STEERING OPTIMIZATION OF AN 8X8 VEHICLE

8X8 BİR ARACIN DÖNÜŞ OPTİMİZASYONU

CAHİT BARTU YAZICI

ASSIST. PROF. DR. –Ing. EMİR KUTLUAY
Supervisor

Submitted to
Graduate School of Science and Engineering of Hacettepe University
As a Partial Fulfillment to the Requirements for the Award of the Degree of Master of Science in Mechanical Engineering.

2019

This work titled **"Steering Optimization of an 8X8 Vehicle"** by **CAHİT BARTU YAZICI** has been approved as a thesis for Degree of **MASTER OF SCIENCE IN MECHANICAL ENGINEERING** by the Examining Committee Members mentioned below.

Assist. Prof. Dr. Selçuk HİMMETOĞLU
Head ....

Assist. Prof. Dr.-ing Emir KUTLUAY
Supervisor ....

Prof. Dr. Y. Samim ÜNLÜSOY
Member ....

Assoc. Prof. Dr. Yiğit YAZICIOĞLU
Member ....

Assoc. Prof. Dr. Ufuk ŞAHİN
Member ....

This thesis has been approved as a thesis for Degree of **MASTER OF SCIENCE IN MECHANICAL ENGINEERING** by Board of Directors of the Institute for Graduate School of Science and Engineering on ...... / .... / ....

Prof. Dr. Menemşe GÜMÜŞDERELİOĞLU
Director of the Institute of
Graduate School of Science and Engineering

To my family…

ETHICS

In this thesis study, prepared in accordance with the spelling rules of Institute of Graduate School of Science and Engineering of Hacettepe University,

I declare that

- all the information and documents have been obtained in the base of the academic rules
- all audio-visual information and results have been presented according to the rules of scientific ethics
- in case of using other works, related studies have been cited in accordance with the scientific standards
- all cited studies have been fully referenced
- I did not do any distortion in the data set
- and any part of this thesis has not been presented as another thesis study at this or any other university.

17.06.2019

Cahit Bartu YAZICI

YAYINLANMA FİKRİ MÜLKİYET HAKKLARI BEYANI

Enstitü tarafından onaylanan lisansüstü tezimin/raporumun tamamını veya herhangi bir kısmını, basılı (kağıt) ve elektronik formatta arşivleme ve aşağıda verilen koşullarla kullanıma açma iznini Hacettepe üniversitesine verdiğimi bildiririm. Bu izinle Üniversiteye verilen kullanım hakları dışındaki tüm fikri mülkiyet haklarım bende kalacak, tezimin tamamının ya da bir bölümünün gelecekteki çalışmalarda (makale, kitap, lisans ve patent vb.) kullanım hakları bana ait olacaktır.

Tezin kendi orijinal çalışmam olduğunu, başkalarının haklarını ihlal etmediğimi ve tezimin tek yetkili sahibi olduğumu beyan ve taahhüt ederim. Tezimde yer alan telif hakkı bulunan ve sahiplerinden yazılı izin alınarak kullanması zorunlu metinlerin yazılı izin alarak kullandığımı ve istenildiğinde suretlerini Üniversiteye teslim etmeyi taahhüt ederim.

Yükseköğretim Kurulu tarafından yayınlanan "***Lisansüstü Tezlerin Elektronik Ortamda Toplanması, Düzenlenmesi ve Erişime Açılmasına İlişkin Yönerge***" kapsamında tezim aşağıda belirtilen koşullar haricince YÖK Ulusal Tez Merkezi / H. Ü. Kütüphaneleri Açık Erişim Sisteminde erişime açılır.

☐ Enstitü / Fakülte yönetim kurulu kararı ile tezimin erişime açılması mezuniyet tarihimden itibaren 2 yıl ertelenmiştir.

☐ Enstitü / Fakülte yönetim kurulu gerekçeli kararı ile tezimin erişime açılması mezuniyet tarihimden itibaren .... ay ertelenmiştir.

☐ Tezim ile ilgili gizlilik kararı verilmiştir.

17/06/2019

CAHİT BARTU YAZICI

ABSTRACT

STEERING OPTIMIZATION OF AN 8X8 VEHICLE

Cahit Bartu YAZICI
Master of Science Degree, Department of Mechanical Engineering
Supervisor: ASSIST. PROF. DR. -Ing. EMİR KUTLUAY

June 2019, 154 pages

In this work, a method for obtaining optimal steering angles for a multi-axle vehicle is proposed. The tire maximum steering angles are optimized for low turning radius and low lateral slip angles. The optimization process is conducted for the low-speed steady-state maneuver. Another method is proposed to investigate the high-speed handling behavior of the vehicle with the optimized wheel steering angles in a double lane change (DLC) maneuver.

The vehicle considered in this work is a prototype 8x8 vehicle with all-wheel steered (AWS) and all-wheel drive (AWD). The 8x8 vehicle has a minimum curb-to-curb turning radius requirement, and the minimum turning radius requirement cannot be met using slip-free turning due to the wheel steering angle limitations. To achieve a turning radius, that is lower than the requirement, tire lateral slip angles must occur. A multi-body dynamics model of the all-wheel steered 8x8 vehicle was created in Adams Car. Adams model was parametrized and automated using a Python code. For the optimization process a genetic algorithm code was written in Python language. The optimization problem is multi-variable constraint optimization, each of the tire maximum steering angle is an optimization parameter and there are physical constraints on the wheel steering angles. For the DLC maneuver, a general open loop steering input function for DLC maneuvers was derived. The genetic algorithm created for the optimization of the wheel steering angles for the turning radius optimization section of the work was revised to optimize the parameters of the general open loop steering input function for the DLC maneuvers.

