MARMARA UNIVERSITY
FACULTY OF ENGINEERING

OPTIMAL DESIGN OF A CAR SUSPENSION SYSTEM

BURAK YILDIZ, İSMAİL ÇAKIRAL

GRADUATION PROJECT REPORT
Department of Mechanical Engineering

Supervisor
Prof. Dr. Mustafa ÖZDEMİR

ISTANBUL, 2025

MARMARA UNIVERSITY
FACULTY OF ENGINEERING

OPTIMAL DESIGN OF A CAR SUSPENSION SYSTEM

BURAK YILDIZ (150421037)
İSMAİL ÇAKIRAL (150420023)

GRADUATION PROJECT REPORT
Department of Mechanical Engineering

Supervisor
Prof. Dr. Mustafa ÖZDEMİR

ISTANBUL, 2025

MARMARA UNIVERSITY
FACULTY OF ENGINEERING

OPTIMAL DESIGN OF A CAR SUSPENSION SYSTEM

by

Burak Yıldız, İsmail Çakıral

June 23, 2025, Istanbul

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

OF

BACHELOR OF SCIENCE

AT

MARMARA UNIVERSITY

The author(s) hereby grant(s) to Marmara University permission to reproduce and to distribute publicly paper and electronic copies of this document in whole or in part and declare that the prepared document does not in any way include copying of previous work on the subject or the use of ideas, concepts, words, or structures regarding the subject without appropriate acknowledgement of the source material.

Signature of Authors..Burak YILDIZ, İsmail ÇAKIRAL

[illegible] [illegible]

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 ÖZDEMİR, for the valuable guidance and advice on preparing this thesis and giving us moral and material support.

June, 2025 Burak YILDIZ

İsmail ÇAKIRAL

CONTEST

ABSTRACT

This study presents the modeling, simulation, and optimization of a quarter-car suspension system to enhance ride comfort. A 2-degree-of-freedom dynamic model was constructed to represent the vertical motions of the sprung and unsprung masses. Three distinct road input profiles step, bump, and sine wave were simulated using MATLAB Simulink. A user-friendly interface was developed via MATLAB App Designer to allow parameter variation and visualization of system responses. Optimization was conducted considering comfort and safety metrics such as RMS acceleration, suspension deflection, and tire deflection. To overcome simulation instability, sigmoid and Bezier curve smoothing techniques were employed. The results identify optimal suspension parameters under various road conditions, offering tool for vehicle suspension design improvement.

SYMBOLS

$\boldsymbol{k_t}$	: stiffness parameters of wheel
$\boldsymbol{k_s}$	: stiffness parameters of suspension
$\boldsymbol{c_s}$	: damper parameters of suspension
$\boldsymbol{m_u}$	: mass of wheel
$\boldsymbol{m_s}$	: mass of quarter car
$\boldsymbol{z_r}$	: vertical displacement of road
$\boldsymbol{z_1}$	: vertical displacement of quarter car body
$\boldsymbol{z_2}$	: vertical displacement of wheel
$\boldsymbol{x_s - x_{us}}$	: suspension deflection
$\boldsymbol{x_{us} - x_r}$	: tire deflection
$\boldsymbol{T}$	: period
$\boldsymbol{L}$	: length
$\boldsymbol{v}$	: velocity
$\boldsymbol{f}$	: frequency

ABBREVIATIONS

RMS	: Root mean square

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Suspension systems are among the most important components of vehicles, such as cars and other vehicles, that play a role in ride comfort, handling, and safety. A suspension system is a vehicle system that provides the driver and passengers with a comfortable ride by absorbing road irregularities and impacts while driving. One of the purposes of the suspension system is to connect the wheels to the chassis of the vehicle.

From a mechanical design perspective, suspension systems are generally divided into two categories: dependent and independent. Examples of independent suspension types include McPherson and Double Wishbone. An example of a dependent suspension type is a solid axle with a leaf spring. Leaf spring suspensions can be used depending on the type of vehicle; for instance, a solid axle with a leaf spring is commonly used for heavy-duty vehicles (Jazar, 2017).

In addition to this mechanical design classification, suspension systems can also be examined based on controllability and adaptability, and they are categorized as passive, semi-active, and active systems. Each of these suspension types has distinct advantages and disadvantages depending on vehicle performance. For example, studies comparing the responses of active and passive suspensions to various road profiles are available in the literature. (Agharkakli, Sabet, & Barouz, 2012).

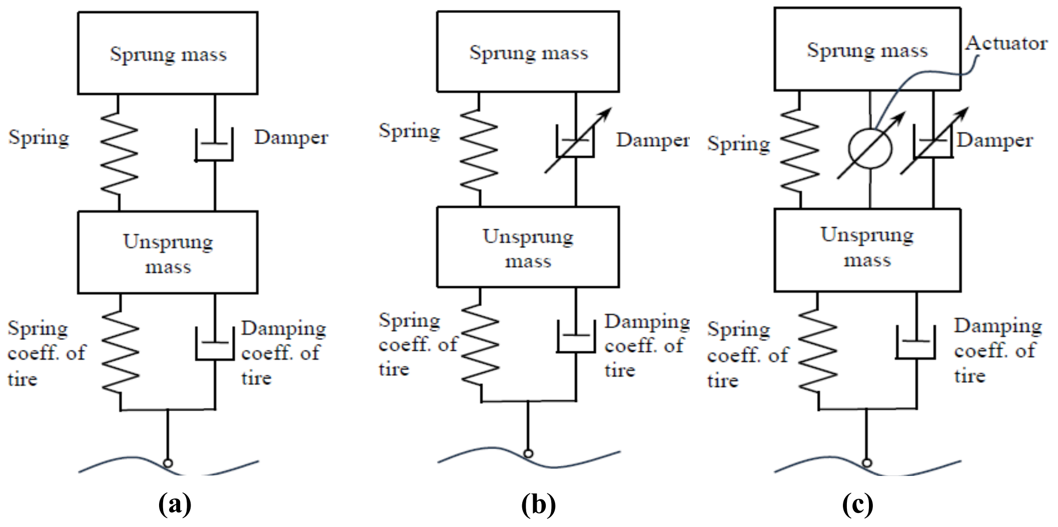


Figure 1. Car suspension types (a) passive, (b) semi-active, (c) active shown in quarter car model (Omar et al., 2017).

Passive suspensions are reliable, do not contain complex structures, and are less costly compared to other types of suspension systems. On the other hand, active suspensions can provide a better response than passive suspensions in terms of ride comfort and handling; however, the number of vehicles using active suspension is limited because these systems are expensive and complex. Another disadvantage of active suspension is that it often includes hydraulic or pneumatic systems. Due to the use of these types of components, an additional power requirement issue arises for these systems (Martins, Esteves, Pina da Silva, & Verdelho, 1999). Furthermore, since active suspension includes more components in addition to those in passive suspension, it results in a loss of space in the vehicle compared to passive suspension shown in Fig. 2.

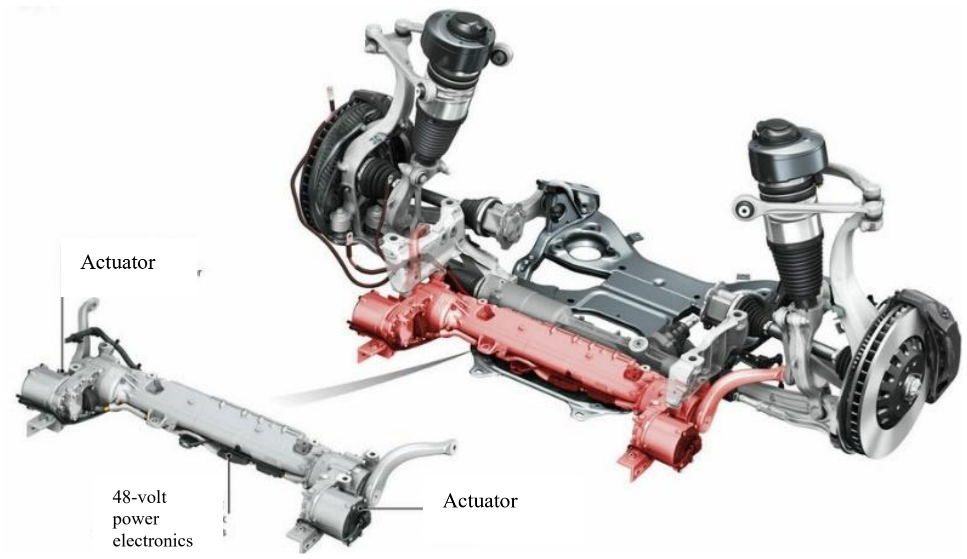


Figure 2. Active suspension illustration (Audi MediaCenter Website, 2019).

Various models are used when mathematically modeling suspensions. For conducting mathematical model of suspension there are several models: quarter car, half car, and full car. The half car model is frequently used in the literature and may be used in place of the full car model to reduce computational complexity (Desai, Guha, & Seshu, 2021). Indeed, the quarter car model is more preferrable for further simplifying the analyses of suspension systems. This model is useful for understanding the impact of road surface irregularities on a single wheel, providing insight into spring and damper performance.

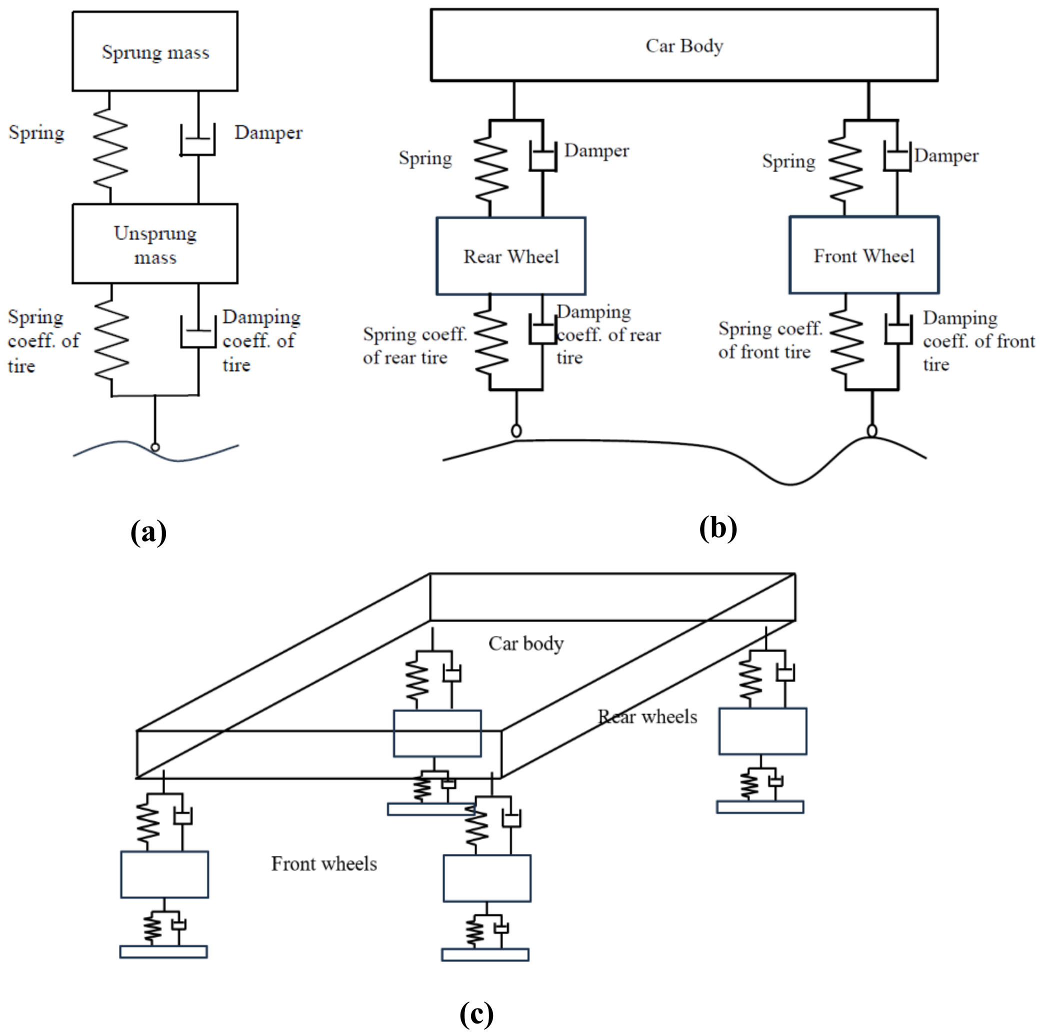


Figure 3. (a) Quarter, (b) half, (c) full car models (Jazar, 2017).

In the modeling of suspension systems, studies can be conducted using several multi-DOF (degrees of freedom) models to capture the complexity of the system in a more realistic way. However, increasing the DOF in suspension system models makes them more complex, which in turn extends simulation time and requires higher computational power. According to a study in the literature, it is generally accepted that an unnecessarily high degree of freedom is not required, due to various simplifying assumptions (Mitschke, 1962).

In this context, the optimization process plays a critical role, allowing us to refine suspension parameters based on different performance indices without introducing unnecessary complexity. Different methods were used for the optimization of the parameters in the study: Genetic Algorithm (GA) and Multi-Objective Optimization. GA allows us to optimize the suspension parameters based on factors such as comfort and road handling (Georgiou, Verros, & Natsiavas, 2007). A genetic algorithm is one of the available methods used to determine optimal parameters for solving complex systems. Alternative optimization methods can also be found in the literature.

In the literature, suspension studies categorize spring and damper components as either linear or nonlinear. Linear suspension systems are used in calculations where the spring and damper forces change linearly, whereas nonlinear suspension systems account for variable system parameters and other non-linearities in their calculations (Mohite, & Mitra, 2018).

2. MODELING

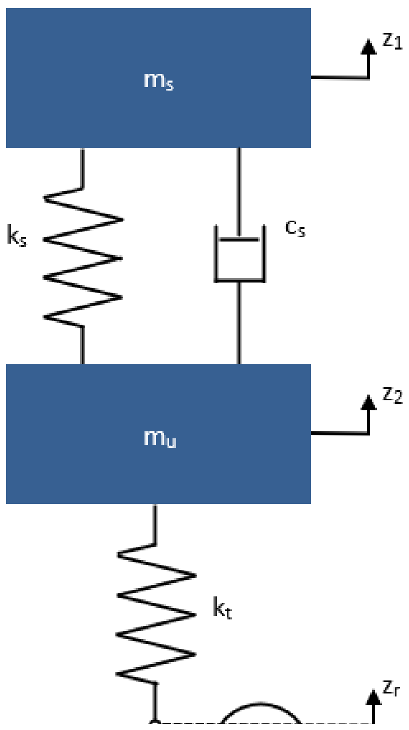


Figure 4. Two Degrees of Freedom Quarter Car Model (Goodarzi, Lu, & Khajepour, 2023).

The quarter car model with 2 degree of freedom is shown in Figure 2. Model includes both unsprung mass and sprung mass vertical motions. The unsprung mass mu represents the wheel and its associated components, and the sprung mass ms represents the corresponding vehicle's body mass to the wheel (approximately a quarter of the body mass). The vertical motions of the sprung and unsprung masses are represented by z_1 and z_2, respectively. Suspension stiffness and damping coefficient are ks and c_s. Tire vertical stiffness is k_t and is damping is usually negligible with respect to the suspension damping (Goodarzi, Lu, & Khajepour, 2023).

Assuming $z_r < z_2(t) < z_1(t)$, free body diagrams for each mass:

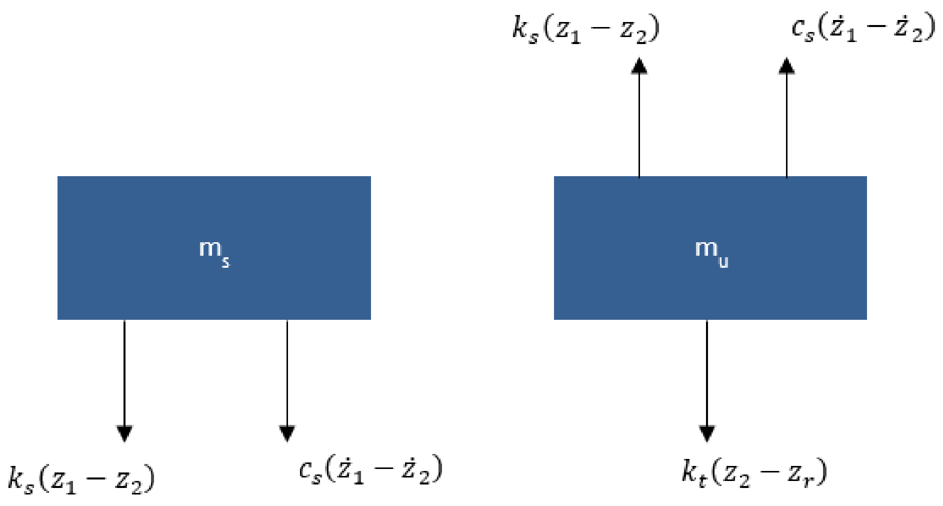


Figure 5. Free body diagram of each mass.

Applying the Newton's Second Law:

$$\sum F = ma$$

$$m_s\ddot{z}_1 = -c_s(\dot{z}_1 - \dot{z}_2) - k_s(z_1 - z_2) \tag{1}$$

$$m_u\ddot{z}_2 = c_s(\dot{z}_1 - \dot{z}_2) + k_s(z_1 - z_2) - k_t(z_2 - z_r) \tag{2}$$

Rearranging the equation (1) and (2):

$$m_s\ddot{z}_1 - c_s\dot{z}_2 + c_s\dot{z}_1 - k_s z_2 + k_s z_1 = 0 \tag{3}$$

$$m_u\ddot{z}_2 + c_s\dot{z}_2 - c_s\dot{z}_1 + (k_s + k_t)z_2 - k_s z_1 = k_t z_r \tag{4}$$

These two equations can be represented as matrix form (Lozia & Zdanowicz, 2016):

$$\begin{bmatrix} m_s & 0 \\ 0 & m_u \end{bmatrix}\begin{bmatrix} \ddot{z}_1 \\ \ddot{z}_2 \end{bmatrix} + \begin{bmatrix} c_s & -c_s \\ -c_s & c_s \end{bmatrix}\begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \end{bmatrix} + \begin{bmatrix} k_s & -k_s \\ -k_s & k_s + k_t \end{bmatrix}\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} 0 \\ k_t z_r \end{bmatrix} \tag{5}$$

The parameters were determined by reviewing previous studies. The fixed parameters were determined as the average of values reported in previous studies. The use of multiple studies will help in reducing the effect of error or outlier values of a single study. Therefore, this study has a more general and balanced parameter set.

Table 1. Example parameters derived from previous studies.

