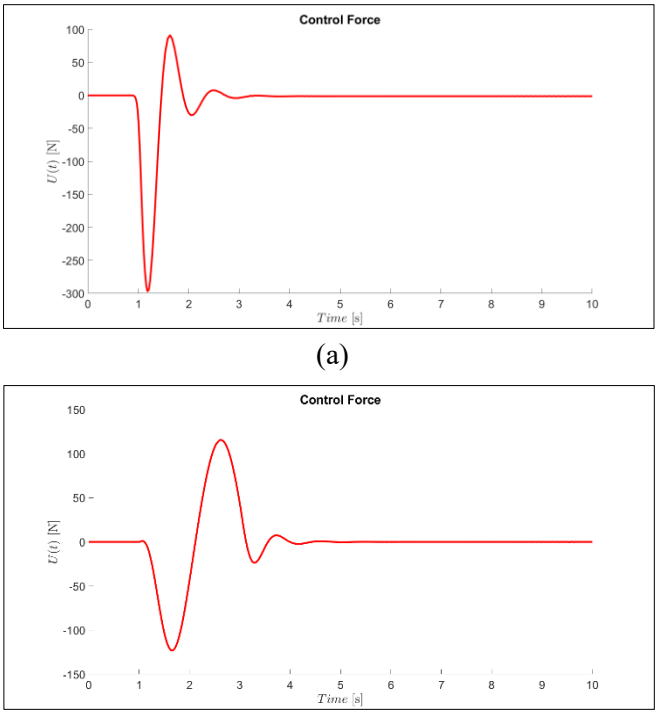


Figure 14 - The Force Applied by the Actuator as a Result of the Step (a) and the Bump (b) Input

In the results, along with the active system, the passive system is presented. The responses of the passive system to the inputs were obtained by simulating the model with a zero gain. If the results are examined, the effect of the active suspension can be realized clearly. Also, it is seen that the criterions set in *Table 1* are satisfied.

About the control force, it is shown in *Figure 14* that the absolute maximum force occurs during the step and bump input approximately as 300 N and 130 N, respectively. (The minus sign in the force means the actuator applies the force at the corresponding instant in the direction opposite to the assumed direction in *Figure 5*). Other studies in this field show that an actuator on the suspension unit can apply a force of 1000 N to 2000 N [16] or even up to 3000 N in sudden impacts (bumps, potholes, etc.) [17]. So, the force levels (*Figure 14*) are compatible with the other studies in this field.

An important point should be noted here. In the step response, a steady-state error may occur depending on both the road profile and control input parameters. This behavior is not observed in *Figure 12*, as the Q and R matrices were selected in a way that prevents the error from appearing. To eliminate the steady-state error entirely, integral action can be combined with the LQR method [23]. However, it is not included in this study.

4 APP DESIGN

Until now, the simulations have been limited to two constant inputs. However, real-world disturbances are much more diverse. To enhance the realism of this study, a wider range of road excitations was incorporated. For this purpose, a MATLAB App Designer-based tool was developed, working in coordination with a SIMULINK model (*Figure 9*), allowing users to configure various input types and associated parameters. Moreover, earlier models overlooked vehicle speed - an essential factor in suspension dynamics. This application addresses that by treating speed as an adjustable variable. The following section is going to present the interface and explain how to use the application.

4.1 Application Interface and Usage

The application initially displays a cover page that includes the project title, student information, and university details (*Figure 15*). This serves as an introduction to the application.

Figure 15 - Cover Page of the Application [24]

When the 'START' button is pressed, a new page requesting the parameters of the quarter car, including the car speed, appears.

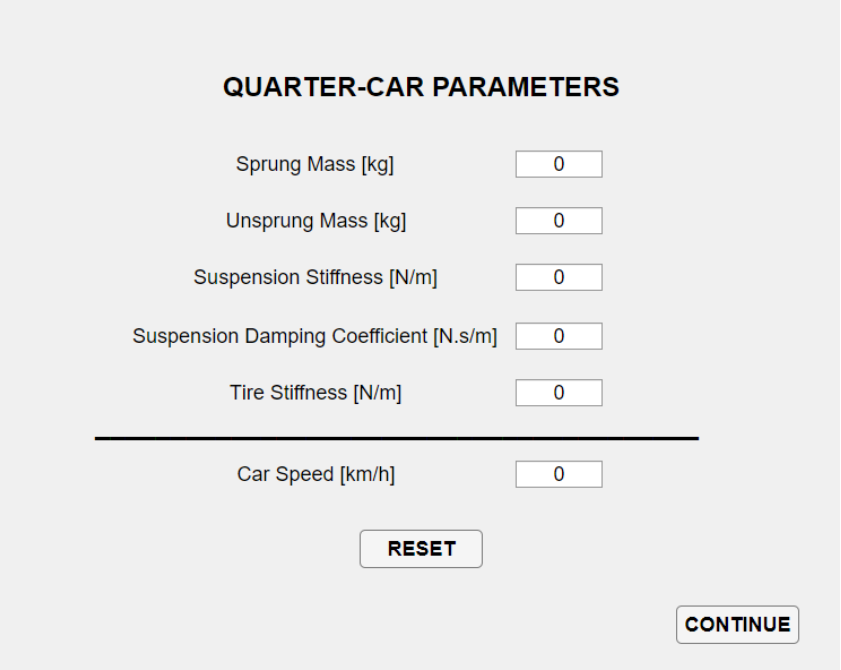


Figure 16 - Quarter-Car Parameters Page of the Application

On this page, the user is asked to enter these parameters in the specified units. Once the blanks are filled in, the user can press the 'CONTINUE' button. In addition to directing the user to the input selection page, this button generates the matrices A, B, C, and D of the state-space model according to the data entered on this page. These matrices are generated using the same approach explained in *3.4*. On the other hand, the 'RESET' button clears all input fields by setting their values to zero, allowing the user to start over.

After this page, the user is directed to a new page to select the road input type.

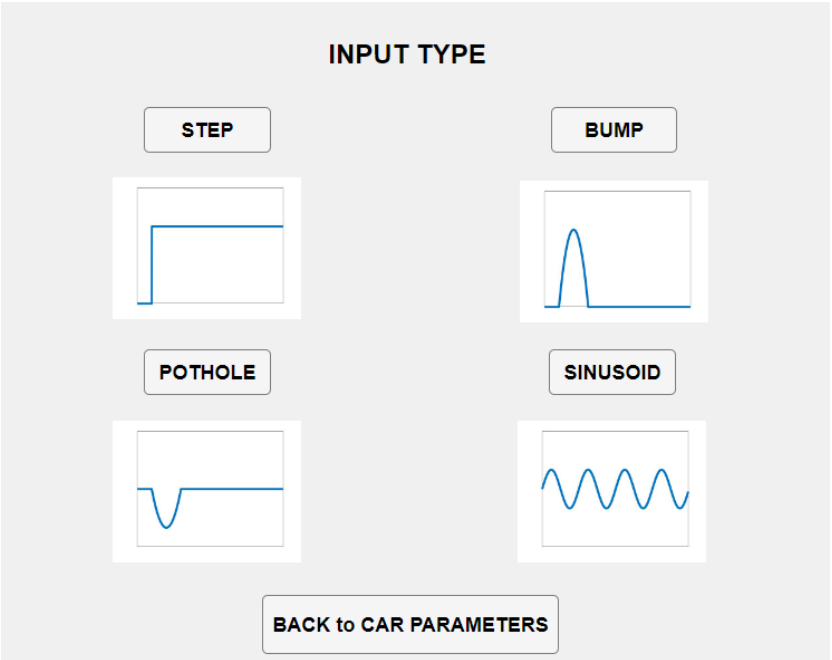


Figure 17 - Input Type Selection Page of the Application

As shown in *Figure 17*, there are four types of input: step, bump, pothole, and sinusoid. The user can select one of these input types by pressing the corresponding button or can return to the previous page by clicking on the button at the bottom of the page.

Selecting an input type directs the user to a new page where he/she can configure the parameters of the chosen input (*Figure 18*).

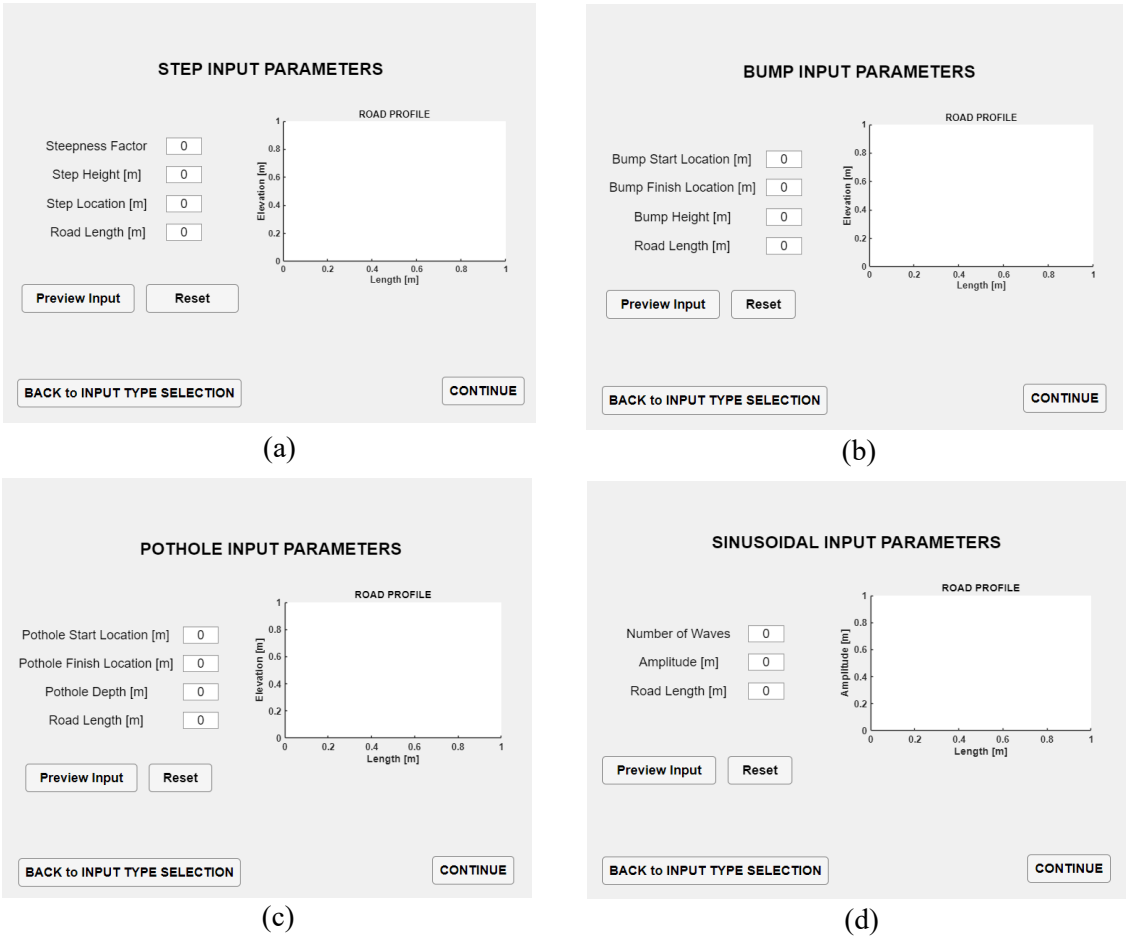


Figure 18 - Page for Adjusting the Parameters of the; (a) Step Input, (a) Bump Input, (c) Pothole Input, (d) Sinusoidal Input

On the step input page, user is prompted to specify the sharpness, position, and magnitude of the step. For the bump input, the interface requires the starting and ending positions along with the bump's height. Similarly, the pothole input asks for the beginning and end points, as well as the depth, to define the road surface. In the sinusoidal input, user must enter the number of waves to be encountered through simulation and their amplitude. Additionally, each input type includes a field called simulation length, representing the total distance to be covered during the simulation.

To prevent discontinuities caused by sharp transitions in the road inputs, smoothing techniques are applied in the background: a sigmoid function for the step input, and cosine-based smoothing to soften the beginning and end segments of the bump and pothole, along with the initial and final portions of the sinusoidal input.

Once all fields are completed, the user should click the ‘Preview Input’ button to visualize the road profile on the existing graph and save the corresponding input. At this stage, the simulation time is also calculated based on the car speed and the simulation length defined by the user, and it is displayed below the plot. If a fresh start is needed, the ‘Reset’ button can be used to clear the graph and set all values to zero. Additionally, the ‘BACK to INPUT TYPE SELECTION’ button enables the user to return to the previous page and choose a different type of input.

After configuring the road input, the user can proceed to the next page (*Figure 19*) by pressing the ‘CONTINUE’ button.

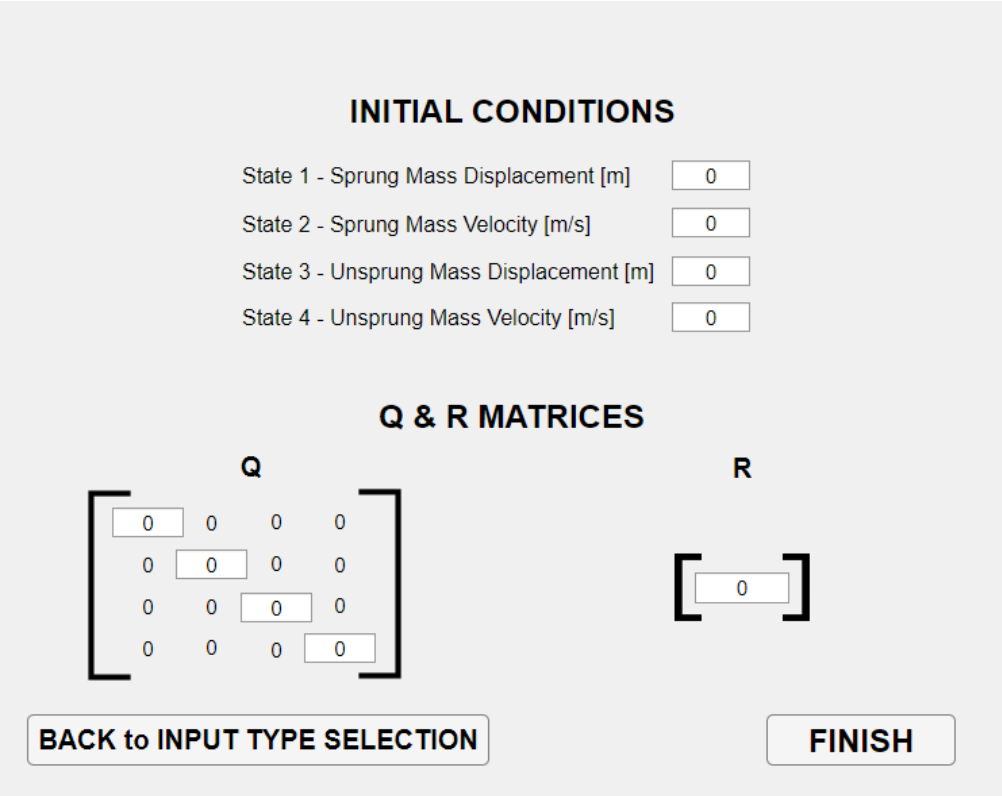


Figure 19 - Setup Page for Initial Conditions and Q & R Matrices

At the top section of this interface, the initial conditions for the system states are entered. Meanwhile, the lower portion is intended to configuring the Q and R matrices, which directly affect the behavior of the LQR controller.

Below the parameter fields, the interface includes two buttons for navigation and execution. The ‘BACK to INPUT TYPE SELECTION’ button returns the user to the previous page to choose a different road input or redesign the currently selected input. The ‘FINISH’ button, on the other hand, uses all parameters entered by the user throughout the application, including those defined on the current page, to run the SIMULINK model and then navigates to a new page where the simulation results are displayed (*Figure 20*), while plotting the system outputs for both active and passive suspension systems across separate figures for detailed analysis (As mentioned before, response of the passive suspension is obtained by simulating the model with zero gain). In addition, it generates an animation for both active

and passive suspension responses using the simulation data and saves it as a video file for visualization (*Figure 21*).

