MARMARA UNIVERSITY
FACULTY OF ENGINEERING

DESIGN OF A SIX-LINK STEERING MECHANISM FOR AUTOMOTIVE VEHICLES

ERHAN TANÇAY

GRADUATION PROJECT REPORT
Department of Mechanical Engineering

Supervisor
Assoc. Prof. Dr. Mustafa ÖZDEMİR

ISTANBUL, 2023

MARMARA UNIVERSITY
FACULTY OF ENGINEERING

DESIGN OF A SIX-LINK STEERING MECHANISM FOR AUTOMOTIVE VEHICLES

ERHAN TANÇAY (150417049)

GRADUATION PROJECT REPORT
Department of Mechanical Engineering

Supervisor
Assoc. Prof. Dr. Mustafa ÖZDEMİR

ISTANBUL, 2023

MARMARA UNIVERSITY
FACULTY OF ENGINEERING

Design of a Six-Link Steering Mechanism for Automotive Vehicles

by

Erhan Tançay

June 05, 2023, Istanbul

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

OF

BACHELOR OF SCIENCE

AT

MARMARA UNIVERSITY

Signature of Author(s) .....:....Erhan Tançay
Department of Mechanical Engineering

Certified By .....Assoc. Prof. Dr. Mustafa Özdemir
Project Supervisor, Department of Mechanical Engineering

Accepted By.....Prof. Dr. Bülent Ekici
Head of the Department of Mechanical Engineering

ACKNOWLEDGEMENT

First of all, I would like to thank my supervisor Associate Professor Dr. Mustafa Özdemir, for the valuable guidance and advice on preparing this thesis and for giving me moral and material support.

June 2023 Erhan Tançay

CONTENTS

ÖZET

Otomotiv araçları için altı bağlantılı direksiyon mekanizması tasarımı

Altı bağlantılı direksiyon mekanizması diğer mekanizmalara göre güçlü ve daha konforludur ancak diğerlerine kıyasla üretimi daha karmaşıktır. Altı bağlantılı mekanizmanın tasarım süreci, çok sayıda hesaplama ve farklı değişkenler gerektirir, mekanizmanın düzgün bir şekilde inşa edilmesi için bu hesaplamalar ve değişkenler gereklidir. Bu denklemleri ve değişkenleri çözmek, optimum sonuçları ve sayıları elde etmeye çalışırken çok zaman alabilir. Bu projemde altı bağlantılı direksiyon mekanizmasının tasarım adımlarını anlatmaya çalıştım. Değişkenleri ve denklemleri tanımladıktan sonra, aşağıdaki bölümlerde tanımlanan Ackerman Koşuluna göre bağlantı uzunluklarını ve bağlantı yerlerini optimize etmeye çalıştım.

ABSTRACT

Design of a six-link steering mechanism for automotive vehicles

The six-bar steering mechanism is strong and comfortable but more complex to build compared to other types of steering mechanisms. The design process of the six-bar mechanism needs a lot of calculations and different variables for comparison. These calculations and variables are needed for properly building the mechanism. Solving these equations and variables may take a lot of time while trying to get optimal results and numbers. In this project, I have tried to explain the design steps of the six-bar steering mechanism. After defining the variables and equations I have tried to optimize the link lengths and joint locations according to Ackerman Condition which is defined in the following sections.

SYMBOLS

a_1 : Link 2 distance
a_2 : Distance from rear tires to the mass center of the vehicle
a_2 : Link 4 distance
b_1 : Link 3 distance
b_2 : Link 5 distances
c_1 : Link 4 distance
c_2 : Link 6 distance
A : Variable defined at equation 4.4
B : Variable defined at equation 4.5
C : Variable defined at equation 4.6
J1 : Variable defined at equation 4.7
J2 : Variable defined at equation 4.8
J3 : Variable defined at equation 4.9
J4 : Variable defined at equation 4.10
J5 : Variable defined at equation 4.11
L : Distance between the front and rear axle, wheelbase
R : Radius of rotation
Rm : Minimum turning radius
w : Distance between front wheels
δ_1 : Input angle of inside tire
δ_2 : Input angle of the outside tire
δ_i : Angle of inside tire
δ_{max} : Maximum steer angle
δ_{ACK} : Ackerman angle
δ_o : Angle of the outside tire
θ_1 : Input angle of the left part
θ_4 : Output angle of the left part
φ_2 : Input angle of the right part
φ_4 : Output angle of the right part

ABBREVIATIONS

ACK : Ackerman

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

The steering system is the system that allows the vehicle to maneuver and move in the desired direction. This system has a big influence on the behavior of the vehicle. This system allows the driver to control the vehicle in desired manner and angle of rotation.

The steering mechanism is generally designed to achieve the condition called Ackermann's Condition. To design an efficient system to preserve both fuel economy and tire life designer should use Ackermann's Condition properties. Ackermann's Condition work in principle to intercept the front wheel axis with the rear wheel axis at the same point [1].

The main system used for steering the vehicles is rack and pinion design. This system is widely used and cheap to manufacture.

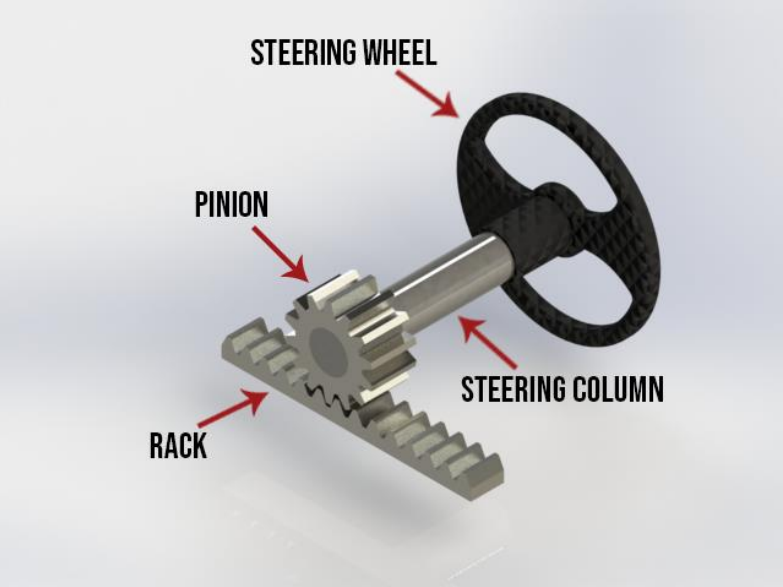

Figure 1: Rack and Pinion Steering System

This system is very economical, simple to manufacture, and easy to operate.

2. STEERING SYSTEM

2.1. What is Steering System

The steering system is the system that allows the driver to maneuver the vehicle in the desired direction. For designing the steering system, we have a couple of existing solutions. Four-bar systems and six-bar systems are the most common solutions to designing this system [2].

2.2. What Are the Components of The Steering System

2.2.1. Steering Wheel

The steering wheel is the component that the driver of the vehicle controls physically. The steering of the vehicle starts with the steering wheel, and this component connects with the steering column.

2.2.2. Steering Column

One tip of this column is surrounded by the wheel and the other connects with the pinion. Column transfers rotation from the steering wheel to the pinion.

2.2.3. Steering Gear

In the steering gear, the pinion takes rotation from the column and transforms into a rack. The advantages of the design are easy to apply multiple steering angles, and more rotation at the tires than the wheel because of the rack and pinion design.

2.3. Steering System Types

2.3.1. Bell-Crank Steering

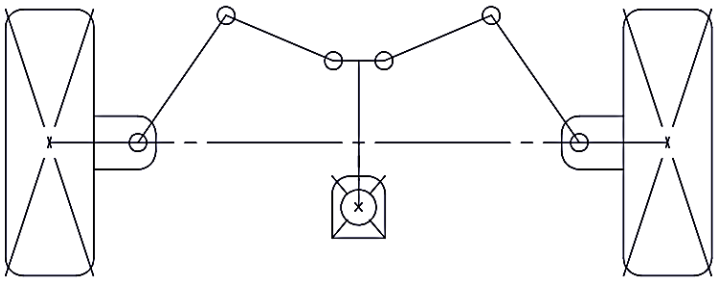

Figure 2: Bell Crank Steering Mechanism Example

This steering system is more commonly used on ATVs and golf carts and some trailers. These days besides the automotive industry bell crank systems are commonly used in light aircraft for control while on the ground [3].

This system changes the direction of motion through an angle and the bar. The angle can be varied from 0 to 360 degrees. There is a straight bar pinned to the center. When one of the arms is pushed bar rotates and pulls the other arm and via all this motion desired direction change is achieved [3].

2.3.2. Rack and Pinion Steering

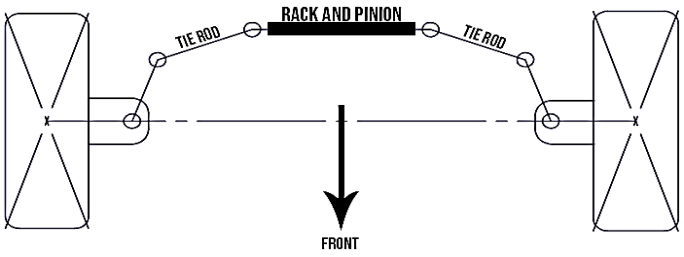

Figure 3: Rack and Pinion

As can be seen in Fig. 1. This system contains a simple gear system to generate desired steering angle. This gear translates circular motion that comes from the steering wheel to linear motion. The gear connects to the shaft and when the steering wheel turn gear turns too. After the gear turns rack starts moving and transfers movement to the tires and desired direction change is achieved [2].