By using this method the steering angles for the tires that results in minimum turning radius and tire lateral slip angles which are capable of performing DLC maneuver at high speeds can be obtained. The methodology developed is especially useful for determining the tire steer angles of the vehicle early in the vehicle design and to create design specifications for the steering system.

Keywords; All-Wheel Steering, Multi-Axle vehicle, Handling, Optimization, Multi-Body Dynamics, Genetic Algorithm.

ÖZET

8x8 BİR ARACIN DÖNÜŞ OPTİMİZASYONU

Cahit Bartu YAZICI
Master of Science Degree, Department of Mechanical Engineering
Tez danismani: Dr. Öğr. Üyesi-Ing EMİR KUTLUAY

Haziran 2019, 154 sayfa

Bu çalışmada, çok akslı bir aracın optimal teker dönüş açılarını elde etmek için bir metod önerilmektedir. Teker maksimum dönüş açıları, düşük dönüş yarıçapı ve düşük teker yanal kayma açıları sağlayacak şekilde optimize edilmiştir. Optimizasyon süreci, düşük hız ve denge durumunda gerçekleştirilmiştir. Optimize edilmiş teker dönüş açılarına sahip aracın, yüksek hızlardaki dönüş davranışını incelemek için, aracı çift şerit değiştirme manevrasından geçirmek için başka bir metod geliştirilmiştir.

Bu çalışmada, tüm tekerlerden tahrik edilen, tüm tekerlerinden yönlendirilebilen prototip bir 8x8 araç ele alınmaktadır. Ele alınan 8x8 aracın dönüş yarıçap isteri, teker dönüş açı limitleri nedeniyle, kayma açısı olmadan sağlanamamaktadır. Dönüş yarıçapı isterini sağlamak için tekerlerde kayma açıları olmak zorundadır. Tüm tekerleri döndürülen 8x8 araç modeli Adams Car programında oluşturulmuştur. Adams Car programında oluşturulan modelin prametrelerini değiştirmek ve otomatize etmek için bir Python kodu yazılmıştır. Optimizasyon süreci için Python dilinde bir genetik algoritma geliştirilmiştir.

Optimizasyon problemi, çok değişkenli ve sınırlandırılmış bir problemdir, her bir teker yönlendirme açısı optimize edilecek bir parametredir ve teker açılari üzerinde fiziksel sınırlar vardır. Çift şerit değiştirme manevrası için genelleştirilmiş açık döngü bir fonksiyon geliştirilmiştir. Dönüş yarıçapı optimizasyonu için geliştirilen genetik algoritma şerit değiştirme manevrası için geliştirilen fonksiyonun parametrelerini optimize etmek için kullanılmıştır.

Bu metodu kullanarak, en düşük dönüş yarıçapını ve düşük teker kayma açılarını sağlayan ve aynı zamanda çift şerit değiştirme menevrası yapabilen teker dönüş açıları bulunabilir. Geliştirilen metod özellikle tasarım aşamasının başlarında teker açılarını belirlemek ve dönüş sistemi tasarım kriteri geliştirmek için kullanılabilir,

Anahtar Kelimeler : Tüm Tekerlerden Yönlendirme, Çok Akslı Araç, Dönüş, Optimizasyon, Çok Kütleli Dinamik, Genetik Algoritma.

ACKNOWLEDGMENTS

First, I would like to express my gratitude to my supervisor, ASSIST. PROF. DR. -Ing. Emir Kutluay, who has guided me during my thesis and helped me with his valuable insights. It has been an invaluable experience to write this thesis under his guidance.

I can not thank my colleagues İsmail Göçer and Çağıl Çiloğlu from FNSS Defense Systems enough for their continuous support and helpful advice. Their thoughts and guidance always lighted my way. Also, I would like to thank Prof. Dr. Y. Samim Ünlüsoy for his valuable feedback and advice.

My endless thanks go to my wife Nagehan Vanlıoğlu Yazıcı, without her, this thesis would never be possible. Her support and sensible advice turn this tough process into a pleasure.

Last but not least, I am eternally indebted to my mother Şengül Tunca who always supported me in every single possible way through my life. I would like to thank my family for their endless support and patience during this thesis.

TABLE OF CONTENT

ABSTRACT i

ÖZET iii

ACKNOWLEDGMENTS v

LIST OF FIGURES xi

LIST OF TABLES xv

LIST OF SYMBOLS xvii

LIST OF ABBREVIATIONS 1

1 INTRODUCTION 3
- 1.1 Problem Definition 3
- 1.2 Aim And Scope Of The thesis 9
- 1.3 Thesis Outline 11

2 LITERATURE SURVEY 13
- 2.1 Introduction 13
- 2.2 Vehicle Dynamics 14
- 2.3 Multi-Axle Vehicle Dynamics 18
- 2.4 Optimization 20
- 2.5 Genetic Algorithms 22
- 2.6 Conclusion 25

3 THEORETICAL BACKGROUND 27
- 3.1 Introduction 27
- 3.2 Bicycle Model 28
- 3.3 Multi-Axle Bicycle Model 31
- 3.4 Multi-body Dynamics Vehicle Model In Adams Car 35
  - 3.4.1 Body 37
  - 3.4.2 Suspension 37