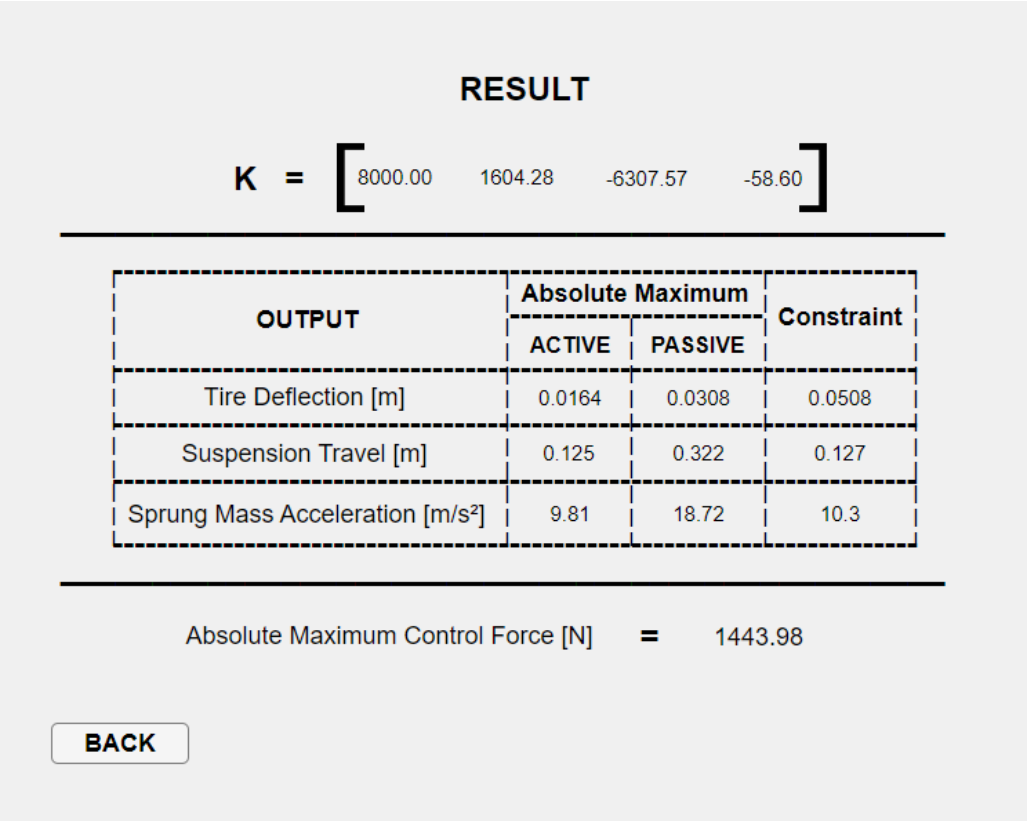


Figure 20 - Result Page of the Application With Sample Results

The results are exhibited in three sections. At the top, the gain matrix K computed based on the system matrices and Q and R matrices, is displayed. In the middle, a table shows the absolute maximum values of the outputs from both active and passive suspension simulations, along with the corresponding constraints listed in *Table 3*. Finally, at the bottom, the absolute maximum control force observed during the simulation is presented. Moreover, a 'BACK' button is provided, allowing the user to re-run the simulation with modified parameters without needing to close the application.

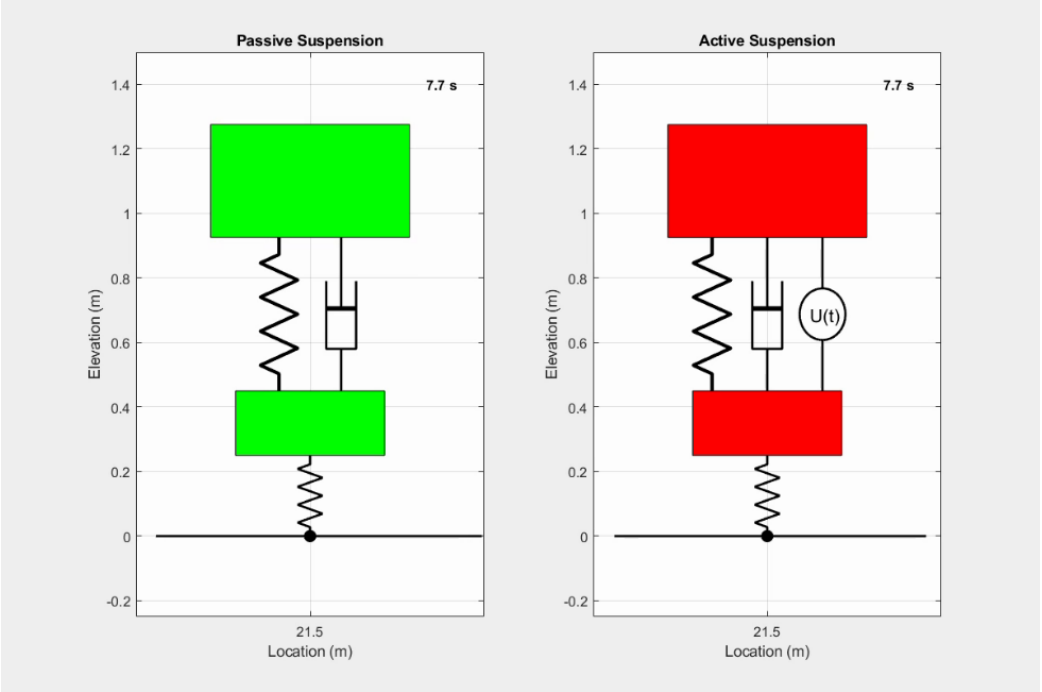


Figure 21 - A Screen Shot Taken From the Animation

In the figure above, a single frame from the animation is displayed. The behaviors of the passive and active suspension systems for the given input are shown on the left and right, respectively. The x-axis indicates the vehicle's position, and the y-axis shows the road elevation. Additionally, the current simulation time is presented in the top-right corner of each visualization. The systems encounter the same section of the road simultaneously, as the simulations progress in parallel. In *Figure 22*, the screen shots taken from different instances of a sample animation are seen. This figure presents the improvement from passive to active suspension obviously.

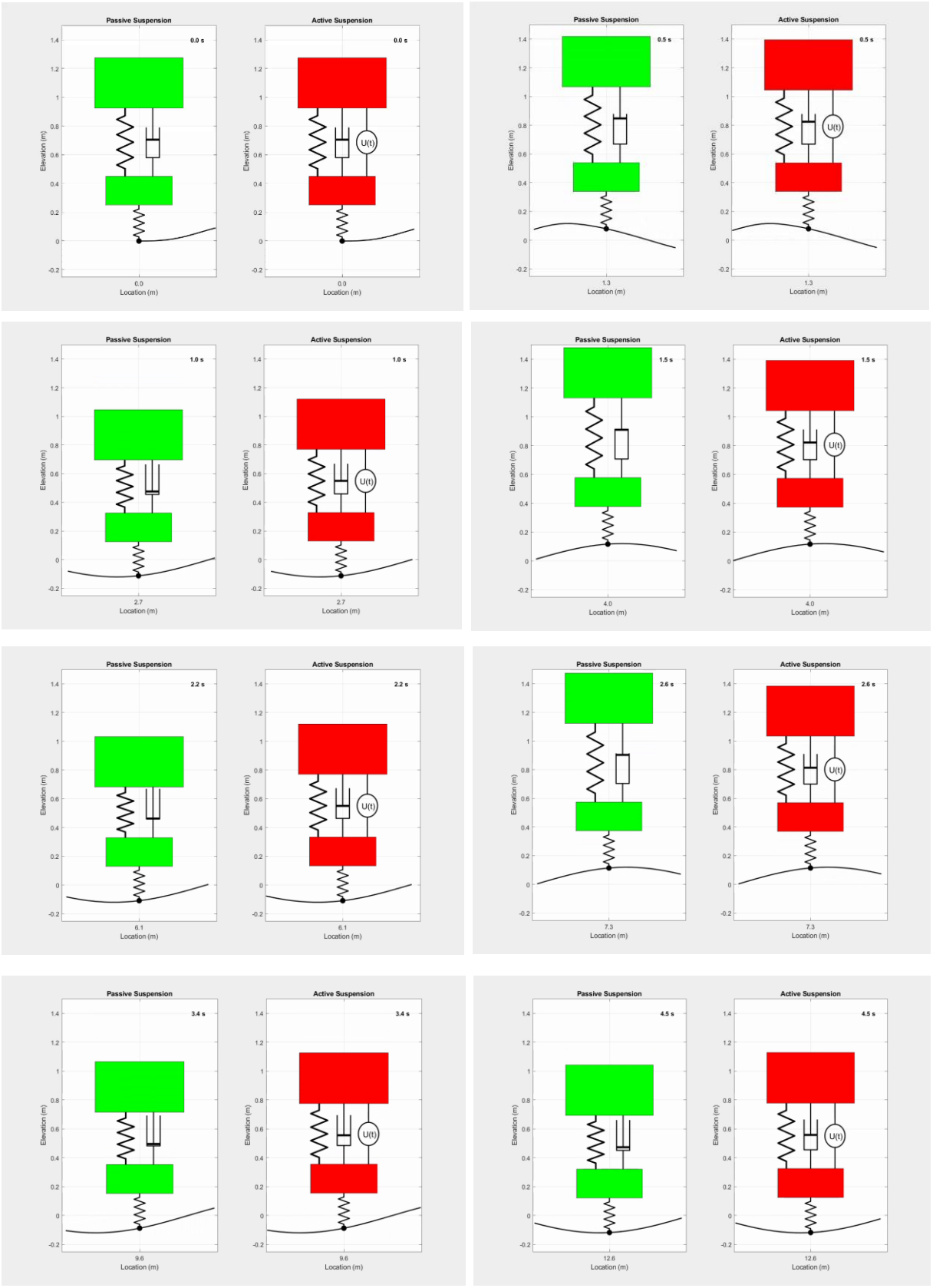


Figure 22 – Figure Showing the Progress of an Animation With a Sample Sinusoidal Input

5 CASE STUDIES

In this section, case studies will be conducted using the application for various road input types, and the corresponding simulation results will be discussed. The vehicle parameters used in the simulations are those listed in *Table 2* and the initial conditions for all states are set to zero.

5.1 Case Study – 1

5.1.1 Simulation Setup

Car Speed = 15 km/h

Road Parameters

- Type: Step Input
- Steepness Factor: 20
- Step Height: 0.15 m
- Step Location: 5 m
- Road Length: 20 m

Q & R Matrices

- $Q = \begin{bmatrix} 2 \times 10^7 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 2 \times 10^7 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix}$
- $R = [9.46 \times 10^{-6}]$