Study	m_s (kg)	m_u (kg)	k_t (N/m)	c_t (N-s/m)
(Agharkakli et al., 2012)	290	59	190000	-
(Alvarez-Sánchez, 2013)	208	28	127200	-
(Florin et al., 2013)	466.5	49.8	135000	1400
(Mitra et al., 2016)	289	30	101428	-
(Salah et al., 2010)	250	35	160000	-
(Mohite, & Mitra, 2018)	295	30	41815.66	-
(Likaj et al., 2010)	637	104	100600	-
(Nagarkar et al., 2016)	290	40	190000	200
(Ghoniem et al., 2020)	350	15	200000	-

$$m_s = \frac{\sum_{i=1}^{N} m_{s,i}}{N} = 326.125\ kg \approx 330\ kg$$

$$m_u = \frac{\sum_{i=1}^{N} m_{u,i}}{N} = 2.615\ kg \approx 45\ kg$$

$$k_t = \frac{\sum_{i=1}^{N} k_{t,i}}{N} = 150528.5\ \mathrm{N/m} \approx 150000\ \mathrm{N/m}$$

Subsequently, the ranges of the design variables were established as follows. The value ranges provided in the table below were determined based on references from studies in the literature. Considering that the simulation will be conducted in a computer environment, these ranges were chosen to be as broad as possible. Since our system is based on a quarter-car model with two degrees of freedom, having wide ranges will not significantly increase the simulation time. This approach ensures sufficient flexibility for a comprehensive analysis.

Table 2. Range of design variables.

Variable	Minimum	Maximum
k_s (N/m)	10000	40000
c_s (N.s/m)	1000	4000

3. OPTIMIZATION CRITERIA

In the literature, various methods have been proposed to optimize parameters such as spring stiffness (k_s) and damping coefficient (c_s) to improve the performance of vehicle suspension systems. In some studies, only a single performance parameter, such as vertical acceleration is optimized which represents passenger comfort. In other hand, other studies adopt a multi-objective approach, simultaneously considering multiple performance criteria such as comfort and road-holding. These criteria are often balanced to achieve an optimal trade-off. These different approaches highlight that the choice of optimization strategy can vary depending on the objectives and application scenarios.

Lozia & Zdanowicz (2016) mentioned multi-objective optimization in their study. In this study, three criteria have been adopted to assess the correctness of selection of parameter but in this study just damping coefficient of c_s is concerned. These three criteria are

- minimization of the measure of vehicle occupants' discomfort, i.e. the standard deviation of sprung mass acceleration, σ_a [m/s2];
- minimization of the safety hazard, measured by the standard deviation of the vertical component of the normal reaction at the tire/road contact, σ_F [N];
- reduction in the working displacements (range of changes in the deflection) of the suspension system to a value lower than the suspension displacement limit r_{zg} [m].

Furthermore, Nagarkar et al. (2024) also studied multi-objective optimization. While optimization RMS method was used, the study focused on parameters for ride comfort and safety, including frequency-weighted RMS acceleration,

$$A_w = \left(\frac{1}{T}\int_0^T [a_w(t)]^2 dt\right)^2 \quad (6)$$

Vibration Dose Value (VDV) can be used as a measure to access whole-body vibration:

$$VDV = \left(\int_0^T [a_w(t)]^4 dt\right)^{\frac{1}{4}} \quad (7)$$

and Maximum Transient Vibration Value (MTTV) is the maximum acceleration value of the cycle:

$$MTTV = \max\left(a_w(t)\right) \tag{8}$$

Additionally, suspension deflection and tire deflection were considered critical for handling and overall performance.

$$\text{Suspension Deflection (SD)} = x_s - x_{us} \tag{9}$$

$$RMS\ SD = \left(\frac{1}{T}\int_0^T [x_s(t) - x_{us}(t)]^2 dt\right)^2 \tag{10}$$

$$\text{Dynamic Tire Deflection(TD)} = x_{us} - x_r \tag{11}$$

$$RMS\ TD = \left(\frac{1}{T}\int_0^T [x_{us}(t) - x_r(t)]^2 dt\right)^2 \tag{12}$$

Constraints such as maximum sprung mass acceleration (4.5 m/s^2), suspension space (125 mm), and tire deflection (0.058 m) ensured safety and functionality.

Some studies have adopted an alternative approach. In these methods, the maximum acceleration value is calculated at each sampling step and the minimum of these maximum values is determined. This strategy aims to improve the suspension performance while at the same time keeping the level of discomfort experienced by the occupants within an acceptable limit and ensuring that the design complies with the specified upper limits. This approach is considered as an important step to improve both the effectiveness and reliability of the suspension system (Baumal et al., 1998).

$$\min f(\{x\}) = max|\ddot{z}_1| \tag{13}$$

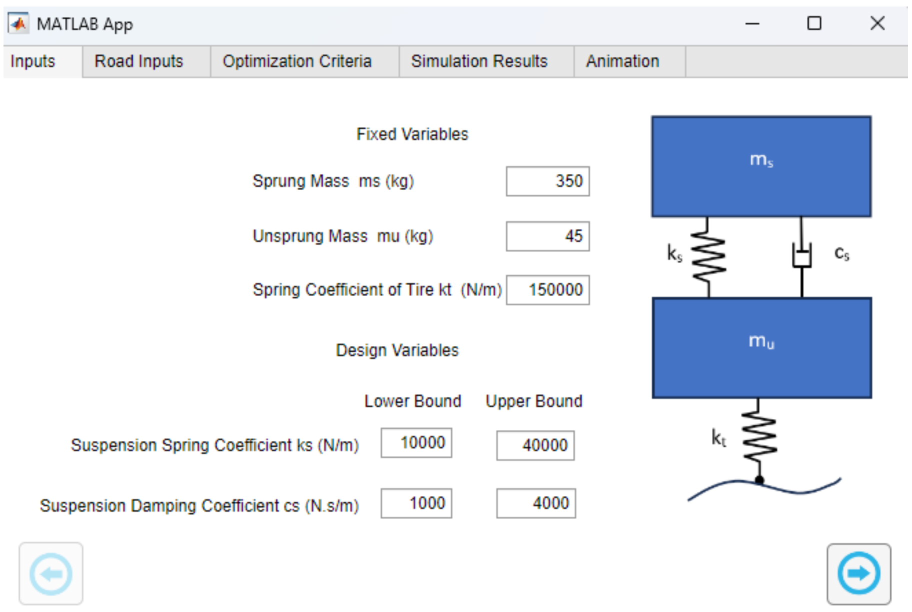
And upper limit is defined as

$$g_1 = f - 10.3\ m/s^2 \leq 0 \tag{14}$$

4. APP DEVELOPING

In this project, an interactive user interface was developed using MATLAB App Designer for the optimization of a quarter-car suspension system. The user can analyze the performance of the suspension system based on the specified input parameters, aiming to determine the most suitable suspension coefficients (k_s and c_s).
The main objective of the project is to identify the parameters that optimize the comfort level of the suspension system. For this purpose, different combinations of spring stiffness (k_s) and damping coefficient (c_s) are tested to find the values that provide the best system response.
The operation of the user interface consists of the following steps:
Defining input parameters (Inputs Section). The user initiates the optimization process by specifying the physical parameters of the vehicle's suspension system. At this stage, the following parameters must be entered as shown in Figure 6:

- Vehicle Body Mass (m_s): The sprung mass value. (kg)
- Wheel Mass (m_{us}): The unsprung mass value. (kg)
- Tire stiffness (k_t): The spring coefficient of tire (N/m)
- Spring Stiffness Range $\left(k_{s,min} - k_{s,max}\right)$: The minimum and maximum values of the suspension spring stiffness. (N/m)
- Damping Coefficient Range $\left(c_{s,min} - c_{s,max}\right)$: The minimum and maximum values of the suspension damping coefficient. (Ns/m)



Fixed Variables

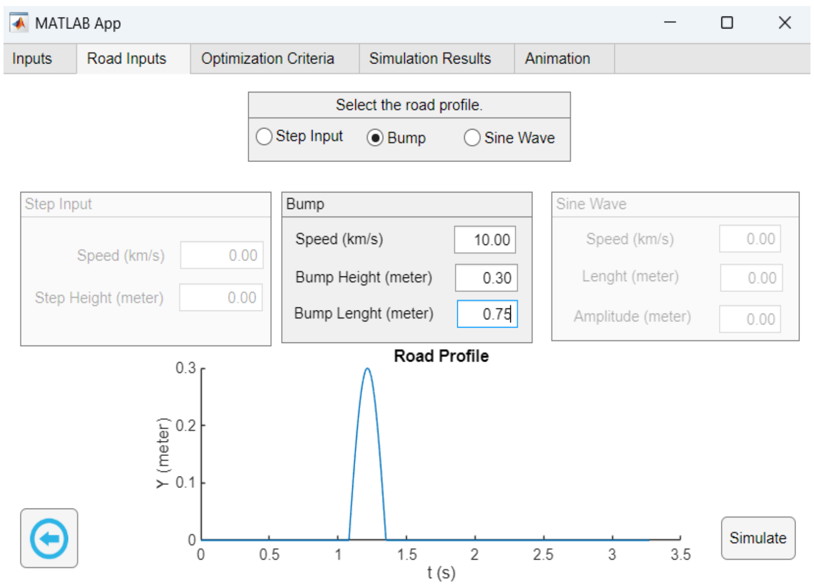
Sprung Mass ms (kg)	350
Unsprung Mass mu (kg)	45
Spring Coefficient of Tire kt (N/m)	150000

Design Variables

	Lower Bound	Upper Bound
Suspension Spring Coefficient ks (N/m)	10000	40000
Suspension Damping Coefficient cs (N.s/m)	1000	4000

Figure 6. Input parameters.

When the forward arrow key is pressed after the parameters are entered, the 'ButtonPushed' code in the appendix works. This button assigns the values entered in the app to the MATLAB workspace. It also warns the user when the values are entered incorrectly. If the user enters the values correctly, then app directs the user to the next tab which is Road Inputs. In this tab, the road profile applied to the suspension system must be defined. This study considers three different road profiles. For instance, a step input has been applied to the suspension system (Nagarkar et al., 2011), whereas others have utilized a sine wave function (Abbas et al., 2013). Last road profile is bump input. The user selects one of three different road types and enters the corresponding parameters as shown in Figure 7.



Select the road profile.

Step Input [✓ Bump] Sine Wave

Step Input	
Speed (km/s)	0.00
Step Height (meter)	0.00

Bump	
Speed (km/s)	10.00
Bump Height (meter)	0.30
Bump Lenght (meter)	0.75

Sine Wave	
Speed (km/s)	0.00
Lenght (meter)	0.00
Amplitude (meter)	0.00

Figure 7.Selection of road types.

Step Input: Represents a sudden height change in the road profile (e.g., The car driving onto the curb). The user specifies the height (m) and vehicle velocity (km/h). Although we had initially planned to use a step function, we realized that we should actually use a sigmoid function instead of a step function. Because when the step function is used, as can be seen in Figure 8, sharp corners are formed. During the development of the app, we realized that these sharp corners cause some of the constraints that we will discuss in the following sections to remain unsolved. For this reason, we chose a sigmoid function is represented by Equation (15), which is similar to the step function but provides a transition with smoothed corners as shown in Figure 9.

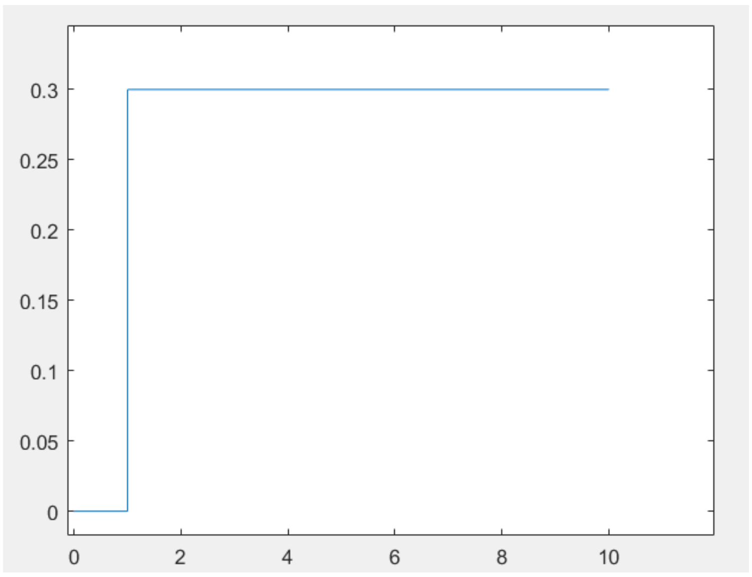


Figure 8. Step function plot.

$$f(x) = \frac{h}{1 + e^{-k(x-x_0)}} \tag{15}$$

Here h is step height, k is smoothing factor, and x_0 is offset of the curve which is where midpoint of curve is placed.

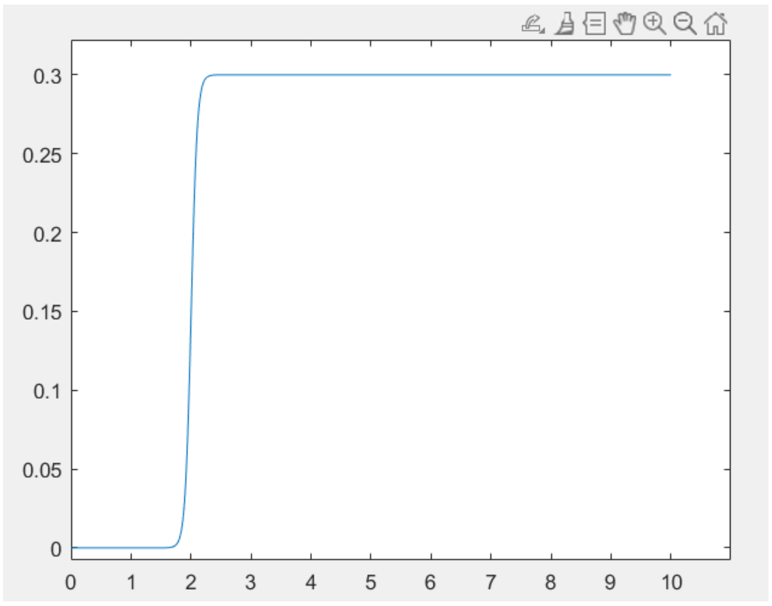


Figure 9. Sigmoid Function.

The user can also observe the road profile as a preview to the step input as values are entered. Each time a new parameter entry is made in the step input section, the 'SpeedkmsEditField_2ValueChanged' or ‘StepHeightmeterEditFieldValueChanged’ code in the appendix runs and user sees the preview.

Bump Input: Simulates the vehicle passing over a bump with a certain height and length. The user specifies the bump height (m), length (m) and vehicle velocity (km/h). The geometric properties of this bump are based on the optimal parameters provided by (Johnson, & Nedzesky, 2004). The road profile is mathematically modeled as a sine wave, representing the vertical displacement of the bump. This sine wave can be expressed by the following equation (Hassaan, 2014):

$$z_r(t) = A\sin(2\pi f t) \tag{16}$$

Here, A is the amplitude of the sine wave, which corresponds to the maximum height of the bump, and f is its frequency.

The period (T) of the sine wave is related to the bump length (L) by the equation (17):

$$\frac{T}{2} = \frac{L}{v} \tag{17}$$

Thus,

$$T = \frac{2L}{v} \tag{18}$$

where v represents the velocity of the vehicle as it passes over the bump.

Thus, the frequency (f) is determined as:

$$f = \frac{1}{T} \tag{19}$$

Substituting Equation (18) to Equation (19), frequency is got;

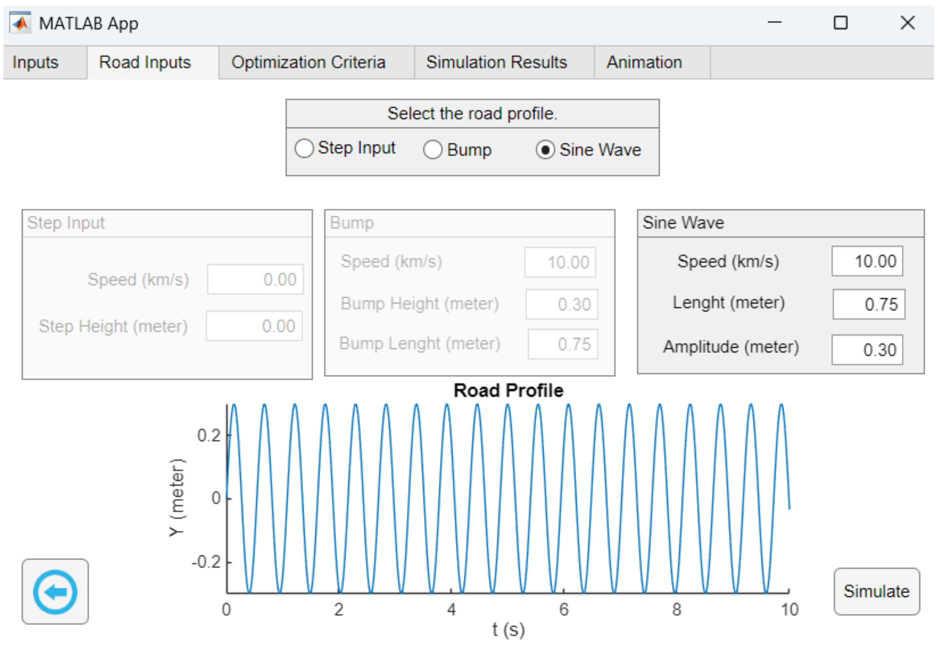
$$f = \frac{v}{2L} \tag{20}$$

This relationship was substituted into Equation (16), leading to:

$$z_r(t) = A\sin\left(2\pi\frac{v}{2L}t\right) = A\sin\left(\frac{v}{L}\pi t\right) \tag{21}$$

But Equation (21) is still wave-formed. Therefore, this wave formed is cut certain points to represent only one vertex shape. Whenever the user enters the inputs, the 'SpeedkmsEditFieldValueChanged', 'BumpHeightmeterEditFieldValueChanged' or 'BumpLenghtmeterEditFieldValueChanged' codes in the appendix runs and the user sees the preview of the bump path type as shown in Figure 7. But in this method sharp corners are occurred at cut points. These sharp corners were smoothed with the quadratic Bezier curve method.

Sine Wave: Represents a continuously undulating road surface. The user specifies the frequency (Hz), amplitude (m) and vehicle velocity (km/h). In this road input, the same logic and same equation we used in bump was used. Here, as the user enters each input, 'SpeedkmsEditField_3ValueChanged', 'LenghtmeterEditFieldValueChanged' or 'AmplitudemeterEditFieldValueChanged' codes in the appendix runs and the user sees the preview.