3.4.3 Powertrain System 39

3.4.4 Brake System 40

3.4.5 Tire 40

3.5 Using Python In Adams 43

3.6 Genetic Algorithm 44

3.6.1 Initial population 46

3.6.2 Fitness 46

3.6.3 Selection 48

3.6.3.1 Basic Selection 48

3.6.3.2 Random Selection 48

3.6.3.3 Roulette Wheel Selection 50

3.6.4 Pairing 51

3.6.4.1 Fittest Individual Pairing 51

3.6.4.2 Random Pairing 52

3.6.4.3 Weighted Random Pairing 52

3.6.5 Mating 53

3.6.5.1 Single Point Mating 54

3.6.5.2 Two Points Mating 55

3.6.5.3 Single Station Mating 55

3.6.6 Mutation 57

3.6.6.1 Reset Mutation 57

3.6.6.2 Gauss Distribution Mutation 58

3.6.7 Next Generation 58

3.6.8 Termination of the Genetic Algorithm 59

3.7 Conclusion 61

4 TURNING RADIUS OPTIMIZATION 63

4.1 Introduction 63

4.2 Maneuver 64

4.3 Adams Command File 66

4.4 Python Code 67

4.4.1 Genetic Algorithm 68

4.4.2 Initial Population 70

4.4.3 Fitness Function 71

4.4.4 Termination Criterion 75

4.5 Current Configuration Results 76

4.6 Optimization Results 78

4.7 Conclusion 86

5 NATO LANE CHANGE SIMULATIONS 89

5.1 Introduction 89

5.2 General Open Loop Steering Input For Lane Change Maneuvers 90

5.3 Nato Lane Change Maneuver 92

5.4 Adams Command File 93

5.5 Python Code 93

5.5.1 Genetic Algorithm 94

5.5.2 Fitness Function 95

5.5.3 Termination Criteria 97

5.6 DLC Study Results 97

5.7 Conclusion 108

6 DISCUSSION AND CONCLUSIONS 111

6.1 Turing Radius Optimization Discussion 111

6.2 Nato Lane Change Results Discussion 113

6.3 Conclusion 115

6.4 Future Work 116

REFERENCES 117

APPENDIX A 121

APPENDIX B 125

APPENDIX C 127

APPENDIX D 139

APPENDIX E 141

LIST OF FIGURES

1.1 Four-wheel vehicle with Ackermann steering. 4

1.2 Examples of multi-axle vehicles with slip-free turning. 4

1.3 Multi-axle vehicle with an Ackermann and improvised steering methods. . . . 5

1.4 Multi-body dynamics vehicle model in Adams Car 6

1.5 Vehicle path during Nato lane change maneuver. 9

3.1 Bicycle model. 28

3.2 Lateral force vs lateral slip. 30

3.3 Lateral force vs lateral slip. 31

3.4 Four-axle bicycle model. 32

3.5 Vehicle path during DLC maneuver. 33

3.6 Steering angle input during the DLC maneuver. 34

3.7 Lateral slip angle during the DLC maneuver. 34

3.8 Lateral acceleration during the DLC maneuver. 35

3.9 Multi-body dynamics vehicle model in Adams Car 36

3.10 Body subsystem. 37

3.11 Double wishbone suspension subsystem in Adams Car. 38

3.12 Modified double wishbone suspension system. 39

3.13 Powertrain subsystem. 39

3.14 Adams help tire use cases.[40] 41

3.15 Adams tire workflow.[40] 42

3.16 Genetic algorithm flow chart. 45

3.17 Example of a generation. 46

3.18 Basic selection example. 49

3.19 Random selection example. 49

3.20 Roulette wheel selection example. 50

3.21 Fittest individual pairing example. 52

3.22 Random pairing example. 53

3.23 Weighted random pairing example. 54

3.24 Single point mating example. 55

3.25 Two points mating example. 56

3.26 Single station mating example. 56
3.27 Reset mutation example. 57
3.28 Gauss distribution example. 58
3.29 2D fitness value visualization. 60

4.1 Turning radius optimization flow chart. 63
4.2 Topdown look at the vehicle during the low-speed steady-state turning maneuver. 65
4.3 Vehicle velocity change during the maneuver. 65
4.4 Steering inputs given to the tires during the maneuver. 66
4.5 Adams command file work flow. 67
4.6 Work flow of the Python code. 68
4.7 Fitness function flow chart. 72
4.8 Tire lateral slip angles during the maneuver. 77
4.9 Maximum fitness change with respect to generation for all of the runs 79
4.10 Average fitness change with respect to generation for all of the optimizations processes. 80
4.11 Comparison of Run 1 and Run 2. 81
4.12 Comparison of Run 1 and Run 5. 82
4.13 Comparison of Run4 and Run6. 83
4.14 Comparison of Run1, Run3 and Run4. 84
4.15 Vehicle paths during the steady-state cornering maneuver. 85
4.16 Maximum slip angles of all the runs. 85

5.1 Steering angle input for the lane change maneuver 91
5.2 Nato lane change maneuver specifications. [2] 92
5.3 Ackermann steering example for one axle. 94
5.4 Fitness function flow chart for the DLC maneuver. 95
5.5 Maximum and average fitness of the original configuration optimization process. 98
5.6 Path of the vehicle with the original steering angle configuration. 99
5.7 Maximum and average fitness of the run 1 configuration 99
5.8 Path of the vehicle with the run 1 steering angle configuration. 100
5.9 Maximum and average fitness of the run 6 configuration 101
5.10 Path of the vehicle with the run 6 steering angle configuration. 101