5.1.2 Results and Discussion

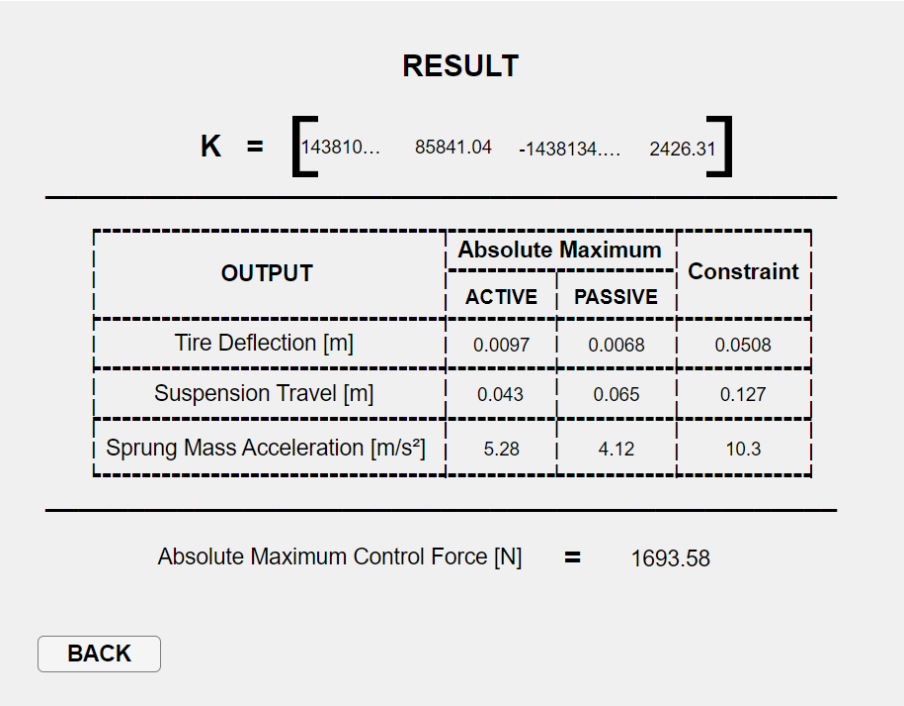


Figure 23 - Results for the Step Input

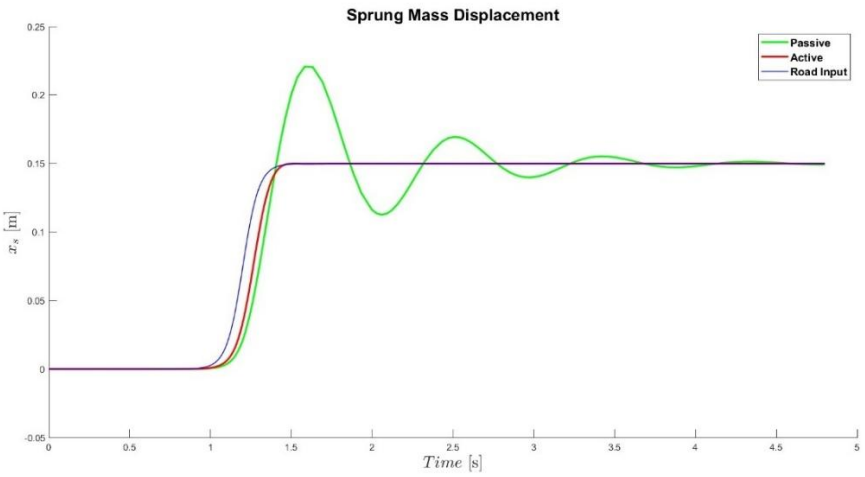


Figure 24 - Sprung Mass Displacement Plot for the Step Input

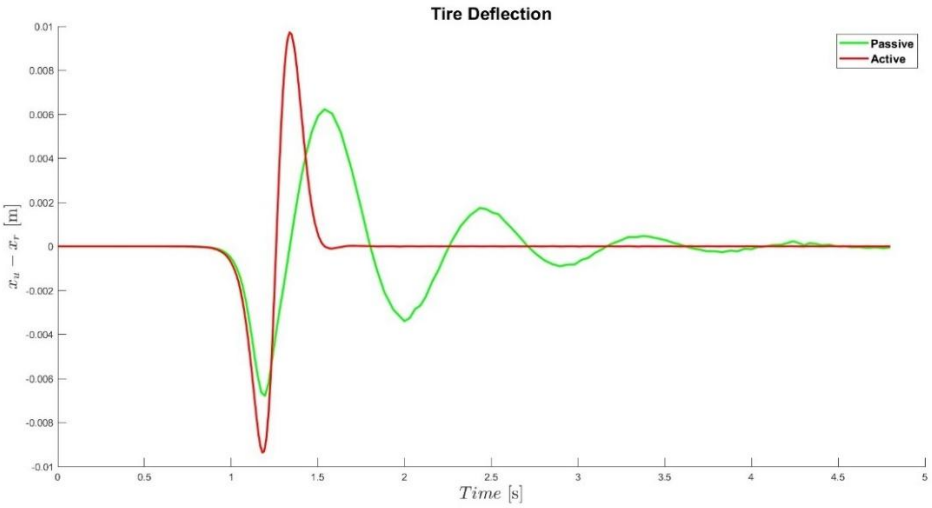


Figure 25 - Tire Deflection Plot for the Step Input

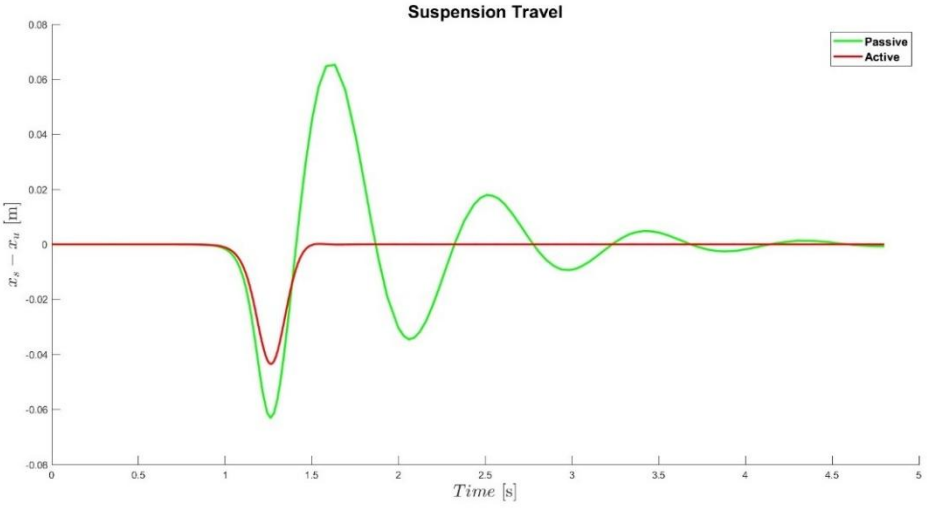


Figure 26 - Suspension Travel Plot for the Step Input

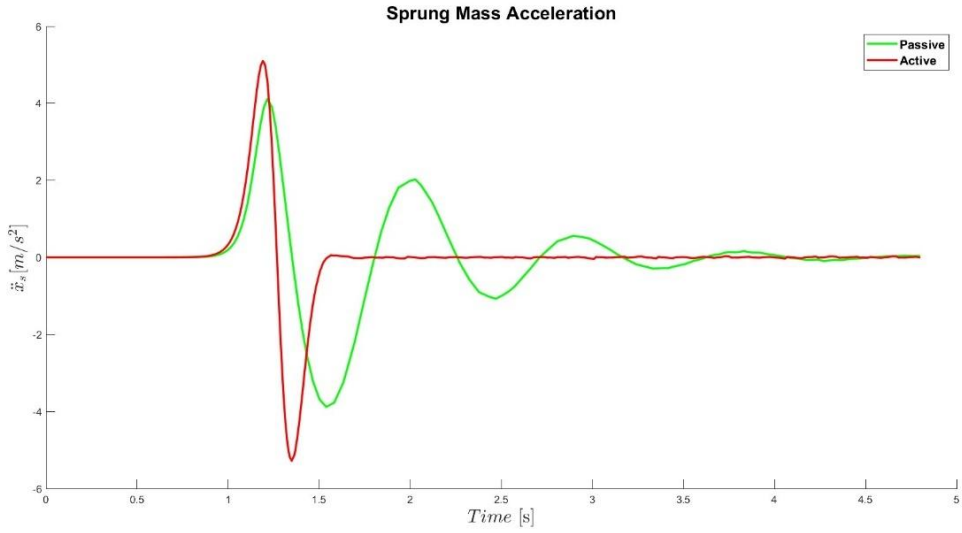


Figure 27 - Sprung Mass Acceleration Plot for the Step Input

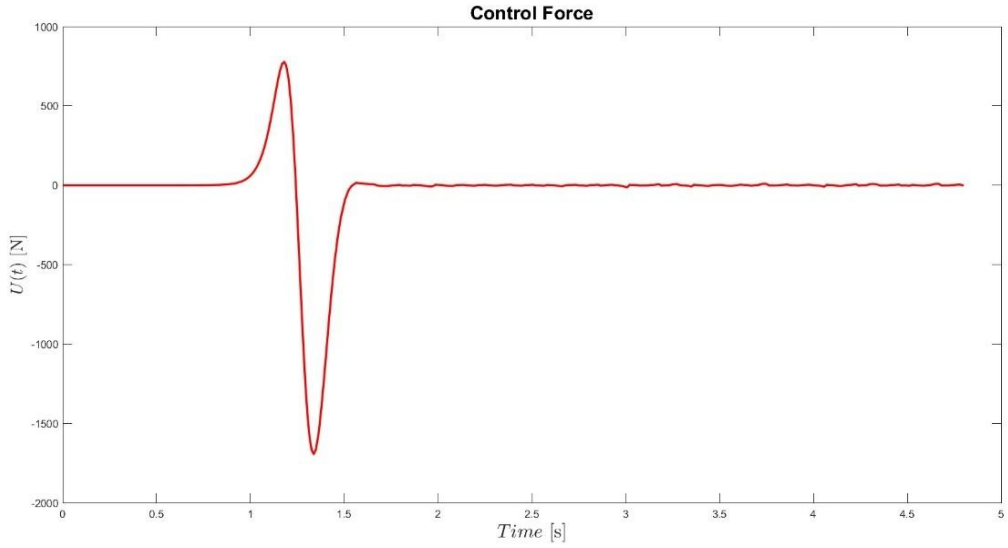


Figure 28 - Control Force Plot for the Step Input

In the results section, it is observed that all performance criteria are satisfied. Although the maximum absolute values of the outputs in the active suspension are generally higher than those in the passive suspension -except for suspension travel-, they remain well below the defined constraints. Moreover, despite the higher peak and trough values, the active suspension exhibits significantly faster settling behavior. The sprung mass displacement plot also shows that the active suspension performs excellently in minimizing oscillations. The control force reaches approximately 1770 N, which is within an acceptable and applicable range.

5.2 Case Study – 2

5.2.1 Simulation Setup

Car Speed = 15 km/h

Road Parameters

- Type: Bump Input
- Bump Start Location: 5 m
- Bump Finish Location: 6.2 m
- Bump Height: 0.15 m
- Road Length: 25 m

Q & R Matrices

- $Q = \begin{bmatrix} 6 \times 10^6 & 0 & 0 & 0 \\ 0 & 6 \times 10^5 & 0 & 0 \\ 0 & 0 & 5 \times 10^6 & 0 \\ 0 & 0 & 0 & 3 \times 10^3 \end{bmatrix}$
- $R = [5 \times 10^{-3}]$

5.2.2 Results and Discussion

RESULT

K = [22157.57 10827.23 -37091.22 -90.31]

OUTPUT	Absolute Maximum		Constraint
	ACTIVE	PASSIVE	
Tire Deflection [m]	0.0200	0.0196	0.0508
Suspension Travel [m]	0.122	0.119	0.127
Sprung Mass Acceleration [m/s²]	9.06	10.06	10.3

Absolute Maximum Control Force [N] = 1617.16

BACK

Figure 29 - Results for the Bump Input

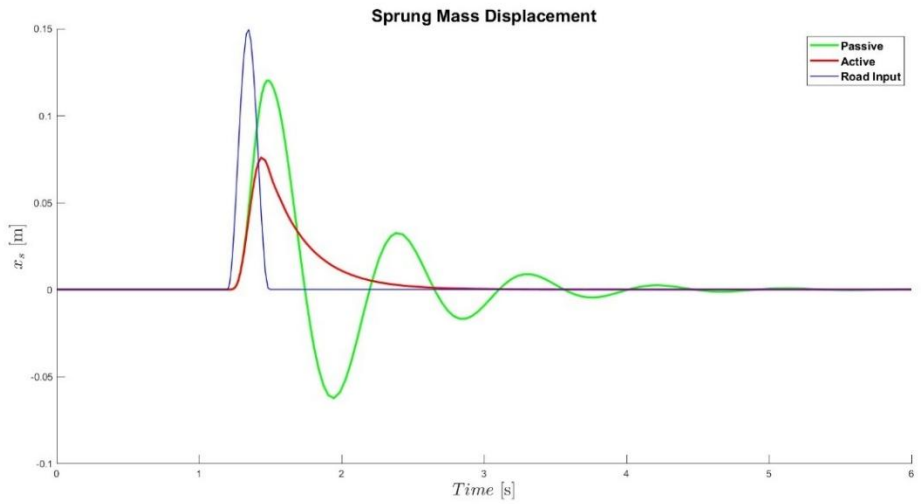


Figure 30 - Sprung Mass Displacement Plot for the Bump Input

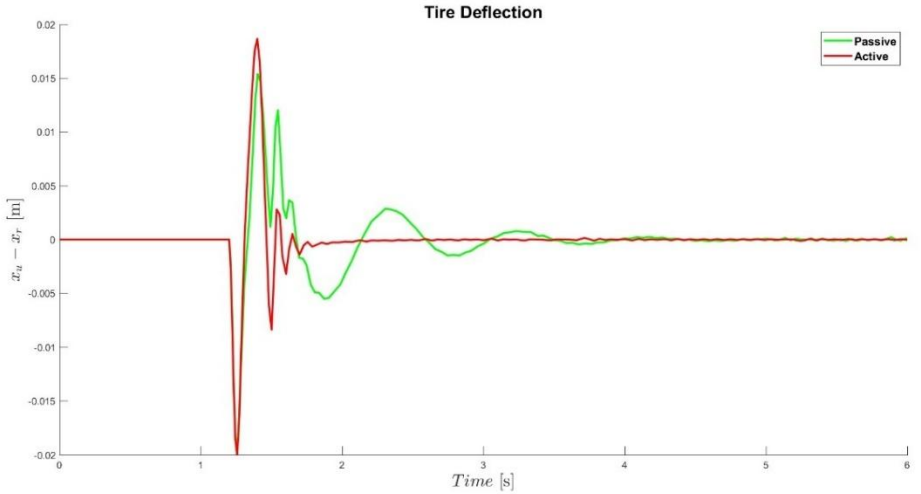


Figure 31 - Tire Deflection Plot for the Bump Input

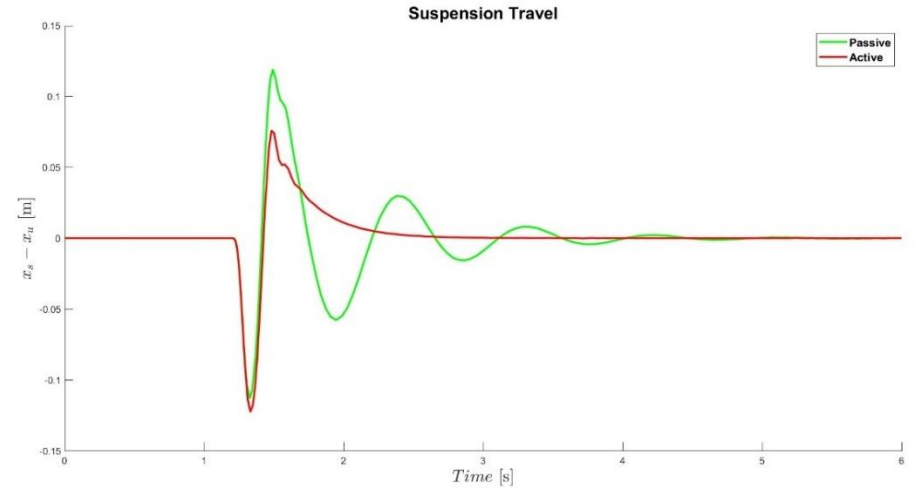


Figure 32 - Suspension Travel Plot for the Bump Input

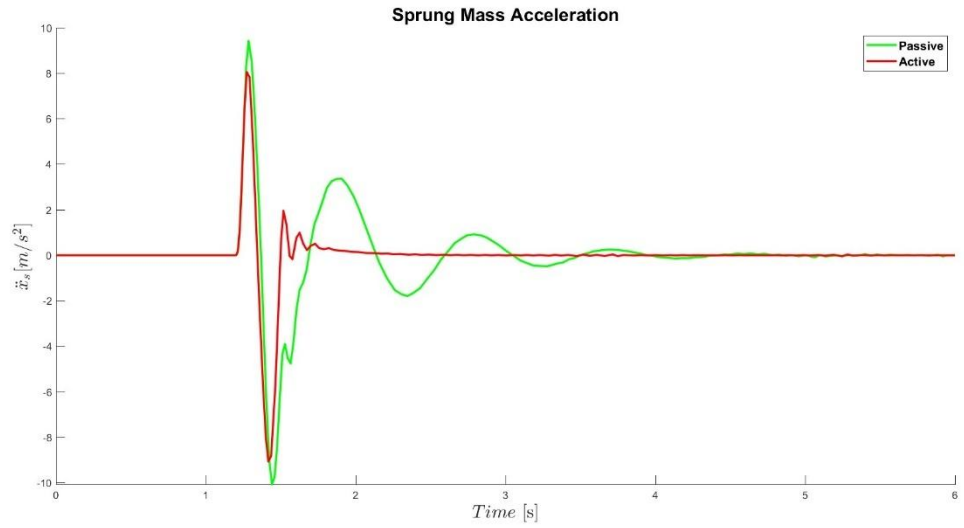


Figure 33 - Sprung Mass Acceleration Plot for the Bump Input

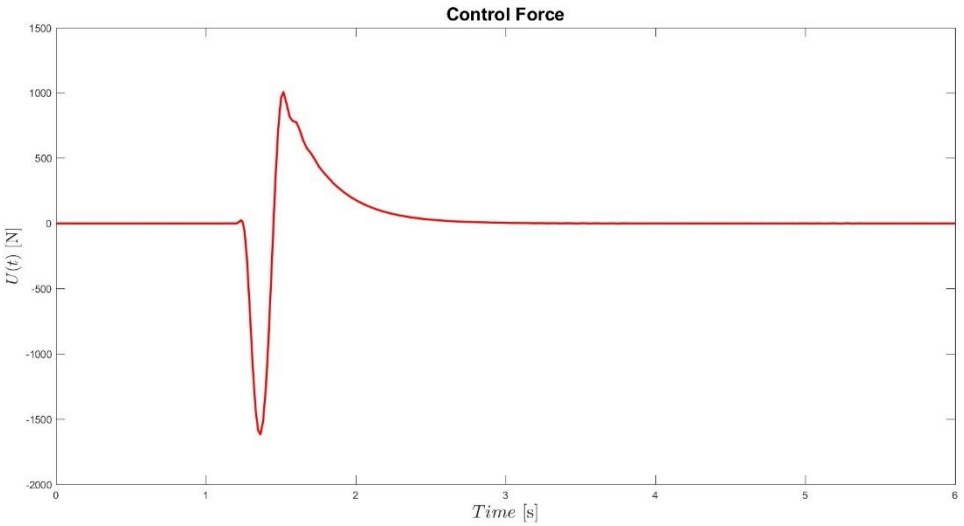


Figure 34 - Control Force Plot for the Bump Input

Under bump input conditions, the absolute maximum values in the active suspension are again higher compared to the passive system; however, they still remain within the limits. Importantly, the sprung mass acceleration -considered as the most critical parameter for ride comfort- is lower in the active suspension than in the passive one. The fast settling behavior continues to be a key strength of the active system, consistent with the step input results. Furthermore, the sprung mass displacement plot indicates a complete absence of oscillation throughout the simulation, which contributes positively to ride comfort. The control force in this case is around 1600 N and remains within a realistic and applicable range.

5.3 Case Study – 3

5.3.1 Simulation Setup

Car Speed = 15 km/h

Road Parameters

- Type: Pothole Input
- Pothole Start Location: 5 m
- Pothole Finish Location: 6.2 m
- Pothole Depth: 0.15 m
- Road Length: 20 m

Q & R Matrices

- $Q = \begin{bmatrix} 5 \times 10^6 & 0 & 0 & 0 \\ 0 & 2 \times 10^5 & 0 & 0 \\ 0 & 0 & 1 \times 10^6 & 0 \\ 0 & 0 & 0 & 1 \times 10^3 \end{bmatrix}$
- $R = [7 \times 10^{-4}]$