2.3.3. What is the Four-Bar Linkage System

Four bar linkage system or mechanism contains four links. 1- Base link from the ground, 2- Input link, 3- Link from input link to output link, 4- Output link, Output angle θ_4 can be calculated with the following equation [4].

$$\theta_4 = 2\tan^{-1}\left(\frac{-B\pm\sqrt{B^2-4AC}}{2A}\right) \quad (2.1)$$

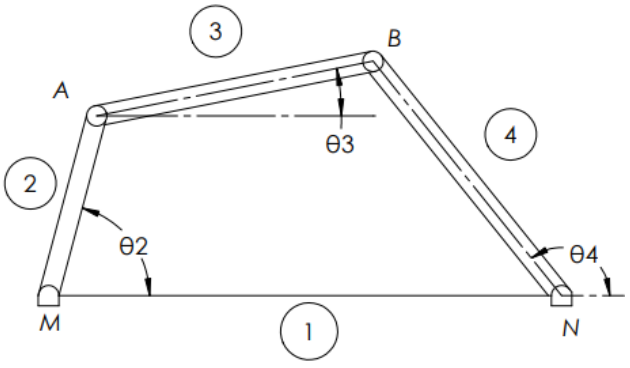

Figure 4: Four-Bar Linkage

In this mechanism, one link is attached to the ground and fixed there. We use the ground link alignment for calculating the rest of the variables [4].

2.3.4. Why Four-Bar Is Disadvantageous and Why Six-Bar Advantageous

Six-bar systems allow us to create larger steering angles needed for bigger vehicles such as trucks or buses [3]. For this problem, cheap four-bar systems create a lack of

rotation for the vehicles in slow maneuvers. The six-bar system with a higher angle of rotation can provide a better solution for these vehicles.

Six-bar systems or multi-link systems contain more linkages than traditional four-bar systems. Because of the design, they are more powerful and can resist more load, and can create a higher angle of rotation. Six-bar steering systems can be like in Fig. 2 or like Fig. 5.

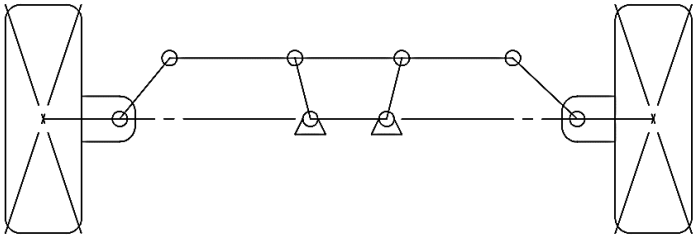

Figure 5: Six-Bar Mechanism as Two Combined Four-Bar Mechanisms

Advantages of a six-bar steering mechanism:

High accuracy: Six-bar steering mechanisms have high accuracy, which can be beneficial for applications where precise steering is required.

Strong design: Six-bar mechanisms are generally strong and able to resist high loads, making them suitable for use in heavy vehicles or off-road applications.

Smooth operation, absorbs shocks: Six-bar mechanisms typically have a smooth operation while driving, it can absorb road shocks better than the four-bar mechanism, which can improve the overall driving experience.

Disadvantages of a four-bar steering mechanism:

Limited range of motion: Four-bar mechanisms have a limited range of motion, they are less suitable for making large steering movements.

Load sensitivity: Four-bar mechanisms can be fragile to changes in load, which can affect their accuracy and performance.

2.3.5. What are the Requirements of a Six-Bar Linkage System

The mechanism consists of six links, which are connected by six joints. To design a six-bar steering mechanism, you will need to consider a couple of factors:

- Load capacity: The mechanism must be able to handle the loads that will be expected to carry on it, and any additional loads that may be occurred during steering.
- Range of motion: The mechanism must provide the desired range of motion to steer the vehicle in the desired direction.
- Link lengths: The lengths of the links will determine the mechanical advantage of the mechanism.

- Joint locations: The locations of the joints will determine the overall geometry of the mechanism.
- Materials: The materials to create the links and joints should be chosen for their qualities.

3. ACKERMAN PRINCIPAL

3.1. Ackerman Steering

This is the condition that is used to find the ideal turning angle for all types of vehicles. The designed mechanism for vehicles has to meet the requirements of this condition [1].

In Ackerman Condition the projectile of the front tires has to meet at the same point on top of the projectile of the rear tires as shown in Fig. 6.

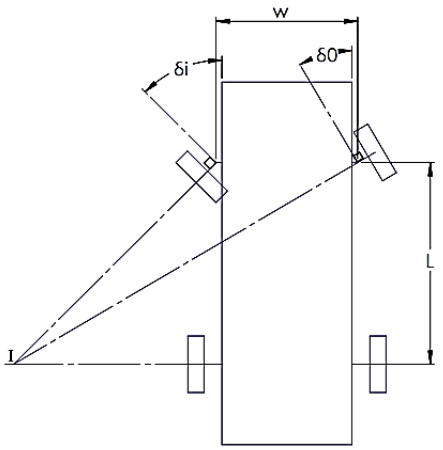

Figure 6: Ackerman Steering Principal

This condition is expressed with the following equation.

$$cot\ \delta_0 - cot\ \delta_i = \frac{w}{L} \tag{3.1}$$

$$\delta_i = angle\ of\ inside\ tire$$

$$\delta_o = angle\ of\ outside\ tire$$

The distance from the front wheel to the rear wheels is called wheelbase and it is represented with L, distance between the front wheels is represented with w.

Distance from the center of the vehicle to point I in Fig. 5 is called as Radius of rotation and can be found with the following equation [4].

$$R = \sqrt{a_2^2 + l^2 cot^2 \delta} \tag{3.2}$$

$$a_2 = distance\ from\ rear\ tyres\ to\ mass\ center\ of\ vehicle$$

$$cot\delta = \frac{cot\delta_0 + cot\delta_i}{2} \tag{3.3}$$

Ackerman's steering principle is ideal only for vehicles that move at low speeds. For race cars, high-speed direction changes, and other steering applications can be used to achieve desired turning radius such as parallel or reverse. But these cars still have to follow Ackerman's principle at low speed. Because of that, we can't say there is one certain optimal steering system we can apply to our designs [4].

4. DESIGNING OF STEERING MECHANISM

I have decided to create a program that optimizes the design steps and numbers for desired given inputs by the user. For this software, I have decided to use Ackerman Equation. Six link steering mechanism is a complex system and can be adjusted to achieve the desired condition such as Ackerman. In this project, we aim to optimize our six-link steering mechanism. To achieve optimization, we take a problem from the book [4] as a reference.

With the Ackerman equation, we can optimize steering angles and linkage lengths. To implement the equation, we need to define base input values such as length, width, and turning radius. After base values are determined, I will start calculating values using equations such as 3.1, 3.2, and 3.3. With these equations and values, it's easy to calculate the angle and linkage lengths for each link. By creating this software, I aim to create a platform that creates efficient and accurate methods for six-bar mechanism design.

I have used these equations to calculate δ_i values.

$$\delta_1 = \theta_2 - (90 - inputleftt) \quad (4.1)$$

$$\delta_2 = \phi 4 - (90 + inputleft) \quad (4.2)$$

$$\theta_4 = 2\tan^{-1}\left(\frac{+B \pm \sqrt{B^2-4AC}}{2A}\right) \quad (4.3)$$

$$A = J_3 - J_1 + (1 - J_2)\cos\theta_2 \quad (4.4)$$

$$B = -2\sin\theta_2 \quad (4.5)$$

$$C = J_1 + J_3 - (1 + J_2)\cos\theta_2 \quad (4.6)$$

$$J_1 = \frac{d_1}{a_1} \quad (4.7)$$

$$J_2 = \frac{d_1}{c_1} \quad (4.8)$$

$$J_3 = \frac{a_1^2 - b_1^2 + c_1^2 + d_1^2}{2a_1c_1} \quad (4.9)$$

$$J_4 = \frac{d_1}{b_1} \quad (4.10)$$

$$J_5 = \frac{c_1^2 - a_1^2 - b_1^2 - d_1^2}{2a_1b_1} \quad (4.11)$$

$$\varphi_4 = 2\tan^{-1}\left(\frac{+B \pm \sqrt{B^2-4AC}}{2A}\right) \quad (4.12)$$

4.1. Optimization of Multi-Link Steering Mechanism

First, I needed to choose the software that I will use for the project. I have decided to use
MATLAB software. Because of its capabilities such as numerical solutions and plot applications that are easily applied with high efficiency. In the following figure, I have divided the mechanism into two parts, in real life bottom part of c_1 and a_2 are connected such as Fig. 8.