5.11 Steering wheel turn percentage for the vehicle with different configurations. . 102
5.12 Velocities of the vehicle with different steering angle configurations during the DLC maneuver. 103
5.13 Throttle demands for all of the configurations. 104
5.14 Lateral acceleration change with respect to time. 104
5.15 Yaw rates of the vehicle with different steering angle configurations during the DLC maneuver. 105
5.16 Roll angles of the vehicle with different steering angle configurations during the DLC maneuver. 105
5.17 Normal rear left tire forces with different steering angle configurations during the DLC maneuver. 106
5.18 Slip angles rear left tire forces with different steering angle configurations during the DLC maneuver. 107

6.1 Paths of all the configurations during the DLC maneuver. 114

LIST OF TABLES

1.1 Number of possible solutions for the turning radius optimization study. 7

4.1 Binary vs floating number representation. 69
4.2 Gene limits. 70
4.3 Maximum wheel steering angles of the original configuration. 77
4.4 Properties of the optimization processes. 78
4.5 Optimization results. 83
4.6 Maximum wheel steering angles. 84

5.1 DLC genetic algorithm configurations. 98
5.2 DLC study results summary. 107

6.1 Turning radius optimization results. 112

LIST OF SYMBOLS

δ : Steering angle

u : Longitudinal velocity

v : Lateral velocity

θ : Yaw angle

$\dot{\theta}$: Yaw rate

$\ddot{\theta}$: Yaw acceleration

l_n : Distance between CoG and n^{th} axle

w : Vehicle width

δ_i : Inner wheel steering angle

δ_o : Outer wheel steering angle

F_{yn} : Lateral force of the tire at the n^{th} axle

α_n : Lateral slip angle of the tire at the n^{th} axle

c_n : Cornering stiffness of the tire at the n^{th} axle

T_{Brake} : Brake torque

BD : Brake demand

$T_{BrakeMax}$: Maximum brake torque

B, C, D, E : Magic formula parameters

m_o : Original value

m_s : Scaled value

σ : Standard deviation

r : Turning radius

x_i : x coordinate of the i^{th} point

y_i : y coordinate of the i^{th} point

α_{maxn} : Normalized maximum tire lateral slip angle

r_n : Normalized vehicle turning radius

n_{run} : Number of simulations

$a1, b2, c1, d1, a2, b2, c2, d2$: Parameters of the steering input function

t : Time

LIST OF ABBREVIATIONS

DLC : Double lane change
4WS : Four wheel steering
AWD : All wheel drive
AWS : All wheel steer
GA : Genetic algorithm
API : Application programming interface
FWS : Front wheel steering
FWD : Front wheel drive
RWD : Rear wheel drive
MPC : Model predictive control
DSTC : Dynamic stability and traction control
CoG : Center of gravity
IH : Impact hardness
ASAGA : Adaptive simulated annealing genetic algorithm
NFN : Neural Fuzzy Network
NATO : North Atlantic Treaty Organization

1 INTRODUCTION

1.1 Problem Definition

Military vehicles with pneumatic tires are expected to operate in urban environments as well as off-road. Urban environments are often very crowded and hard to operate in with big and heavy vehicles. To be useful in urban environments, users often demand a specific minimum turning radius.

Passenger vehicles are often designed to achieve Ackermann steering. As Harrer and Pfeffer [1] claimed; "Modern road vehicles are almost exclusively steered at the front wheels with a so-called Ackermann steering.". Ackermann steering is a kinematic condition that is defined between the inner and outer wheels that allow the tires to turn slip-free, the figure visualizing the Ackermann steering can be seen in Figure 1.1. This condition ensures that all the wheels roll freely with no lateral slip angle. High lateral slip angles will cause the tires to wear faster, and this may not be a big concern in race cars where the tires are not expected to last longer than the races, so steering geometries that do not follow a traditional Ackermann steering like parallel steering may be used to achieve better handling performance. However, in commercial and military vehicles where the vehicle is expected to operate for a very long time, tire degradation becomes crucial.

Where R is the distance from the turning radius to the vehicle center of gravity, w is the vehicle width, l is the distance between the front and rear axles, δ_i is the steering angle of the inner tire, and the δ_o is the steering angle of the outer tire.

For a vehicle which has Ackermann steering, the center of rotation for low-speed maneuvers is the point where all the normal lines from tires directions of motion cross each other. For

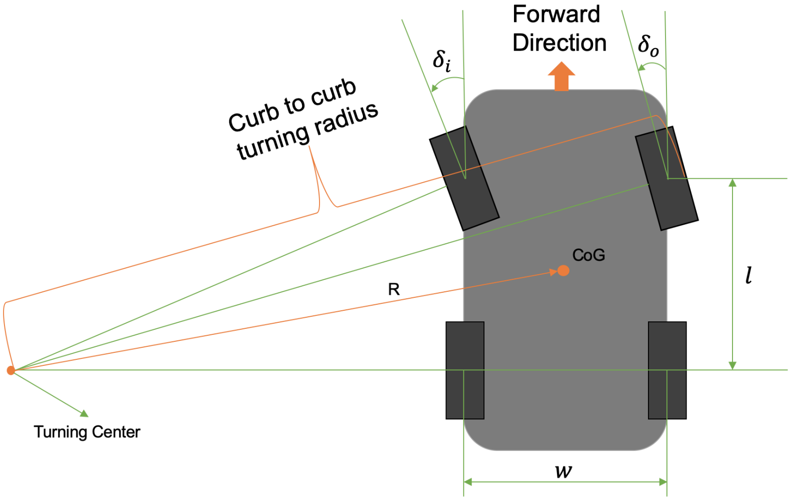


Figure 1.1: Four-wheel vehicle with Ackermann steering.

a multi-axle vehicle, Ackermann steering can only be achieved using multi-axle steering. To achieve slip-free turning a multi-axle vehicle that has n axles must be steered from at least $n-1$ axles. Figure 1.2 shows a few examples of multi-axle slip free turning.