5.3.2 Results and Discussion

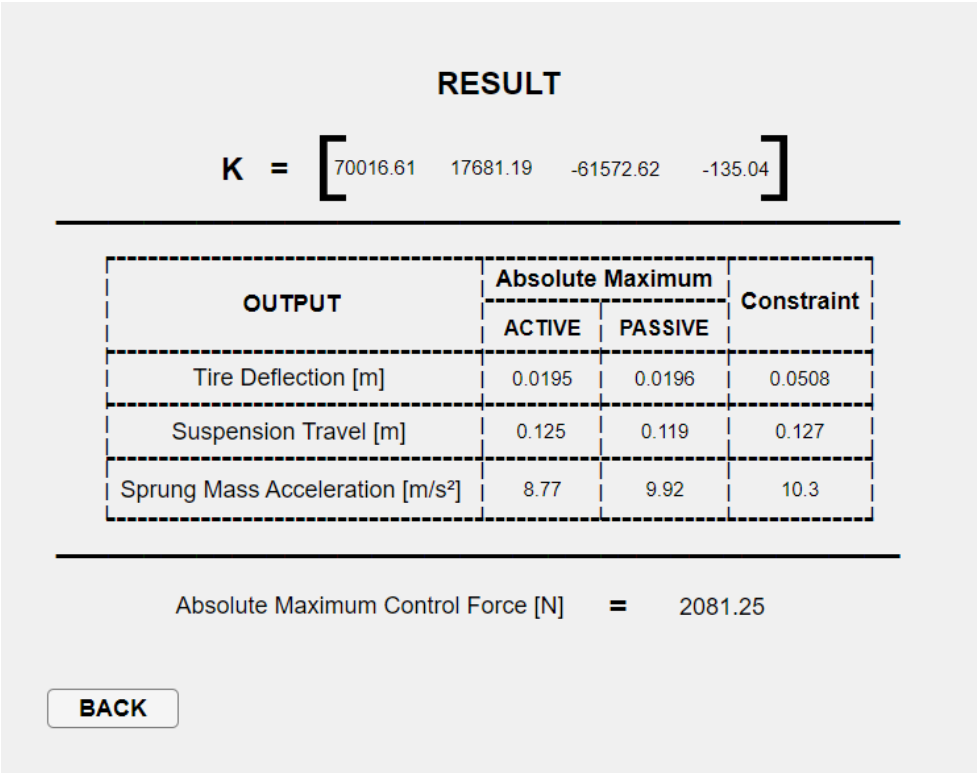


Figure 35 – Results for the Pothole Input

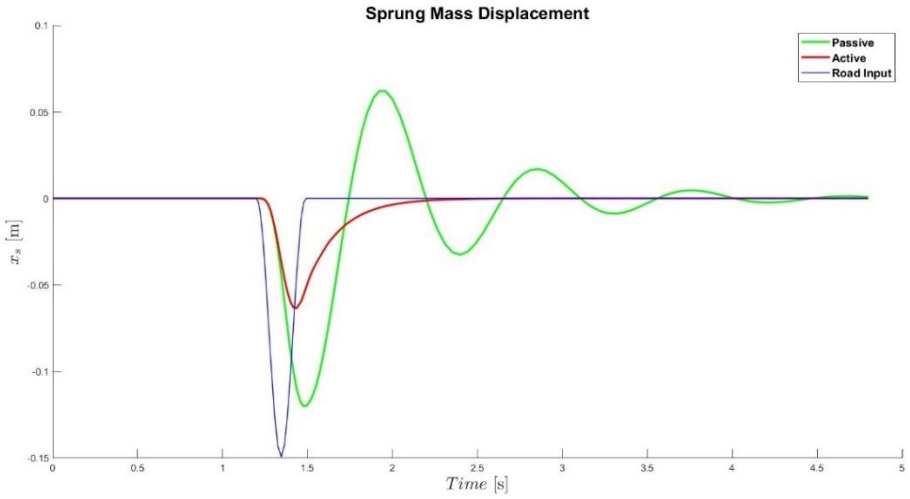


Figure 36 - Sprung Mass Displacement Plot for the Pothole Input

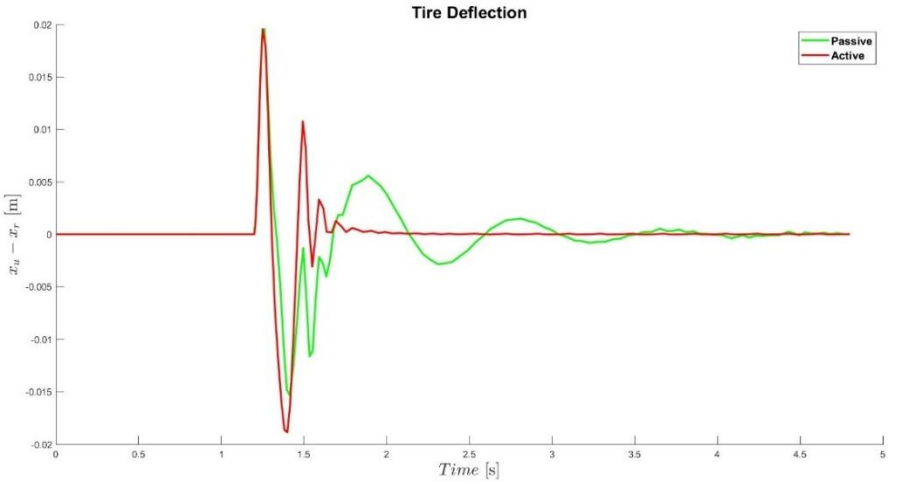


Figure 37 - Tire Deflection Plot for the Pothole Input

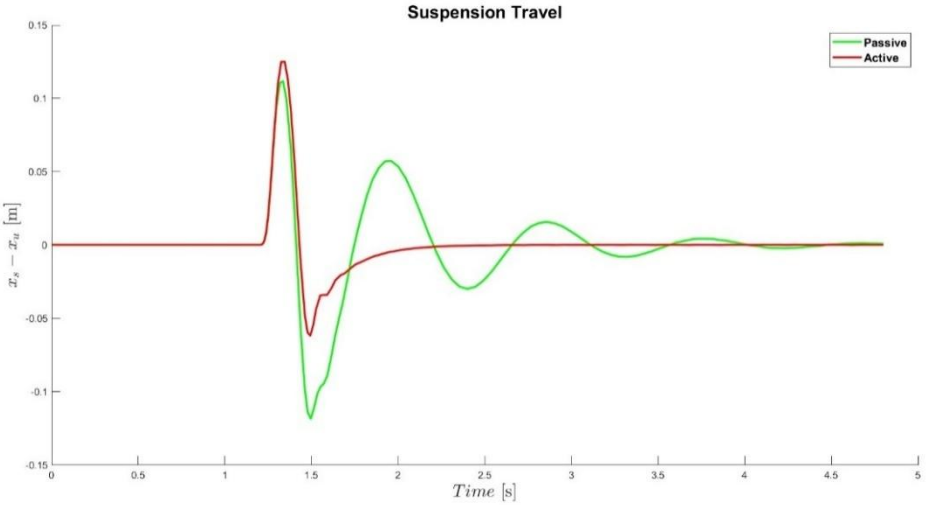


Figure 38 - Suspension Travel Plot for the Pothole Input

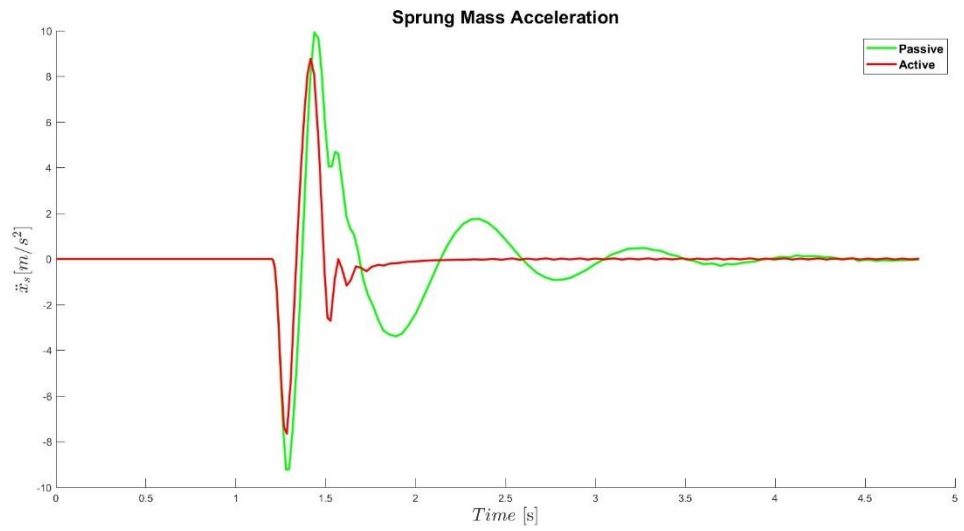


Figure 39 - Sprung Mass Acceleration Plot for the Pothole Input

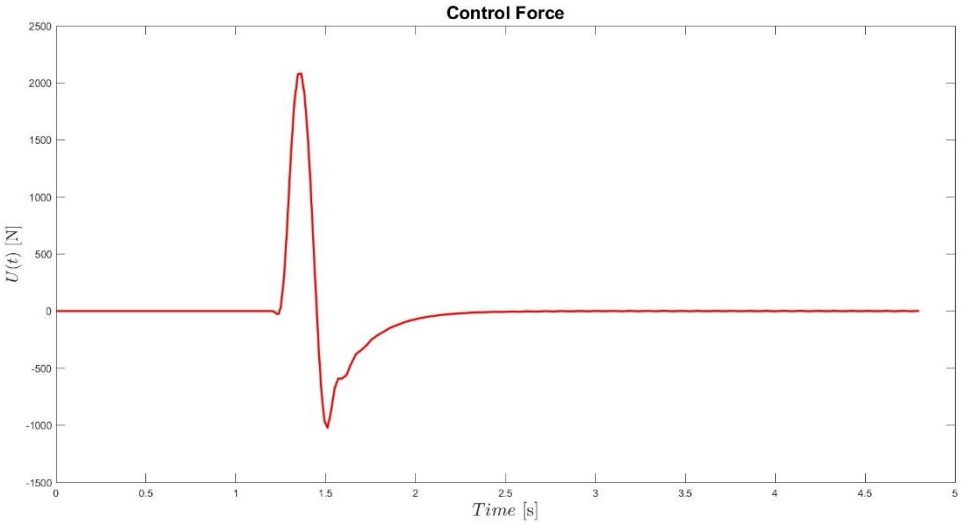


Figure 40 - Control Force Plot for the Pothole Input

Under pothole input conditions, the active suspension system clearly demonstrates a significant performance advantage. It effectively reduces both the vertical displacement and acceleration of the vehicle body, which are key factors in passenger comfort. In fact, the vertical movement transmitted to the occupants is approximately halved compared to the disturbance magnitude when using the active system. Additionally, the active suspension successfully prevents the car body from oscillating even after the wheels pass over the obstacle, which plays a crucial role in enhancing both comfort and safety. The required control force remains realistic, staying below 2000 N.

5.4 Case Study – 4

5.4.1 Simulation Setup

Car Speed = 10 km/h

Road Parameters

- Type: Sinusoidal Input
- Number of Waves: 8
- Amplitude: 0.07 m
- Road Length: 20 m

Q & R Matrices

- $Q = \begin{bmatrix} 3 \times 10^5 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 3 \times 10^5 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix}$
- $R = [1 \times 10^{-4}]$

5.4.2 Results and Discussion

RESULT

$$K = \begin{bmatrix} 41061.37 & 4696.69 & -25754.09 & -223.95 \end{bmatrix}$$

OUTPUT	Absolute Maximum		Constraint
	ACTIVE	PASSIVE	
Tire Deflection [m]	0.0138	0.0308	0.0508
Suspension Travel [m]	0.084	0.322	0.127
Sprung Mass Acceleration [m/s²]	7.43	18.72	10.3

Absolute Maximum Control Force [N] = 1984.54

BACK

Figure 41 - Results for the Sinusoidal Input

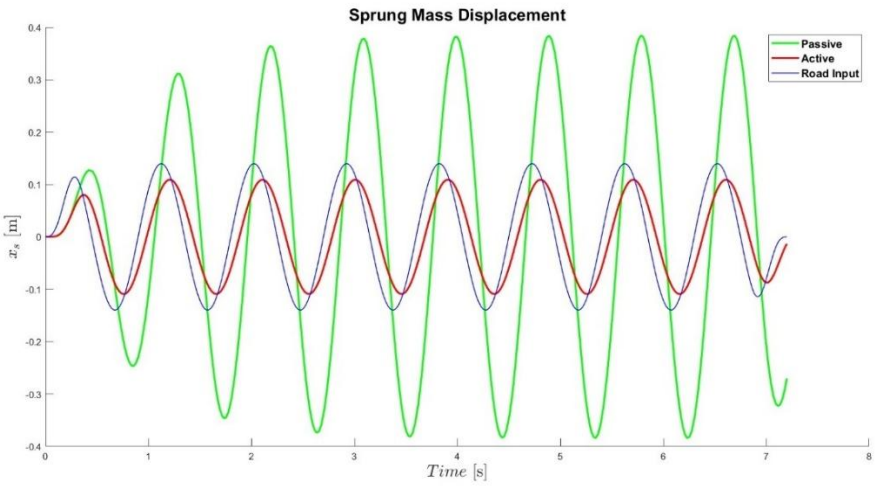


Figure 42 - Sprung Mass Displacement Plot for the Sinusoidal Input

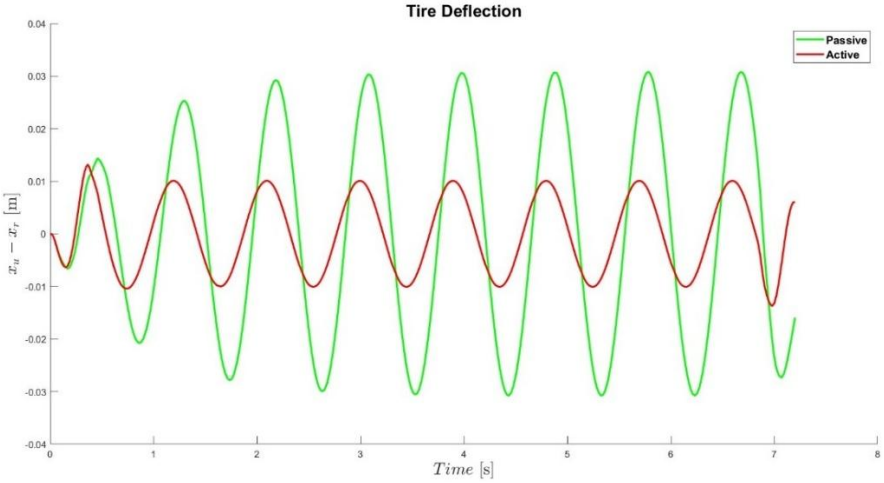


Figure 43 - Tire Deflection Plot for the Sinusoidal Input

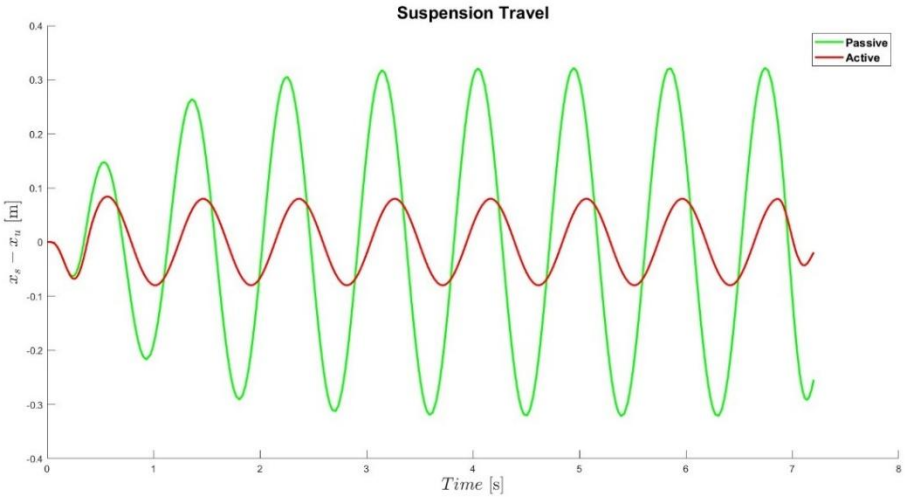


Figure 44 - Suspension Travel Plot for the Sinusoidal Input

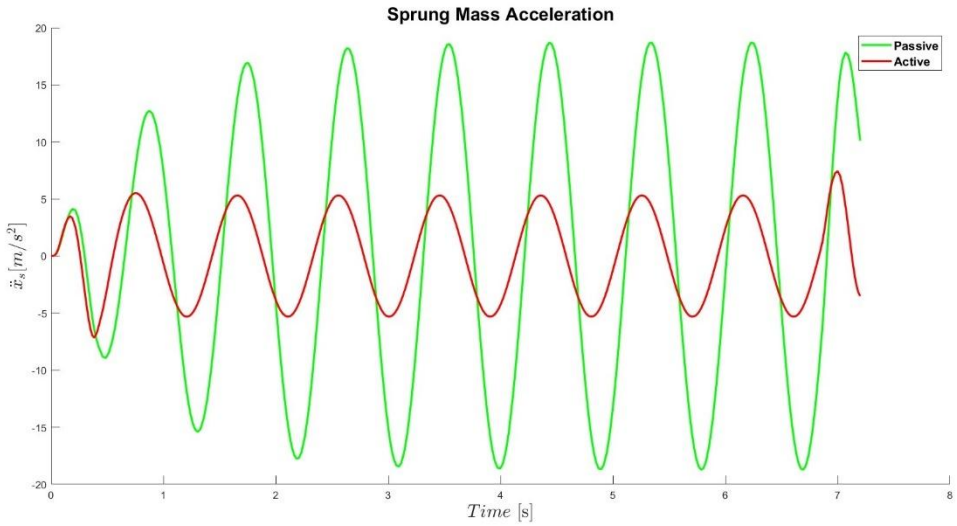


Figure 45 - Sprung Mass Acceleration Plot for the Sinusoidal Input

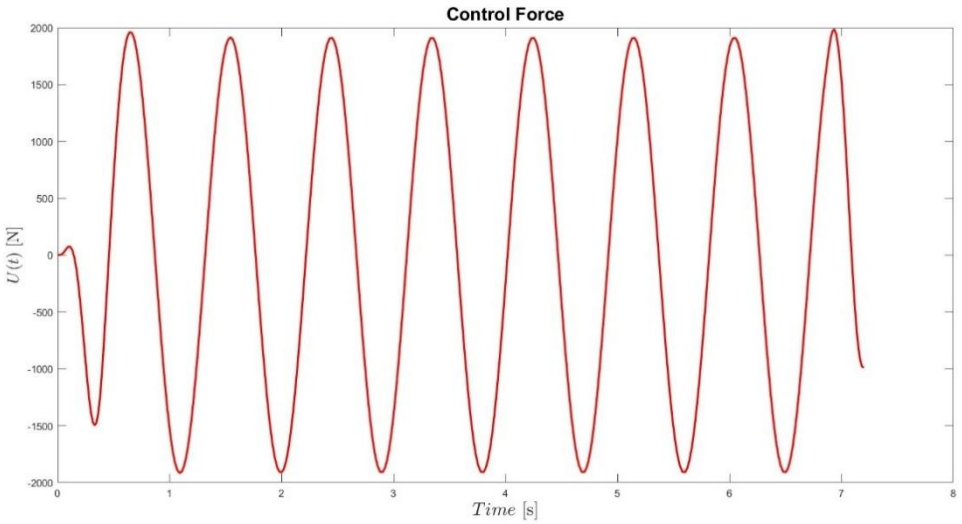


Figure 46 - Control Force Plot for the Sinusoidal Input

The case with sinusoidal road disturbances reflects common driving conditions encountered in everyday life. The active suspension system once again proves its robustness by effectively mitigating road-induced vibrations, significantly improving ride comfort and safety. The disturbances transmitted to the vehicle body are reduced to nearly one-third of those experienced with the passive suspension. Among all evaluated input scenarios, this case yields the best performance based on defined criteria. Moreover, the actuation force remains within a practical range, making the system suitable for real-world applications.