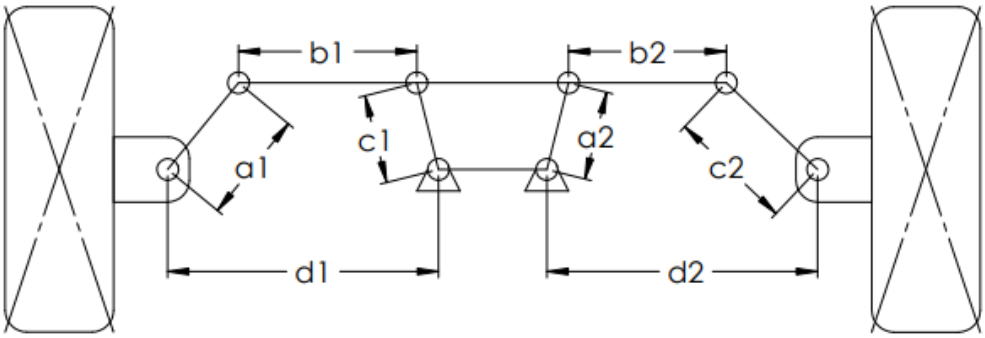

Figure 7: Six-Bar Mechanism With a,b,c,d Variables Inserted

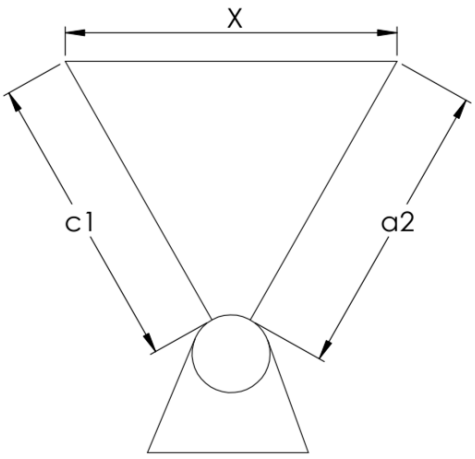

Figure 8: X Distance Representation

For this project, I needed to put base values in the code as a variable for calculation such as width, length, and a_2(a_2 = distance from rear tires to the mass center of the vehicle) Fig. 6.

$$a_2 = 0 \cdot 45l \tag{4.1.1}$$

With these values, we add the input left value as “54.6” as a base value. [4]

4.2. Calculating δ_{max}

For calculating δ_{max} I have used Equations (3.1), and (3.2). According to the general view in the industry, I have picked my minimum turning radius as 10m.

For calculating δ_{max} I have solved the equation (3.2) for $R_m = 10, a_2 = 0.45 * l, and\ l = 4.8m$ After all the calculations I have found my $\delta_{max} = 0.23713$ rad $= 13.5865°$

4.3. Finding Inner and Outer Steer Angles

To determine $\delta_i \; and \; \delta_o$ I have used the following equations:

$$R_1 = l * cot\delta_m \tag{4.3.1}$$

$$\delta_i \; = tan^{-1}\left(\frac{l}{R_1 - \frac{w}{2}}\right) = 14.4247° \tag{4.3.2}$$

$$\delta_o \; = tan^{-1}\left(\frac{l}{R_1 + \frac{w}{2}}\right) = 12.8388° \tag{4.3.3}$$

This result shows us mechanism at least needs to achieve a $14.4247°$ turning radius. For being safe we need to pick a higher number than this to optimize. For this mechanism, I will pick $\mp 15°$

4.4. Steer Angles of Left and Right Links

To calculate steer angles, I will use the following equations:

$$\delta_1 = \; \theta_2 - (90 - inputleft) \tag{4.4.1}$$
$$\delta_2 = \; \varphi_4 - (90 + inputright) \tag{4.4.2}$$

I have picked my input left angle as 54.6°

4.5. Creating a Table From the Variables Calculated

After calculating the necessary values, I have put them on the table to represent better and clarify further calculations.

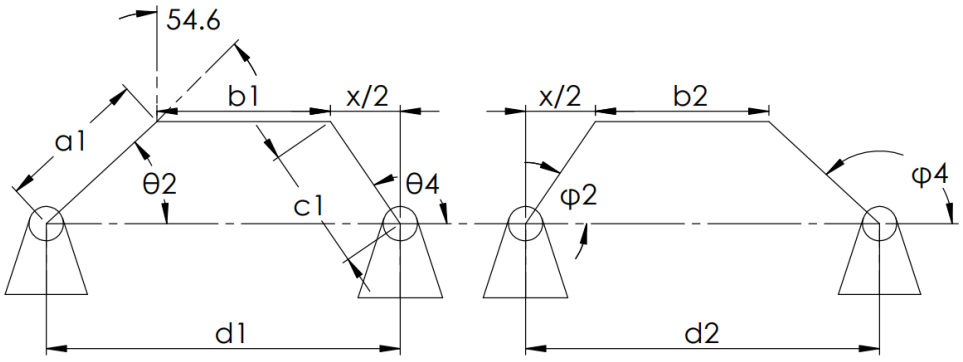

Figure 9: Joint Angles Two Four-Bar Linkage

Table 1. Initial Calculations

LEFT LINKAGE		
Link	Length	Angle
1	$d_1 = 1.2$	180
2	$a_1 = \frac{0.22}{\cos(54.6)}$	θ_2
3	$b_1 = 1.2 - 0.22 * (tan54.6) - \frac{x}{2}$	θ_3
4	$c_1 = \sqrt{0.22^2 + \frac{x^2}{4}}$	$\boldsymbol{\theta_4}$
RIGHT LINKAGE		
Link	Length	Angle
1	$d_2 = 1.2$	180
2	$a_2 = \sqrt{0.22^2 + \frac{x^2}{4}}$	$\varphi_2 = \boldsymbol{\theta_4} - 2tan^{-1}\left(\frac{x}{0.44}\right)$
3	$b_2 = 1.2 - 0.22 * (tan54.6) - \frac{x}{2}$	φ_3
4	$c_2 = 0.22/\cos(54.6)$	φ_4

4.6. Calculating $\boldsymbol{\theta_4}$

First I have created a "while loop" inside MATLAB for the desired number of iterations. I have calculated a_1, b_1, c_1 and d_1 values and all the J values for use in Equation 4.3. For base input angle and $x = 0$ I have found these values as follows:

Table 2. J Values for θ_4

$\boldsymbol{a_1}$	$\boldsymbol{b_1}$	$\boldsymbol{c_1}$	$\boldsymbol{d_1}$	$\boldsymbol{J_1}$	$\boldsymbol{J_2}$	$\boldsymbol{J_3}$	$\boldsymbol{J_4}$	$\boldsymbol{J_5}$
0.3798	0.8904	0.22	1.2	3.1597	5.4545	5.0254	1.3477	-3.4431

Using these lengths and j values I have calculated

$$\theta_4\ values\ for\ input\ angle \mp 15 =$$

Table 3. θ_4 values for steer angle ∓ 15

$\boldsymbol{\delta_1}$**(angle)**	$\boldsymbol{\theta_4}$**(angle)**
-15	78.8695
-13.125	79.8136
-11.25	80.8931
-9.375	82.1035
-7.5	83.441
-5.625	84.9019
-3.75	86.4834
-1.875	88.1832
0	90
1.875	91.9332
3.75	93.9835
5.625	96.1525
7.5	98.4437
9.375	100.8622
11.25	103.4152
13.125	106.1129
15	108.9695

4.7. Calculating $\boldsymbol{\varphi_4}$

Again I have created a "while loop" for the desired number of iterations
I have calculated a_2, b_2, c_2 and d_2 values and all the J values for use in Equation 4.11
For base input angle and $x = 0$ I have found these values as follows:

Table 4. J values for φ_4

$\boldsymbol{a_2}$	$\boldsymbol{b_2}$	$\boldsymbol{c_2}$	$\boldsymbol{d_2}$	$\boldsymbol{J_1}$	$\boldsymbol{J_2}$	$\boldsymbol{J_3}$	$\boldsymbol{J_4}$	$\boldsymbol{J_5}$
0.2200	0.8904	0.3798	1.2	5.4545	3.1597	5.0254	1.3477	-5.4545

For calculating these values, I need to change $cos\ \theta_2$ with $cos\ \boldsymbol{\varphi_2}$.

$$\varphi_2 = \theta_4 - 2tan^{-1}\frac{x}{a_2 * 2} \tag{4.7.1}$$

Using these lengths and j values I have calculated:

$$\boldsymbol{\varphi_2}\ values\ for\ input\ angle \mp 15 =$$

Table 5. φ_2 values for steer angle ∓ 15

δ_1(angle)	φ_2(angle)
-15	78.86946
-13.125	79.81359
-11.25	80.89308
-9.375	82.10354
-7.5	83.44098
-5.625	84.9019
-3.75	86.48338
-1.875	88.18322
0	90
1.875	91.93322
3.75	93.98347
5.625	96.15254
7.5	98.44375
9.375	100.8622
11.25	103.4152
13.125	106.1129
15	108.9695

4.8. Calculating δ_2

By using equation 4.2 I have calculated δ_2 value as follows for the input angle $\mp 15 =$

Table 6. δ_2 values for the steer angle ∓ 15

δ_1(angle)	δ_2(angle)
-15	-9.58
-13.125	-8.86
-11.25	-8.02
-9.375	-7.06
-7.5	-5.97
-5.625	-4.73
-3.75	-3.33
-1.875	-1.77
0	0.00
1.875	2.00
3.75	4.29
5.625	6.96
7.5	10.20
9.375	14.44
11.25	22.21
15	25.89

4.9. Calculating δ_{ACK}

I have calculated δ_{ACK} using equation 3.1, using δ_0 $as = \delta_{ACK}$ after solving the equation with corresponding values I have found δ_{ACK} values as follows:

Table 7. δ_{ACK} values for steer angle ∓ 15

δ_1(angle)	δ_{ACK}(angle)
-15	-17.19
-13.125	-14.79
-11.25	-12.46
-9.375	-10.20
-7.5	-8.02
-5.625	-5.91
-3.75	-3.88
-1.875	-1.91
0	0.00
1.875	1.84
3.75	3.63
5.625	5.36
7.5	7.04
9.375	8.67
11.25	10.25
13.125	11.80
15	-17.19

4.10. Creating Plot for $\delta_1 \; vs \; \delta_2, \delta_{Ackerman}$

I have created the plot for $\delta_1 \; vs \; \delta_2, \delta_{Ackerman}$:

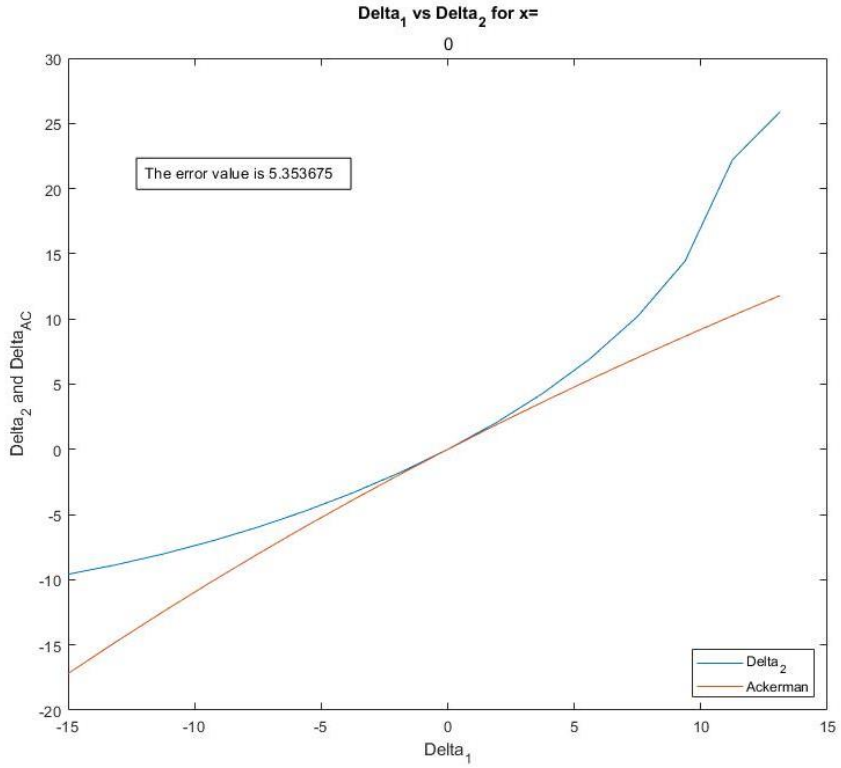

Figure 10: Plot for δ_1 vs δ_2, $\delta_{Ackerman}$ for x=0

4.11. Calculating Error Value

To calculate the error value I have used these two equations:

$$\Delta = \delta_2 - \delta_{Ackerman} \tag{4.11.1}$$

$$e = \sqrt{\frac{\sum \Delta^2}{n}} \tag{4.11.2}$$

After using equations I have found the error value for x=0 as $e = 5.353678$

4.12. Creating Error Graph

Error graphs can be useful for comparing results that are obtained for different X values. To calculate the error graph we need to have a multiple x results. We need an error graph to quickly compare different x values. I have created an error graph to help the user choose the optimal x value in the range with a different code that will be explained in the next chapter.

5. OPTIMIZATION FOR A RANGE OF X VALUES

In the previous chapter, I calculated the error value and link lengths using a single x value.
In this chapter, I will work with a range of x values and try to help the user for picking the x value that creates the lowest error. In the code, I have used the same steps and equations that I mentioned in Chapter-4. But unlike the other chapter, In this chapter, I have calculated the range of x values that the user specified.

First I request user input for steering input angle (θ), car length (l), car width (w), start of the range, end of the range, increment to start(step input), and minimum turning radius (Rm).

After collecting all the inputs from the user, I followed the steps in Chapter 4 with all the X values in the range. This code/software needs to help the user for finding the optimal x value with a minimum error when compared with the Ackerman equation. To achieve this, after all the calculations are done software gives all the graphs with corresponding x values and the error graph as the error value(Eq. 4.11.2) vs x value as a meter.

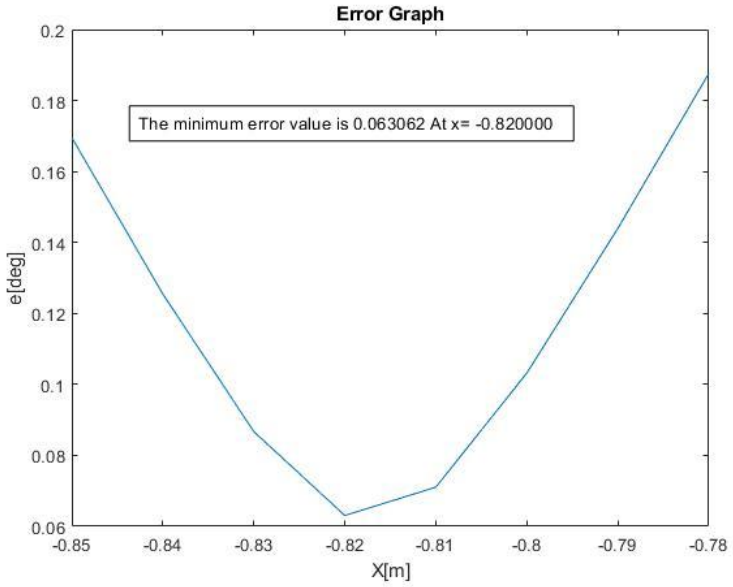

Figure 11: Error Graph for a range of x=(-0.85,-0.78)

After looking at this graph users can easily pick the optimal x value for the design criteria that they are determined at the beginning. For my base values, the optimal x value can be found as -0.82.

6. USER GUIDE

6.1. How Does the Program Work

The program uses previously mentioned steps and equations for making necessary calculations

Step 1:

Taking inputs from users for steering input angle (θ), car length (l), car width (w), the x value, Minimum turning radius (Rm)

Step 2:

The program calculates the distance from the rear tires to the mass center of the vehicle with user input. Then calculate the maximum steer angle for each wheel and decide the optimization angle.

Step 3:

The program calculates all the necessary angles, J values, a,b,c, and d values.

Step 4:

The program calculates the error value for corresponding user inputs

Step 5:

After all the necessary calculations are finished program put the delta variables and error values in the graph.

6.2. Input Guide

Users have 5 input option

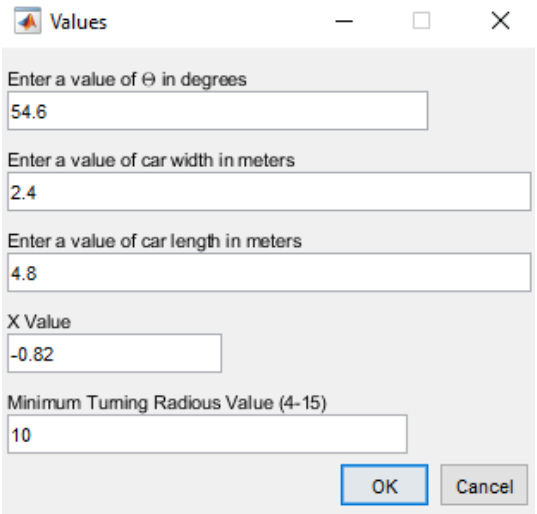

Figure 12: Input Window

Steering input angle (θ)(degree): This is the angle "54.6" in Figure 8, I recommend picking an angle between 0-90 for the correct result.

Car width (w) (meter): This is the width(w) shown in Figure 6. Value has to be bigger than zero

Car length (l) (meter): This is the length(L) shown in Figure 6. Value has to be bigger than zero

X Value: This is the value shown in Figure 6 and Figure 7, which can be negative or positive.

Minimum Turning Radius (Rm) (meter): The minimum radius at which a car can turn a full circle.

All inputs need to be a number.