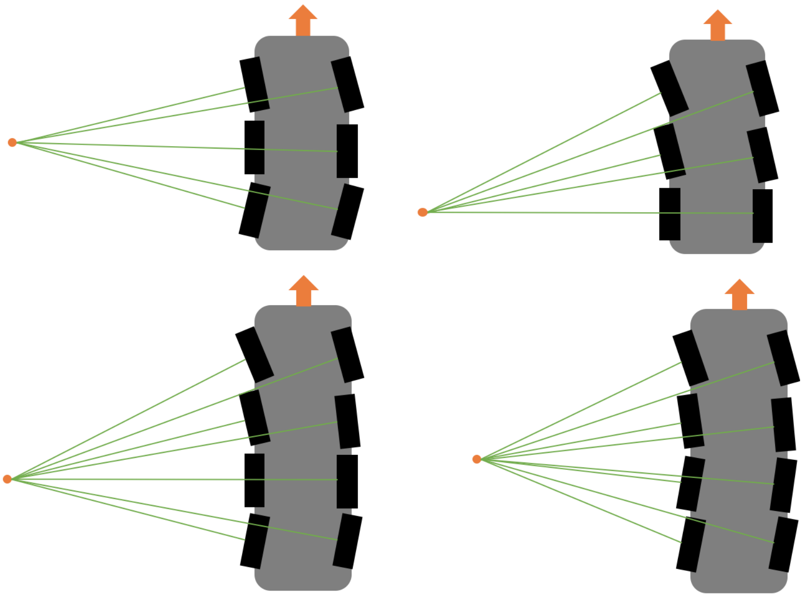


Figure 1.2: Examples of multi-axle vehicles with slip-free turning.

For a four-wheel vehicle the relationship between the inner and outer wheel steering angle can be written as;

$$tan(\delta_i) = \frac{l}{R - w/2} \tag{1.1}$$

also,

$$tan(\delta_o) = \frac{l}{R - w/2} \tag{1.2}$$

Since vehicle width (w) is a positive number, the steering angle of the inner tire (δ_i) is always higher than the steering angle of the outer tire (δ_o) for Ackermann steering. For a vehicle that has Ackermann steering geometry, the maximum steering angle of the inner tire the restricts the minimum turning radius of the vehicle. In some cases, the minimum turning radius that can be achieved by Ackermann steering does not meet the requirements due to steering angle limitations. The turning radius can be lowered using other methods such as parallel steering, but tire degradation and vehicle handling behavior at high speeds become a concern for multi-axle vehicles. So to achieve a specific turning radius and relatively low lateral slip angles, vehicle specific steering methods are used. In Figure 1.3, a multi-axle vehicle with Ackermann steering and an improvised vehicle specific steering geometry is given.

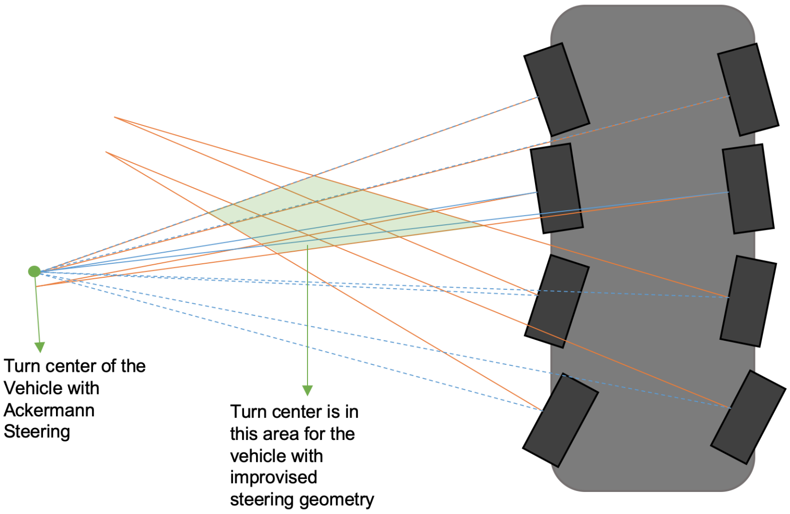

Turn center of the Vehicle with Ackermann Steering
Turn center is in this area for the vehicle with improvised steering geometry

Figure 1.3: Multi-axle vehicle with an Ackermann and improvised steering methods.

In this work, a prototype 8x8 vehicle with AWS is taken into consideration. Because of the maximum steering angle limit of the inner tires, slip-free turning does not provide an acceptable turning radius. So unconventional steering methods are needed to achieve a specified minimum turning radius and relatively low lateral slip angles. Calculating turning radius and tire lateral slip angles for steering methods other than Ackermann steering is not a straight forward process. In this work multi-body dynamics model of the vehicle is used to simulate the low-speed steady-state turning maneuver to calculate the turning radius and the tire lateral slip angles. The multi-body dynamics model used in this work can be seen in Figure 1.4. The multi-body dynamics model of the vehicle was created without a steering system. Instead, each suspension is modeled with a dummy part in the upper kingpin ball joint. This part allows replacing the ball joint between the wheel end and the upper control arm with two revolute joints, one for each degree of freedom which is wheel travel and steering the wheel about the kingpin axes. By turning the revolute joint about the kingpin axes, the tire can be steered. This modeling technique allows each wheel to be steered individually without a steering system. Details of the suspension system are discussed in later chapters.