Select the road profile.
Step Input | Bump | [✓ Sine Wave]

Step Input
Speed (km/s) 0.00
Step Height (meter) 0.00

Bump
Speed (km/s) 10.00
Bump Height (meter) 0.30
Bump Lenght (meter) 0.75

Sine Wave
Speed (km/s) 10.00
Lenght (meter) 0.75
Amplitude (meter) 0.30

Figure 10. Sine input road profile and its preview on the app.

In addition to the road input tab, other types of road profiles can be added to this tab as options. In this study, three types of road profiles: step, bump, and sine were selected as they are commonly used in vehicle dynamics simulations and effectively represent various driving conditions. However, different road profiles can also be incorporated into the system depending on the purpose of the analysis. For example, random road profiles, ISO-defined road roughness classes, or custom user-defined profiles can be integrated to simulate more complex or real-world scenarios for more inclusive studies.

After the user selects the road profile of his choice and presses the simulate button ‘SimulateButton_2Pushed’ code in appendix is run, then it runs the script file of whichever road profile he used. In other words, we have 3 separate scripts for 3 different road profiles which ‘bumpsc.m’, ‘stepsc.m’, ‘sinesc.m’ in appendix. These scripts could have been edited in a single file, but in our study, we preferred to write separate scripts in order to detect the problems that arise during the writing phase of the code more easily. In the future development phase of this study, these scripts can be combined and the code can be converted into a simpler, compact code.

Also, when the 'SimulateButton_2Pushed' code is run, if bump road input is selected, the 'bezier.m' file in the appendix also runs. This 'bezier.m' file is used to smooth the edges of the sine function used in the bump road profile mentioned earlier and transfers the curves to Simulink.

In this script, the start (P_1) and end (P_2) points of the smoothing are determined and then the derivative of Equation (21) at the end point of the smoothing is calculated. This derivative gives the slope of the equation of the line on which our 3rd control point (P_3) and 2nd control point (P_2) lie, as shown in Figure 11. Then our 3rd control point was determined from this straight-line segment so that all control points for the quadratic Bezier curve were known. The same process was done for the other edge to be smoothed and thus the smoothing process was done.

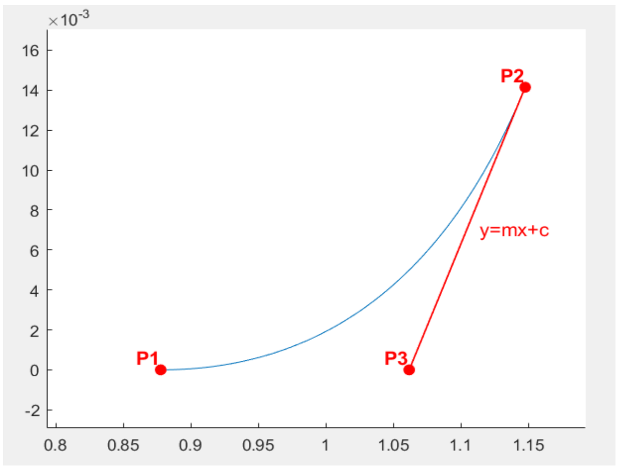


Figure 11. Illustration of how control points of quadratic Bezier curve are determined.

In fact, this 'bezier.m' function could have been written directly into the ‘bumpsc.m’ file. However, as we mentioned before, breaking the code step by step into functional script files makes it much easier to detect potential errors and react to them during the development process. Since each function works in isolation in its own file, this structure provides a significant advantage in terms of both debugging and readability of the code.

Since the values assigned via MATLAB App Designer cannot be directly accessed by Simulink, intermediary script files were utilized in this process which they are ‘bumpsc.m’, ‘stepsc.m’, ‘sinesc.m’. These scripts receive the required values from App Designer and assign them to the MATLAB workspace. Once the necessary values are transferred to the workspace, the corresponding Simulink file is executed based on the selected road profile: ‘sine.slx’, ‘step.slx’ and ‘bump.slx’. This approach enables smooth data transfer between App Designer and Simulink, allowing the simulation of user-defined scenarios without compatibility issues.

In Figures 12, 13 and 14 the block diagrams for each road profile are constructed. The blocks after the road input are the same for each Simulink file. The ‘MATLAB Function Block’ we used in these files is also the same and the code is given in the appendix as ‘MATLAB Function1’. This function block contains the equation of motion in Equation (5) and Simulink solves the equation accordingly.

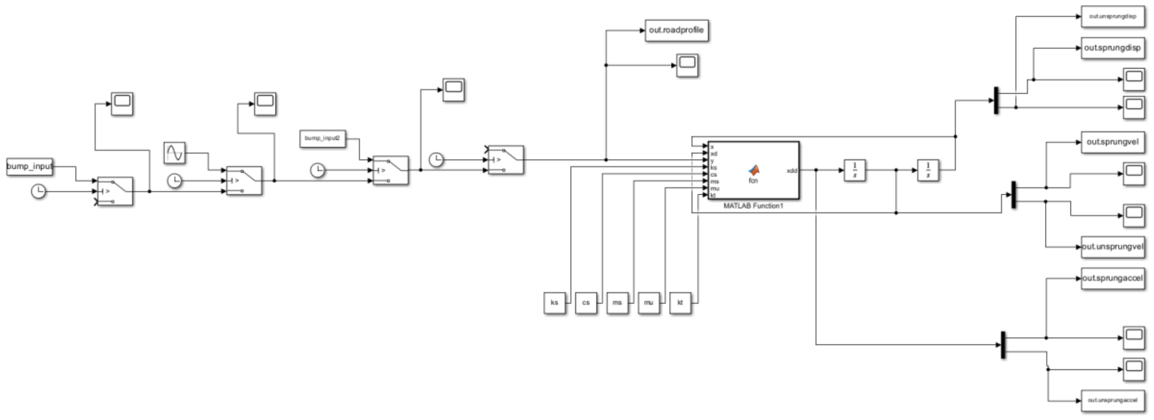


Figure 12. Block diagram of bump.slx.

As can be seen in the figure above, the road input for bump is a bit more complicated than other Simulink block diagrams. This is because as we said before, we use Bezier curve for smoothing in bump input. Basically, first the straight line comes from $y = 0$ along the x-axis, then the Bezier curve comes from the appropriate point which is calculated in the switch block, then the sine wave comes after the first Bezier curve and sine wave continues to the other side where is second Bezier curve placed and then continues as a straight line.

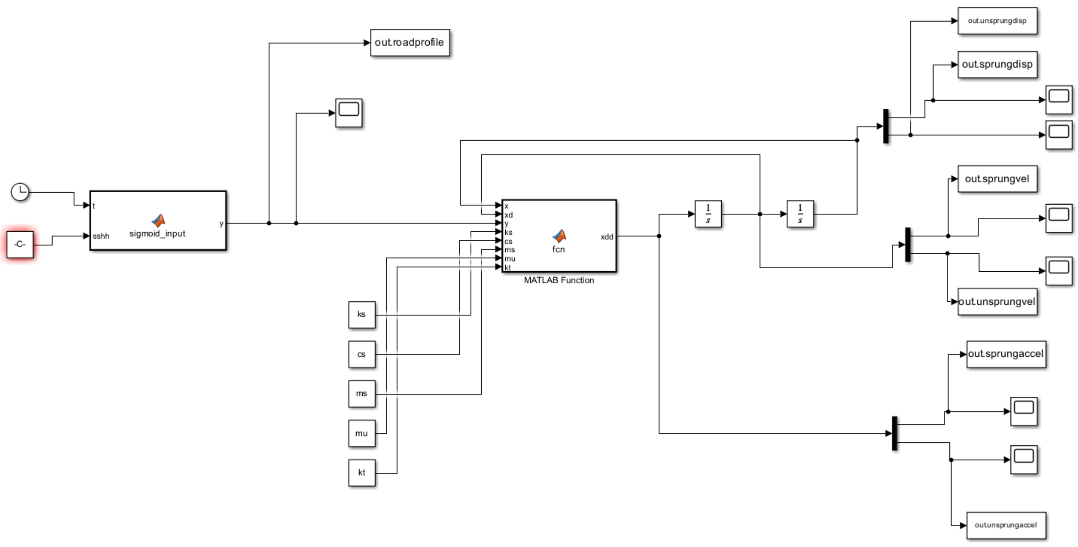


Figure 13. Block diagram of step.slx.

In the previous sections, we mentioned that we use sigmoid function instead of step input. In Figure 13, since the sigmoid function does not have a specific block, the sigmoid input was determined using the MATLAB function block specified in the appendix as 'Sigmoid Input'. A constant block containing the step height and a clock block to convert the sigmoid function into a timeseries function were used in this Simulink, thus successfully defining the road input.

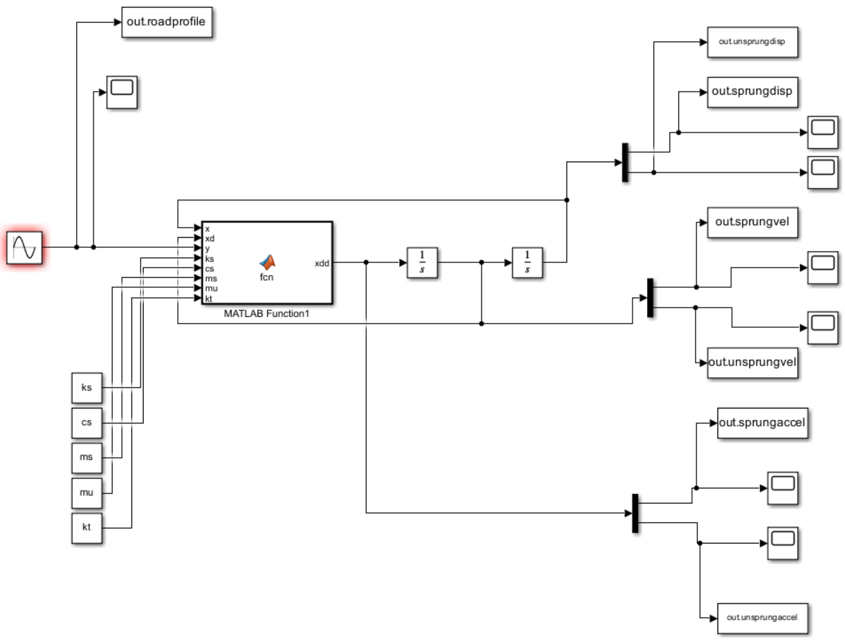
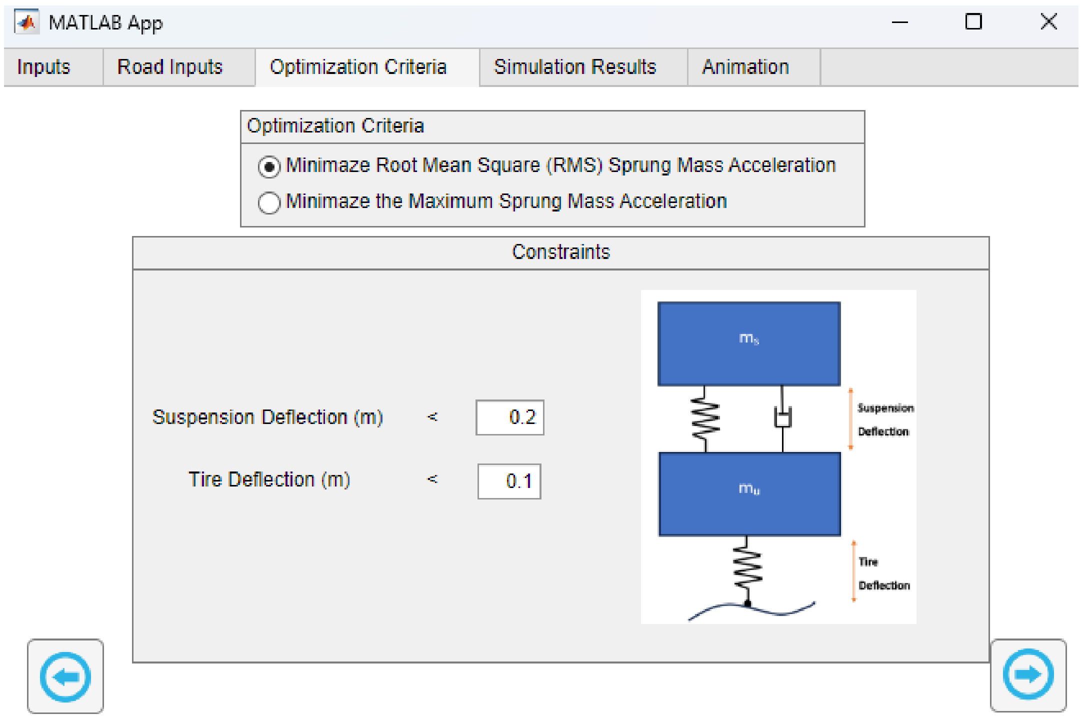


Figure 14. Block diagram of sine.slx.

In Figure 14, the system is constructed using the sine wave block. Moreover, for each k_s and c_s simulations are done and simulation results are transferred to the workspace with 'To Workspace' blocks. In the model settings, ode4 (Runge-Kutta) was used. Also, during development of app, we observed that simulations were done automatically with step size 10^{-4}, but this made the animation part too long and slow. So, we reduced the step size in the simulation settings and started using a step size of 10^{-2}. As a result, the animation part, which we will discuss about in the next sections, started to work more smoothly. When simulation is done app directs the user to third tab which is 'Optimization Criteria' as shown in Figure 15.

In this section, the user selects which performance metric will be applied during optimization. Two main comfort-based criteria are available: minimizing the root mean square (RMS) of the sprung mass acceleration or minimizing the maximum acceleration of the sprung mass. In addition, two more physical constraints are presented to the user: suspension deflection and tire deflection. These constraints limit how much the suspension and the tire can deflect.



Optimization Criteria

[✓ Minimaze Root Mean Square (RMS) Sprung Mass Acceleration]

Minimaze the Maximum Sprung Mass Acceleration

Constraints

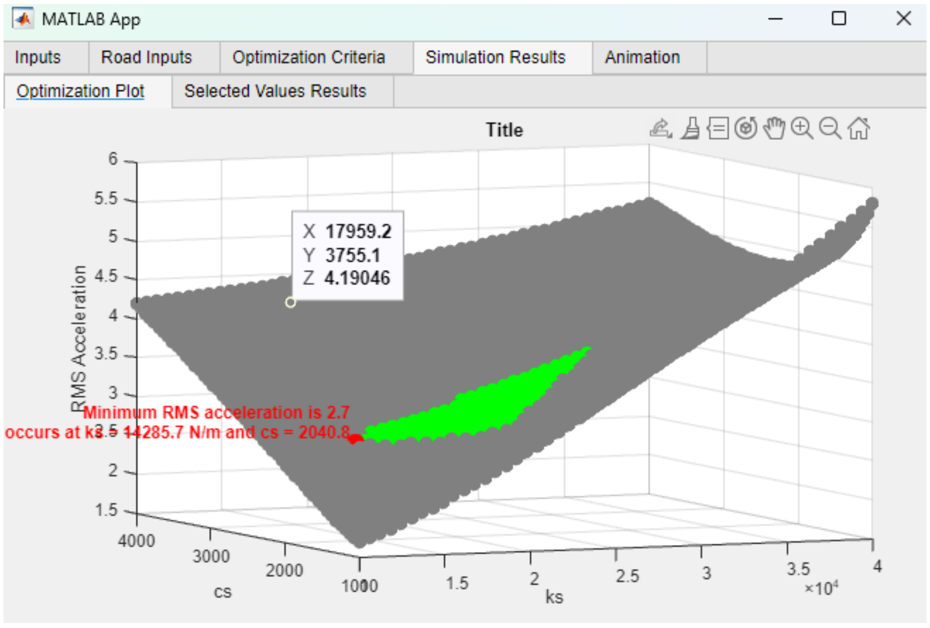
Suspension Deflection (m)	<	0.2
Tire Deflection (m)	<	0.1

Figure 15. Optimization Criteria tab.

If the user presses the forward arrow button after selecting the method and entering constrain values, the 'Button_3Pushed' code in the appendix runs. This code takes the simulation results transferred from Simulink to the workspace into its own workspace and then identifies the simulations that do not meet the constrains by using Equations 9 and 11.

If the RMS method is selected, it saves the combination with the minimum RMS value by using Equation 6, if the minimizing the maximum acceleration method is selected, it saves the minimum of the maximum sprung mass acceleration values by using Equation 13. The app then switches to the other tab, the simulation tab.

In Figure 16, the results of the simulations are displayed as a surface plot. The green points represent solutions that satisfy the specified constraints, while the gray points indicate solutions that do not meet the constraints. The point shown in red represents the optimal solution. There is also a detailed description of the optimum point on the red point, which k_s and c_s values are selected and what the optimized value is. In Figure 16, we used the RMS method as an example.



```
X 17959.2
Y 3755.1
Z 4.19046
```

Minimum RMS acceleration is 2.7
occurs at ks = 14285.7 N/m and cs = 2040.8

Figure 16. Simulation results plot.

There is also another tab inside the simulation results tab as shown in Figure 17. In this tab acceleration, velocity and displacement plots of sprung and unsprung masses for the selected values of k_s and c_s can be accessed. If the user wants to see the plots separated or combined, user can press the ‘Separated View’ button, which runs the appropriate 'SeperatedViewButtonPushed', 'SeperatedViewButton_2Pushed' or 'SeperatedViewButton_3Pushed' codes on whichever tab the user is on and provides whichever type of plot the user wants to observe.

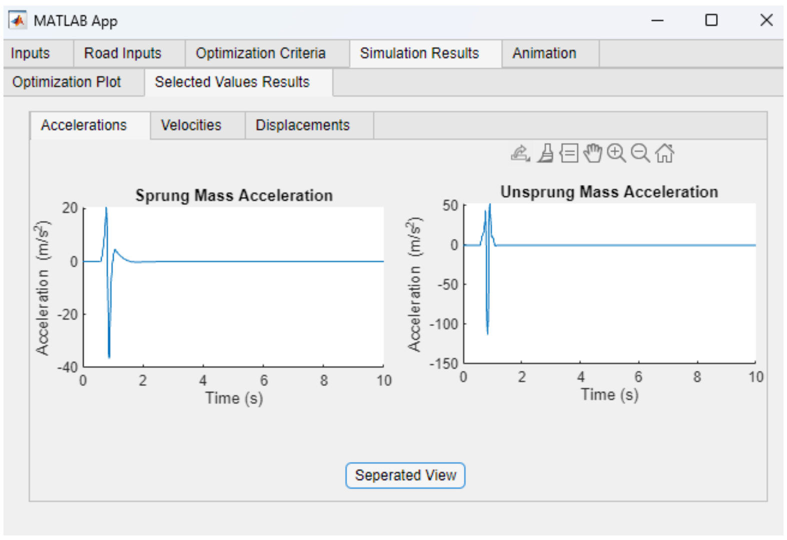


Figure 17. Sprung mass and Unsprung mass acceleration, displacement, and velocity for the optimal Ks and Cs values.