6.3. How Can I Use the Program

After opening the program enter the values you have picked to the related spaces as follows steering input angle (θ), car length (l), car width (w), the x value, Minimum turning radius (Rm)

For base input, you will get a pop-up showing an error value like this:

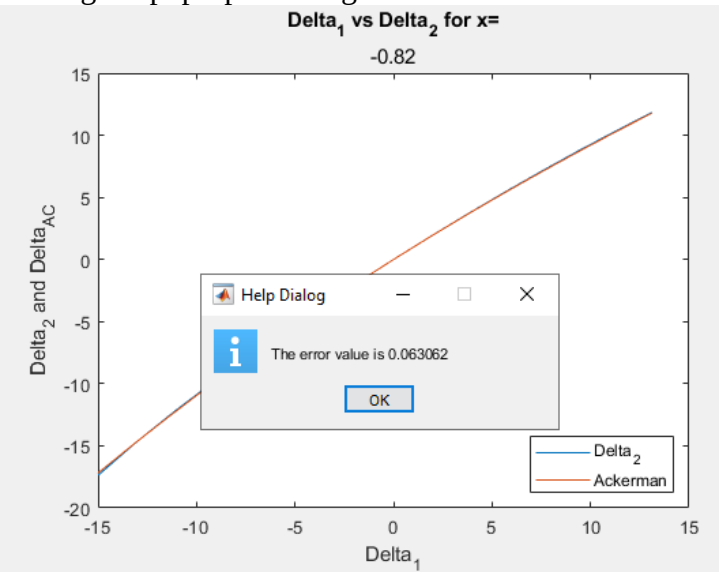

Figure 13: Result Window

After click ok button final result will look like this:

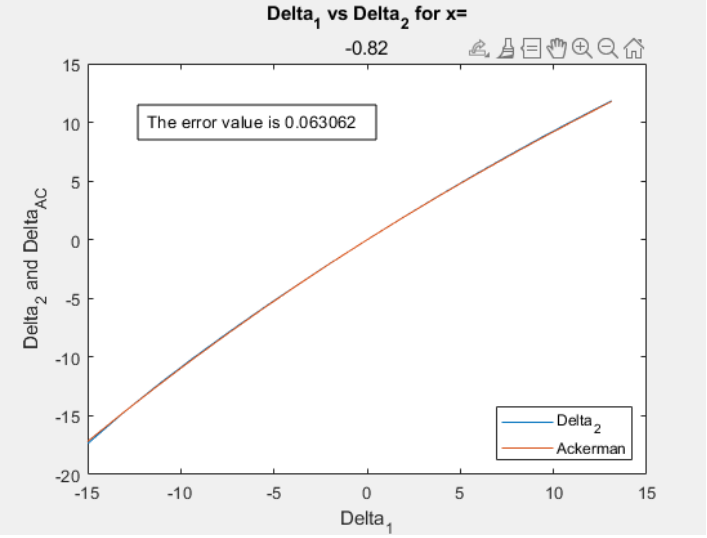

Figure 14: Final Result

6.4. Input Guide for Range of X

Users have 7 input options:

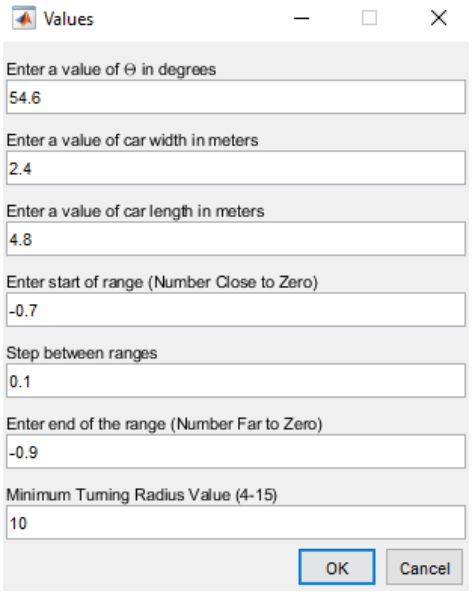

Figure 15: Input Window For Range

Steering input angle (θ)(degree): This is the angle "54.6" in Figure 8, I recommend picking an angle between 0-90 for the correct result.

Car width (w) (meter): This is the width(w) shown in Figure 6. Value has to be bigger than zero

Car length (l) (meter): This is the length(L) shown in Figure 6. Value has to be bigger than zero

Enter the start of the range: This is the start value for the X range. This value needed to be the value that is close to zero. If positive values will be used this warning need to be followed carefully to prevent error.

Step between ranges: This is the increment that will be added to numbers starting from range start and followed until reaching range end.

For example:
0.1 step, range: -0.7, -0.9
Steps: -0.7, -0.8, -0.9
0.05 step, range: -0.7, -0.9
Steps: -0.7, -0.75, -0.8, -0.85, -0.9

A higher step number will create a smoother error graph and will be much more accurate but it will take longer to finalize calculations. A lower step number will give a more rough but faster result.

Enter end of the range: This is the end value for the X range. This value needed to be the value that is far from zero. If positive values will be used this warning need to be followed carefully to prevent error.

Minimum Turning Radius (Rm) (meter): The minimum radius at which a car can turn a full circle.

Both range input needs to be in the same sign. Zero shouldn’t be used. All inputs need to be a number.

7. CONCLUSION

This project aims to create MATLAB software that quickly calculates the values to be used in the design of the six-link steering mechanism by using user-defined inputs. Six-link steering mechanism has complex steps of calculations, with this software calculation time can be drastically lowered. Using the “range of x” version, the software can help the user to pick the optimal x distance between determined joints and the user can check the error values compared to Ackerman Principle.

The advantage of this software is, it is more efficient and more reliable compared to other methods such as handmade calculation. The design of a six-link steering mechanism needs a series of calculations, and values obtained from solving one equation are used in another equation until the final result is obtained. Solving a problem like this by hand may take hours and this is only for a single x value. With the help of this software, this calculation can be done in seconds for each desired x value and with desired input values. With the second version, this calculation can be done within a range determined. And error values obtained from these calculations can be used to determine the optimal distance between joints. With lowering design time for the steering mechanism, excess time can be used in another part of the development process.

When the advancements in technology are considered, in the future this software can be part of greater software for a vehicle design program and can be integrated with other programs that are focused on vehicle design subjects.

The results obtained from the software are accurate and fast, UI is direct and clear. The drawbacks of this software are several.

1- UI could have been more user-friendly and more graphically pleasing.
2- Both codes could have been implemented in a single exe file.
3- The range version of the software can work with only a single mathematical sign for start and end input.

The design of software never ends, and new features and implementations always can be added.

Overall, in this project, I used Ackerman Principal and solved the complex equations in MATLAB, the project gave the user fast and accurate results, with the options for custom inputs.

REFERENCES

[1] Neider Nadid Romero-Núñez and Rafael Eduardo Tuiran-Villalba, "Optimization Method Of A Six-Bar Steering Mechanism Formulated With Natural Coordinates," *Dyna,* vol. 85, no. 207, pp. 168-173, 2018.

[2] P. Pfeffer, M. Harrer and D. Johnston, "Interaction Of Vehicle And Steering System Regarding On-Centre Handling," *Vehicle System Dynamics,* vol. 46, no. 5, pp. 413-428, 2008.

[3] P. Simionescu, "Initial Estimates In The Design Of Central-Lever Steering Linkages," *Journal Of Mechanical Design,* vol. 124, no. 4, pp. 646-651, 2002.

[4] R. N. Jazar, Vehicle Dynamics: Theory and Applications, New York: Springer, 2008.

[5] S. Lende, "Six Bar Steering Mechanism," *International Journal Of Advance Research And Innovative Ideas In Education,* vol. 3, no. 5, pp. 1560-1564, 2017.

[6] Chunguo Zhou, Tao Hua, Huiwu Wang and Xiuyu Wang, "Design and Error Analysis of The Linkage Type," *Proceedings of the 2019 International Conference on Modeling, Simulation, Optimization and Numerical Techniques (SMONT 2019),* vol. 165, pp. 219-225, April 2019.

[7] K. S. Mowade and S. M. Mowade, "Six-Bar Steering Mechanism," *Global Journal Of Engineering Science And Researches,* pp. 206-212, 2018.

APPENDICES

The MATLAB Code For Single X Value

```
clc
clear
```

Optimization of the multi-link steering mechanism

Inputs

```
syms delta_1 delta_2 dm;

% Base Values
% theta=54.6
% w=2.4;  %%Width of the vehicle in meter
% l=4.8;  %% Length of the vehicle in meter
% x=0.82;
% Rm=10;  %% Minimum radius for full circle turn in meter

prompt = {'Enter a value of \Theta in degrees',...
    'Enter a value of car width in meters',...
    'Enter a value of car length in meters','X Value','Minimum Turning Radius Value (4-
15)'};

dlgtitle = 'Values';
dims = [1 40; 1 50; 1 50; 1 20; 1 38];
definput = {'54.6','2.4','4.8','-0.82','10'};
opts.Interpreter = 'tex';
start = inputdlg(prompt,dlgtitle,dims,definput,opts);

inputleft=cell2mat(start(1));
inputleft=str2num(inputleft);
inputright=inputleft;

w=cell2mat(start(2));
w=str2num(w);

l=cell2mat(start(3));
l=str2num(l);

test=cell2mat(start(4));
test=str2num(test);

Rm=cell2mat(start(5));
Rm=str2num(Rm)

x=test;

a2=0.45*l;

inputleft_base=inputleft;
inputright=54.6;
distancep_c = 0.22;  % distance from C tp P
theta2=90-inputleft;
theta2base=theta2;
```

Calculating Delta Maximum

```
cot(delta_2)-cot(delta_1)== w/l;   %%d2=delta angle 2, d1=delta angle 1
% Rm=10;                   %%Vehicle must turn with Rm radious

sol=solve(sqrt((a2.^2)+((l.^2)*(cot(dm))))==Rm,dm);

dm=vpa(sol);
dm=double(dm);               %%Turning sym to double for able to calculate
% dm=0.23713;
```

Finding Inner and Outer Steer Angles

```
R1=l*cot(dm(dm>0));
di=atand(l/(R1-(w/2)));  %% Inner delta angle
ditop=ceil(di);
dibottom=floor(di);

    if ditop-di > 0.5
        di=ceil(di);    %% To be safe, we try to optimize the mechanism for upper angle
    else
        di=di+1;     %%% To be safe, we try to optimize the mechanism for greater angle
than original
    end

do=atand(l/(R1+(w/2))); %% Outer delta angle
```