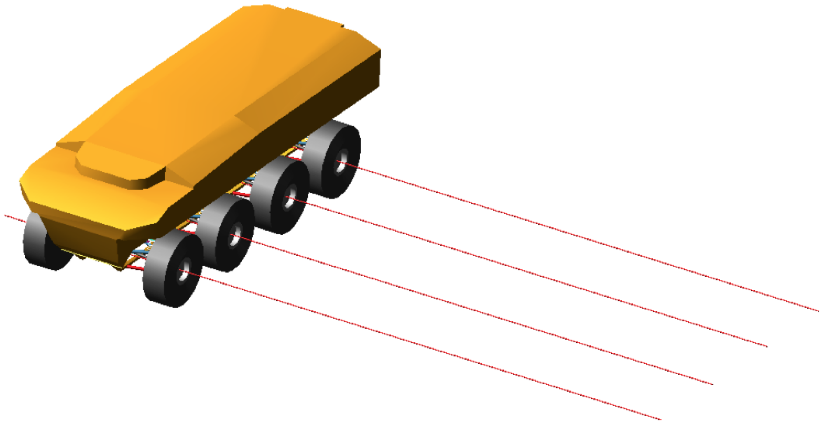


Figure 1.4: Multi-body dynamics vehicle model in Adams Car

Adams software has Python API Support. This feature allows the user to execute Python files in Adams software to create and modify design objects in Adams. Using this feature, together with the Adams command language, simulations can be executed in Adams. In this work wheel steering angles are changed using Python code, and Adams command language is used to run and gather the results of multi-body dynamics vehicle simulations. These simulations results are then used in Python for the optimization process.

The optimization problem in the first part of this work is a multi-variable and constrained optimization problem since there are eight wheel steering angles to be optimized, and maximum wheel steering angles are constrained. The objective of the optimization is to minimize turning radius and keep the tire lateral slip angles of the tires to a minimum. The cost function used in the optimization consists of the vehicle turning radius and tire lateral slip angles. In Table 1.1, the number of possible solutions for the problem is given. In this table, front four tires are assumed to be steered in the direction of the turn, and the rear tires are steered in opposite directions (which is often the case for low-speed maneuvers for AWS vehicles), and the possible options are calculated with a 0.1 degree increment. Each tire is assumed to be capable of being steered inside the constrained range.

Table 1.1: Number of possible solutions for the turning radius optimization study.

	Lower Limit (deg)	Upper Limit (deg)	Number of Possibilities
δ_{i1}	-30	0	300
δ_{i2}	-30	0	300
δ_{i3}	0	30	300
δ_{i4}	0	30	300
δ_{o1}	-30	0	300
δ_{o2}	-30	0	300
δ_{o3}	0	30	300
δ_{o4}	0	30	300

Total Possible Options	$6.561 * 10^{19}$

Where δ_{in} is the inner wheel steering angle of the n^{th} axle, and δ_{on} is the outer wheel steering angle of the n^{th} axle. As seen in Table 1.1, the number of possible solutions with 0.1-degree increment results in a very high number of possible solutions. With such a high number of possible solutions, brute force optimization methods would be highly inefficient. Minimizing only the turning radius even with this many possible solutions is not such a hard problem. Steering all of the tires to maximum possible steering angle should give the lowest turning radius. However, lateral slip angles of the tires are also considered and selecting maximum possible steering angles for all of the tires would result in unacceptable tire lateral slip angles and cause rapid tire wear. Introducing tire lateral slip angles to the cost function complicates the function resulting in local optima to appear in the eight-dimensional solution space. Local optima cause difficulties for optimization algorithms, as they can get stuck in one of the local optima. Due to their abilities to search the search space and not to get stuck at the local optima, genetic algorithms are chosen for the optimization problem.

Genetic algorithms are powerful optimization tools that are inspired by nature. The genetic algorithm begins the optimization process by creating an initial population of candidate solutions. After the initial population of candidate solutions is evaluated, some of them are selected to move up to the next generation. These selected solutions are paired with each other to create offsprings for the next generation. Before the next generation is complete random mutations occur in the candidate solutions. The selection process allows fitter solutions to move up. Pairing solutions with higher fitness values with each other have a chance to produce even better offsprings. Moreover, random mutations allow the algorithm to continuously search the solution space so the algorithm would not get stuck on local optima.

Vehicles that have AWS usually change the steering ratio of the rear tires as a function of velocity or yaw rate, because steering the rear tires in the opposite direction of the turn at high velocities will cause the vehicle to become unstable. To overcome this problem, some AWS controllers steer the rear tires in the same direction as the front tires with increasing speed. This strategy reduces the yaw gain of the vehicle and increases stability. However, in this study, the AWS controller of the vehicle investigated does not steer the rear tires after a specific speed limit. So the handling behavior of the vehicle with the optimized steering angles for the high-velocity region must be investigated. For this purpose, a DLC maneuver is considered. The DLC maneuver chosen for this study is called the Nato Lane Change [2], and it is a maneuver made explicitly for military vehicles. In this maneuver, the dimensions of the road are determined by the dimensions of the vehicle, so large vehicles like 8x8 vehicles do not have a disadvantage. During this maneuver, the vehicle has a constant longitudinal velocity. Vehicle path during a DLC maneuver can be seen in Figure 1.5.