After examining the plots, the user can switch to the ‘Animation’ tab to understand the system better, this tab is shown in Figure 18. When the user presses the animate button in this tab, the ‘AnimateButtonPushed’ code in the appendix runs. This code separately runs the script file ‘animationbump.m’, ‘animationsine.m’ or ‘animationstep.m’ in the appendix according to the road input. These animation script files were developed based on the study by Mendes (Mendes, 2022) found in the literature, and were modified for our own study.

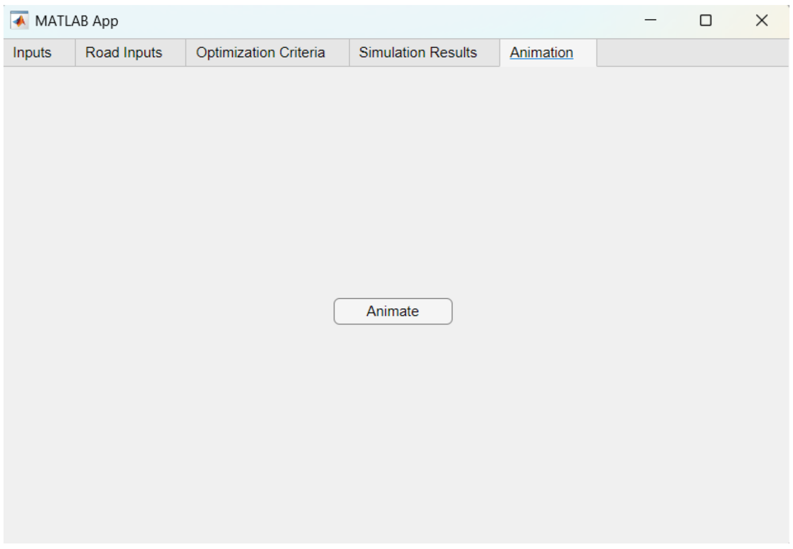


Figure 18. Animation tab.

In addition, these script files require two additional MATLAB function files provided in the Appendix: plotDamper.m and plotSpring.m. These function scripts run automatically within the animation scripts using appropriate parameter values and provide visual representations of the spring and damper elements. As illustrated in Figure 19, the animation of the system allows the user to easily observe the point at which the system reaches its maximum deflection, enabling a better understanding of its dynamic behavior. Furthermore, by modifying the parameters of the road input, the user can investigate how the system responds under different operating conditions, which offers valuable insights for the design and optimization of the suspension system.

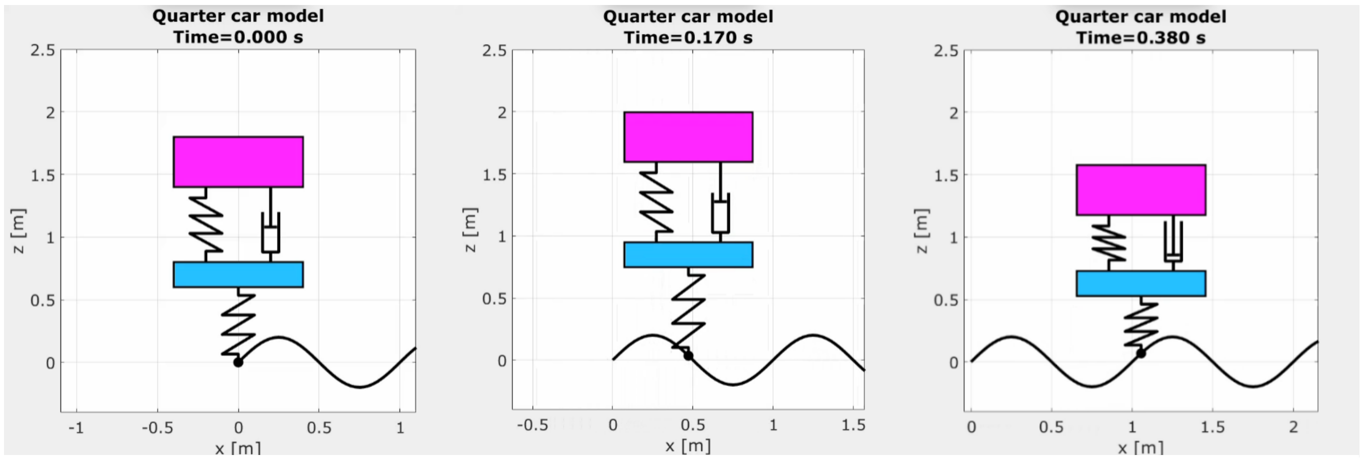


Figure 19. As an example, under a bump input, the system's position can be observed at its equilibrium state (t = 0 s), at the time it approximately reaches maximum deflection (t ≈ 0.170 s), and at the point of maximum compression (t ≈ 0.380 s).

5. RESULTS AND DISCUSSION

The selected ranges for spring stiffness (k_s) and damping coefficient (c_s) were divided into 50 equal parts, resulting in 2500 simulations in total. If desired, the user can change the number of divisions and ranges from both the script and the app interface by manually. The input parameters were determined by taking the average of values used in previous studies (see Table 1 and Table 2).

- $m_s = 350\ kg$
- $m_u = 45\ kg$
- $k_t = 150000\ N/m$
- $k_s = 10000 - 40000\ N/m$
- $c_s = 1000 - 4000\ N.s/m$

The first stage of our simulation was carried out based on the step input. The road parameters are:

- $Step\ height = 0.15\ m$
- $Speed = 10\ km/h$

Under these conditions, the RMS acceleration was used to analyze the dynamic response of the system. Additionally, to evaluate the system performance, the constraints were defined as suspension deflection < 0.08 m and tire deflection < 0.033 m.

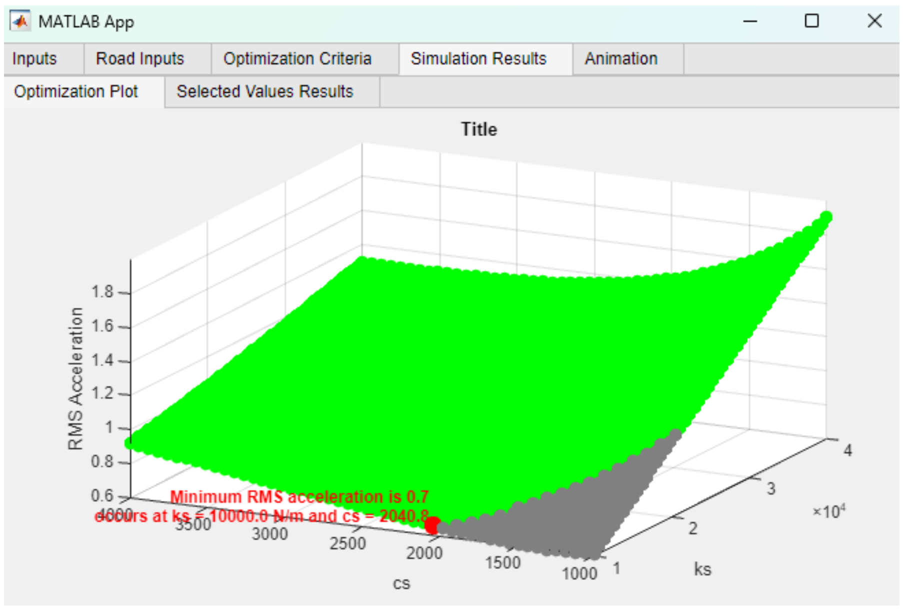

Minimum RMS acceleration is 0.7
occurs at ks = 10000.0 N/m and cs = 2040.8

Figure 20. Result plot of step input with 0.15 height and vehicle velocity of 10 km/h.

The graph above shows the optimization results of the simulation. Stiffness and damping coefficient were divided into 50 equal steps within the specified ranges, resulting in a totally of 2500 different combinations. For each combination, the system's RMS acceleration was calculated, and compliance with the defined constraints suspension deflection < 0.08 m and tire deflection < 0.033 m was evaluated.

In the simulations, when the constraints are tightened meaning the maximum allowable deflections are reduced the values of (k_s) and (c_s) need to increase to meet these conditions. This indicates that the system requires a stiffer suspension structure. As a result of this change, the green area on the graph representing the combinations that satisfy the constraints becomes narrower, meaning the number of suitable k_s-c_s combinations decrease.

The red-marked dot represents the optimal Ks and Cs combination, which achieves the minimum RMS acceleration value (0.7 m/s^2) while also satisfying the defined constraints. At this optimum point:

- $\boldsymbol{k_s = 10000\, N/m}$
- $\boldsymbol{c_s = 2040.8\, N.s/m}$

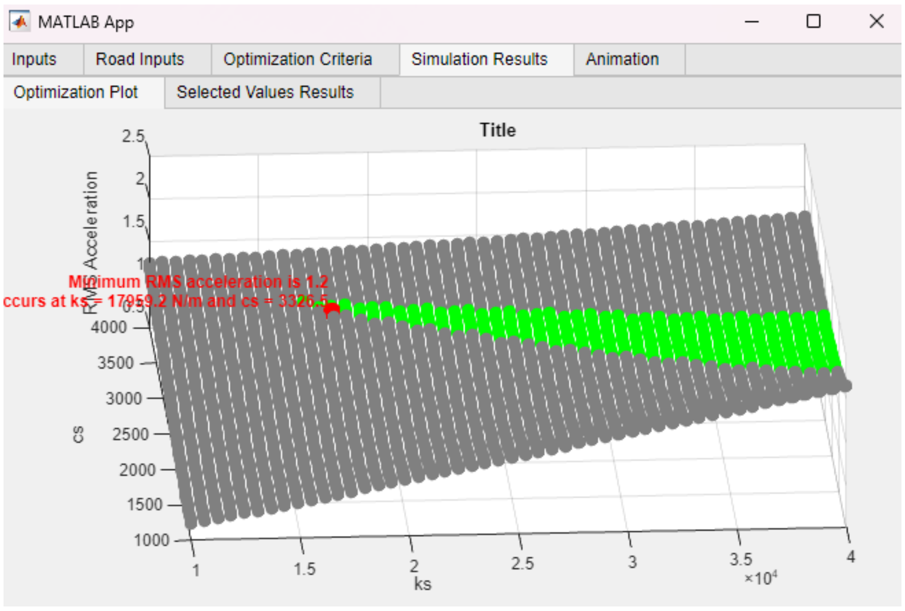
When the vehicle speed was changed to 15 km/h under the same condition, it was observed that the system was exposed to more sudden and higher-amplitude dynamic effects.

In the simulations, the green region significantly decreased as the speed increased, as shown in Figure 21. The main reason for this is that higher speeds lead to increased suspension deflections. As observed in the graph, this change reduces the number of suitable parameter combinations and shifts the optimal point toward higher values of spring and damping coefficients.

Optimum results for 15 km/h are:

- **Minimum RMS acceleration value is 1.2 m/s^2**
- $\boldsymbol{k_s = 17959.2\, N/m}$
- $\boldsymbol{c_s = 3326.5\, N.s/m}$

Our optimum result for 15 km/h speed: The RMS acceleration value increased with increasing speed. The RMS acceleration measured as 0.7 m/s^2 at 10 km/h increased to 1.2 m/s^2 when the speed increased to 15 km/h. The main reason for this increase is that at higher speeds, the response time of the suspension system to road disturbances decreases, leading to higher acceleration. At the same time, the response of the system also changes with the increase in speed, especially the increases observed in the suspension stiffness (k_s) and damping coefficient (c_s) values indicate that the system is trying to adapt to these new dynamics. However, these increases cannot completely compensate for the increase in acceleration. Therefore, the increase in speed causes the suspension system to enter a more challenging operating range in terms of comfort.



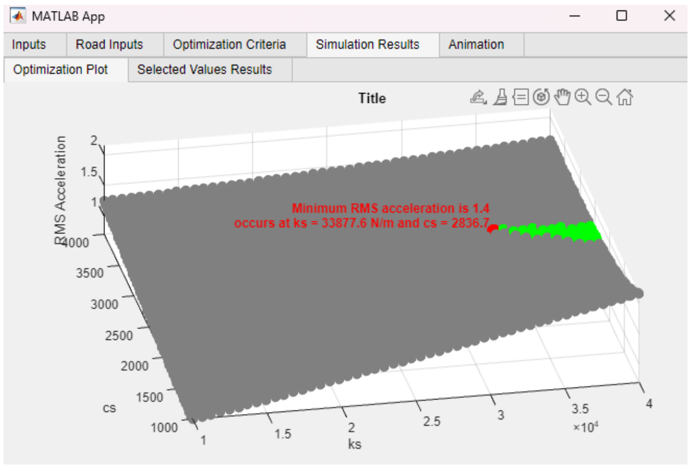
Minimum RMS acceleration is 1.2
ccurs at ks = 17959.2 N/m and cs = 3326.5

Figure 21. Optimization results under the same conditions except speed is 15 km/h.

To see the effect of constraints on the system the constraints were further tightened to suspension deflection < 0.075 m and tire deflection < 0.033 m, the simulation was still able to produce results; however, the feasible solution space became extremely limited. As a result, the surface plot displayed only a few green points, while most of the combinations violated the constraints and were shown as gray points.

Optimization results when suspension deflection is decreased from 0.08 m to 0.075 m:

- **Minimum RMS acceleration value is 1.4 m/s²**
- $\boldsymbol{k_s \ = \ 33877.6\, N/m}$
- $\boldsymbol{c_s \ = \ 2836.7\, N.s/m}$



Minimum RMS acceleration is 1.4
occurs at ks = 33877.6 N/m and cs = 2836.7

Figure 22. Optimization results after reducing the suspension deflection constraint from 0.08 m to 0.075 m.

In this case, reducing the suspension deflection limit from 0.08 m to 0.075 m caused the system to operate under tighter geometric constraints and this change led to a significant increase in the suspension stiffness (k_s) value. The increase in k_s from 17959.2 to 33877.6 is due to the fact that the system requires a much stiffer spring characteristic to provide the same dynamic performance in a more limited range of motion. This is because when the deflection limit decreases, the physical distance over which the suspension can absorb impacts decreases; in this case, the system needs to absorb the same energy over a shorter distance. This is achieved directly by increasing the spring coefficient. On the other hand, the damping coefficient (c_s) controls more the velocity-related forces and affects the vibration behavior of the system rather than the deflection limit. Therefore, lowering the deflection limit had a more limited effect on the c_s, with only a slight change in value from 3326.5 to 2836.7. This difference shows that the c_s is more sensitive to the system geometry, while the c_s is more related to dynamic energy dissipation and vibration management.

When the constraints were slightly tightened to suspension deflection < 0.075 m and tire deflection < 0.03 m, the simulation results did not yield any feasible solution, and the surface plot was composed entirely of gray points.

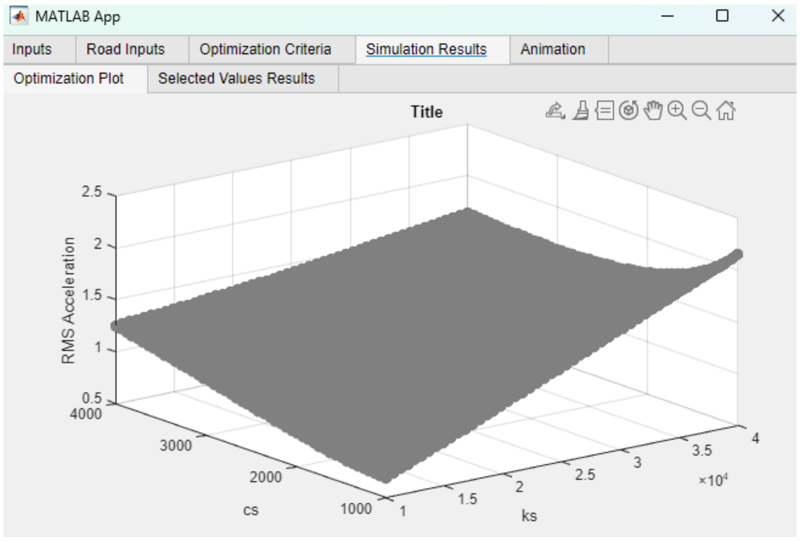


Figure 23. Optimization results after reducing the tire deflection constraint from 0.033 m to 0.03 m.

In the second stage of the simulation, a bump input was used. The bump parameters are:

- $Bump\ height = 0.30\ m$
- $Bump\ length = 0.75\ m$
- $Speed = 15\ km/h$

For the selected road profile, we chose to use RMS acceleration to analyze the system's dynamic behavior. To reduce the motion of the masses and improve comfort, the constraints were set as suspension deflection < 0.2 m and tire deflection < 0.1 m. optimal combination that achieves the lowest RMS acceleration is (see Figure 24):

- **Minimum RMS acceleration value is 2.7 m/s²**
- $\boldsymbol{k_s = 14285.7\ N/m}$
- $\boldsymbol{c_s = 2040.8\ N.s/m}$

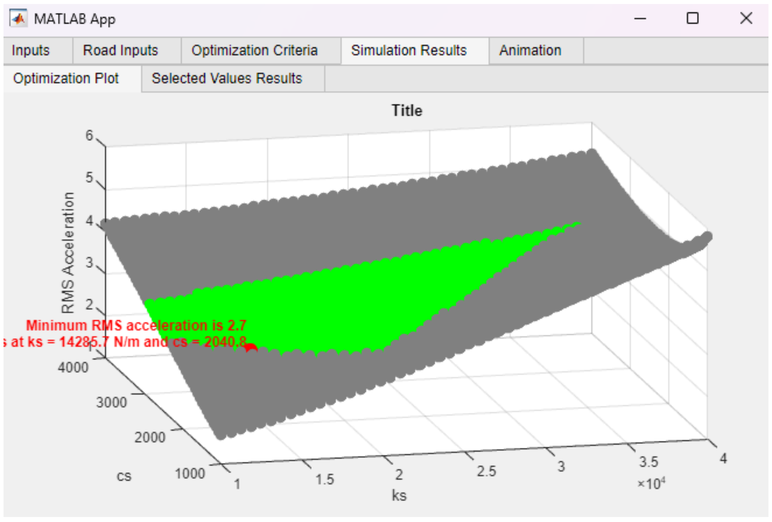


Figure 24. Result plot of bump input with 0.30m height, 0.75m length and vehicle velocity of 15 km/h.