6 CONCLUSION

Suspension systems play a key role in vehicle dynamics by reducing the impact of road irregularities on passengers and improving the contact between tires and the road surface. A well-designed suspension system not only enhances ride comfort but also contributes to vehicle stability and safety. With increasing expectations for performance and comfort, more advanced solutions such as actively controlled suspension systems have become an important area of research.

In this study, an active suspension controller based on Linear-Quadratic Regulator (LQR) theory was designed and tested on a quarter-car model using MATLAB and SIMULINK environments. The goal was to reduce body vibrations, improve comfort, and maintain safe values for suspension travel and tire forces. The model was initially simulated under two basic road disturbances (step and bump inputs) without considering vehicle speed as a factor. These early results demonstrated the advantages of the active suspension system over its passive counterpart, especially in terms of faster settling time and reduced acceleration.

However, it was later realized that real-world conditions involve a wider variety of road profiles and that vehicle speed has a direct impact on suspension dynamics. To address these limitations, a MATLAB App Designer-based tool was developed. This application not only expanded the road input library to include pothole and sinusoidal disturbances in addition to step and bump, but also introduced vehicle speed as a user-defined parameter that affects simulation behavior. The interface allows users to customize the road input parameters, initial conditions, Q and R matrices, and observe system performance through plots, comparative tables, and animations. One notable feature is the ability to generate and save the simulation as a video. This not only facilitates effective presentation and reporting of the results, but also makes the tool valuable for educational purposes by providing a clear visualization of the suspension behavior under various conditions.

Using this tool, additional case studies were conducted to evaluate system performance under different types of road inputs. Among the test cases, the most significant improvement in ride comfort was observed under the sinusoidal road input, which is the only continuous input existing in the tool, where the active suspension notably reduced oscillations and transmitted vibrations compared to the passive setup.

Although the quarter-car model simplifies the real system by considering only vertical motion, it offers a manageable and computationally efficient framework for initial controller design and analysis. Nevertheless, the model is based on idealized assumptions such as linear dynamics and full-state feedback which may differ from real-world conditions. For future work, the controller design can be improved by incorporating integral action through a Linear-Quadratic Integral (LQI) approach. This would help eliminate steady-state errors that may arise under constant road disturbances like step inputs, further enhancing ride quality and system robustness. Additionally, the use of simulation tools and virtual testing environments, such as this developed application, offers a significant cost advantage by

reducing the need for physical prototypes and experimental setups in the early stages of research.

In conclusion, this project successfully demonstrates the feasibility and effectiveness of LQR-based active suspension systems, both through theoretical modeling and interactive simulation. The developed application provides a flexible and visual platform for analyzing suspension dynamics and supports further research and educational exploration in the field of vehicle control systems.

7 REFERENCES

[1] BOULAARAS, Z., AOUICHE, A., & CHAFAA, K. Robust Control of a Quarter Car Active Suspension System Using LQR Approach.

[2] Simionescu, P. A., & Norton, R. L. (2023). On the History of Early Automobile Suspension Systems. Advances in Mechanism and Machine Science, 1012–1022.

[3] Jazar, R. N. (2008). Vehicle Dynamics: Theory and Application. Springer US.

[4] Omar, M., El-kassaby, M. M., & Abdelghaffar, W. (2017). A universal suspension test rig for electrohydraulic active and passive automotive suspension system. *Alexandria Engineering Journal*, *56*(4), 359–370.

[5] R., N., T., R. B., Sayyad, J., & Biswas, S. (2024). A comparative analysis and evaluating active suspension system controllers with Coppeliasim. *2024 6th International Conference on Energy, Power and Environment (ICEPE)*, 1–6.

[6] Jiregna, I., & Sirata, G. (2020). A review of the vehicle suspension system. *Journal of Mechanical and Energy Engineering*, *4*(2).

[7] Hansen, C.H. and Snyder, S.D., Active Control of Noise and Vibration, Chapter 12, Chapman and Hall, ISBN 0 419 19390 1, 1997.

[8] Rajamani, R. (2012). Vehicle Dynamics and Control. Springer US.

[9] Koch, Guido & Kloiber, Tobias. (2014). Driving State Adaptive Control of an Active Vehicle Suspension System. Control Systems Technology, IEEE Transactions on. 22. 44-57. 10.1109/TCST.2013.2240455.

[10] TURAN, A., & AGGÜMÜŞ, H. (2022). Optimal PID Controller Design Based on Proportional Gain for Quarter Vehicle Model. European Journal of Science and Technology.

[11] Alp Arslan, T., Aysal, F. E., Çelik, İ., Bayrakçeken, H., & Öztürk, T. N. (2022). Quarter Car Active Suspension System Control Using Fuzzy Controller. Engineering Perspective, 2(4), 33–39.

[12] Nguyen Tuan Anh, N. T. A. (2020). Control an Active Suspension System by Using PID and LQR Controller. International Journal of Mechanical and Production Engineering Research and Development, 10(3), 7003–7012.

[13] Mahala, K. M., Gadkari, P., & Deb, A. (2009). Mathematical models for designing vehicles for ride comfort. In *ICORD 09: Proceedings of the 2nd International Conference on Research into Design, Bangalore, India 07.-09.01. 2009*.

[14] Kaleemullah, M., Faris, W. F., & Hasbullah, F. (2011). Design of robust H_∞, fuzzy and LQR controller for active suspension of a quarter car model. 2011 4th International Conference on Mechatronics (ICOM), 1–6.

[15] Zhao, W., & Gu, L. (2023). Hybrid Particle Swarm Optimization Genetic LQR Controller for Active Suspension. Applied Sciences, 13(14), 8204.

[16] Enders, E., Burkhard, G., & Munzinger, N. (2020). Analysis of the Influence of Suspension Actuator Limitations on Ride Comfort in Passenger Cars Using Model Predictive Control. Actuators, 9(3), 77. https://doi.org/10.3390/act9030077

[17] Qiu, H., Al-Nussairi, A. K. J., Chevinli, Z. S., Singh Sawaran Singh, N., Chyad, M. H., Yu, J., & Maesoumi, M. (2025). Integrating digital twins with neural networks for adaptive control of automotive suspension systems. Scientific Reports, 15(1). https://doi.org/10.1038/s41598-025-91243-1

[18] International Organization for Standardization. (1997). *Mechanical vibration and shock — Evaluation of human exposure to whole-body vibration — Part 1: General requirements* (Standard No. ISO 2631-1:1997, 2nd ed.).

[19] Baumal, A. E., McPhee, J. J., & Calamai, P. H. (1998). Application of genetic algorithms to the design optimization of an active vehicle suspension system. Computer Methods in Applied Mechanics and Engineering, 163(1–4), 87–94. https://doi.org/10.1016/s0045-7825(98)00004-8

[20] Guarnaccia, L., Bevilacqua, R., & Pastorelli, S. P. (2016). Suboptimal LQR-based spacecraft full motion control: Theory and experimentation. Acta Astronautica, 122, 114–136. https://doi.org/10.1016/j.actaastro.2016.01.016

[21] Ye, H., & Zheng, L. (2019). Comparative study of semi-active suspension based on LQR control and H2/H∞ multi-objective control. 2019 Chinese Automation Congress (CAC), 3901–3906. https://doi.org/10.1109/cac48633.2019.8996771

[22] Amman, H. M., & Neudecker, H. (1997). Numerical solutions of the algebraic matrix Riccati equation. *Journal of Economic Dynamics and Control*, *21*(2-3), 363-369.

[23] Anjali, B. S., Vivek, A., & Nandagopal, J. L. (2016). Simulation and Analysis of Integral LQR Controller for Inner Control Loop Design of a Fixed Wing Micro Aerial Vehicle (MAV). Procedia Technology, 25, 76–83. https://doi.org/10.1016/j.protcy.2016.08.083

[24] References for the images on the page:

- https://beaudaniels.com/car-suspensions
- https://www.hrsprings.com.au/hr-suspension-coilovers/
- https://mozaracing.com/blog/2024/02/02/mozas-innovations-in-direct-drive-and-vehicle-suspension-technology/
- https://www.import-car.com/abc-suspension-rocking-motion-when-stationary-with-mode-d-engaged-and-engine-running/

[25] *ME4051 Automotive Engineering Lecture Notes*. [Unpublished lecture notes]. Marmara University, 2025.

[26] James Allison (2025). Animation of a Quarter-Car Automotive Suspension (https://www.mathworks.com/matlabcentral/fileexchange/35478-animation-of-a-quarter-car-automotive-suspension), MATLAB Central File Exchange. Retrieved June 12, 2025.

8 APPENDIX

8.1 Code for Preliminary Study

```
function xdotdot = fcn(xdot,x,x_r)

ms = 300;
mu = 50;
ks = 16000;
cs = 1000;
kt = 190000;

M = [ms 0;0 mu];
C = [cs -cs;-cs cs];
K = [ks -ks;-ks kt+ks];
F = [0;kt*x_r];

xdotdot = inv(M)*(F - C*xdot - K*x);
```

8.2 Code for Step Input

In the code below; `t_step` and `h` expresses the step time and the step height, respectively. By changing the variable `k`, the sharpness of the input can be arranged. The larger the `k`, the greater the input sharpness. Formulation in *3.5.1*, uses 1 for `t_step`, 0.1 for `h`, and 25 for `k`.

```
t = linspace(0, 10, 1000)';
u_r = h./(1 + exp(-k*(t - t_step)));
```

8.3 Code for Bump Input

In the code below; `h`, `t_rise`, and `t_fall` corresponds to the height, start time, and the finish time of the bump input, respectively. Formulation given in *3.5.2*, uses 0.1, 1, and 3 for the `h`, `t_rise`, and `t_fall`, respectively.

```
t = linspace(0, 10, 1000)';
u_r = zeros(size(t));

for i = 1:length(t)
    if t(i) >= t_rise && t(i) <= t_fall
        u_r(i) = (h / 2) * (1 - cos(2 * pi * (t(i) - t_rise) / (t_fall -
t_rise)));
    else
        u_r(i) = 0;
    end
end
```

8.4 Code for Simulation

```
clear;
clc;

% PARAMETERS

ms = 300;      % Sprung Mass [kg]
mu = 50;       % Unsprung Mass [kg]
ks = 16000;    % Suspension Stiffness [N/m]
```

```
c = 1000;     % Suspension Damping [N.s/m]
kt = 190000;  % Tire Stiffness [N/m]

% STATE SPACE MODEL

A = [   0      1         0       0   ;
     -ks/ms  -c/ms     ks/ms    c/ms ;
        0      0         0       1   ;
      ks/mu   c/mu -(ks+kt)/mu -c/mu];

B = [   0       0   ;
        0      1/ms ;
        0       0   ;
      kt/mu   -1/mu];

C = [   1     0     0    0    ;
        1     0    -1    0    ;
     -ks/ms -c/ms ks/ms c/ms  ;
        0     0     1    0   ];

D = [  0   0   ;
       0   0   ;
       0  1/ms ;
      -1   0  ];

sys = ss(A,B,C,D);

% INITIAL CONDITIONS

x0 = [0; 0; 0; 0];

% Q and R MATRICES

Q = [10900   0    0    0 ;
       0    10    0    0 ;
       0     0  10000  0 ;
       0     0    0   10];

R = 0.0001;

K_LQR = lqr(A,B(:,2),Q,R);

% COMPARISON BETWEEN PASSIVE and ACTIVE SYSTEMS

%% STEP INPUT

close all;

t = linspace(0, 10, 1000)';
k = 25;
t_step = 1;
h = 0.1;
u_r = h./(1 + exp(-k*(t - t_step)));

K = zeros(size(K_LQR));
passive = sim('eng_project');

K = K_LQR;
active = sim('eng_project');
```

```
figure(1)
hold on
plot(passive.y.Time, passive.y.Data(:,1), 'g', 'LineWidth',4);
plot(active.y.Time, active.y.Data(:,1), 'r', 'LineWidth',4);
plot(active.y.Time, active.input.Data, 'b', 'LineWidth',2);
title('Sprung Mass Displacement','FontSize',40,'FontWeight','Bold');
xlabel('$Time$ [s]','FontSize',30,'Interpreter','latex');
ylabel('$x_s$ [m]','FontSize',30,'FontWeight','Bold','Interpreter','latex');
legend('Passive','Active','Road Input','FontSize',30,'FontWeight','Bold');
set(gca, 'FontSize', 25)

figure(2)
hold on
plot(passive.y.Time, passive.y.Data(:,2), 'g', 'LineWidth',4);
plot(active.y.Time, active.y.Data(:,2), 'r', 'LineWidth',4);
title('Suspension Travel','FontSize',40,'FontWeight','Bold');
xlabel('$Time$ [s]','FontSize',30,'Interpreter', 'latex');
ylabel('$x_s - x_u$ [m]','FontSize',30,'Interpreter','latex');
legend('Passive','Active','FontSize',30,'FontWeight','Bold');
set(gca, 'FontSize', 25)

figure(3)
hold on
plot(passive.y.Time, passive.y.Data(:,3), 'g', 'LineWidth',4);
plot(active.y.Time, active.y.Data(:,3), 'r', 'LineWidth',4);
title('Sprung Mass Acceleration','FontSize',40,'FontWeight','Bold');
xlabel('$Time$ [s]','FontSize',30,'Interpreter','latex');
ylabel('$\ddot{x}_{s} [m/s^2]$','FontSize',30,'Interpreter', 'latex');
legend('Passive','Active','FontSize',30,'FontWeight','Bold');
set(gca, 'FontSize', 25)

figure(4)
hold on
plot(active.u.Time, active.u.Data, 'r', 'LineWidth',4);
title('Control Force','FontSize',40,'FontWeight','Bold');
xlabel('$Time$ [s]','FontSize',30,'Interpreter', 'latex');
ylabel('$U(t)$ [N]','FontSize',30,'Interpreter','latex');
set(gca, 'FontSize', 25)

figure(5)
hold on
plot(passive.y.Time, passive.y.Data(:,4), 'g', 'LineWidth',4);
plot(active.y.Time, active.y.Data(:,4), 'r', 'LineWidth',4);
title('Tire Deflection','FontSize',40,'FontWeight','Bold');
xlabel('$Time$ [s]','FontSize',30,'Interpreter','latex');
ylabel('$x_u - x_r$ [m]','FontSize',30,'Interpreter', 'latex');
legend('Passive','Active','FontSize',30,'FontWeight','Bold');
set(gca, 'FontSize', 25)

%% BUMP INPUT

clf;
h = 0.1;      % Maximum bump height (m)
t_rise = 1;   % Time when the bump starts to rise (s)
t_fall = 3;   % Time when the bump returns to ground level (s)

t = linspace(0, 10, 1000)';
```

```
u_r = zeros(size(t));

for i = 1:length(t)
    if t(i) >= t_rise && t(i) <= t_fall
        u_r(i) = (h / 2) * (1 - cos(2 * pi * (t(i) - t_rise) / (t_fall -
t_rise)));
    else
        u_r(i) = 0;
    end
end

K = zeros(size(K_LQR));
passive = sim('eng_project');

K = K_LQR;
active = sim('eng_project');


figure(1)
hold on
plot(passive.y.Time, passive.y.Data(:,1), 'g', 'LineWidth',4);
plot(active.y.Time, active.y.Data(:,1), 'r', 'LineWidth',4);
plot(active.y.Time, active.input.Data, 'b', 'LineWidth',2);
title('Sprung Mass Displacement','FontSize',40,'FontWeight','Bold');
xlabel('$Time$ [s]','FontSize',30,'Interpreter','latex');
ylabel('$x_s$ [m]','FontSize',30,'FontWeight','Bold','Interpreter','latex');
legend('Passive','Active','Road Input','FontSize',30,'FontWeight','Bold');
set(gca, 'FontSize', 25)

figure(2)
hold on
plot(passive.y.Time, passive.y.Data(:,2), 'g', 'LineWidth',4);
plot(active.y.Time, active.y.Data(:,2), 'r', 'LineWidth',4);
title('Suspension Travel','FontSize',40,'FontWeight','Bold');
xlabel('$Time$ [s]','FontSize',30,'Interpreter', 'latex');
ylabel('$x_s - x_u$ [m]','FontSize',30,'Interpreter','latex');
legend('Passive','Active','FontSize',30,'FontWeight','Bold');
set(gca, 'FontSize', 25)

figure(3)
hold on
plot(passive.y.Time, passive.y.Data(:,3), 'g', 'LineWidth',4);
plot(active.y.Time, active.y.Data(:,3), 'r', 'LineWidth',4);
title('Sprung Mass Acceleration','FontSize',40,'FontWeight','Bold');
xlabel('$Time$ [s]','FontSize',30,'Interpreter','latex');
ylabel('$\ddot{x}_{s} [m/s^2]$','FontSize',30,'Interpreter', 'latex');
legend('Passive','Active','FontSize',30,'FontWeight','Bold');
set(gca, 'FontSize', 25)

figure(4)
hold on
plot(active.u.Time, active.u.Data, 'r', 'LineWidth',4);
title('Control Force','FontSize',40,'FontWeight','Bold');
xlabel('$Time$ [s]','FontSize',30,'Interpreter', 'latex');
ylabel('$U(t)$ [N]','FontSize',30,'Interpreter','latex');
set(gca, 'FontSize', 25)

figure(5)
hold on
```

```
plot(passive.y.Time, passive.y.Data(:,4), 'g', 'LineWidth',4);
plot(active.y.Time, active.y.Data(:,4), 'r', 'LineWidth',4);
title('Tire Deflection','FontSize',40,'FontWeight','Bold');
xlabel('$Time$ [s]','FontSize',30,'Interpreter','latex');
ylabel('$x_u - x_r$ [m]','FontSize',30,'Interpreter', 'latex');
legend('Passive','Active','FontSize',30,'FontWeight','Bold');
set(gca, 'FontSize', 25)
```