Steer angles of Left and Right links

```
theta2 = sym('theta2');
phi4  = sym('phi4');
delta_1 = theta2-(90-inputleft);
delta_2 = phi4-(90+inputright);  %%phi angle
```

Putting Variables to Table

```
T = table('Size',[13 3],'VariableTypes',{'string','string','string'},...
    'VariableNames',{'Link','Length','Angle'});

T.Link(1) = '---';
T.Length(1) = 'Left Linkage';
T.Angle(1) = '---';

T.Link(2) = '1';
T.Length(2) = sprintf('d1=%f', w/2);
T.Angle(2) = '180';

a1=distancep_c/cosd(inputleft);
T.Link(3) = '2';
T.Length(3) = a1;
T.Angle(3) = 'Theta2';

b1=(w/2)-(distancep_c*tand(inputleft))-(x/2);
T.Link(4) = '3';
T.Length(4) = sprintf('%f-x/2', b1);
```

```
T.Angle(4) = 'Theta3';

T.Link(5) = '4';
T.Length(5) = sprintf('sqrt(%f+(x^2)/4)', distancep_c);
T.Angle(5) = 'Theta4';

T.Link(6) = '---';
T.Length(6) = '---';
T.Angle(6) = '---';

T.Link(7) = '---';
T.Length(7) = 'Right Linkage';
T.Angle(7) = '---';

T.Link(8) = 'Link';
T.Length(8) = 'Length';
T.Angle(8) = 'Angle';

T.Link(9) = '-----';
T.Length(9) = '-----';
T.Angle(9) = '-----';

T.Link(10) = '1';
T.Length(10) = sprintf('d1=%f', w/2);
T.Angle(10) = '180';

T.Link(11) = '4';
T.Length(11) = sprintf('sqrt(%f+(x^2)/4)', distancep_c);
T.Angle(11) = sprintf('phi2=Theta4-(2*tan^-1)*x/(%f*2)', distancep_c);

b2=b1;
T.Link(12) = '5';
T.Length(12) = sprintf('%f-x/2', b2);
T.Angle(12) = 'phi3';

c2=distancep_c/cosd(inputright);
T.Link(13) = '6';
T.Length(13) = sprintf('%f', c2);
T.Angle(13) = 'phi4';

disp(T)
```

Link	Length	Angle
"---"	"Left Linkage"	"---"
"1"	"d1=1.200000"	"180"
"2"	"0.37978"	"Theta2"
"3"	"0.890430-x/2"	"Theta3"
"4"	"sqrt(0.220000+(x^2)/4)"	"Theta4"
"---"	"---"	"---"
"---"	"Right Linkage"	"---"
"Link"	"Length"	"Angle"
"-----"	"-----"	"-----"
"1"	"d1=1.200000"	"180"
"4"	"sqrt(0.220000+(x^2)/4)"	"phi2=Theta4-(2*tan^-1)*x/(0.220000*2)"
"5"	"0.890430-x/2"	"phi3"
"6"	"0.379781"	"phi4"

Calculation of D2 Value For Given D1 Value

```
x=test;
theta2=linspace(theta2base-di,theta2base+di,17);

theta4=sym('theta4');
d1=w/2;
d2=w/2;

iteration=1;
val=1;
```

Theta 4

```
while iteration<18;

    a1=distancep_c/cosd(inputleft);
    b1=(w/2)-(distancep_c*tand(inputleft))-(x/2);
    c1=sqrt((distancep_c)^2+((x^2)/4));
    d1=w/2;

    J1=d1/a1;
    J2=d1/c1;
    J3=((a1^2)-(b1^2)+(c1^2)+(d1^2))/(2*a1*c1);
    J4=d1/b1;
    J5=(c1^2-d1^2-a1^2-b1^2)/(2*a1*b1);

    A=J3-J1+((1-J2)*cosd(theta2(iteration)));
    B=-2*sind(theta2(iteration));
    C=J1+J3-((1+J2)*cosd(theta2(iteration)));

    a1values(iteration)=a1;
    b1values(iteration)=b1;
    c1values(iteration)=c1;

    t1=(-B+sqrt(B^2-(4*A*C)));
    t2=(-B-sqrt(B^2-(4*A*C)));

    theta4=[(2*atand(t2/(2*A)))];
    theta4values(iteration)=theta4;
    iteration=iteration + 1;
    val=val+2;


end

theta4=theta4values;
```

Phi 4 and Delta 2

```
phi2=(theta4-(2*atand(x/(distancep_c*2))));

inputleft=inputleft_base;
theta2=linspace(theta2base-15,theta2base+15,17);
delta_1 = theta2-(90-inputleft);
```

```
iteration=1;
val=1;
x=test;

while iteration<17;


    a2=sqrt(((distancep_c)^2)+((x^2)/4));
    b2=(w/2)-(distancep_c*tand(inputright))-(x/2);
    c2=distancep_c/cosd(inputright);

    J1=(d2/a2);
    J2=(d2/c2);
    J3=((a2.^2)-(b2.^2)+(c2.^2)+((d2.^2)))/(2*a2*c2);
    J4=(d2/b2);
    J5=((c2.^2)-(d2.^2)-(a2.^2)-(b2.^2))/(2*a2*b2);

    if isnumeric(phi2)==0
        phi2=eval(phi2);
        phi2=subs(double(phi2));

    end

    A=(J3)-J1+((1-J2)*cosd(phi2(iteration)));

    B=-2*sind(phi2(iteration));
    C=J1+J3-((1+J2)*cosd(phi2(iteration)));



    a2values(iteration)=a2;
    b2values(iteration)=b2;
    c2values(iteration)=c2;

    t1=(-B+sqrt(B.^2-(4.*A.*C)));
    t2=(-B-sqrt(B.^2-(4.*A.*C)));
    phi4=[(2*atand(t2/(2*A)))];
    phi4values(iteration)=phi4;

    delta_2=phi4-(90+inputright);
    delta2values(iteration)=delta_2(1);

    iteration=iteration+1;

end

phi4=phi4values;
    delta_2=delta2values;
    n = numel(delta_2);
```

Plotting The First Graph

```
plot(delta_1(1:n),delta_2)
title('Delta_1 vs Delta_2 for x=',test)
xlabel('Delta_1')
ylabel('Delta_2 and Delta_A_C')
syms del_acker;
```

```
x=1;
while x < numel(delta_1)
    ack(x)=solve(cotd(del_acker)-cotd(delta_1(x)) == w/l, del_acker);
    x=x+1;

    if delta_1(x) == 0;
        x=x+1;
    end

end
ack=eval(ack);
hold on
plot(delta_1(1:n),ack)
legend('Delta_2','Ackerman','Location','southeast')
hold off
```

```
Warning: Imaginary parts of complex X and/or Y arguments ignored.
```

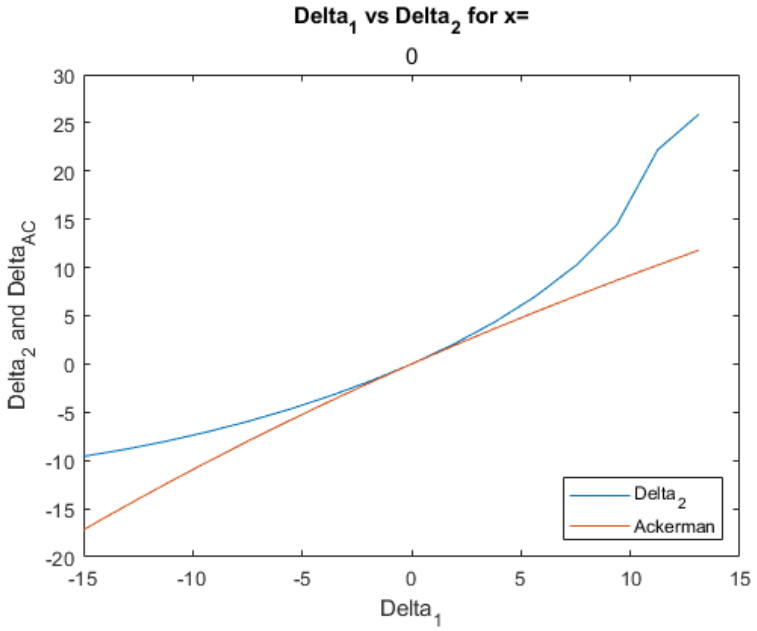

Figure 16: Delta 1 vs Delta 2 For x=0

Error Calculation

```
errord=delta_2-ack;
error=sqrt(sum(errord.^2)/numel(ack));
message = sprintf('The error value is %f', error);

uiwait(helpdlg(message));
annotation('textbox',[.2 .5 .3 .3],'String',message,'FitboxToText','on')
```