Our optimization criterion is changed to minimizing the peak sprung-mass acceleration from the RMS method. Since road input and constraints were not changed the plot in Figure 25 is the same as the plot in Figure 24, only the optimal point has changed. Optimal combination is (see Figure 25):

- **Minimum peak acceleration value is 24.3 m/s²**
- $\boldsymbol{k_s = 21020.4\ N/m}$
- $\boldsymbol{c_s = 1612.2\ N.s/m}$

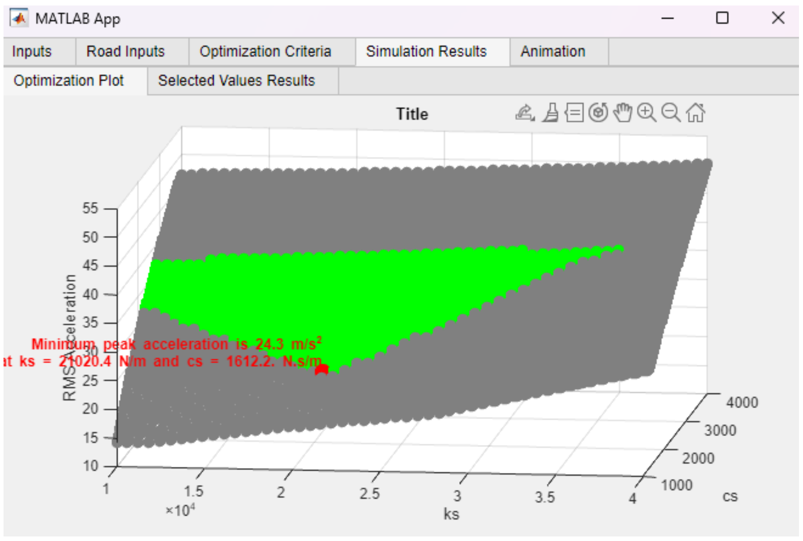


Figure 25. Optimization results under the same conditions, except that the optimization criterion is minimizing the sprung mass acceleration instead of RMS acceleration.

The optimization based on the RMS acceleration criterion resulted in a softer and more damped structure with a k_s value of 14285.7 and a c_s value of 2040.8, as the system tries to minimize overall vibration levels. This results in an overall more comfortable ride along the road, although the maximum acceleration may be higher in the sudden impacts (e.g. a bump entry). By changing the optimization criterion to minimum peak acceleration, the system focuses on reducing the peak acceleration at the moment of impact. In this case, the k_s value increased to 21020.4, meaning the suspension became stiffer, while the c_s value decreased to 1612.2. The stiffer spring helps to reduce peak acceleration by absorbing the impact in less time, while the lower damping coefficient allows the system to recover faster. This change shows that targeting peak acceleration instead of RMS acceleration creates a structure that is more resistant to short-term shocks, compromising the overall comfort characteristics of the system.

The third stage of our simulation was carried out based on the sine input. The road parameters are:

- $Sine\ height = 0.25\ m$
- $Sine\ length = 0.70\ m$
- $Speed = 10\ km/h$

For the selected road profile, which is the sine wave, RMS acceleration was used to analyze the system's dynamic behavior. To reduce the motion of the masses and improve comfort, the constraints were set as suspension deflection < 0.3 m and tire deflection < 0.2 m. Although these constraints may seem larger compared to those used for the bump profile, the sine wave has a different nature. The sine wave causes the vehicle to oscillate continuously and regularly, which leads to the system being exposed to vibrations for a longer period. While the bump input results in a short and sudden reaction, the sine wave creates a longer and more continuous effect. Therefore, the smaller constraints used for the bump input do not yield suitable results under the sine wave input. For the system to function properly, wider constraints were defined. Optimal combination is (see Figure 26):

- **Minimum RMS acceleration value is 47.0 m/s²**
- $\boldsymbol{k_s \;=\; 40{,}000\; N/m}$
- $\boldsymbol{c_s \;=\; 3204.1\; N.s/m}$

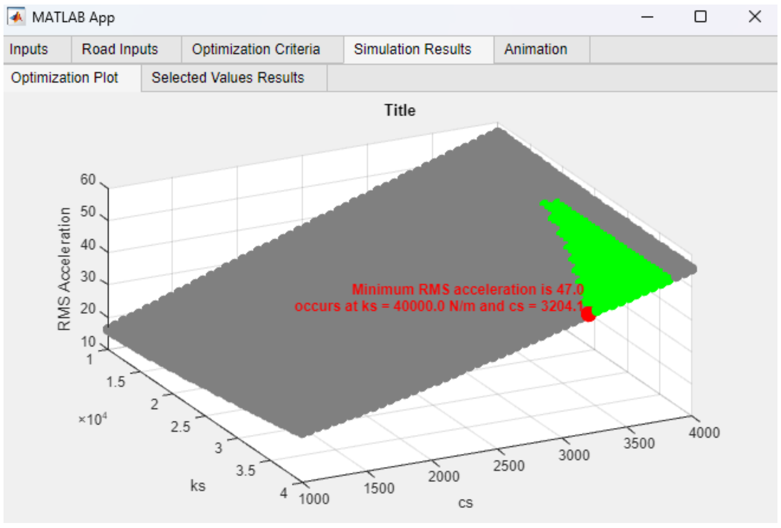


Figure 26. Result plot of sine wave input with 0.25m height, 0.70m length and vehicle velocity of 10 km/h.

As can be seen in Figure 26, optimized k_s value was at the limit of the k_s value we initially set. To prevent this, we increased the maximum value of 40000 N/m to 80000 N/m. Thus, when we simulated the system again, we obtained the plot in Figure 27.

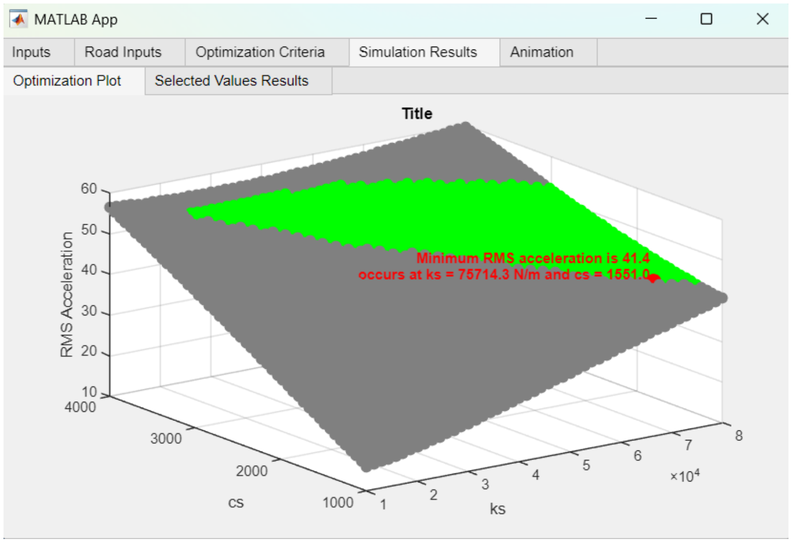


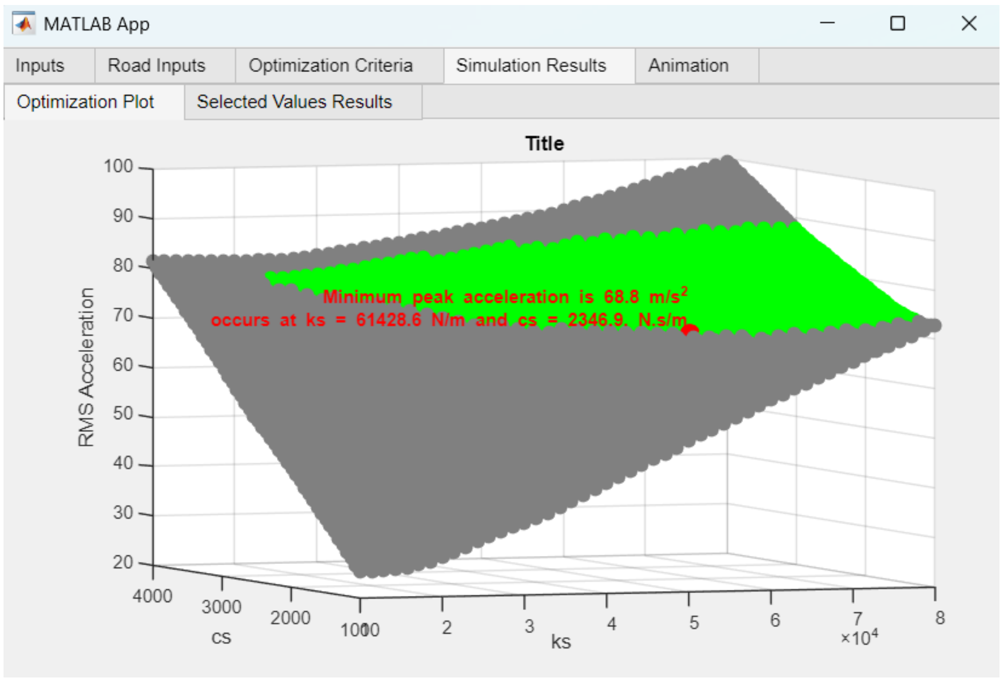
Figure 27. New optimization plot for wide range of $\boldsymbol{k_s}$.

Thus, new optimal combination is:

- **Minimum RMS acceleration value is 41.4 m/s²**
- $\boldsymbol{k_s = 75714.3\ N/m}$
- $\boldsymbol{c_s = 1551\ N.s/m}$

When a sine wave road profile is used, the system is exposed to continuous and regular vibrations. This means the system is constantly oscillating throughout the entire simulation. As a result, focusing only on the peak acceleration does not significantly change the system's overall behavior, because the excitation is continuous. In other words, the system's peak response and its average behavior are quite close under a sine wave input. As shown in Figure 28, there is no significant change in k_s and c_s values. In addition, the difference in acceleration values in the previous road inputs was not observed in the sine wave and the RMS acceleration value increased from 41.4 m/s^2 to 68.8 m/s^2. And optimal combination becomes:

- $\boldsymbol{k_s = 61428.6\ N/m}$
- $\boldsymbol{c_s = 2346.9\ N.s/m}$



Minimum peak acceleration is 68.8 m/s^2
occurs at ks = 61428.6 N/m and cs = 2346.9. N.s/m

Figure 28. Optimization plot for minimize peak acceleration method was applied.

However, when the suspension deflection limit is reduced from 0.3 m to 0.29 m and the tire deflection limit from 0.2 m to 0.19 m, this seemingly small difference creates a significant restriction for the system. Since suspension movement already occurs within a limited physical range, narrowing this range by even a centimeter leads to the elimination of many parameter combinations. With this constraint range optimal combination is (see Figure 29):

- $k_s = 68571.4\ N/m$
- $c_s = 2653.1\ N.s/m$

Also, minimum peak acceleration value is 76.8 m/s^2 (see Figure 29).

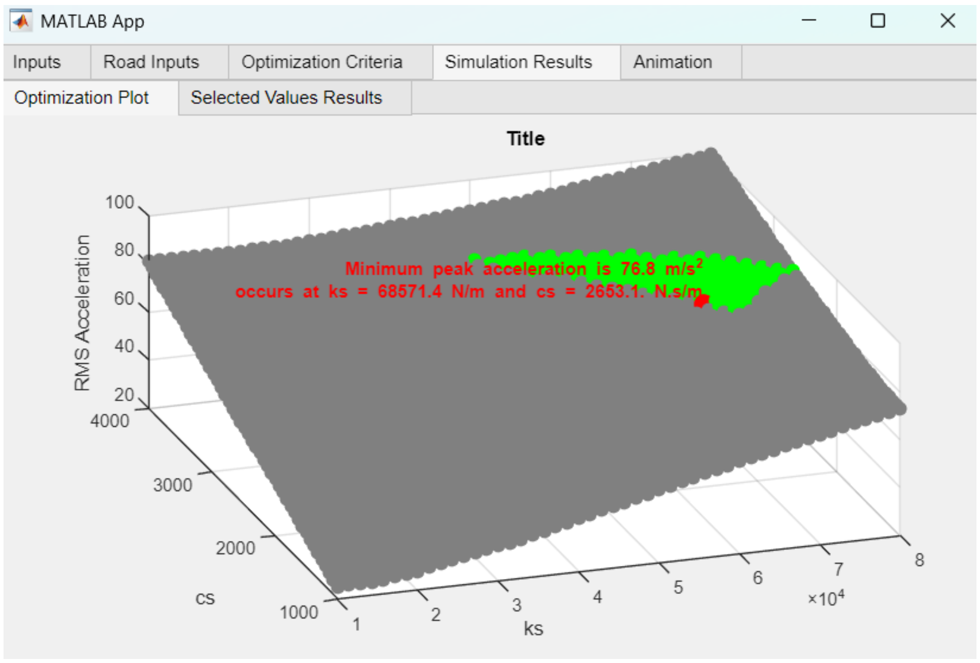


Figure 29. Optimization results after reducing the suspension deflection constraint from 0.30 m to 0.29 m and the tire deflection constraint from 0.20 m to 0.19 m.

6. CONCLUSION

In this project, we focused on improving ride comfort in a quarter-car suspension system through modeling, simulation, and optimization techniques. The entire study was built around a user-oriented MATLAB App Designer interface that allowed users to easily modify input parameters, choose road profiles, define physical constraints, and observe results interactively. Our primary goal was not to develop a complex, multi-functional vehicle system, but rather to understand how suspension parameters can be adjusted to minimize the discomfort caused by road irregularities.

We specifically concentrated on optimizing two key parameters of the suspension system: the spring stiffness (k_s) and the damping coefficient (c_s). We strictly focused on ride comfort. This choice allowed us to simplify the problem and go deeper into understanding how different combinations of k_s and c_s affect the vibrations experienced by passengers. For this purpose, we used both RMS (root mean square) acceleration and peak acceleration of the sprung mass as performance metrics, depending on the selected optimization mode.

To test the behavior of the system under various road conditions, three typical input profiles step, bump, and sine wave were implemented. Each input profile presented a different kind of disturbance to the vehicle. During simulations, we observed how the system responded to each road input and how changes in speed or constraint tightness influenced comfort. For instance, increasing vehicle speed led to higher RMS acceleration values, which indicated a decrease in ride quality. Similarly, narrowing the limits for suspension or tire deflection forced the system to adopt stiffer settings, which often reduced comfort.

One of the key contributions of this work is the interactive application that visualizes not only the results of simulations but also the system's dynamic behavior through animations. These visualizations allowed for an intuitive understand of how different suspension setups influence passenger comfort. Additionally, the use of smoothing functions like sigmoid curves and Bezier transitions helped us resolve common simulation issues caused by sharp discontinuities in road inputs.

Furthermore, this study incurred no additional cost, as all modeling, simulation, and interface development processes were carried out using the MATLAB license provided by our university. This makes the application an accessible and sustainable tool for similar educational environments.

In summary, this project demonstrated how a simplified yet thoughtfully constructed quarter-car model can be used to optimize suspension settings for maximum ride comfort. While we did not aim to address handling, road-holding, or safety trade-offs, our focused approach enabled us to extract meaningful and practical insights to passenger comfort. The developed tool provides a strong base for further academic exploration or real-world adaptation in comfort-focused suspension design tasks.

REFERENCES

Abbas, W., Emam, A., Badran, S., Shebl, M., & Abouelatta, O. (2013). Optimal seat and suspension design for a half-car with driver model using genetic algorithm.

Agharkakli, A., Sabet, G. S., & Barouz, A. (2012). Simulation and analysis of passive and active suspension system using quarter car model for different road profile. International Journal of Engineering Trends and Technology, 3(5), 636-644.

Alvarez-Sánchez, E. (2013). A quarter-car suspension system: car body mass estimator and sliding mode control. Procedia Technology, 7, 208-214.

Mendes, A. de S. (2022). Quarter car model (https://github.com/andresmendes/Quarter-car-model/releases/tag/1.0.2), GitHub. Retrieved June 5, 2025.

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.

Desai, R., Guha, A., & Seshu, P. (2020). Investigations on the Human Body and Seat Suspension Response Using Quarter, Half and Full Car Models. New Advances in Mechanisms, Mechanical Transmissions and Robotics, 507–516. https://doi.org/10.1007/978-3-030-60076-1_46

Georgiou, G., Verros, G., & Natsiavas, S. (2007). Multi-objective optimization of quarter-car models with a passive or semi-active suspension system. Vehicle System Dynamics, 45(1), 77–92. https://doi.org/10.1080/00423110600812925

Ghoniem, M., Awad, T., & Mokhiamar, O. (2020). Control of a new low-cost semi-active vehicle suspension system using artificial neural networks. *Alexandria Engineering Journal*, *59*(5), 4013-4025.

Goodarzi, A., Lu, Y., & Khajepour, A. (2023). Analysis and Design of Suspension Mechanisms. Vehicle Suspension System Technology and Design, 27–70. https://doi.org/10.1007/978-3-031-21804-0_3

Grillneder S. Multifaceted personality: predictive active suspension in the A8 flagship model. Ingolstadt: Audi MediaCenter; 2019 Jul 18. Available from: https://www.audi-mediacenter.com/en/press-releases/multifaceted-personality-predictive-active-suspension-in-the-a8-flagshipmodel-11905

Hassaan, G. A. (2014). Car dynamics using quarter model and passive suspension, part I: effect of suspension damping and car speed. *International Journal of Computer Techniques*, *1*(2), 1-9.

Jazar, R. N. (2017). Vehicle Dynamics. Springer International Publishing. https://doi.org/10.1007/978-3-319-53441-1

ME4051 Automotive Engineering Lecture Notes. [Unpublished lecture notes]. Marmara University (2025).

L. Johnson and A. Nedzesky, "A comparative study of speed humps, speed slots and speed cushions", Institution of Transportation Engineers, 2004.

Likaj, R., Shala, A., Bruqi, M., & Qelaj, M. (2010, September). Optimal design of quarter car vehicle suspension system. In 14th International Research/Expert Conference (pp. 11-18).

Lozia, Z., & Zdanowicz, P. (2016). Optimization of damping in the passive automotive suspension system with using two quarter-car models. IOP Conference Series: Materials Science and Engineering, 148, 012014. https://doi.org/10.1088/1757-899x/148/1/012014

Martins, I., Esteves, M., Pina da Silva, F., & Verdelho, P. (n.d.). Electromagnetic hybrid active-passive vehicle suspension system. 1999 IEEE 49th Vehicular Technology Conference (Cat. No.99CH36363), 3, 2273–2277. https://doi.org/10.1109/vetec.1999.778470

Mitra, A. C., Desai, G. J., Patwardhan, S. R., Shirke, P. H., Kurne, W. M., & Banerjee, N. (2016). Optimization of passive vehicle suspension system by genetic algorithm. *Procedia Engineering*, *144*, 1158-1166.