8.5 Codes in the App Designer Based – Tool

8.5.1 Functions for Generating the Animation [26]

```
function plotsuspa(app, x_a, road_x_a, road_z_a, curr_x_a, umf_a)

    cla(app.activeAxes); hold(app.activeAxes, 'on');

    z0_a = x_a(1); % road elevation
    z1_a = x_a(2); % unsprung mass cm deviation
    z2_a = x_a(3); % sprung mass cm deviation
    t_a = x_a(4);  % current time

    % ========================
    % Geometric suspension parameters
    % ========================
    h1_a = 0.35;    % resting position of unsprung cm
    h2_a = 1.1;     % resting position of sprung cm
    h3_a = 0.2;     % height of unsprung mass block
    h4_a = 0.35;    % height of sprung mass block
    w1_a = 0.6;     % width of unsprung mass block
    w2_a = 0.8;     % width of sprung mass block
    w3_a = 0.1;     % width of tire spring
    w4_a = 0.15;    % width of suspension spring
    w5_a = 0.5;     % spring/damper spacing

    % Plotting parameter
    fw_a = 0.7;     % half of figure width

    % ========================
    % Preliminary calculations:
    % ========================
    x0_r_a = z0_a;                  % tire spring base position
    x0_s_a = h1_a + z1_a + h3_a/2;     % suspension spring base position
    x0_t_a = h1_a + z1_a - h3_a/2;     % unsprung mass block base position
    x0_b_a = h2_a + z2_a - h4_a/2;     % sprung mass block base position
    L1_a = x0_t_a - x0_r_a;          % tire spring length
    L2_a = x0_b_a - x0_s_a;          % suspension spring length


    if ~isempty(road_x_a) && ~isempty(road_z_a) && length(road_x_a) ==
length(road_z_a)
        x_start_a = max([curr_x_a - fw_a, min(road_x_a)]);
        x_end_a = min([curr_x_a + fw_a, max(road_x_a)]);
        mask_a = (road_x_a >= x_start_a) & (road_x_a <= x_end_a);
        if any(mask_a) && sum(mask_a) <= length(road_z_a)
            plot(app.activeAxes, road_x_a(mask_a) - curr_x_a,
road_z_a(mask_a) * umf_a, 'k-', 'LineWidth', 2);
        end
        xlim(app.activeAxes, [-fw_a, fw_a]);
    end
```

```
            axis(app.activeAxes, [-fw_a fw_a -0.25 1.5]);

            text(app.activeAxes, 2*fw_a/3, 1.4, [num2str(t_a, '%2.1f') ' s'],
'FontWeight', 'bold', 'FontSize', 12);

            % =========================
            % Plot unsprung mass block
            % =========================
            x0t_a = [0; x0_t_a];
            x1t_a = x0t_a + [-w1_a/2; 0];
            x2t_a = x0t_a + [-w1_a/2; h3_a];
            x3t_a = x0t_a + [w1_a/2; h3_a];
            x4t_a = x0t_a + [w1_a/2; 0];
            fill(app.activeAxes, [x1t_a(1) x2t_a(1) x3t_a(1) x4t_a(1)], [x1t_a(2)
x2t_a(2) x3t_a(2) x4t_a(2)], [1 0 0]);

            % =========================
            % Plot sprung mass block
            % =========================
            x0b_a = [0; x0_b_a];
            x1b_a = x0b_a + [-w2_a/2; 0];
            x2b_a = x0b_a + [-w2_a/2; h4_a];
            x3b_a = x0b_a + [w2_a/2; h4_a];
            x4b_a = x0b_a + [w2_a/2; 0];
            fill(app.activeAxes, [x1b_a(1) x2b_a(1) x3b_a(1) x4b_a(1)], [x1b_a(2)
x2b_a(2) x3b_a(2) x4b_a(2)], [1 0 0]);

            % =========================
            % Plot tire spring
            % =========================
            x0r_a = [0; x0_r_a];
            plot(app.activeAxes, x0r_a(1), x0r_a(2), 'ko', 'MarkerSize', 10,
'MarkerFaceColor', 'k');

            u_a = L1_a/9;
            x1r_a = x0r_a + [0; u_a];
            x2r_a = x0r_a + [-w3_a/2; 3/2*u_a];
            x3r_a = x2r_a + [w3_a; u_a];
            x4r_a = x3r_a + [-w3_a; u_a];
            x5r_a = x4r_a + [w3_a; u_a];
            x6r_a = x5r_a + [-w3_a; u_a];
            x7r_a = x6r_a + [w3_a; u_a];
            x8r_a = x7r_a + [-w3_a; u_a];
            x9r_a = x8r_a + [w3_a/2; u_a/2];
            x10r_a = x9r_a + [0; u_a];

            plot(app.activeAxes, [x0r_a(1) x1r_a(1) x2r_a(1) x3r_a(1) x4r_a(1)
x5r_a(1) x6r_a(1) x7r_a(1) x8r_a(1) x9r_a(1) x10r_a(1)], ...
                 [x0r_a(2) x1r_a(2) x2r_a(2) x3r_a(2) x4r_a(2) x5r_a(2) x6r_a(2)
x7r_a(2) x8r_a(2) x9r_a(2) x10r_a(2)], ...
                 'k-', 'LineWidth', 2);

            % =========================
            % Plot suspension spring
            % =========================
            x0s_a = [-4*w5_a/9; x0_s_a];
            u_a = L2_a/9;
            x1s_a = x0s_a + [0; u_a];
```

```
            x2s_a = x0s_a + [-w4_a/2; 3/2*u_a];
            x3s_a = x2s_a + [w4_a; u_a];
            x4s_a = x3s_a + [-w4_a; u_a];
            x5s_a = x4s_a + [w4_a; u_a];
            x6s_a = x5s_a + [-w4_a; u_a];
            x7s_a = x6s_a + [w4_a; u_a];
            x8s_a = x7s_a + [-w4_a; u_a];
            x9s_a = x8s_a + [w4_a/2; u_a/2];
            x10s_a = x9s_a + [0; u_a];

            plot(app.activeAxes, [x0s_a(1) x1s_a(1) x2s_a(1) x3s_a(1) x4s_a(1)
x5s_a(1) x6s_a(1) x7s_a(1) x8s_a(1) x9s_a(1) x10s_a(1)], ...
                 [x0s_a(2) x1s_a(2) x2s_a(2) x3s_a(2) x4s_a(2) x5s_a(2) x6s_a(2)
x7s_a(2) x8s_a(2) x9s_a(2) x10s_a(2)], ...
                 'k-', 'LineWidth', 3);

            % Number of points
            n_ac = 1000;

            % Center of the actuator
            c_ac = [4*w5_a/9 (x0_t_a+h3_a+x0_b_a)/2];

            % Radius
            r_ac = 0.08;

            % Running variable
            t_ac = linspace(0,2*pi,n_ac);

            x_ac = c_ac(1) + 1.15*r_ac*sin(t_ac);
            y_ac = c_ac(2) + r_ac*cos(t_ac);

            % Plot Actuator
            plot(app.activeAxes, x_ac,y_ac, 'k-', 'LineWidth', 2);
            plot(app.activeAxes, [4*w5_a/9 4*w5_a/9], [x0_b_a
((x0_t_a+h3_a+x0_b_a)/2)+r_ac], 'k-', 'LineWidth', 2);
            plot(app.activeAxes, [4*w5_a/9 4*w5_a/9], [c_ac(2)-r_ac x0_t_a+h3_a],
'k-', 'LineWidth', 2);

            text(app.activeAxes, c_ac(1)-0.05, c_ac(2), 'U(t)', 'FontSize', 16)

            % =========================
            % Plot suspension damper
            % =========================

            f_a = 0.13;
            x0d_a = [0; x0_s_a+f_a];
            a_a = 0.7*(h2_a-h1_a-h3_a/2-h4_a);
            b_a = L2_a-0.22;
            c_a = 0.4*w4_a;

            x1d_a = x0d_a + [-c_a; a_a];
            x2d_a = x0d_a + [-c_a; 0];
            x3d_a = x0d_a + [c_a; 0];
            x4d_a = x0d_a + [c_a; a_a];
            x5d_a = x0d_a + [-c_a; b_a-f_a];
            x6d_a = x0d_a + [c_a; b_a-f_a];
            x7d_a = x0d_a + [0; L2_a-f_a];
            x8d_a = x0d_a + [0; b_a-f_a];
```

```
            plot(app.activeAxes, [x1d_a(1) x2d_a(1) x3d_a(1) x4d_a(1)], [x1d_a(2)
x2d_a(2) x3d_a(2) x4d_a(2)], 'k-', 'LineWidth', 2);
            plot(app.activeAxes, [x5d_a(1) x6d_a(1)], [x5d_a(2) x6d_a(2)], 'k-',
'LineWidth', 4);
            plot(app.activeAxes, [x7d_a(1) x8d_a(1)], [x7d_a(2) x8d_a(2)], 'k-',
'LineWidth', 2);
            plot(app.activeAxes, [0 0], [x0_s_a x0_s_a+f_a], 'k-', 'LineWidth',
2);

            % =========================
            xticks(app.activeAxes, 0);
            xticklabels(app.activeAxes, {sprintf('%.1f', curr_x_a)});
            xlabel(app.activeAxes, 'Location (m)', 'FontSize', 12);
            ylabel(app.activeAxes, 'Elevation (m)', 'FontSize', 12);
            grid(app.activeAxes, 'on');

            box(app.activeAxes, 'on');
            set(app.activeAxes, 'FontSize', 12);
            drawnow;
        end

        function plotsuspp(app, x_p, road_x_p, road_z_p, curr_x_p, umf_p)

            cla(app.passiveAxes); hold(app.passiveAxes, 'on');

            z0_p = x_p(1); % road elevation
            z1_p = x_p(2); % unsprung mass cm deviation (tekerlek)
            z2_p = x_p(3); % sprung mass cm deviation (gövde)
            t_p = x_p(4);  % current time

            % =========================
            % Geometric suspension parameters
            % =========================
            h1_p = 0.35;    % resting position of unsprung cm
            h2_p = 1.1;     % resting position of sprung cm
            h3_p = 0.2;     % height of unsprung mass block
            h4_p = 0.35;    % height of sprung mass block
            w1_p = 0.6;     % width of unsprung mass block
            w2_p = 0.8;     % width of sprung mass block
            w3_p = 0.1;     % width of tire spring
            w4_p = 0.15;    % width of suspension spring
            w5_p = 0.5;     % spring/damper spacing

            % Plotting parameter
            fw_p = 0.7;     % half of figure width

            % =========================
            % Preliminary calculations:
            % =========================
            x0_r_p = z0_p;                  % tire spring base position
            x0_s_p = h1_p + z1_p + h3_p/2;     % suspension spring base position
            x0_t_p = h1_p + z1_p - h3_p/2;     % unsprung mass block base position
            x0_b_p = h2_p + z2_p - h4_p/2;     % sprung mass block base position
            L1_p = x0_t_p - x0_r_p;          % tire spring length
            L2_p = x0_b_p - x0_s_p;          % suspension spring length


            if ~isempty(road_x_p) && ~isempty(road_z_p) && length(road_x_p) ==
length(road_z_p)
```

```
                x_start_p = max([curr_x_p - fw_p, min(road_x_p)]);
                x_end_p = min([curr_x_p + fw_p, max(road_x_p)]);
                mask_p = (road_x_p >= x_start_p) & (road_x_p <= x_end_p);
                if any(mask_p) && sum(mask_p) <= length(road_z_p)
                    plot(app.passiveAxes, road_x_p(mask_p) - curr_x_p,
road_z_p(mask_p) * umf_p, 'k-', 'LineWidth', 2);
                end
                xlim(app.passiveAxes, [-fw_p, fw_p]);
            end

            axis(app.passiveAxes, [-fw_p fw_p -0.25 1.5]);

            text(app.passiveAxes, 2*fw_p/3, 1.4, [num2str(t_p, '%2.1f') ' s'],
'FontWeight', 'bold', 'FontSize', 12);

            % ========================
            % Plot unsprung mass block
            % ========================
            x0t_p = [0; x0_t_p];
            x1t_p = x0t_p + [-w1_p/2; 0];
            x2t_p = x0t_p + [-w1_p/2; h3_p];
            x3t_p = x0t_p + [w1_p/2; h3_p];
            x4t_p = x0t_p + [w1_p/2; 0];
            fill(app.passiveAxes, [x1t_p(1) x2t_p(1) x3t_p(1) x4t_p(1)], [x1t_p(2)
x2t_p(2) x3t_p(2) x4t_p(2)], [0 1 0]);

            % ========================
            % Plot sprung mass block
            % ========================
            x0b_p = [0; x0_b_p];
            x1b_p = x0b_p + [-w2_p/2; 0];
            x2b_p = x0b_p + [-w2_p/2; h4_p];
            x3b_p = x0b_p + [w2_p/2; h4_p];
            x4b_p = x0b_p + [w2_p/2; 0];
            fill(app.passiveAxes, [x1b_p(1) x2b_p(1) x3b_p(1) x4b_p(1)], [x1b_p(2)
x2b_p(2) x3b_p(2) x4b_p(2)], [0 1 0]);

            % ========================
            % Plot tire spring
            % ========================
            x0r_p = [0; x0_r_p];
            plot(app.passiveAxes, x0r_p(1), x0r_p(2), 'ko', 'MarkerSize', 10,
'MarkerFaceColor', 'k');

            u_p = L1_p/9;
            x1r_p = x0r_p + [0; u_p];
            x2r_p = x0r_p + [-w3_p/2; 3/2*u_p];
            x3r_p = x2r_p + [w3_p; u_p];
            x4r_p = x3r_p + [-w3_p; u_p];
            x5r_p = x4r_p + [w3_p; u_p];
            x6r_p = x5r_p + [-w3_p; u_p];
            x7r_p = x6r_p + [w3_p; u_p];
            x8r_p = x7r_p + [-w3_p; u_p];
            x9r_p = x8r_p + [w3_p/2; u_p/2];
            x10r_p = x9r_p + [0; u_p];

            plot(app.passiveAxes, [x0r_p(1) x1r_p(1) x2r_p(1) x3r_p(1) x4r_p(1)
x5r_p(1) x6r_p(1) x7r_p(1) x8r_p(1) x9r_p(1) x10r_p(1)], ...
```

```
                [x0r_p(2) x1r_p(2) x2r_p(2) x3r_p(2) x4r_p(2) x5r_p(2) x6r_p(2)
x7r_p(2) x8r_p(2) x9r_p(2) x10r_p(2)], ...
                'k-', 'LineWidth', 2);

            % =========================
            % Plot suspension spring
            % =========================
            x0s_p = [-w5_p/4; x0_s_p];
            u_p = L2_p/9;
            x1s_p = x0s_p + [0; u_p];
            x2s_p = x0s_p + [-w4_p/2; 3/2*u_p];
            x3s_p = x2s_p + [w4_p; u_p];
            x4s_p = x3s_p + [-w4_p; u_p];
            x5s_p = x4s_p + [w4_p; u_p];
            x6s_p = x5s_p + [-w4_p; u_p];
            x7s_p = x6s_p + [w4_p; u_p];
            x8s_p = x7s_p + [-w4_p; u_p];
            x9s_p = x8s_p + [w4_p/2; u_p/2];
            x10s_p = x9s_p + [0; u_p];

            plot(app.passiveAxes, [x0s_p(1) x1s_p(1) x2s_p(1) x3s_p(1) x4s_p(1)
x5s_p(1) x6s_p(1) x7s_p(1) x8s_p(1) x9s_p(1) x10s_p(1)], ...
                [x0s_p(2) x1s_p(2) x2s_p(2) x3s_p(2) x4s_p(2) x5s_p(2) x6s_p(2)
x7s_p(2) x8s_p(2) x9s_p(2) x10s_p(2)], ...
                'k-', 'LineWidth', 3);

            % =========================
            % Plot suspension damper
            % =========================
            f_p = 0.13;
            x0d_p = [w5_p/4; x0_s_p+f_p];
            a_p = 0.7*(h2_p-h1_p-h3_p/2-h4_p);
            b_p = L2_p-0.22;
            c_p = 0.4*w4_p;

            x1d_p = x0d_p + [-c_p; a_p];
            x2d_p = x0d_p + [-c_p; 0];
            x3d_p = x0d_p + [c_p; 0];
            x4d_p = x0d_p + [c_p; a_p];
            x5d_p = x0d_p + [-c_p; b_p-f_p];
            x6d_p = x0d_p + [c_p; b_p-f_p];
            x7d_p = x0d_p + [0; L2_p-f_p];
            x8d_p = x0d_p + [0; b_p-f_p];

            plot(app.passiveAxes, [x1d_p(1) x2d_p(1) x3d_p(1) x4d_p(1)], [x1d_p(2)
x2d_p(2) x3d_p(2) x4d_p(2)], 'k-', 'LineWidth', 2);
            plot(app.passiveAxes, [x5d_p(1) x6d_p(1)], [x5d_p(2) x6d_p(2)], 'k-',
'LineWidth', 4);
            plot(app.passiveAxes, [x7d_p(1) x8d_p(1)], [x7d_p(2) x8d_p(2)], 'k-',
'LineWidth', 2);
            plot(app.passiveAxes, [w5_p/4 w5_p/4], [x0_s_p x0_s_p+f_p], 'k-',
'LineWidth', 2);

            % =========================
            xticks(app.passiveAxes, 0);
            xticklabels(app.passiveAxes, {sprintf('%.1f', curr_x_p)});
            xlabel(app.passiveAxes, 'Location (m)', 'FontSize', 12);
            ylabel(app.passiveAxes, 'Elevation (m)', 'FontSize', 12);
            grid(app.passiveAxes, 'on');
```

```
        box(app.passiveAxes, 'on');
        set(app.passiveAxes, 'FontSize', 12);
        drawnow;
    end
```