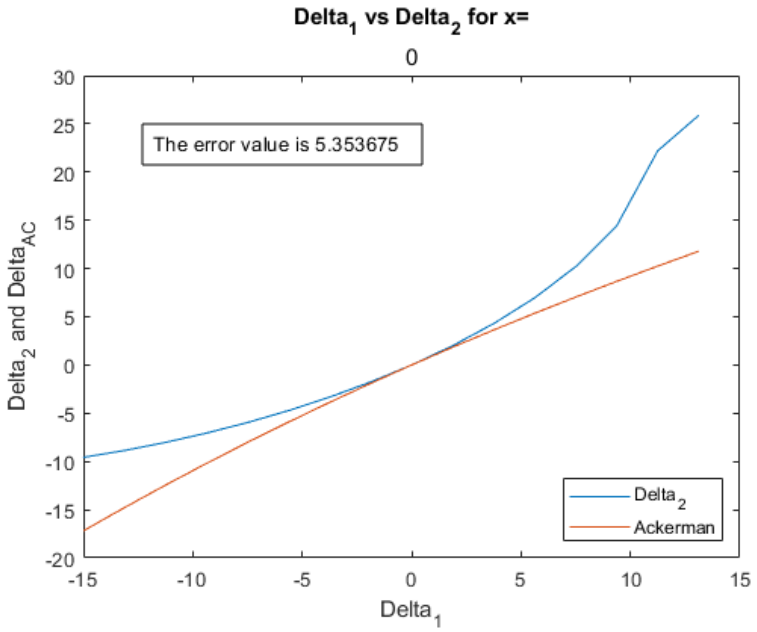

Figure 17: Delta 1 vs Delta 2 For x=0 with error

The MATLAB Code For X Values in Range

```
clc
clear
```

Optimization of multi-link steering mechanism

Inputs

```
syms delta_1 delta_2 dm;

% w=2.4;  %%Width of vehicle
% l=4.8;  %% Length of vehicle

prompt = {'Enter a value of \Theta in degrees',...
    'Enter a value of car width in meters',...
    'Enter a value of car length in meters','Enter start of range (Number Close to
Zero)',...
    'Step between ranges','Enter end of the range (Number Far to Zero)','Minimum Turning
Radius Value (4-15)'};

dlgtitle = 'Values';
dims = [1 50; 1 50; 1 50; 1 50; 1 50 ;1 50; 1 50];
definput = {'54.6','2.4','4.8','-0.7','0.1','-0.9','10'};
opts.Interpreter = 'tex';
start = inputdlg(prompt,dlgtitle,dims,definput,opts);

inputleft=cell2mat(start(1));
inputleft=str2num(inputleft);
inputright=inputleft;

w=cell2mat(start(2));
w=str2num(w);
```

```
l=cell2mat(start(3));
l=str2num(l);

rangestart=cell2mat(start(4));
rangestart=str2num(rangestart);

rangeend=cell2mat(start(6));
rangeend=str2num(rangeend);

step=cell2mat(start(5));
step=str2num(step);

Rm=cell2mat(start(7));
Rm=str2num(Rm);

a2=0.45*l;

inputleft_base=inputleft;
distancep_c = 0.22;  % distance from C tp P
theta2=90-inputleft;
theta2base=theta2;
```

Calculating Delta Maximum

```
cot(delta_2)-cot(delta_1)== w/l;   %%d2=delta angle 2, d1=delta angle 1
% Rm=10;                    %%Vehicle must turn with Rm radious

sol=solve(sqrt((a2.^2)+((l.^2)*(cot(dm))))==Rm,dm);

dm=vpa(sol);
dm=double(dm);               %%Turning sym to double for able to calculate
% dm=0.23713;
```

Finding Inner and Outer Steer Angles

```
R1=l*cot(dm(dm>0));
di=atand(l/(R1-(w/2)));  %% Inner delta angle
ditop=ceil(di);
dibottom=floor(di);

    if ditop-di > 0.5
        di=ceil(di);    %% To be safe, we try to optimize the mechanism for upper angle
    else
        di=di+1;     %%% To be safe, we try to optimize the mechanism for greater angle
than original
    end

do=atand(l/(R1+(w/2))); %% Outer delta angle
```

Steer angles of Left and Right links

```
theta2 = sym('theta2');
phi4  = sym('phi4');
delta_1 = theta2-(90-inputleft);
delta_2 = phi4-(90+inputright);  %%phi angle
```

Putting Variables to Table

```
T = table('Size',[13 3],'VariableTypes',{'string','string','string'},...
    'VariableNames',{'Link','Length','Angle'});

T.Link(1) = '---';
T.Length(1) = 'Left Linkage';
T.Angle(1) = '---';

T.Link(2) = '1';
T.Length(2) = sprintf('d1=%f', w/2);
T.Angle(2) = '180';

a1=distancep_c/cosd(inputleft);
T.Link(3) = '2';
T.Length(3) = a1;
T.Angle(3) = 'Theta2';

x=sym('x');
b1=w/2-distancep_c*tand(inputleft);
T.Link(4) = '3';
T.Length(4) = sprintf('%f-x/2', b1);
T.Angle(4) = 'Theta3';

T.Link(5) = '4';
T.Length(5) = sprintf('sqrt(%f+(x^2)/4)', distancep_c);
T.Angle(5) = 'Theta4';

T.Link(6) = '---';
T.Length(6) = '---';
T.Angle(6) = '---';

T.Link(7) = '---';
T.Length(7) = 'Right Linkage';
T.Angle(7) = '---';

T.Link(8) = 'Link';
T.Length(8) = 'Length';
T.Angle(8) = 'Angle';

T.Link(9) = '-----';
T.Length(9) = '-----';
T.Angle(9) = '-----';

T.Link(10) = '1';
T.Length(10) = sprintf('d1=%f', w/2);
T.Angle(10) = '180';

T.Link(11) = '4';
T.Length(11) = sprintf('sqrt(%f+(x^2)/4)', distancep_c);
T.Angle(11) = sprintf('phi2=Theta4-(2*tan^-1)*x/(%f*2)', distancep_c);

b2=b1;
T.Link(12) = '5';
T.Length(12) = sprintf('%f-x/2', b2);
T.Angle(12) = 'phi3';

c2=distancep_c/cosd(inputright);
```

```
T.Link(13) = '6';
T.Length(13) = sprintf('%f', c2);
T.Angle(13) = 'phi4';

disp(T)
```

Link	Length	Angle
"---"	"Left Linkage"	"---"
"1"	"d1=1.200000"	"180"
"2"	"0.37978"	"Theta2"
"3"	"0.890430-x/2"	"Theta3"
"4"	"sqrt(0.220000+(x^2)/4)"	"Theta4"
"---"	"---"	"---"
"---"	"Right Linkage"	"---"
"Link"	"Length"	"Angle"
"-----"	"-----"	"-----"
"1"	"d1=1.200000"	"180"
"4"	"sqrt(0.220000+(x^2)/4)"	"phi2=Theta4-(2*tan^-1)*x/(0.220000*2)"
"5"	"0.890430-x/2"	"phi3"
"6"	"0.379781"	"phi4"

Calculation of D2 Value For Given D1 Value

```
test=rangeend;
x=test;
theta2=linspace(theta2base-di,theta2base+di,17);

theta4=sym('theta4');
d1=w/2;
d2=w/2;

iteration=1;
val=1;
```

Theta 4

```
counter=1;
errorcount=1;
erroral1=0;

if rangestart < 0
    test=rangeend;
    x=test;
    while test<rangestart;
while iteration<18;


    a1=distancep_c/cosd(inputleft);
    b1=(w/2)-(distancep_c*tand(inputleft))-(x/2);
    c1=sqrt((distancep_c)^2+((x^2)/4));
    d1=w/2;

    J1=d1/a1;
    J2=d1/c1;
    J3=((a1^2)-(b1^2)+(c1^2)+(d1^2))/(2*a1*c1);
```

```
    J4=d1/b1;
    J5=(c1^2-d1^2-a1^2-b1^2)/(2*a1*b1);



    A=J3-J1+((1-J2)*cosd(theta2(iteration)));
    B=-2*sind(theta2(iteration));
    C=J1+J3-((1+J2)*cosd(theta2(iteration)));


    a1values(iteration)=a1;
    b1values(iteration)=b1;
    c1values(iteration)=c1;


    t1=(-B+sqrt(B^2-(4*A*C)));
    t2=(-B-sqrt(B^2-(4*A*C)));


    theta4=[(2*atand(t2/(2*A)))];
    theta4values(iteration)=theta4;
    iteration=iteration + 1;
    val=val+2;

end

theta4=theta4values;
```

Phi 4

```
phi2=(theta4-(2*atand(x/(distancep_c*2))));



inputleft=inputleft_base;
theta2=linspace(theta2base-15,theta2base+15,17);
delta_1 = theta2-(90-inputleft);


iteration=1;
val=1;
x=test;

while iteration<17;


    a2=sqrt(((distancep_c)^2)+((x^2)/4));
    b2=(w/2)-(distancep_c*tand(inputright))-(x/2);
    c2=distancep_c/cosd(inputright);

    J1=(d2/a2);
    J2=(d2/c2);
    J3=((a2.^2)-(b2.^2)+(c2.^2)+((d2.^2)))/(2*a2*c2);
    J4=(d2/b2);
    J5=((c2.^2)-(d2.^2)-(a2.^2)-(b2.^2))/(2*a2*b2);
```

```
    if isnumeric(phi2)==0
        phi2=eval(phi2);
        phi2=subs(double(phi2));

    end

    A=(J3)-J1+((1-J2)*cosd(phi2(iteration)));



    B=-2*sind(phi2(iteration));
    C=J1+J3-((1+J2)*cosd(phi2(iteration)));


    a2values(iteration)=a2;
    b2values(iteration)=b2;
    c2values(iteration)=c2;

    t1=(-B+sqrt(B.^2-(4.*A.*C)));
    t2=(-B-sqrt(B.^2-(4.*A.*C)));
    phi4=[(2*atand(t2/(2*A)))];
    phi4values(iteration)=phi4;



    delta_2=phi4-(90+inputright);
    delta2values(iteration)=delta_2(1);



    iteration=iteration+1;

end


phi4=phi4values;
    delta_2=delta2values;
    n = numel(delta_2);
```