Mitschke, M. (1962). Influence of Road and Vehicle Dimensions on the Amplitude of Body Motions and Dynamic Wheel Loads (Theoretical and Experimental Vibration Investigations). SAE Technical Paper Series. https://doi.org/10.4271/620541

Mohite, A. G., & Mitra, A. C. (2018). Development of Linear and Non-linear Vehicle Suspension Model. Materials Today: Proceedings, 5(2), 4317–4326. https://doi.org/10.1016/j.matpr.2017.11.697

Nagarkar, M., Bhalerao, Y., Sashikumar, S., Hase, V., Navthar, R., Zaware, R., ... & Surner, N. (2024). Multi-objective optimization and experimental investigation of quarter car suspension system. *International Journal of Dynamics and Control*, *12*(5), 1222-1238.

Nagarkar, M. P., Patil, G. J. V., & Patil, R. N. Z. (2016). Optimization of nonlinear quarter car suspension–seat–driver model. *Journal of advanced research*, *7*(6), 991-1007.

Nagarkar, M. P., Vikhe, G. J., Borole, K. R., & Nandedkar, V. M. (2011). Active control of quarter car suspension system using linear quadratic regulator. *International Journal of Automotive and Mechanical Engineering*, *3*, 364-372.

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. https://doi.org/10.1016/j.aej.2017.01.024

Salah, A., Abbas, W., & Abouelatta, O. B. (2010). Design of optimal linear suspension for quarter car with human model using genetic algorithms. *Res. Bull. Jordan*, *11*, 42-51.

APPENDIX

AnimateButtonPushed

```
if app.SineWaveButton.Value == 1

                run("animationsine.m")

elseif app.BumpButton.Value == 1

                run("animationbump.m")

elseif app.StepInputButton.Value == 1

                run("animationstep.m")

end
```

animationbump.m

```
Ks  = evalin('base', 'optimized_ks');                          % Spring constant
suspension    [N/m]
Cs  = evalin('base', 'optimized_cs');                         % Spring constant
tire          [N/m]

% Animation model
L0_s    = 0.8;                      % Spring relaxed suspension     [m]
L0_u    = 0.6;                      % Spring relaxed tire           [m]
h_s     = 0.4;                      % Height of the sprung block    [m]
h_u     = 0.2;                      % Height of the unsprung block  [m]
a       = 0.8;                      % Width of the blocks           [m]
l_win   = 3;                        % Length window analysis        [m]

% Video
tF      = 10;                        % Final time                    [s]

simOut.tout = evalin('base', 'simOut.tout');
bumpvelocity = evalin('base', 'bumpvelocity')*1000/3600;
simOut = evalin('base', 'simOut');
kt = evalin('base', 'kt');
mu = evalin('base', 'mu');
ms = evalin('base', 'ms');
lamda = evalin('base','lamda');
smoothingfactor = evalin('base','smoothingfactor');

time    = simOut.tout;               % Time                          [s]

%% Road
% Concatenating
X_r = simOut.tout*bumpvelocity;
Z_r = simOut.roadprofile.signals.values;

%figure
%hold on ; box on ; grid on ; axis equal
%plot(X_r,Z_r,'k','LineWidth',2)
%xlabel('Distance x [m]')
%ylabel('Distance z [m]')
%title('Input')
```

```
%% Simulation

%  State space model
A = [ 0               1         0       0       ;
      -(Ks+kt)/mu      -Cs/mu     Ks/mu    Cs/mu    ;
      0               0         0       1       ;
      Ks/ms            Cs/ms      -Ks/ms   -Cs/ms   ];
B = [ 0     ;
      kt/mu  ;
      0     ;
      0     ];
C = [ 1 0 0 0 ;
      0 0 1 0 ];
D = [0 ; 0];

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

% Input
vel = bumpvelocity*1000/3600;                            % Longitudinal speed
of the car [m/s]
lon_pos = vel*time;                    % Longitudinal position of the car [m]
%u_vet = z_u;

%[y,time,x] = lsim(sys,u_vet,time);

% Sprung mass absolute vertical position (lower center point)
z_s = simOut.sprungdisp.signals.values + L0_u + L0_s;
% Unsprung mass absolute vertical position (lower center point)
z_u = simOut.unsprungdisp.signals.values + L0_u;

u_vet = simOut.unsprungdisp.signals.values+L0_u;
%u_vet = simOut.unsprungdisp.signals.values;

[y,time,x] = lsim(sys,u_vet,time);

%figure
%hold on ; grid on ; box on
%plot(X_r,z_s)
%plot(X_r,z_u)
%xlabel('Time [s]')
%ylabel('Vertical coordinate [m]')
%legend('z_s','z_u')

%% Animation

color = cool(6); % Colormap

figure
% set(gcf,'Position',[50 50 1280 720])  % YouTube: 720p
% set(gcf,'Position',[50 50 854 480])   % YouTube: 480p
set(gcf,'Position',[50 50 640 640])     % Social

% Create and open video writer object
v = VideoWriter('quarter_car_model.mp4','MPEG-4');
v.Quality   = 100;
%v.FrameRate = fR;
open(v);
```

```
t_start = (4*lamda)+((smoothingfactor));
t_end = (5*lamda)-((smoothingfactor));
t_animationstart = t_start-1;
t_animationend = t_start+1;

if t_animationstart <= 0
    t_animationstart = 0.01;
else
    t_animationstart = t_animationstart;
end

for i=int32(t_animationstart*100):int32(t_animationend*100)
%length(time)

    cla

    % Instant position
    x_inst = X_r(i);

    % Road passing by:
    set(gca,'xlim',[x_inst-l_win/2    x_inst+l_win/2],'ylim',[-0.1 -0.1+l_win])
    hold on ; grid on ; box on %; axis equal (Dont work well. It drifts
vertically)
    plot(X_r,Z_r,'k','LineWidth',3)

    set(gca,'FontName','Verdana','FontSize',16)
    title(["Quarter car model",strcat('Time=',num2str(time(i),'%.3f'),' s')])

    % Sprung mass plot
    fill([x_inst-a/2 x_inst+a/2 x_inst+a/2 x_inst-a/2],[z_s(i) z_s(i)
z_s(i)+h_s z_s(i)+h_s],color(6,:),'LineWidth',2)

    % Unsprung mass plot
    fill([x_inst-a/2 x_inst+a/2 x_inst+a/2 x_inst-a/2],[z_u(i) z_u(i)
z_u(i)+h_u z_u(i)+h_u],color(2,:),'LineWidth',2)

    % Spring
    plotSpring(L0_u,L0_s,h_u,u_vet,z_s,z_u,i,x_inst,Z_r)

    % Damper
    plotDamper(L0_s,h_u,z_s,z_u,i,x_inst)

    % Tire
    plot(x_inst,Z_r(i),'ko','MarkerFacecolor','k','MarkerSize',10)

    xlabel('x [m]')
    ylabel('z [m]')

    frame = getframe(gcf);
    writeVideo(v,frame);

end


for i=int32(t_animationstart*100):int32(t_animationend*100)
%length(time)

    cla
    % Instant position
    x_inst = X_r(i);

    % Road passing by:
    set(gca,'xlim',[x_inst-l_win/2    x_inst+l_win/2],'ylim',[-0.1 -0.1+l_win])
    hold on ; grid on ; box on %; axis equal (Dont work well. It drifts
vertically)
    plot(X_r,Z_r,'k','LineWidth',3)

    set(gca,'FontName','Verdana','FontSize',16)
    title(["Quarter car model",strcat('Time=',num2str(time(i),'%.3f'),' s')])

    % Sprung mass plot
    fill([x_inst-a/2 x_inst+a/2 x_inst+a/2 x_inst-a/2],[z_s(i) z_s(i)
z_s(i)+h_s z_s(i)+h_s],color(6,:),'LineWidth',2)

    % Unsprung mass plot
    fill([x_inst-a/2 x_inst+a/2 x_inst+a/2 x_inst-a/2],[z_u(i) z_u(i)
z_u(i)+h_u z_u(i)+h_u],color(2,:),'LineWidth',2)

    % Spring
    plotSpring(L0_u,L0_s,h_u,u_vet,z_s,z_u,i,x_inst,Z_r)

    % Damper
    plotDamper(L0_s,h_u,z_s,z_u,i,x_inst)

    % Tire
    plot(x_inst,Z_r(i),'ko','MarkerFacecolor','k','MarkerSize',10)

    xlabel('x [m]')
    ylabel('z [m]')

    frame = getframe(gcf);
    writeVideo(v,frame);

end

for i=int32(t_animationstart*100):int32(t_animationend*100)
%length(time)

    cla

    % Instant position
    x_inst = X_r(i);

    % Road passing by:
    set(gca,'xlim',[x_inst-l_win/2    x_inst+l_win/2],'ylim',[-0.1 -0.1+l_win])
    hold on ; grid on ; box on %; axis equal (Dont work well. It drifts
vertically)
    plot(X_r,Z_r,'k','LineWidth',3)

    set(gca,'FontName','Verdana','FontSize',16)
    title(["Quarter car model",strcat('Time=',num2str(time(i),'%.3f'),' s')])

    % Sprung mass plot
    fill([x_inst-a/2 x_inst+a/2 x_inst+a/2 x_inst-a/2],[z_s(i) z_s(i)
z_s(i)+h_s z_s(i)+h_s],color(6,:),'LineWidth',2)

    % Unsprung mass plot
    fill([x_inst-a/2 x_inst+a/2 x_inst+a/2 x_inst-a/2],[z_u(i) z_u(i)
z_u(i)+h_u z_u(i)+h_u],color(2,:),'LineWidth',2)

    % Spring
    plotSpring(L0_u,L0_s,h_u,u_vet,z_s,z_u,i,x_inst,Z_r)

    % Damper
    plotDamper(L0_s,h_u,z_s,z_u,i,x_inst)

    % Tire
    plot(x_inst,Z_r(i),'ko','MarkerFacecolor','k','MarkerSize',10)

    xlabel('x [m]')
    ylabel('z [m]')

    frame = getframe(gcf);
    writeVideo(v,frame);

end

close(v);
```

animationsine.m

```
Ks  = evalin('base', 'optimized_ks');                          % Spring constant
suspension    [N/m]
Cs  = evalin('base', 'optimized_cs');                         % Spring constant
tire           [N/m]

% Animation model
L0_s    = 0.8;                    % Spring relaxed suspension     [m]
L0_u    = 0.6;                    % Spring relaxed tire           [m]
h_s     = 0.4;                    % Height of the sprung block    [m]
h_u     = 0.2;                    % Height of the unsprung block  [m]
a       = 0.8;                    % Width of the blocks           [m]
l_win   = 2.2;                    % Length window analysis        [m]

% Video
tF      = 10;                      % Final time                     [s]

simOut.tout = evalin('base', 'simOut.tout');
sinevelocity = evalin('base', 'sinespeed')*1000/3600;
simOut = evalin('base', 'simOut');
kt = evalin('base', 'kt');
mu = evalin('base', 'mu');
ms = evalin('base', 'ms');
sineheight=evalin('base','sineheight');

time    = simOut.tout;             % Time                           [s]

%% Road
% Concatenating
X_r = simOut.tout*sinevelocity;
Z_r = simOut.roadprofile.signals.values;

%figure
%hold on ; box on ; grid on ; axis equal
%plot(X_r,Z_r,'k','LineWidth',2)
%xlabel('Distance x [m]')
```

```
%ylabel('Distance z [m]')
%title('Input')

%% Simulation

%  State space model
A = [ 0               1         0       0       ;
      -(Ks+kt)/mu      -Cs/mu     Ks/mu    Cs/mu    ;
      0               0         0       1       ;
      Ks/ms             Cs/ms      -Ks/ms    -Cs/ms   ];
B = [ 0     ;
      kt/mu  ;
      0     ;
      0     ];
C = [ 1 0 0 0 ;
      0 0 1 0 ];
D = [0 ; 0];

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

% Input
vel = sinevelocity*1000/3600;                           % Longitudinal speed
of the car [m/s]
lon_pos = vel*time;                   % Longitudinal position of the car [m]
%u_vet = z_u;

%[y,time,x] = lsim(sys,u_vet,time);

% Sprung mass absolute vertical position (lower center point)
z_s = simOut.sprungdisp.signals.values + L0_u + L0_s;
% Unsprung mass absolute vertical position (lower center point)
z_u = simOut.unsprungdisp.signals.values + L0_u;

%u_vet = interp1(X_r,Z_r,lon_pos)';
%u_vet = simOut.unsprungdisp.signals.values;
u_vet = simOut.unsprungdisp.signals.values+L0_u;

[y,time,x] = lsim(sys,u_vet,time);

%figure
%hold on ; grid on ; box on
%plot(X_r,z_s)
%plot(X_r,z_u)
%xlabel('Time [s]')
%ylabel('Vertical coordinate [m]')
%legend('z_s','z_u')

%% Animation

color = cool(6); % Colormap

figure
% set(gcf,'Position',[50 50 1280 720])  % YouTube: 720p
% set(gcf,'Position',[50 50 854 480])   % YouTube: 480p
set(gcf,'Position',[50 50 640 640])     % Social

% Create and open video writer object
v = VideoWriter('quarter_car_model.mp4','MPEG-4');
v.Quality   = 100;
```

```
%v.FrameRate = fR;
open(v);

for i=1:length(time)           %length(time)

    cla

    % Instant position
    x_inst = X_r(i);

    % Road passing by:
    set(gca,'xlim',[x_inst-l_win/2    x_inst+l_win/2],'ylim',[-(sineheight*2)
2.5])
    hold on ; grid on ; box on %; axis equal (Dont work well. It drifts
vertically)
    plot(X_r,Z_r,'k','LineWidth',3)

    set(gca,'FontName','Verdana','FontSize',16)
    title(["Quarter car model",strcat('Time=',num2str(time(i),'%.3f'),' s')])

    % Sprung mass plot
    fill([x_inst-a/2 x_inst+a/2 x_inst+a/2 x_inst-a/2],[z_s(i) z_s(i)
z_s(i)+h_s z_s(i)+h_s],color(6,:),'LineWidth',2)

    % Unsprung mass plot
    fill([x_inst-a/2 x_inst+a/2 x_inst+a/2 x_inst-a/2],[z_u(i) z_u(i)
z_u(i)+h_u z_u(i)+h_u],color(2,:),'LineWidth',2)

    % Spring
    plotSpring(L0_u,L0_s,h_u,u_vet,z_s,z_u,i,x_inst,Z_r)

    % Damper
    plotDamper(L0_s,h_u,z_s,z_u,i,x_inst)

    % Tire
    plot(x_inst,Z_r(i),'ko','MarkerFacecolor','k','MarkerSize',10)

    xlabel('x [m]')
    ylabel('z [m]')

    frame = getframe(gcf);
    writeVideo(v,frame);

end
```