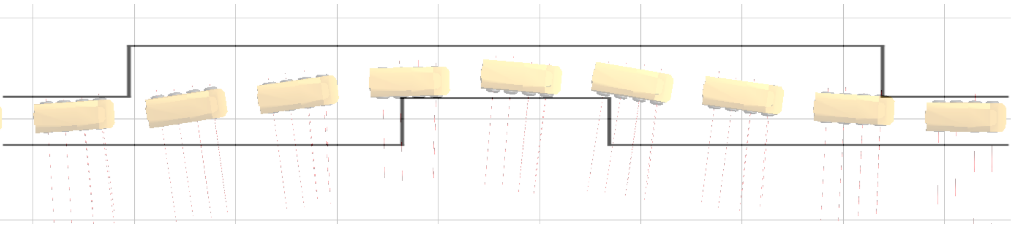


Figure 1.5: Vehicle path during Nato lane change maneuver.

As mentioned, the multi-body dynamics vehicle model has no steering system, and the tires are steered individually. This method provides flexibility for the steady-state turning maneuvers, but due to the lack of a steering system, Adams Car vehicle steering controllers do not work. To steer the vehicle, a mathematical function that represents the steering input during a DLC maneuver was created. Each of the front tires is steered using this mathematical function, while the steering angles of the rear tires are kept at zero. The parameters of the steering input function are optimized to control the steering of the vehicle through the DLC maneuver. To find the optimal parameters for the steering input function, a genetic algorithm was used. Using this method, the DLC maneuver can be performed for a vehicle that does not have a steering system. The steering input acts as an open loop controller for the vehicle. By using the two methods together, optimal wheel steering angles for both low-speed and high-speed maneuvers can be obtained.

1.2 Aim And Scope Of The thesis

Aim of this work is to develop a method for finding the optimal wheel steering angles that result in minimum turning radius and minimum tire lateral slip angles. To calculate the turning radius and lateral tire slip angles, a multi-body dynamics vehicle model was used. The multi-body dynamics model of the vehicle was created in Adams Car software. Turning radius calculations are conducted by simulating a steady state, constant velocity maneuver at a low longitudinal velocity where the steering input is applied to the wheels. The vehicle used in the optimization process is an 8x8 vehicle with AWS, so there are eight parameters to be optimized, which are the maximum steering angles of the tires. The optimization process is done with a genetic algorithm written in Python language.

After the maximum steering angles of the tires are optimized for low-speed maneuvers, another method was created to investigate the high speed handling characteristics of the vehicle coupled with the optimized wheel steering angles. In this study, Nato lane change maneuver was chosen as the maneuver in which the high-speed handling behavior of the vehicle will be investigated. The Nato Lane Change maneuver is a high-speed DLC maneuver in which the road boundaries are determined according to the vehicle dimensions. The maximum steering angles that were optimized to give low turning radius and low lateral slip angles are used to create restrictions representing a steering system in this maneuver. The steering angle for the front inner tire was obtained by the general open loop DLC input function. The general open loop DLC steering input function is a trigonometric function that yields the steering input necessary for a DLC maneuver. The general DLC steering input function was derived for this specific problem and will be discussed in the Nato Lane Change Simulations chapter. The other three wheel steering angles of the front two axles were obtained by assuming perfect Ackermann steering with no steering error. The rear two axles are not steered during the DLC maneuver. The parameters of the trigonometric function used for creating steering input for the inner front tire is optimized using a genetic algorithm. The genetic algorithm code that is used to optimize the turning radius of the vehicle is revised to optimize the parameters of steering inputs.

By using these two methods together, in the early stages of vehicle design optimal steering angle that causes low turning radius and low lateral slip, which is also capable of completing Nato Lane Change maneuver can be found. In the early stages of vehicle development, DLC maneuvers can be conducted to investigate the high-speed handling behavior of the vehicle with different steering angle configurations, before the design of the steering system. After the optimal wheel steering angles are found, these maximum wheel steering angles can be used as design criteria for the steering system mechanism design.

1.3 Thesis Outline

This chapter presents the structure, aim and scope of the thesis. The chapter is concluded with the thesis outline.

The second chapter is the literature survey. In this section, previous research done on the related subjects to this work are given. The chapter includes research on vehicle dynamics, multi-axle vehicle dynamics, optimization, and genetic algorithms.

The third chapter states the theoretical background of the thesis. In this section, the theories of the concepts used in the thesis are discussed. The section starts with simple bicycle models and multi-axle bicycle model. Then moves on to the multi-body dynamics vehicle modeling in Adams Car. Adams Python API and its capabilities are discussed in this section as well. The chapter is concluded with genetic algorithm theory.

In the fourth chapter, turning radius optimization is presented. The Python code and Adams command language code that are used to run the optimization process are presented. The genetic algorithm used for the turning radius optimization is reviewed in this section as well. The cost function used for the optimization process is given, and the section is concluded with the presentation of simulation results.

The fifth chapter includes the Nato Lane Change chapter. The steering input function derived for DLC maneuvers is presented in this section. The genetic algorithm used for the optimization of the steering input is discussed. The cost function created for the DLC maneuver is given, and the section is concluded with the results of the DLC maneuver.

The last chapter includes discussion and conclusion. In this section results of the turning radius optimization and the DLC maneuver is presented. Possible future work is also discussed.

2 LITERATURE SURVEY

2.1 Introduction

This chapter presents pioneering work in the fields of vehicle dynamics, multi-axle vehicle dynamics, optimization, and genetic algorithms. By doing so, the thesis aims to construct a proper background to analyze and observe its case study in the intersecting viewpoint of these fields. This chapter is divided into four sections to display the significance of these works for this thesis better. The sections discussed in this chapter are;

- Vehicle dynamics section
- Multi-axle vehicle dynamics section
- Optimization section
- Genetic algorithm section

The first section focuses on the field of vehicle dynamics. In this section, some of the significant research on vehicle dynamics, which is essential for this thesis is presented. This section mainly focuses on the handling aspect of vehicles with four wheels. The second section is the multi-axle vehicle dynamics section. In this section, research that investigates the multi-axle vehicle handling behavior is given. After the multi-axle vehicle dynamics section, the optimization section is presented. A brief introduction to optimization is given before significant research on vehicle dynamics optimization. After this section, literature about the genetic algorithms is given. The research about genetic algorithm usage about the on the vehicle dynamics field is presented in the genetic algorithm section.