8.5.2 Running the Tool

```
    function startupFcn(app)
        close all;


        screenSize = get(0, 'ScreenSize');
        appSize = [app.a.Position(3), app.a.Position(4)];
        newPosition = [
            (screenSize(3) - appSize(1)) / 2,
            (screenSize(4) - appSize(2)) / 2,
            appSize(1),
            appSize(2)
        ];
        app.a.Position = newPosition;

        app.InputType.Visible = 'off';
        app.StepParameters.Visible = 'off';
        app.BumpParameters.Visible = 'off';
        app.CarParameters.Visible = 'off';
        app.QandR.Visible = 'off';
        app.Result.Visible = 'off';
        app.PotholeParameters.Visible = 'off';
        app.SinusoidParameters.Visible = 'off';
        app.Cover.Visible = 'on';

        img = uiimage(app.QandR);
        img.ImageSource = 'Qleftbracket.png';
        img.Position = [83 65 50 200];

        img = uiimage(app.QandR);
        img.ImageSource = 'Qrightbracket.png';
        img.Position = [283 63 50 200];

        app.Figure1Data = imread('Step_Input.png');
        app.Figure2Data = imread('Pothole_Input.png');
        app.Figure3Data = imread('Bump_Input.png');
        app.Figure4Data = imread('Sinusoid_Input.png');

        app.Image.ImageSource = app.Figure1Data;
        app.Image3.ImageSource = app.Figure2Data;
        app.Image2.ImageSource = app.Figure3Data;
        app.Image4.ImageSource = app.Figure4Data;
    end
```

8.5.3 Cover Page

8.5.3.1 ‘START’ Button

```
    function STARTButtonPushed(app, event)
        app.Cover.Visible = 'off';
        app.CarParameters.Visible = 'on';

        img = uiimage(app.QandR);
```

```
    img.ImageSource = 'Rleftbracket.png';
    img.Position = [512 137 50 50];

    img = uiimage(app.QandR);
    img.ImageSource = 'Rrightbracket.png';
    img.Position = [592 137 50 50];

    img = uiimage(app.Result);
    img.ImageSource = 'Rleftbracket.png';
    img.Position = [232 446 50 50];

    img = uiimage(app.Result);
    img.ImageSource = 'Rrightbracket.png';
    img.Position = [567 446 50 50];
end
```

8.5.4 Quarter-Car Parameters Page

8.5.4.1 'RESET' Button

```
function RESETButtonPushed(app, event)
    app.SprungMasskgEditField.Value = 0;
    app.UnsprungMasskgEditField.Value = 0;
    app.SuspensionStiffnessNmEditField.Value = 0;
    app.SuspensionDampingCoefficientNsmEditField.Value = 0;
    app.TireStiffnessNmEditField.Value = 0;
    app.CarSpeedkmhEditField.Value = 0;
end
```

8.5.4.2 'CONTINUE' Button

```
function CONTINUEButtonPushed(app, event)
    ms = app.SprungMasskgEditField.Value;
    mu = app.UnsprungMasskgEditField.Value;
    ks = app.SuspensionStiffnessNmEditField.Value;
    c = app.SuspensionDampingCoefficientNsmEditField.Value;
    kt = app.TireStiffnessNmEditField.Value;
    app.car_speed = (app.CarSpeedkmhEditField.Value*1000)/3600;

    app.A = [   0      1         0       0   ;
             -ks/ms  -c/ms     ks/ms    c/ms ;
                0      0         0       1   ;
              ks/mu   c/mu -(ks+kt)/mu -c/mu];

    app.B = [   0       0   ;
                0      1/ms ;
                0       0   ;
              kt/mu   -1/mu];

    app.C = [   1     0     0    0   ;
                1     0    -1    0   ;
             -ks/ms -c/ms ks/ms c/ms ;
                0     0     1    0  ];

    app.D = [   0    0   ;
                0    0   ;
                0   1/ms ;
               -1    0  ];

    app.CarParameters.Visible = 'off';
    app.InputType.Visible = 'on';
```

```
end
```

8.5.5 Input Type Page

8.5.5.1 ‘STEP’ Button

```
function STEPButtonPushed(app, event)
    app.InputType.Visible = 'off';
    app.StepParameters.Visible = 'on';
end
```

8.5.5.2 ‘BUMP’ Button

```
function BUMPButtonPushed(app, event)
    app.InputType.Visible = 'off';
    app.BumpParameters.Visible = 'on';
end
```

8.5.5.3 ‘POTHOLE’ Button

```
function POTHOLEButtonPushed(app, event)
    app.InputType.Visible = 'off';
    app.PotholeParameters.Visible = 'on';
end
```

8.5.5.4 ‘SINUSOID’ Button

```
function SINUSOIDButtonPushed(app, event)
    app.InputType.Visible = 'off';
    app.SinusoidParameters.Visible = 'on';
end
```

8.5.5.5 ‘BACK to CAR PARAMETERS’ Button

```
function BACKtoCARPARAMETERSButtonPushed(app, event)
    app.InputType.Visible = 'off';
    app.CarParameters.Visible = 'on';
end
```

8.5.6 Step Input Parameters Page

8.5.6.1 ‘Preview Input’ Button

```
function PreviewInputButton_2Pushed(app, event)
    k = app.SteepnessFactorEditField.Value;
    h = app.StepHeightmEditField.Value;
    ls = app.StepLocationmEditField.Value;
    lengthsim = app.RoadLengthmEditField.Value;

    l_x = (0:0.01:lengthsim)';
    x_u_r = h ./ (1 + exp(-k * (l_x - ls)));

    plot(app.StepPlot, l_x, x_u_r, 'LineWidth', 3);

    xMax = max(l_x);
    yMax = max(abs(x_u_r));

    targetRatio = 50;

    if (xMax / yMax) > targetRatio
        yLimit = [-xMax / targetRatio, xMax / targetRatio];
        xLimit = [0, xMax];
    else
        xLimit = [0, yMax * targetRatio];
        yLimit = [-yMax, yMax];
    end
```

```
        app.StepPlot.XLim = xLimit;
        app.StepPlot.YLim = yLimit;

        app.tsim = lengthsim / app.car_speed;
              app.simtimestep.Text = ['Simulation Time = ' sprintf('%.2f',
    app.tsim) ' s'];
        app.t_r = (0:0.01:app.tsim)';
        app.u_r = h ./ (1 + exp(-k * (app.t_r - (ls / app.car_speed))));
    end
```

8.5.6.2 'Reset' Button

```
    function ResetButton_2Pushed(app, event)
        app.SteepnessFactorEditField.Value = 0;
        app.StepHeightmEditField.Value = 0;
        app.StepLocationmEditField.Value = 0;
        app.RoadLengthmEditField.Value = 0;

        cla(app.StepPlot)
    end
```

8.5.6.3 'BACK to INPUT TYPE SELECTION' Button

```
    function BACKtoINPUTTYPESELECTIONButton_5Pushed(app, event)
        app.StepParameters.Visible = 'off';
        app.InputType.Visible = 'on';
    end
```

8.5.6.4 'CONTINUE' Button

```
    function CONTINUEButton_3Pushed(app, event)
        app.StepParameters.Visible = 'off';
        app.QandR.Visible = 'on';
    end
```

8.5.7 Bump Input Parameters Page

8.5.7.1 'Preview Input' Button

```
    function PreviewInputButtonPushed(app, event)
        h = app.BumpHeightmEditField.Value;
        l_rise = app.BumpStartLocationmEditField.Value;
        l_fall = app.BumpFinishLocationmEditField.Value;
        lengthsim = app.RoadLengthmEditField_2.Value;

        l_x = (0:0.01:lengthsim)';
        x_u_r = zeros(size(l_x));

        for i = 1:length(l_x)
            if l_x(i) >= l_rise && l_x(i) <= l_fall
                x_u_r(i) = (h / 2) * (1 - cos(2 * pi * (l_x(i) - l_rise) /
(l_fall - l_rise)));
            else
                x_u_r(i) = 0;
            end
        end

        plot(app.BumpPlot, l_x, x_u_r, 'LineWidth', 3)

        xMax = max(l_x);
        yMax = max(abs(x_u_r));

        targetRatio = 50;
```

```
            if (xMax / yMax) > targetRatio
                yLimit = [-xMax / targetRatio, xMax / targetRatio];
                xLimit = [0, xMax];
            else
                xLimit = [0, yMax * targetRatio];
                yLimit = [-yMax, yMax];
            end

            app.BumpPlot.XLim = xLimit;
            app.BumpPlot.YLim = yLimit;

            app.tsim = lengthsim / app.car_speed;
            app.simtimebump.Text = ['Simulation Time = ' sprintf('%.2f', app.tsim)
' s'];
            app.t_r = (0:0.01:app.tsim)';
            app.u_r = zeros(size(app.t_r));

            for i = 1:length(app.t_r)
                if app.t_r(i) >= l_rise / app.car_speed && app.t_r(i) <= l_fall /
app.car_speed
                    app.u_r(i) = (h / 2) * (1 - cos(2 * pi * (app.t_r(i) - (l_rise
/ app.car_speed)) / ((l_fall - l_rise) / app.car_speed)));
                else
                    app.u_r(i) = 0;
                end
            end
        end
```

8.5.7.2 'Reset' Button

```
        function ResetButtonPushed(app, event)
            app.BumpHeightmEditField.Value = 0;
            app.BumpStartLocationmEditField.Value = 0;
            app.BumpFinishLocationmEditField.Value = 0;
            app.RoadLengthmEditField_2.Value = 0;

            cla(app.BumpPlot)
        end
```

8.5.7.3 'BACK to INPUT TYPE SELECTION' Button

```
        function BACKtoINPUTTYPESELECTIONButton_4Pushed(app, event)
            app.BumpParameters.Visible = 'off';
            app.InputType.Visible = 'on';
        end
```

8.5.7.4 'CONTINUE' Button

```
        function CONTINUEButton_4Pushed(app, event)
            app.BumpParameters.Visible = 'off';
            app.QandR.Visible = 'on';
        end
```

8.5.8 Pothole Input Parameters Page

8.5.8.1 'Preview Input' Button

```
        function PreviewInputButton_3Pushed(app, event)
            h = app.PotholeDepth.Value;
            l_start = app.PotholeStartLocation.Value;
            l_finish = app.PotholeFinishLocation.Value;
            lengthsim = app.RoadLengthmEditField_3.Value;
```

```
            l_x = (0:0.01:lengthsim)';
            x_u_r = zeros(size(l_x));

            for i = 1:length(l_x)
                if l_x(i) >= l_start && l_x(i) <= l_finish
                    x_u_r(i) = (h / 2) * -(1 - cos(2 * pi * (l_x(i) - l_start) /
(l_finish - l_start)));
                else
                    x_u_r(i) = 0;
                end
            end

            plot(app.PotholePlot, l_x, x_u_r, 'LineWidth', 3)

            xMax = max(l_x);
            yMax = max(abs(x_u_r));

            targetRatio = 50;

            if (xMax / yMax) > targetRatio
                yLimit = [-xMax / targetRatio, xMax / targetRatio];
                xLimit = [0, xMax];
            else
                xLimit = [0, yMax * targetRatio];
                yLimit = [-yMax, yMax];
            end

            app.PotholePlot.XLim = xLimit;
            app.PotholePlot.YLim = yLimit;

            app.tsim = lengthsim / app.car_speed;
            app.simtimepoth.Text = ['Simulation Time = ' sprintf('%.2f', app.tsim)
' s'];
            app.t_r = (0:0.01:app.tsim)';
            app.u_r = zeros(size(app.t_r));

            for i = 1:length(app.t_r)
                if app.t_r(i) >= l_start / app.car_speed && app.t_r(i) <= l_finish
/ app.car_speed
                    app.u_r(i) = (h / 2) * -(1 - cos(2 * pi * (app.t_r(i) -
l_start / app.car_speed) / ((l_finish - l_start) / app.car_speed)));
                else
                    app.u_r(i) = 0;
                end
            end
        end
```

8.5.8.2 ‘Reset’ Button

```
        function ResetButton_3Pushed(app, event)
            app.PotholeStartLocation.Value = 0;
            app.PotholeFinishLocation.Value = 0;
            app.PotholeDepth.Value = 0;
            app.RoadLengthmEditField_3.Value = 0;

            cla(app.PotholePlot)
        end
```