animationstep.m

```
Ks  = evalin('base', 'optimized_ks');                            % Spring constant
suspension     [N/m]
Cs  = evalin('base', 'optimized_cs');                           % Spring constant
tire           [N/m]

% Animation model
L0_s    = 0.8;                         % Spring relaxed suspension     [m]
L0_u    = 0.6;                         % Spring relaxed tire           [m]
h_s     = 0.4;                         % Height of the sprung block    [m]
h_u     = 0.2;                         % Height of the unsprung block  [m]
a       = 0.8;                         % Width of the blocks           [m]
l_win   = 3;                           % Length window analysis        [m]
```

```
% Video
tF      = 10;                          % Final time                      [s]

simOut.tout = evalin('base', 'simOut.tout');
stepvelocity = evalin('base', 'stepspeed')*1000/3600;
simOut = evalin('base', 'simOut');
kt = evalin('base', 'kt');
mu = evalin('base', 'mu');
ms = evalin('base', 'ms');
stepheight = evalin('base', 'stepheight');

time    = simOut.tout;                 % Time                            [s]

%% Road
% Concatenating
X_r = simOut.tout*stepvelocity;
%Z_r = simOut.roadprofile.signals.values;
Z_r = stepheight ./ (1 + exp(-10 * (X_r - 5)));
%b = 10;    % eğim (ne kadar keskin olsun)
%c = 5;    % offset (sigmoid’in orta noktası)
%y = sshh ./ (1 + exp(-(b*ssss/5) * (t - (c/ssss))));;

%figure
%hold on ; box on ; grid on ; axis equal
%plot(X_r,Z_r,'k','LineWidth',2)
%xlabel('Distance x [m]')
%ylabel('Distance z [m]')
%title('Input')

%% Simulation

%  State space model
A = [ 0               1         0       0       ;
      -(Ks+kt)/mu      -Cs/mu     Ks/mu    Cs/mu    ;
      0               0         0       1       ;
      Ks/ms            Cs/ms      -Ks/ms    -Cs/ms    ];
B = [ 0     ;
      kt/mu  ;
      0     ;
      0     ];
C = [ 1 0 0 0 ;
      0 0 1 0 ];
D = [0 ; 0];

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

% Input
vel = stepvelocity*1000/3600;                             % Longitudinal speed
of the car [m/s]
lon_pos = vel*time;                  % Longitudinal position of the car [m]
%u_vet = z_u;

%[y,time,x] = lsim(sys,u_vet,time);

% Sprung mass absolute vertical position (lower center point)
z_s = simOut.sprungdisp.signals.values + L0_u + L0_s;
% Unsprung mass absolute vertical position (lower center point)
z_u = simOut.unsprungdisp.signals.values + L0_u;
```

```
u_vet = simOut.unsprungdisp.signals.values+L0_u;
%u_vet = simOut.unsprungdisp.signals.values;

[y,time,x] = lsim(sys,u_vet,time);

%figure
%hold on ; grid on ; box on
%plot(X_r,z_s)
%plot(X_r,z_u)
%xlabel('Time [s]')
%ylabel('Vertical coordinate [m]')
%legend('z_s','z_u')

%% Animation

color = cool(6); % Colormap

figure
% set(gcf,'Position',[50 50 1280 720])  % YouTube: 720p
% set(gcf,'Position',[50 50 854 480])   % YouTube: 480p
set(gcf,'Position',[50 50 640 640])     % Social

% Create and open video writer object
v = VideoWriter('quarter_car_model.mp4','MPEG-4');
v.Quality   = 100;
%v.FrameRate = fR;
open(v);

t_midbump = 5/stepvelocity;
t_animationstart = t_midbump-1;

if t_animationstart <= 0
    t_animationstart = 0.01;
else
    t_animationstart = t_animationstart;
end

t_animationend = t_midbump+1;

for i=int32(t_animationstart*100):int32(t_animationend*100)
%length(time)

    cla

    % Instant position
    x_inst = X_r(i);

    % Road passing by:
    set(gca,'xlim',[x_inst-l_win/2    x_inst+l_win/2],'ylim',[-0.1 -0.1+l_win])
    hold on ; grid on ; box on %; axis equal (Dont work well. It drifts
vertically)
    plot(X_r,Z_r,'k','LineWidth',3)

    set(gca,'FontName','Verdana','FontSize',16)
    title(["Quarter car model",strcat('Time=',num2str(time(i),'%.3f'),' s')])

    % Sprung mass plot
    fill([x_inst-a/2 x_inst+a/2 x_inst+a/2 x_inst-a/2],[z_s(i) z_s(i)
z_s(i)+h_s z_s(i)+h_s],color(6,:),'LineWidth',2)

    % Unsprung mass plot
    fill([x_inst-a/2 x_inst+a/2 x_inst+a/2 x_inst-a/2],[z_u(i) z_u(i)
z_u(i)+h_u z_u(i)+h_u],color(2,:),'LineWidth',2)

    % Spring
    plotSpring(L0_u,L0_s,h_u,u_vet,z_s,z_u,i,x_inst,Z_r)

    % Damper
    plotDamper(L0_s,h_u,z_s,z_u,i,x_inst)

    % Tire
    plot(x_inst,Z_r(i),'ko','MarkerFacecolor','k','MarkerSize',10)

    xlabel('x [m]')
    ylabel('z [m]')

    frame = getframe(gcf);
    writeVideo(v,frame);

end


for i=int32(t_animationstart*100):int32(t_animationend*100)
%length(time)

    cla

    % Instant position
    x_inst = X_r(i);

    % Road passing by:
    set(gca,'xlim',[x_inst-l_win/2    x_inst+l_win/2],'ylim',[-0.1 -0.1+l_win])
    hold on ; grid on ; box on %; axis equal (Dont work well. It drifts
vertically)
    plot(X_r,Z_r,'k','LineWidth',3)

    set(gca,'FontName','Verdana','FontSize',16)
    title(["Quarter car model",strcat('Time=',num2str(time(i),'%.3f'),' s')])

    % Sprung mass plot
    fill([x_inst-a/2 x_inst+a/2 x_inst+a/2 x_inst-a/2],[z_s(i) z_s(i)
z_s(i)+h_s z_s(i)+h_s],color(6,:),'LineWidth',2)

    % Unsprung mass plot
    fill([x_inst-a/2 x_inst+a/2 x_inst+a/2 x_inst-a/2],[z_u(i) z_u(i)
z_u(i)+h_u z_u(i)+h_u],color(2,:),'LineWidth',2)

    % Spring
    plotSpring(L0_u,L0_s,h_u,u_vet,z_s,z_u,i,x_inst,Z_r)

    % Damper
    plotDamper(L0_s,h_u,z_s,z_u,i,x_inst)

    % Tire
    plot(x_inst,Z_r(i),'ko','MarkerFacecolor','k','MarkerSize',10)
    xlabel('x [m]')
    ylabel('z [m]')

    frame = getframe(gcf);
    writeVideo(v,frame);

end

for i=int32(t_animationstart*100):int32(t_animationend*100)
%length(time)

    cla

    % Instant position
    x_inst = X_r(i);

    % Road passing by:
    set(gca,'xlim',[x_inst-l_win/2    x_inst+l_win/2],'ylim',[-0.1 -0.1+l_win])
    hold on ; grid on ; box on %; axis equal (Dont work well. It drifts 
vertically)
    plot(X_r,Z_r,'k','LineWidth',3)

    set(gca,'FontName','Verdana','FontSize',16)
    title(["Quarter car model",strcat('Time=',num2str(time(i),'%.3f'),' s')])

    % Sprung mass plot
    fill([x_inst-a/2 x_inst+a/2 x_inst+a/2 x_inst-a/2],[z_s(i) z_s(i) 
z_s(i)+h_s z_s(i)+h_s],color(6,:),'LineWidth',2)

    % Unsprung mass plot
    fill([x_inst-a/2 x_inst+a/2 x_inst+a/2 x_inst-a/2],[z_u(i) z_u(i) 
z_u(i)+h_u z_u(i)+h_u],color(2,:),'LineWidth',2)

    % Spring
    plotSpring(L0_u,L0_s,h_u,u_vet,z_s,z_u,i,x_inst,Z_r)

    % Damper
    plotDamper(L0_s,h_u,z_s,z_u,i,x_inst)

    % Tire
    plot(x_inst,Z_r(i),'ko','MarkerFacecolor','k','MarkerSize',10)

    xlabel('x [m]')
    ylabel('z [m]')

    frame = getframe(gcf);
    writeVideo(v,frame);

end

close(v);
```

bezier.m

```
bumpheight = evalin('base', 'bumpheight');
bumplength = evalin('base', 'bumplength');
bumpvelocity = evalin('base', 'bumpvelocity');
f = evalin('base', 'f');
lamda = evalin('base', 'lamda');
```

```
syms t
y = bumpheight*sin(2*pi*f*t);

DIFF = diff(y);

smoothingfactor = lamda*25/100;
assignin('base', 'smoothingfactor', smoothingfactor)

p1x = (4*lamda)-(3*(smoothingfactor));
p1y = 0;

p3x = (4*lamda)+((smoothingfactor));
p3y = double(subs(y,t,(4*lamda)+((smoothingfactor))));

m = double(subs(DIFF,t,p3x));
c = p3y-(m*p3x);

p2x = -c/m;
p2y = 0;

n=3;
n1=n-1;

[p]= [p1x p1y; p2x p2y; p3x p3y];

for    i=0:1:n1
sigma(i+1)=factorial(n1)/(factorial(i)*factorial(n1-i));  % for calculating
(x!/(y!(x-y)!)) values
end

l=[];
UB=[];

for u=0:0.002:1
for d=1:n
UB(d)=sigma(d)*((1-u)^(n-d))*(u^(d-1));
end
l=cat(1,l,UB);                                      %catenation
end

P=l*p;

line(P(:,1),P(:,2))

bump_input = timeseries(P(:,2), P(:,1));
assignin('base', 'bump_input', bump_input)
%line(p(:,1),p(:,2))

p4x = 2*(4.5*lamda) - p3x;
p4y = p3y;

p6x = 2*(4.5*lamda) - p1x;
p6y = p1y;

p5x = 2*(4.5*lamda) - p2x;
p5y = p2y;

[p1]= [p4x p4y; p5x p5y; p6x p6y];
```

```
for    i=0:1:n1
sigma(i+1)=factorial(n1)/(factorial(i)*factorial(n1-i));  % for calculating
(x!/(y!(x-y)!)) values
end

l1=[];
UB1=[];

for u=0:0.002:1
for d=1:n
UB1(d)=sigma(d)*((1-u)^(n-d))*(u^(d-1));
end
l1=cat(1,l1,UB1);                                          %catenation
end

P1=l1*p1;

line(P1(:,1),P1(:,2))

bump_input2 = timeseries(P1(:,2), P1(:,1));
assignin('base', 'bump_input2', bump_input2)
```

BumpHeightmeterEditFieldValueChanged

```
A = app.BumpHeightmeterEditField.Value;
L = app.BumpLenghtmeterEditField.Value;
v = app.SpeedkmsEditField.Value*1000/3600;

assignin('base', 'bumplength',app.BumpLenghtmeterEditField.Value);
assignin('base', 'bumpheight',app.BumpHeightmeterEditField.Value);
assignin('base', 'bumpvelocity',app.SpeedkmsEditField.Value);

f = v/(2*L);
assignin('base', 'f', f)

lamda = L/v;
assignin('base', 'lamda', lamda)
lamda = round(lamda,2);

t = 0:0.01:(lamda+3);

loc1 = find(round(t, 2) == round(5*lamda, 2));
loc2 = find(round(t, 2) == round(4*lamda, 2));

y = A*sin(2*pi*f*t);
y(loc1:end)=0;
y(1:loc2)=0;

plot(t,y);
```

BumpLengthmeterEditFieldValueChanged

```
A = app.BumpHeightmeterEditField.Value;
L = app.BumpLenghtmeterEditField.Value;
v = app.SpeedkmsEditField.Value*1000/3600;

assignin('base', 'bumplength',app.BumpLenghtmeterEditField.Value);
assignin('base', 'bumpheight',app.BumpHeightmeterEditField.Value);
assignin('base', 'bumpvelocity',app.SpeedkmsEditField.Value);

f = v/(2*L);
assignin('base', 'f', f)

lamda = L/v;
assignin('base', 'lamda', lamda)
lamda = round(lamda,2);

t = 0:0.01:(lamda+3);

loc1 = find(round(t, 2) == round(5*lamda, 2));
loc2 = find(round(t, 2) == round(4*lamda, 2));

y = A*sin(2*pi*f*t);
y(loc1:end)=0;
y(1:loc2)=0;

plot(t,y);
```

bumpsc.m

```
ks_lowerb = evalin('base', 'ks_lowerb');
ks_upperb = evalin('base', 'ks_upperb');

cs_lowerb = evalin('base', 'cs_lowerb');
cs_upperb = evalin('base', 'cs_upperb');

ks_values = linspace(ks_lowerb, ks_upperb, 10);
assignin('base', 'ks_values', ks_values);
cs_values = linspace(cs_lowerb, cs_upperb, 10);
assignin('base', 'cs_values', cs_values);

as = zeros(1001,length(ks_values)*length(cs_values));
aus = zeros(1001,length(ks_values)*length(cs_values));

vs = zeros(1001,length(ks_values)*length(cs_values));
vus = zeros(1001,length(ks_values)*length(cs_values));

ds = zeros(1001,length(ks_values)*length(cs_values));
dus = zeros(1001,length(ks_values)*length(cs_values));

i=0;

% Simulasyon döngüsü
for ks = ks_values
    for cs = cs_values
        % Workspace'e değerleri ata
        assignin('base', 'ks', ks);
        assignin('base', 'cs', cs);

        % Simulink modelini çalıştır
        %simOut = sim("step.slx", 'ReturnWorkspaceOutputs', 'on', 'base');
        simOut = sim('bump.slx', 'SrcWorkspace', 'base');
        assignin('base', 'simOut', simOut);

        % Çıkış verilerini al
        i=i+1;
        % Accelerations
        sma = simOut.sprungaccel.signals.values;
        usma = simOut.unsprungaccel.signals.values;
        as(:,i) = sma;
        assignin('base', 'as', as);
        aus(:,i) = usma;
        assignin('base', 'aus', aus);


        % Velocities
        smv = simOut.sprungvel.signals.values;
        usmv = simOut.unsprungvel.signals.values;
        vs(:,i) = smv;
        assignin('base', 'vs', vs);
        vus(:,i) = usmv;
        assignin('base', 'vus', vus);

        %Displacements
        rd = simOut.roadprofile.signals.values;
        assignin('base', 'rd', rd);
        smd = simOut.sprungdisp.signals.values;
        usmd = simOut.unsprungdisp.signals.values;
        ds(:,i) = smd;
        assignin('base', 'ds', ds);
        dus(:,i) = usmd;
        assignin('base', 'dus', dus);

    end
end
```

<u>ButtonPushed</u>

```
if app.mslamp.Visible == "on"
                app.mslamp.Visible = "off";
            end
            if app.mulamp.Visible == "on"
                app.mulamp.Visible = "off";
            end
            if app.ktlamp.Visible == "on"
                app.ktlamp.Visible = "off";
            end
            if app.kslamp.Visible == "on"
                app.kslamp.Visible = "off";
            end
            if app.cslamp.Visible == "on"
                app.cslamp.Visible = "off";
            end
            if app.InputsmustbevalidLamp.Visible == "on"
                app.InputsmustbevalidLamp.Visible = "off";
            end
            if app.InputsmustbevalidLamp_2.Visible == "on"
                app.InputsmustbevalidLamp_2.Visible = "off";
            end
            if app.InputsmustbevalidLampLabel.Visible == "on"
                app.InputsmustbevalidLampLabel.Visible= "off";
            end

            if app.SprungMassmskgEditField.Value > 0 &&
app.UnsprungMassmukgEditField.Value > 0 &&
app.SpringCoefficientofTirektNmEditField.Value > 0 ...
```

```
                    && app.SuspensionSpringCoefficientlowervalue.Value >= 0 &&
app.SuspensionDampingCoefficientlower.Value >= 0 ...
                    && app.SuspensionSpringCoefficientuppervalue.Value >
app.SuspensionSpringCoefficientlowervalue.Value && ...
                    app.SuspensionDampingCoefficientupper.Value >
app.SuspensionDampingCoefficientlower.Value
                app.TabGroup.SelectedTab = app.RoadInputs;
                app.mslamp.Visible = "off";
                app.mulamp.Visible = "off";
                app.ktlamp.Visible = "off";
                app.kslamp.Visible = "off";
                app.cslamp.Visible = "off";
                app.InputsmustbevalidLamp.Visible = "off";
                app.InputsmustbevalidLamp_2.Visible = "off";
                app.InputsmustbevalidLampLabel.Visible= "off";

                assignin('base', 'ms', app.SprungMassmskgEditField.Value);
                assignin('base', 'mu', app.UnsprungMassmukgEditField.Value);
                assignin('base', 'kt',
app.SpringCoefficientofTirektNmEditField.Value);
                assignin('base', 'ks_lowerb',
app.SuspensionSpringCoefficientlowervalue.Value);
                assignin('base', 'ks_upperb',
app.SuspensionSpringCoefficientuppervalue.Value);
                assignin('base', 'cs_lowerb',
app.SuspensionDampingCoefficientlower.Value);
                assignin('base', 'cs_upperb',
app.SuspensionDampingCoefficientupper.Value);


            else
                app.InputsmustbevalidLamp_2.Visible = 'on';
                app.InputsmustbevalidLamp.Visible = 'on';
                app.InputsmustbevalidLampLabel.Visible = 'on';
            end

            if app.SprungMassmskgEditField.Value <= 0
                app.mslamp.Visible = "on";
            end
            if app.UnsprungMassmukgEditField.Value <= 0
                app.mulamp.Visible = "on";
            end
            if app.SpringCoefficientofTirektNmEditField.Value <= 0
                app.ktlamp.Visible = "on";
            end
            if app.SuspensionSpringCoefficientlowervalue.Value < 0 ||
app.SuspensionSpringCoefficientuppervalue.Value <=
app.SuspensionSpringCoefficientlowervalue.Value
                app.kslamp.Visible = "on";
            end
            if app.SuspensionDampingCoefficientlower.Value < 0 ||
app.SuspensionDampingCoefficientupper.Value <=
app.SuspensionDampingCoefficientlower.Value
                app.cslamp.Visible = "on";
            end
```

Button_3Pushed

```
if app.MinimazetheMaximumSprungMassAccelerationButton.Value == 1
                i=1;
                kscs=zeros(10*10,2);

                ks_values = 
linspace(app.SuspensionSpringCoefficientlowervalue.Value, 
app.SuspensionSpringCoefficientuppervalue.Value, 10);
                cs_values = 
linspace(app.SuspensionDampingCoefficientlower.Value, 
app.SuspensionDampingCoefficientupper.Value, 10);

                for kss = ks_values
                    for css = cs_values
                        kscs(i,:)=[kss,css];
                        kscs1(i,:)=[kss,css];
                        i=i+1;
                    end

                end

                as = evalin("base","as");
                aus = evalin("base","aus");
                ds = evalin("base","ds");
                dus = evalin("base","dus");
                vs = evalin("base","vs");
                vus = evalin("base","vus");
                rd = evalin("base","rd");

                kscs(:,3)=max(abs(as));
                kscs1(:,3)=max(abs(as));

                %
                %tire deflection
                tiredeflection = abs(dus-rd);

                %maksimum = max(tiredeflection);

                %suspension deflection
                suspensiondeflection = abs(ds-dus);

                %maksiumum2 = max(suspensiondeflection);

                colsToDelete = any(tiredeflection > 
app.TireDeflectionmEditField.Value, 1); % 1: sütun bazlı işlem
                colsToDelete2 = any(suspensiondeflection > 
app.SuspensionDeflectionmEditField.Value, 1); % 1: sütun bazlı işlem

                deletecolumn = colsToDelete2 | colsToDelete;
                columnnumber = find(deletecolumn==1);
                columnnumber = sort(columnnumber, 'descend');

                columnnumber1 = find(deletecolumn==0);
                columnnumber1 = sort(columnnumber1, 'descend');

                tiredeflection(:, deletecolumn) = [];
                suspensiondeflection(:, deletecolumn) = [];
                %
                for ii = columnnumber
                    kscs(ii,:) = [];
                end

                for kkkk = columnnumber1
                    kscs1(kkkk,:) = [];
                end

                x = kscs(:,1);
                y = kscs(:,2);
                z = kscs(:,3);

                x1 = kscs1(:,1);
                y1 = kscs1(:,2);
                z1 = kscs1(:,3);

                scatter3(app.UIAxes2,x,y,z,50,"green","filled")

                %colorbar(app.UIAxes2);
                grid(app.UIAxes2, 'on');
                hold(app.UIAxes2, 'on');

                scatter3(app.UIAxes2,x1,y1,z1,50,[0.5 0.5 0.5],"filled")

                [minVal, minIdx] = min(z);
                scatter3(app.UIAxes2, x(minIdx), y(minIdx), minVal, 100, 'r',
'filled');
                text(app.UIAxes2,x(minIdx), y(minIdx), minVal, sprintf('Minimum
peak acceleration is %.1f m/s^2\noccurs at ks = %.1f N/m and cs = %.1f. N.s/m',
minVal, x(minIdx), y(minIdx)), ...
                'Color', 'red', 'FontSize', 12, 'FontWeight', 'bold', ...
                'VerticalAlignment', 'bottom', 'HorizontalAlignment', 'right');

                hold(app.UIAxes2, 'off');

            elseif
app.MinimazeRootMeanSquareRMSSprungMassAccelerationButton.Value == 1

                i=1;
                kscs=zeros(10*10,2);

                ks_values =
linspace(app.SuspensionSpringCoefficientlowervalue.Value,
app.SuspensionSpringCoefficientuppervalue.Value, 10);
                cs_values =
linspace(app.SuspensionDampingCoefficientlower.Value,
app.SuspensionDampingCoefficientupper.Value, 10);


                for kss = ks_values
                    for css = cs_values
                        kscs(i,:)=[kss,css];
                        kscs1(i,:)=[kss,css];
                        i=i+1;
                    end

                end

                as = evalin("base","as");
                aus = evalin("base","aus");
                ds = evalin("base","ds");
                dus = evalin("base","dus");
                vs = evalin("base","vs");
                vus = evalin("base","vus");
                rd = evalin("base","rd");

                kscs(:,3)=rms(as);
                kscs1(:,3)=rms(as);

                %
                %tire deflection
                tiredeflection = abs(dus-rd);

                %maksimum = max(tiredeflection);

                %suspension deflection
                suspensiondeflection = abs(ds-dus);

                %maksiumum2 = max(suspensiondeflection);

                colsToDelete = any(tiredeflection >
app.TireDeflectionmEditField.Value, 1); % 1: sütun bazlı işlem
                colsToDelete2 = any(suspensiondeflection >
app.SuspensionDeflectionmEditField.Value, 1); % 1: sütun bazlı işlem

                deletecolumn = colsToDelete2 | colsToDelete;
                columnnumber = find(deletecolumn==1);
                columnnumber = sort(columnnumber, 'descend');

                columnnumber1 = find(deletecolumn==0);
                columnnumber1 = sort(columnnumber1, 'descend');

                tiredeflection(:, deletecolumn) = [];
                suspensiondeflection(:, deletecolumn) = [];
                %
                for ii = columnnumber
                    kscs(ii,:) = [];
                end

                for kkkk = columnnumber1
                    kscs1(kkkk,:) = [];
                end

                x = kscs(:,1);
                y = kscs(:,2);
                z = kscs(:,3);

                x1 = kscs1(:,1);
                y1 = kscs1(:,2);
                z1 = kscs1(:,3);

                scatter3(app.UIAxes2,x,y,z,50,"green","filled")

                grid(app.UIAxes2, 'on');
                hold(app.UIAxes2, 'on');

                scatter3(app.UIAxes2,x1,y1,z1,50,[0.5 0.5 0.5],"filled")
                [minVal, minIdx] = min(z);
                scatter3(app.UIAxes2, x(minIdx), y(minIdx), minVal, 100, 'r',
'filled');
                text(app.UIAxes2,x(minIdx), y(minIdx), minVal, sprintf('Minimum
RMS acceleration is %.1f \noccurs at ks = %.1f N/m and cs = %.1f.', minVal,
x(minIdx), y(minIdx)), ...
                'Color', 'red', 'FontSize', 12, 'FontWeight', 'bold', ...
                'VerticalAlignment', 'bottom', 'HorizontalAlignment', 'right');

                hold(app.UIAxes2, 'off');

            end

            optimized_cs = y(minIdx);
            assignin('base', 'optimized_cs',optimized_cs);
            optimized_ks = x(minIdx);
            assignin('base', 'optimized_ks',optimized_ks);

            tout = evalin("base","simOut.tout");

            plot(app.Acceleration,tout,as(:,minIdx))
            hold(app.Acceleration,"on")
            plot(app.Acceleration,tout,aus(:,minIdx))
            hold(app.Acceleration,"off")
            plot(app.AccelerationUnsp,tout,aus(:,minIdx))
            plot(app.AccelerationSprung,tout,as(:,minIdx))
            app.AccelerationSprung.Children.Visible = "off";
            app.AccelerationUnsp.Children.Visible = "off";

            plot(app.Velocity,tout,vs(:,minIdx))
            hold(app.Velocity,"on")
            plot(app.Velocity,tout,vus(:,minIdx))
            hold(app.Velocity,"off")
            plot(app.VelocityUnsp,tout,vus(:,minIdx))
            plot(app.VelocitySprung,tout,vs(:,minIdx))
            app.VelocitySprung.Children.Visible = "off";
            app.VelocityUnsp.Children.Visible = "off";

            plot(app.Displacement,tout,ds(:,minIdx))
            hold(app.Displacement,"on")
            plot(app.Displacement,tout,dus(:,minIdx))
            hold(app.Displacement,"off")
            plot(app.DisplacementUnsp,tout,dus(:,minIdx))
            plot(app.DisplacementSprung,tout,ds(:,minIdx))
            app.DisplacementSprung.Children.Visible = "off";
            app.DisplacementUnsp.Children.Visible = "off";

            app.TabGroup.SelectedTab = app.SimulationResultsTab;
```