Plotting The First Graph

delta_1=eval(delta_1)

```
figure
plot(delta_1(1:n),delta_2)
title('Delta_1 vs Delta_2 for x=',test)
xlabel('Delta_1')
ylabel('Delta_2 and Delta_A_C')
syms del_acker;
x=1;
while x < numel(delta_1)
    ack(x)=solve(cotd(del_acker)-cotd(delta_1(x)) == w/l, del_acker);
    x=x+1;

    if delta_1(x) == 0;
        x=x+1;
    end
```

```
end

if isscalar(ack) == 1;
    ack=eval(ack)
else
end
    hold on
plot(delta_1(1:n),ack)
legend('Delta_2','Ackerman','Location','southeast')
hold off
```

Error Calculation

```
errord=delta_2-ack;
error=sqrt(sum(errord.^2)/numel(ack));
erroall(errorcount)=error;
errorcount=errorcount+1;
test=test+step;
x=test;
iteration=1;
plot1=(rangeend:step:rangestart);
perc=numel(plot1)+1;
percentage=linspace(0,100,perc);
disp([num2str(percentage(errorcount)), '% Progress'])
delta_2=0;
delta2values=0;
```

```
4.7619% Progress
9.5238% Progress
14.2857% Progress
19.0476% Progress
23.8095% Progress
28.5714% Progress
33.3333% Progress
38.0952% Progress
42.8571% Progress
47.619% Progress
52.381% Progress
57.1429% Progress
61.9048% Progress
66.6667% Progress
71.4286% Progress
76.1905% Progress
80.9524% Progress
85.7143% Progress
90.4762% Progress
95.2381% Progress
```

```
end
else

test=rangestart;
x=test;
```

```
errorcount=1;

    while test<rangeend;
while iteration<18;
    a1=distancep_c/cosd(inputleft);
    b1=(w/2)-(distancep_c*tand(inputleft))-(x/2);
    c1=sqrt((distancep_c)^2+((x^2)/4));
    d1=w/2;

    J1=d1/a1;
    J2=d1/c1;
    J3=((a1^2)-(b1^2)+(c1^2)+(d1^2))/(2*a1*c1);
    J4=d1/b1;
    J5=(c1^2-d1^2-a1^2-b1^2)/(2*a1*b1);


    A=J3-J1+((1-J2)*cosd(theta2(iteration)));
    B=-2*sind(theta2(iteration));
    C=J1+J3-((1+J2)*cosd(theta2(iteration)));

    a1values(iteration)=a1;
    b1values(iteration)=b1;
    c1values(iteration)=c1;


    t1=(-B+sqrt(B^2-(4*A*C)));
    t2=(-B-sqrt(B^2-(4*A*C)));


    theta4=[(2*atand(t2/(2*A)))];
    theta4values(iteration)=theta4;
    iteration=iteration + 1;
    val=val+2;

end

theta4=theta4values;
```

Phi 4

```
phi2=(theta4-(2*atand(x/(distancep_c*2))));



inputleft=inputleft_base;
theta2=linspace(theta2base-15,theta2base+15,17);
delta_1 = theta2-(90-inputleft);


iteration=1;
val=1;
x=test;

while iteration<17;

    a2=sqrt(((distancep_c)^2)+((x^2)/4));
    b2=(w/2)-(distancep_c*tand(inputright))-(x/2);
    c2=distancep_c/cosd(inputright);
```

```
    J1=(d2/a2);
    J2=(d2/c2);
    J3=((a2.^2)-(b2.^2)+(c2.^2)+((d2.^2)))/(2*a2*c2);
    J4=(d2/b2);
    J5=((c2.^2)-(d2.^2)-(a2.^2)-(b2.^2))/(2*a2*b2);

    if isnumeric(phi2)==0
        phi2=eval(phi2);
        phi2=subs(double(phi2));

    end

    A=(J3)-J1+((1-J2)*cosd(phi2(iteration)));
    B=-2*sind(phi2(iteration));
    C=J1+J3-((1+J2)*cosd(phi2(iteration)));


    a2values(iteration)=a2;
    b2values(iteration)=b2;
    c2values(iteration)=c2;


    t1=(-B+sqrt(B.^2-(4.*A.*C)));
    t2=(-B-sqrt(B.^2-(4.*A.*C)));
    phi4=[(2*atand(t2/(2*A)))];
    phi4values(iteration)=phi4;

    delta_2=phi4-(90+inputright);
    delta2values(iteration)=delta_2(1);

    iteration=iteration+1;

end


phi4=phi4values;
    delta_2=delta2values;
    n = numel(delta_2);
```

Plotting The First Graph

delta_1=eval(delta_1)

```
figure
plot(delta_1(1:n),delta_2)
title('Delta_1 vs Delta_2 for x=',test)
xlabel('Delta_1')
ylabel('Delta_2 and Delta_A_C')
syms del_acker;
x=1;
while x < numel(delta_1)
    ack(x)=solve(cotd(del_acker)-cotd(delta_1(x)) == w/l, del_acker);
    x=x+1;

    if delta_1(x) == 0;
        x=x+1;
    end
```

```
end

if isscalar(ack) == 1;
    ack=eval(ack)
else
end
    hold on
plot(delta_1(1:n),ack)
legend('Delta_2','Ackerman','Location','southeast')
hold off
```

Error Calculation

```
errord=delta_2-ack;
error=sqrt(sum(errord.^2)/numel(ack));
erroall(errorcount)=error;
errorcount=errorcount+1;
test=test+step;
x=test;
iteration=1;
plot1=(rangestart:step:rangeend);
perc=numel(plot1)+1;
percentage=linspace(0,100,perc);
disp([num2str(percentage(errorcount)), '% Progress'])
test;
delta_2;
delta_2=0;
delta2values=0;
```

```
end
end

if rangestart < 0;
    legend('Delta_2','Ackerman','Location','southeast')
    hold off
    figure
    plot1=rangeend:step:rangestart;

    [mm,location]=(min(erroall));
    message = sprintf('The minimum error value is %f At x =
    %f',min(erroall),plot1(location));
    uiwait(helpdlg(message));
    annotation('textbox',[.2 .5 .3 .3],'String',message,'FitboxToText','on')

    if numel(plot1) == numel(erroall)
       plot(rangeend:step:rangestart,erroall)
    elseif numel(plot1) < numel(erroall)
        plot(rangeend:step:rangestart+step,erroall)
    else
        plot(rangeend:step:rangestart-step,erroall)
    end
    title('Error Graph')
    xlabel('X[m]');
    ylabel('e[deg]');
```

```
else
    legend('Delta_2','Ackerman','Location','southeast')
    hold off
    figure
    plot1=rangestart:step:rangeend;

    [mm,location]=(min(errorall));
    message = sprintf('The minimum error value is %f At x =
    %f',min(errorall),plot1(location));
    uiwait(helpdlg(message));
    annotation('textbox',[.2 .5 .3 .3],'String',message,'FitboxToText','on')

    if numel(plot1) == numel(errorall)
       plot(rangestart:step:rangeend,errorall)
    elseif numel(plot1) < numel(errorall)
        plot(rangestart:step:rangeend+step,errorall)
    else
        plot(rangestart:step:rangeend-step,errorall)
    end
    title('Error Graph')
    xlabel('X[m]');
    ylabel('e[deg]');
end
```

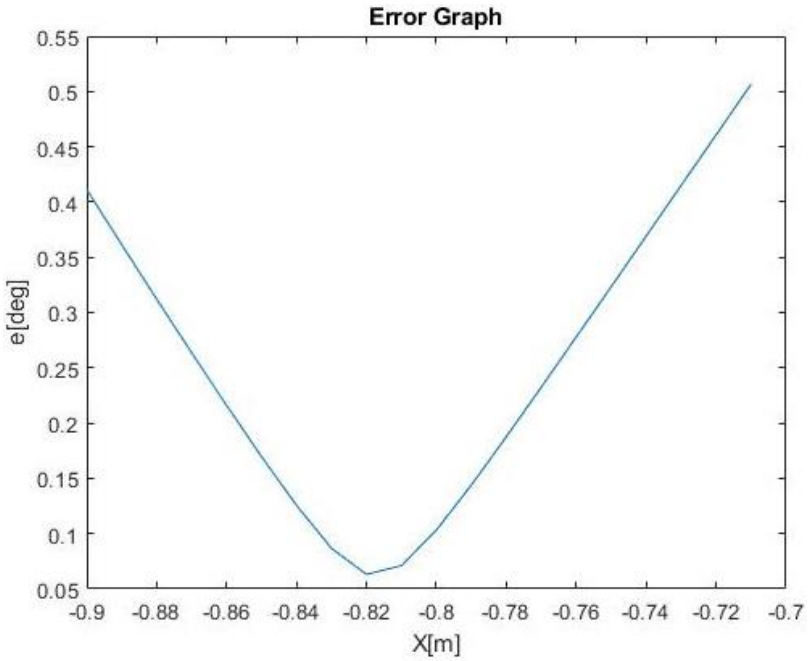

Figure 18: Error Graph for a range of x=(-0.7,-0.9)

Published with MATLAB® R2020b