8.5.8.3 'BACK to INPUT TYPE SELECTION' Button

```
function BACKtoINPUTTYPESELECTIONButton_3Pushed(app, event)
    app.PotholeParameters.Visible = 'off';
    app.InputType.Visible = 'on';
end
```

8.5.8.4 'CONTINUE' Button

```
function CONTINUEButton_5Pushed(app, event)
    app.PotholeParameters.Visible = 'off';
    app.QandR.Visible = 'on';
end
```

8.5.9 Sinusoidal Input Parameters Page

8.5.9.1 'Preview Input' Button

```
function PreviewInputButton_4Pushed(app, event)
    n_w = app.NumberofWaves.Value;
    h_w = app.Amplitude.Value;
    lengthsim = app.RoadLengthmEditField_4.Value;

    smooth_percentage = 0.05;
    smooth_length = lengthsim * smooth_percentage;

    l_x = (0:0.01:lengthsim)';
    f_x = (2*pi*n_w)/lengthsim;
    x_u_r = (2*h_w)*sin(f_x*l_x);

    window = ones(size(l_x));

    start_idx = l_x < smooth_length;
    window(start_idx) = 0.5 - 0.5*cos(pi*l_x(start_idx)/smooth_length);

    end_idx = l_x > (lengthsim - smooth_length);
    window(end_idx) = 0.5 - 0.5*cos(pi*(lengthsim -
l_x(end_idx))/smooth_length);

    x_u_r_smoothed = x_u_r .* window;

    plot(app.SinusoidPlot, l_x, x_u_r_smoothed, 'LineWidth', 3)

    xMax = max(l_x);
    yMax = max(abs(x_u_r_smoothed));

    targetRatio = 50;

    if (xMax / yMax) > targetRatio
        yLimit = [-xMax / targetRatio, xMax / targetRatio];
        xLimit = [0, xMax];
    else
        xLimit = [0, yMax * targetRatio];
        yLimit = [-yMax, yMax];
    end

    app.SinusoidPlot.XLim = xLimit;
    app.SinusoidPlot.YLim = yLimit;

    app.tsim = lengthsim / app.car_speed;
    app.simtimesin.Text = ['Simulation Time = ' sprintf('%.2f', app.tsim)
' s'];
```

```
            app.t_r = (0:0.01:app.tsim)';
            f_t = (2*pi*n_w) / app.tsim;

            window_time = ones(size(app.t_r));

            start_idx_time = app.t_r < (app.tsim * smooth_percentage);
            window_time(start_idx_time) = 0.5 -
0.5*cos(pi*app.t_r(start_idx_time)/(app.tsim*smooth_percentage));

            end_idx_time = app.t_r > (app.tsim * (1-smooth_percentage));
            window_time(end_idx_time) = 0.5 - 0.5*cos(pi*(app.tsim -
app.t_r(end_idx_time))/(app.tsim*smooth_percentage));

            app.u_r = (2*h_w) * sin(f_t * app.t_r) .* window_time;
        end
```

8.5.9.2 'Reset' Button

```
        function ResetButton_4Pushed(app, event)
            app.NumberofWaves.Value = 0;
            app.Amplitude.Value = 0;
            app.RoadLengthmEditField_4.Value = 0;

            cla(app.SinusoidPlot)
        end
```

8.5.9.3 'BACK to INPUT TYPE SELECTION' Button

```
        function BACKtoINPUTTYPESELECTIONButton_2Pushed(app, event)
            app.SinusoidParameters.Visible = 'off';
            app.InputType.Visible = 'on';
        end
```

8.5.9.4 'CONTINUE' Button

```
function CONTINUEButton_2Pushed(app, event)
            app.SinusoidParameters.Visible = 'off';
            app.QandR.Visible = 'on';
        end
```

8.5.10 Initial Conditions and Q & R Matrices Page

8.5.10.1 'BACK to INPUT TYPE SELECTION' Button

```
       function BACKtoINPUTTYPESELECTIONButtonPushed(app, event)
            app.QandR.Visible = 'off';
            app.InputType.Visible = 'on';
        end
```

8.5.10.2 'FINISH' Button

```
        function FINISHButtonPushed(app, event)
            app.Q = zeros(4,4);
            app.Q(1,1) = app.Q11.Value;
            app.Q(2,2) = app.Q22.Value;
            app.Q(3,3) = app.Q33.Value;
            app.Q(4,4) = app.Q44.Value;

            app.R = app.R11.Value;

            app.x0 = zeros(4,1);
            app.x0(1) = app.State1SprungMassDisplacementmEditField.Value;
            app.x0(2) = app.State2SprungMassVelocitymsEditField.Value;
            app.x0(3) = app.State3UnsprungMassDisplacementmEditField.Value;
            app.x0(4) = app.State4UnsprungMassVelocitymsEditField.Value;
```

```
            app.K_LQR = lqr(app.A,app.B(:,2),app.Q,app.R);

            modelName = 'eng_project_app_sim';

            if ~bdIsLoaded(modelName)
                load_system(modelName);
            end

            input_data = [app.t_r(:) app.u_r(:)];


            simInputActive = Simulink.SimulationInput(modelName);
            simInputActive = simInputActive...
                .setExternalInput(input_data)...
                .setBlockParameter([modelName '/State-Space'], 'A',
mat2str(app.A))...
                .setBlockParameter([modelName '/State-Space'], 'B',
mat2str(app.B))...
                .setBlockParameter([modelName '/State-Space'], 'C',
mat2str([eye(4);app.C]))...
                .setBlockParameter([modelName '/State-Space'], 'D',
mat2str([zeros(4,2);app.D]))...
                .setBlockParameter([modelName '/State-Space'], 'x0',
mat2str(app.x0))...
                .setBlockParameter([modelName '/gain'], 'Gain', mat2str(-
app.K_LQR))...
                .setModelParameter('StopTime', num2str(app.tsim));

            active = sim(simInputActive);


            simInputPassive = Simulink.SimulationInput(modelName);
            simInputPassive = simInputPassive...
                .setExternalInput(input_data)...
                .setBlockParameter([modelName '/State-Space'], 'A',
mat2str(app.A))...
                .setBlockParameter([modelName '/State-Space'], 'B',
mat2str(app.B))...
                .setBlockParameter([modelName '/State-Space'], 'C',
mat2str([eye(4);app.C]))...
                .setBlockParameter([modelName '/State-Space'], 'D',
mat2str([zeros(4,2);app.D]))...
                .setBlockParameter([modelName '/State-Space'], 'x0',
mat2str(app.x0))...
                .setBlockParameter([modelName '/gain'], 'Gain', 'zeros(1,4)')...
                .setModelParameter('StopTime', num2str(app.tsim));

            passive = sim(simInputPassive);

            app.K11.Text = sprintf('%.2f', app.K_LQR(1));
            app.K12.Text = sprintf('%.2f', app.K_LQR(2));
            app.K13.Text = sprintf('%.2f', app.K_LQR(3));
            app.K14.Text = sprintf('%.2f', app.K_LQR(4));

            app.SprungMassAccelerationms2Label.Text = ['Sprung Mass Acceleration
[m/s' char(178) ']'];

            app.actidef.Text = sprintf('%.4f', max(abs(active.y.Data(:,4))));
```

```
            app.patidef.Text = sprintf('%.4f', max(abs(passive.y.Data(:,4))));
            app.acsustr.Text = sprintf('%.3f', max(abs(active.y.Data(:,2))));
            app.pasustr.Text = sprintf('%.3f', max(abs(passive.y.Data(:,2))));
            app.acspacc.Text = sprintf('%.2f', max(abs(active.y.Data(:,3))));
            app.paspacc.Text = sprintf('%.2f', max(abs(passive.y.Data(:,3))));
            app.confor.Text = sprintf('%.2f', max(abs(active.u.Data)));


            app.QandR.Visible = 'off';
            app.Result.Visible = 'on';

            screen_size = get(0, 'ScreenSize');
            screen_width = screen_size(3);
            screen_height = screen_size(4);

            fig_width = 400;
            fig_height = 300;

            figure(1)
            set(gcf, 'Position', [(screen_width-fig_width)/2 (screen_height-
fig_height)/2 fig_width fig_height]);
            ax1 = axes;
            hold on
            plot(ax1, passive.y.Time, passive.y.Data(:,1), 'g', 'LineWidth',2);
            plot(ax1, active.y.Time, active.y.Data(:,1), 'r', 'LineWidth',2);
            plot(ax1, active.y.Time, active.input.Data, 'b', 'LineWidth',1);
            title(ax1, 'Sprung Mass
Displacement','FontSize',18,'FontWeight','Bold');
            xlabel(ax1, '$Time$ [s]','FontSize',18,'Interpreter','latex');
            ylabel(ax1, '$x_s$
[m]','FontSize',18,'FontWeight','Bold','Interpreter','latex');
            legend(ax1, 'Passive','Active','Road
Input','FontSize',12,'FontWeight','Bold');

            figure(2)
            set(gcf, 'Position', [(screen_width-fig_width)/2 (screen_height-
fig_height)/2 fig_width fig_height]);
            ax2 = axes;
            hold on
            plot(ax2, passive.y.Time, passive.y.Data(:,2), 'g', 'LineWidth',2);
            plot(ax2, active.y.Time, active.y.Data(:,2), 'r', 'LineWidth',2);
            title(ax2, 'Suspension Travel','FontSize',18,'FontWeight','Bold');
            xlabel(ax2, '$Time$ [s]','FontSize',18,'Interpreter', 'latex');
            ylabel(ax2, '$x_s - x_u$ [m]','FontSize',18,'Interpreter','latex');
            legend(ax2, 'Passive','Active','FontSize',12,'FontWeight','Bold');

            figure(3)
            set(gcf, 'Position', [(screen_width-fig_width)/2 (screen_height-
fig_height)/2 fig_width fig_height]);
            ax3 = axes;
            hold on
            plot(ax3, passive.y.Time, passive.y.Data(:,3), 'g', 'LineWidth',2);
            plot(ax3, active.y.Time, active.y.Data(:,3), 'r', 'LineWidth',2);
            xlabel(ax3, '$Time$ [s]','FontSize',18,'Interpreter','latex');
            ylabel(ax3, '$\ddot{x}_{s} [m/s^2]$','FontSize',18,'Interpreter',
'latex');
            title(ax3, 'Sprung Mass
Acceleration','FontSize',18,'FontWeight','Bold');
            legend(ax3, 'Passive','Active','FontSize',12,'FontWeight','Bold');
```

```
            figure(4)
            set(gcf, 'Position', [(screen_width-fig_width)/2 (screen_height-
fig_height)/2 fig_width fig_height]);
            ax4 = axes;
            hold on
            plot(ax4, passive.y.Time, passive.y.Data(:,4), 'g', 'LineWidth',2);
            plot(ax4, active.y.Time, active.y.Data(:,4), 'r', 'LineWidth',2);
            xlabel(ax4, '$Time$ [s]','FontSize',18,'Interpreter','latex');
            ylabel(ax4, '$x_u - x_r$ [m]','FontSize',18,'Interpreter', 'latex');
            title(ax4, 'Tire Deflection','FontSize',18,'FontWeight','Bold');
            legend(ax4, 'Passive','Active','FontSize',12,'FontWeight','Bold');

            figure(5)
            set(gcf, 'Position', [(screen_width-fig_width)/2 (screen_height-
fig_height)/2 fig_width fig_height]);
            ax5 = axes;
            plot(ax5, active.u.Time, active.u.Data, 'r', 'LineWidth',2);
            xlabel(ax5, '$Time$ [s]','FontSize',18,'Interpreter', 'latex');
            ylabel(ax5, '$U(t)$ [N]','FontSize',16,'Interpreter','latex');
            title(ax5, 'Control Force','FontSize',18,'FontWeight','Bold');

            app.animFigure = figure('Position', [100, 100, 1200, 800], 'Visible',
'off');
            set(app.animFigure, 'DoubleBuffer', 'on');
            set(0, 'DefaultFigureRenderer', 'painters');

            global animation_cancelled;
            animation_cancelled = false;

            h = waitbar(0, 'Generating animation...', 'Name', 'Progress', ...
                'CreateCancelBtn', 'setappdata(gcbf,''canceling'',1)');
            setappdata(h, 'canceling', 0);

            videoFileName = 'suspension_comparison.mp4';
            writerObj = [];
            try
                writerObj = VideoWriter(videoFileName, 'MPEG-4');
                writerObj.FrameRate = 30;
                open(writerObj);
            catch
                if ishandle(h), delete(h); end
                if ishandle(app.animFigure), delete(app.animFigure); end
                return;
            end

            try
                app.passiveAxes = subplot(1, 2, 1, 'Parent', app.animFigure);
                title(app.passiveAxes, 'Passive Suspension', 'FontWeight', 'bold',
'FontSize', 14);
                app.activeAxes = subplot(1, 2, 2, 'Parent', app.animFigure);
                title(app.activeAxes, 'Active Suspension', 'FontWeight', 'bold',
'FontSize', 14);
            catch
                if ishandle(h), delete(h); end
                if ishandle(app.animFigure), delete(app.animFigure); end
                if ~isempty(writerObj), close(writerObj); end
                return;
            end
```

```
            try
                z0_a = active.input.Data;
                z1_a = active.y.Data(:,1) - active.y.Data(:,2);
                z2_a = active.y.Data(:,1);
                t_a = active.y.Time;
                road_x_a = t_a * app.car_speed;
                road_z_a = z0_a;
                umf_a = 1;
                z0_p = passive.input.Data;
                z1_p = passive.y.Data(:,1) - passive.y.Data(:,2);
                z2_p = passive.y.Data(:,1);
                t_p = passive.y.Time;
                road_x_p = t_p * app.car_speed;
                road_z_p = z0_p;
                umf_p = 1;
                min_t_start = min(t_a(1), t_p(1));
                max_t_end = max(t_a(end), t_p(end));
                num_frames = min(length(t_a), length(t_p));
                common_time = linspace(min_t_start, max_t_end, num_frames);
                z0_a_interp = interp1(t_a, z0_a, common_time, 'linear', 'extrap');
                z1_a_interp = interp1(t_a, z1_a, common_time, 'linear', 'extrap');
                z2_a_interp = interp1(t_a, z2_a, common_time, 'linear', 'extrap');
                road_x_a_interp = common_time * app.car_speed;
                z0_p_interp = interp1(t_p, z0_p, common_time, 'linear', 'extrap');
                z1_p_interp = interp1(t_p, z1_p, common_time, 'linear', 'extrap');
                z2_p_interp = interp1(t_p, z2_p, common_time, 'linear', 'extrap');
                road_x_p_interp = common_time * app.car_speed;
                speed_factor = 0.3;
            catch
                if ishandle(h), delete(h); end
                if ishandle(app.animFigure), delete(app.animFigure); end
                if ~isempty(writerObj), close(writerObj); end
                return;
            end

            try
                for i = 1:length(common_time)
                    if getappdata(h, 'canceling') || ~ishandle(h)
                        animation_cancelled = true;
                        break;
                    end

                    try
                        waitbar(i/length(common_time), h);
                    catch
                        animation_cancelled = true;
                        break;
                    end

                    try
                        plotsuspp(app, [z0_p_interp(i), z1_p_interp(i),
z2_p_interp(i), common_time(i)], ...
                            road_x_p, road_z_p, road_x_p_interp(i), umf_p);
                        plotsuspa(app, [z0_a_interp(i), z1_a_interp(i),
z2_a_interp(i), common_time(i)], ...
                            road_x_a, road_z_a, road_x_a_interp(i), umf_a);
                    catch
                        continue;
```

```
                    end

                    try
                        frame = getframe(app.animFigure);
                        writeVideo(writerObj, frame);
                    catch
                        continue;
                    end

                    if i < length(common_time)
                        dt = common_time(i+1) - common_time(i);
                        pause(dt / speed_factor);
                    end
                end
            catch
            end

            try, if ishandle(h), delete(h); end, catch, end
            try, if ~isempty(writerObj), close(writerObj); end, catch, end
            try, if ishandle(app.animFigure), close(app.animFigure); end, catch,
end

            if ~animation_cancelled && exist(videoFileName, 'file')
                videoPath = fullfile(pwd, videoFileName);
                try
                    if ispc
                        winopen(videoPath);
                    elseif ismac
                        system(['open "', videoPath, '" &']);
                    else
                        system(['xdg-open "', videoPath, '" &']);
                    end
                catch
                end
            elseif animation_cancelled && exist(videoFileName, 'file')
                try, delete(videoFileName); catch, end
            end
        end
```

8.5.11 Results Page

8.5.11.1 ‘BACK’ Button

```
        function BACKButtonPushed(app, event)
            app.Result.Visible = 'off';
            app.QandR.Visible = 'on';
        end
```