MATLAB Function1

```
function xdd = fcn(x,xd,y,ks,cs,ms,mu,kt)

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

xdd=inv(M)*(F-C*xd-K*x);
```

plotDamper.m

```
function plotDamper(L0_2,h1,z_s,z_u,i,x_inst)

    rodLowerPct = 0.1;      % Length lower rod percentage of total gap
    rodUpperPct = 0.4;      % Length upper rod percentage of total gap
    cylinderPct = 0.4;      % Length cylinder porcentagem of total gap
    damper_line_wid  = 3;   % Damper line width

    % Damper geometry
    c = 0.2+x_inst;         % Longitudinal position
    w= 0.05;                % Width

    % rod attached to unsprung mass
    rod_1_X = [c c];
    rod_1_Z = [z_u+h1 z_u+h1+rodLowerPct*L0_2];

    % Damper base cylinder - rod - base
    c_X =   [
                c-w
                c-w
                c+w
                c+w
            ];

    c_Z =   [
                z_u(i) + h1 + rodLowerPct*L0_2 + cylinderPct*L0_2
                z_u(i) + h1 + rodLowerPct*L0_2
                z_u(i) + h1 + rodLowerPct*L0_2
                z_u(i) + h1 + rodLowerPct*L0_2 + cylinderPct*L0_2
            ];

    % rod attached to sprung mass
    rod2X = [c c];
    rod2Z = [z_s z_s-rodUpperPct*L0_2];
    % Piston inside cylinder
    pistonX = [c-0.8*w c+0.8*w];
    pistonZ = [z_s-rodUpperPct*L0_2 z_s-rodUpperPct*L0_2];

    % Iteration values
    rod1Zval = rod_1_Z(i,:);
    rod2Zval = rod2Z(i,:);
    pistonZVal = pistonZ(i,:);

    % PLOT
    % rods
    plot(rod_1_X,rod1Zval,'k','LineWidth',damper_line_wid)
    plot(rod2X,rod2Zval,'k','LineWidth',damper_line_wid)
    % Damper parts
    plot(pistonX,pistonZVal,'k','LineWidth',damper_line_wid)
    plot(c_X,c_Z,'k','LineWidth',damper_line_wid)

end
```

plotSpring.m

```
function plotSpring(L0_u,L0_s,h_u,u_vet,z_s,z_u,i,x_inst,Z_r)

    rodPct      = 0.11;     % Length rod percentage of total L0
```

```
springPct   = 1/3;       % Spring pitch percentage of total gap
spring_wid  = 3;         % Spring line width

aa = rodPct*L0_u;
k = ((u_vet(i)-Z_r(i))-(2*aa))/3;

% Spring 1 and 2 length without rods
L_s = (z_s - (z_u+h_u)) - 2*rodPct*L0_s;
L_u = (z_u - u_vet')     - 2*rodPct*L0_u;

% Spring sprung suspension geometry
c_s = x_inst-0.2;        % Longitudial position
w_s = 0.1;               % Width

% Spring unsprung tire geometry
c_u = x_inst;            % Longitudial position
w_u = 0.1;               % Width

% Spring unsprung tire
spring_u_X = [
            c_u                                             % Start
            c_u                                             % rod
            c_u+w_u                                         % Part 1
            c_u-w_u                                         % Part 2
            c_u+w_u                                         % Part 3
            c_u-w_u                                         % Part 4
            c_u+w_u                                         % Part 5
            c_u-w_u                                         % Part 6
            c_u                                             % Part 7
            c_u                                             % rod/End
            ];

   spring_u_Z = [
            u_vet(i)                                        % Start
            Z_r(i)+  aa+k+k+k                          % rod
            Z_r(i)+  aa+k+k+k                          % Part 1
            Z_r(i)+  aa+k+k       % Part 2
            Z_r(i)+  aa+k+k       % Part 3
            Z_r(i)+  aa+k       % Part 4
            Z_r(i)+  aa+k      % Part 5
            Z_r(i)+  aa        % Part 6
            Z_r(i)+  aa        % Part 7
            Z_r(i)       % rod/End
           ];

% Spring sprung suspension
spring_s_X = [
            c_s                                             % Start
            c_s                                             % rod
            c_s+w_s                                         % Part 1
            c_s-w_s                                         % Part 2
            c_s+w_s                                         % Part 3
            c_s-w_s                                         % Part 4
            c_s+w_s                                         % Part 5
            c_s-w_s                                         % Part 6
            c_s                                             % Part 7
            c_s                                             % rod/End
            ];
```

```
        spring_s_Z = [
                z_u(i)+h_u                                    % Start
                z_u(i)+h_u +   rodPct*L0_s                    % rod
                z_u(i)+h_u +   rodPct*L0_s                    % Part 1
                z_u(i)+h_u +   rodPct*L0_s +   springPct*L_s(i) % Part 2
                z_u(i)+h_u +   rodPct*L0_s +   springPct*L_s(i) % Part 3
                z_u(i)+h_u +   rodPct*L0_s + 2*springPct*L_s(i) % Part 4
                z_u(i)+h_u +   rodPct*L0_s + 2*springPct*L_s(i) % Part 5
                z_u(i)+h_u +   rodPct*L0_s + 3*springPct*L_s(i) % Part 6
                z_u(i)+h_u +   rodPct*L0_s + 3*springPct*L_s(i) % Part 7
                z_u(i)+h_u + 2*rodPct*L0_s + 3*springPct*L_s(i) % rod/End
               ];

      % PLOT
      plot(spring_u_X,spring_u_Z,'k','LineWidth',spring_wid)
      plot(spring_s_X,spring_s_Z,'k','LineWidth',spring_wid)

  end
```

SeperatedViewButtonPushed

```
if app.Acceleration.Visible == "on"
                app.Acceleration.Visible = "off";
                app.AccelerationSprung.Visible = "on";
                app.AccelerationUnsp.Visible = "on";
                set(app.Acceleration.Children, 'Visible', 'off');
                app.AccelerationSprung.Children.Visible = "on";
                app.AccelerationUnsp.Children.Visible = "on";

            elseif  app.AccelerationSprung.Visible == "on" &&
app.AccelerationUnsp.Visible == "on"
                app.Acceleration.Visible = "on";
                app.AccelerationSprung.Visible = "off";
                app.AccelerationUnsp.Visible = "off";
                set(app.Acceleration.Children, 'Visible', 'on');
                app.AccelerationSprung.Children.Visible = "off";
                app.AccelerationUnsp.Children.Visible = "off";
            end
```

SeperatedViewButton_2Pushed

```
if app.Velocity.Visible == "on"
                app.Velocity.Visible = "off";
                app.VelocitySprung.Visible = "on";
                app.VelocityUnsp.Visible = "on";
                set(app.Velocity.Children, 'Visible', 'off');
                app.VelocitySprung.Children.Visible = "on";
                app.VelocityUnsp.Children.Visible = "on";

            elseif  app.VelocitySprung.Visible == "on" &&
app.VelocityUnsp.Visible == "on"
                app.Velocity.Visible = "on";
                app.VelocitySprung.Visible = "off";
                app.VelocityUnsp.Visible = "off";
                set(app.Velocity.Children, 'Visible', 'on');
                app.VelocitySprung.Children.Visible = "off";
                app.VelocityUnsp.Children.Visible = "off";
            end
```

SeperatedViewButton_3Pushed

```
if app.Displacement.Visible == "on"
                app.Displacement.Visible = "off";
                app.DisplacementSprung.Visible = "on";
                app.DisplacementUnsp.Visible = "on";
                set(app.Displacement.Children, 'Visible', 'off');
                app.DisplacementSprung.Children.Visible = "on";
                app.DisplacementUnsp.Children.Visible = "on";

            elseif  app.DisplacementSprung.Visible == "on" &&
app.DisplacementUnsp.Visible == "on"
                app.Displacement.Visible = "on";
                app.DisplacementSprung.Visible = "off";
                app.DisplacementUnsp.Visible = "off";
                set(app.Displacement.Children, 'Visible', 'on');
                app.DisplacementSprung.Children.Visible = "off";
                app.DisplacementUnsp.Children.Visible = "off";
end
```

Sigmoid Input

```
function y = sigmoid_input(t,sshh)
    b = 20;    % eğim (ne kadar keskin olsun)
    c = 2;    % offset (sigmoid'in orta noktası)

    y = sshh ./ (1 + exp(-b * (t - c)));
end
```

SimulateButton_2Pushed

```
if app.SineWaveButton.Value == 1

                %assignin('base', 'frequ',freq);
                assignin('base', 'ampli',app.AmplitudemeterEditField.Value);

                run("sinesc.m")

            elseif app.BumpButton.Value == 1

                run("bezier.m")
                run("bumpsc.m")

            elseif app.StepInputButton.Value == 1

                assignin('base',
'stepheight',app.StepHeightmeterEditField.Value);
                assignin('base', 'stepspeed',app.SpeedkmsEditField_2.Value);

                run("stepsc.m")

            end

            app.TabGroup.SelectedTab = app.OptimizationCriteriaTab;
```

sinesc.m

```
ks_lowerb = evalin('base', 'ks_lowerb');
ks_upperb = evalin('base', 'ks_upperb');

cs_lowerb = evalin('base', 'cs_lowerb');
cs_upperb = evalin('base', 'cs_upperb');

ks_values = linspace(ks_lowerb, ks_upperb, 10);
assignin('base', 'ks_values', ks_values);
cs_values = linspace(cs_lowerb, cs_upperb, 10);
assignin('base', 'cs_values', cs_values);

as = zeros(1001,length(ks_values)*length(cs_values));
aus = zeros(1001,length(ks_values)*length(cs_values));

vs = zeros(1001,length(ks_values)*length(cs_values));
vus = zeros(1001,length(ks_values)*length(cs_values));

ds = zeros(1001,length(ks_values)*length(cs_values));
dus = zeros(1001,length(ks_values)*length(cs_values));

i=0;

% Simulasyon döngüsü
for ks = ks_values
    for cs = cs_values
        % Workspace'e değerleri ata
        assignin('base', 'ks', ks);
        assignin('base', 'cs', cs);

        % Simulink modelini çalıştır
        simOut = sim('sine.slx', 'SrcWorkspace', 'base');
        assignin('base', 'simOut', simOut);

        % Çıkış verilerini al
        i=i+1;
        % Accelerations
        sma = simOut.sprungaccel.signals.values;
        usma = simOut.unsprungaccel.signals.values;
        as(:,i) = sma;
        assignin('base', 'as', as);
        aus(:,i) = usma;
        assignin('base', 'aus', aus);


        % Velocities
        smv = simOut.sprungvel.signals.values;
        usmv = simOut.unsprungvel.signals.values;
        vs(:,i) = smv;
        assignin('base', 'vs', vs);
        vus(:,i) = usmv;
        assignin('base', 'vus', vus);

        %Displacements
        rd = simOut.roadprofile.signals.values;
        assignin('base', 'rd', rd);
        smd = simOut.sprungdisp.signals.values;
        usmd = simOut.unsprungdisp.signals.values;
        ds(:,i) = smd;
        assignin('base', 'ds', ds);
        dus(:,i) = usmd;
        assignin('base', 'dus', dus);

    end
end
```

SpeedkmsEditFieldValueChanged

```
A = app.BumpHeightmeterEditField.Value;
L = app.BumpLenghtmeterEditField.Value;
v = app.SpeedkmsEditField.Value*1000/3600;

assignin('base', 'bumplength',app.BumpLenghtmeterEditField.Value);
assignin('base', 'bumpheight',app.BumpHeightmeterEditField.Value);
assignin('base', 'bumpvelocity',app.SpeedkmsEditField.Value);

f = v/(2*L);
assignin('base', 'f', f)

lamda = L/v;
assignin('base', 'lamda', lamda)
lamda = round(lamda,2);

t = 0:0.01:(lamda+3);

loc1 = find(round(t, 2) == round(5*lamda, 2));
loc2 = find(round(t, 2) == round(4*lamda, 2));

y = A*sin(2*pi*f*t);
y(loc1:end)=0;
y(1:loc2)=0;

plot(t,y);
```

SpeedkmsEditField_2ValueChanged

```
t=0:0.01:6;
            stepinputheight = app.StepHeightmeterEditField.Value;% maksimum
değer
            stepspeed = app.SpeedkmsEditField_2.Value;

            b = 20;    % eğim (ne kadar keskin olsun)
            c = 2;    % offset (sigmoid'in orta noktası)

            y = stepinputheight ./ (1 + exp(-b * (t - c)));

            plot(app.UIAxes,t,y);
```

StepHeightmeterEditFieldValueChanged

```
t=0:0.01:6;
            stepinputheight = app.StepHeightmeterEditField.Value;% maksimum
değer
            stepspeed = app.SpeedkmsEditField_2.Value;

            b = 20;    % eğim (ne kadar keskin olsun)
            c = 2;    % offset (sigmoid'in orta noktası)
            y = stepinputheight ./ (1 + exp(-b * (t - c)));

            plot(app.UIAxes,t,y);
```

stepsc.m

```
ks_lowerb = evalin('base', 'ks_lowerb');
ks_upperb = evalin('base', 'ks_upperb');

cs_lowerb = evalin('base', 'cs_lowerb');
cs_upperb = evalin('base', 'cs_upperb');

ks_values = linspace(ks_lowerb, ks_upperb, 10);
assignin('base', 'ks_values', ks_values);
cs_values = linspace(cs_lowerb, cs_upperb, 10);
assignin('base', 'cs_values', cs_values);

as = zeros(1001,length(ks_values)*length(cs_values));
aus = zeros(1001,length(ks_values)*length(cs_values));

vs = zeros(1001,length(ks_values)*length(cs_values));
vus = zeros(1001,length(ks_values)*length(cs_values));

ds = zeros(1001,length(ks_values)*length(cs_values));
dus = zeros(1001,length(ks_values)*length(cs_values));

i=0;

% Simulasyon döngüsü
for ks = ks_values
    for cs = cs_values
        % Workspace'e değerleri ata
        assignin('base', 'ks', ks);
        assignin('base', 'cs', cs);

        % Simulink modelini çalıştır
        %simOut = sim("step.slx", 'ReturnWorkspaceOutputs', 'on', 'base');
        simOut = sim('step.slx', 'SrcWorkspace', 'base');
        assignin('base', 'simOut', simOut);

        % Çıkış verilerini al
        i=i+1;
        % Accelerations
        sma = simOut.sprungaccel.signals.values;
        usma = simOut.unsprungaccel.signals.values;
        as(:,i) = sma;
        assignin('base', 'as', as);
        aus(:,i) = usma;
        assignin('base', 'aus', aus);


        % Velocities
        smv = simOut.sprungvel.signals.values;
        usmv = simOut.unsprungvel.signals.values;
        vs(:,i) = smv;
        assignin('base', 'vs', vs);
        vus(:,i) = usmv;
        assignin('base', 'vus', vus);
        %Displacements
        rd = simOut.roadprofile.signals.values;
        assignin('base', 'rd', rd);
        smd = simOut.sprungdisp.signals.values;
        usmd = simOut.unsprungdisp.signals.values;
        ds(:,i) = smd;
        assignin('base', 'ds', ds);
        dus(:,i) = usmd;
        assignin('base', 'dus', dus);

    end
end
```