2.2 Vehicle Dynamics

Vehicle dynamics is a vast subject with many subsections. However, the main three parts can be distinguished as ride, performance, and handling. All three parts of vehicle dynamics are vast subjects with many research potential. There are many books and articles written on vehicle dynamics; in this chapter, some of the literature is given. Very intuitive books covering all parts of vehicle dynamics were written by Gillespie [3] and Jazar [4]. In 2002, Wong [5] published a vehicle dynamics book with an emphasis on terra-mechanics. Pacejka [6] published a vehicle dynamics book with a focus on the tires. Harrer and Pfeffer [1] published a book on vehicle steering.

In 2017, Marzbani et. al. [7] investigated the effects of the variable longitudinal velocity of the vehicle on the transient and steady-state turning. To be able to compare the transient and steady-state handling characteristics of a vehicle, two maneuvers were simulated using the same vehicle model. In the first maneuver, the vehicle is moving with a constant longitudinal velocity and the steering input is given to the system as a step steer angle change. The second maneuver is a lane change maneuver. In this work mathematical formulation for finding the dynamic center of rotation is given.

In their article, Menhour et al. [8] proposed three different vehicle steering control strategies. Two vehicle models were used, the first model is a vehicle handling model with two degrees of freedom, which are the lateral and yaw movements, the second model is a non-linear four-wheel vehicle model with four degrees of freedom, this model is used to test the control methods proposed. The first control method proposed is an indirect method with multiple PID controller controlling two-degree-of-freedom. The second control method proposed is a direct method with robust switching two-PID controllers with LMI–LQR. Also, the last control method proposed is sliding-mode control (SMC). Robustness of the three proposed controllers was validated by experiment.

All-wheel steering is a commonly used method to enhance the handling capabilities of vehicles. Although front wheel steering (FWS) has been used almost exclusively in vehicles with two axles all-wheel steering has been worked on for a long time, and there is much research done on the subject. Furukawa et al. [9] have summarized all-wheel steering from vehicle dynamics and control perspective. In this comprehensive work effects of all-wheel steering is investigated. The article starts with a historical background and moves on to vehicle dynamics aspects of all-wheel steering. Mathematical equations that are used to examine all-wheel steering, as well as the block diagrams, are given. Control strategies and main research areas are discussed as well. Authors also give examples of the steering strategies used by different car manufacturers.

Sano et al. [10] proposed a control strategy for all-wheel steering systems. In this work, rear tires are steered as a function of the steering angle of the front tires. The rear tires are steered in the same direction as the front tires when the steering input is small; this allows the vehicle to have better handling characteristics at high velocities. At low-velocity maneuvers like parking where the steering angle input is high, the rear tires are steered in the opposite direction of the front tires, and this strategy allows the vehicle to be more maneuverable at low-velocity maneuvers. The vehicle stability when using this all-wheel steering strategy was investigated both mathematical formulations and proving ground tests. Comparison between the existing steering systems and the proposed system is also given. The article then goes into detail about how to find the proposed functions depending on the vehicle characteristics. The main findings of this research can be summarized as;

- Steer angle function systems can achieve the same effect as the vehicle speed function based systems.
- Three crucial aspects of the steer angle function systems are steering angle ratio at small steering angle range, maximum steering angle at the opposite direction at high steering angle range and changes in the difference between the steering angles of the front and rear tires.
- Steer angle function systems can be realized by mechanical linkages and a mechanism that inverts the input as necessary.
- Experiment results show all-wheel steered vehicle using this strategy may be better than front wheel steered the vehicle in several aspects.

Takiguchi et al. [11] proposed a control system to control the steering ratio in an all-wheel steering system optimally. The main objectives of the proposed system are to have high maneuverability at low speeds, help the steering effort at moderate speeds and to reach high stability at high-speed maneuvers. To observe the effect of the proposed control strategy on the system, the frequency response of the vehicle is investigated. To be able to observe the performance of the system, the proposed controller was implemented in 1989 Mazda 625 and vehicle tests were conducted. High steering angle low-speed tests like parking and u-turning were conducted as well as high-speed slalom tests. From the results of these test, the author concluded that handling characteristics can be improved by matching the yaw rate and lateral acceleration phase lag, to able to utilize all-wheel steering properly the steering ratio between the front and the rear must be varied concerning vehicle speed. The tests conducted using the prototype vehicle showed improved handling and stability at high speeds and improved parking performance at low-speed maneuvers.

Fukada [12] proposed a slip angle estimation method to be used in vehicle stability control systems. There are two main methods for slip estimation. The first method is direct integration, this method does is not affected by the tire parameters and road friction, but it is open to sensing errors. The second method is estimating the slip angle using a vehicle model; this method is beneficial for the linear region but lacks the accuracy for non-linear regions. The method proposed in this work is a combination of the two methods with a yaw-velocity feedback algorithm. Experiment results show that the proposed method can estimate the slip angle in DLC maneuver accurately and in J-turn maneuver it predicts a smaller value.

Peng and Yang [13] provided a comparison of five different DLC maneuvers. ISO 3888-1 and ISO 3888-2 standards, as well as the Nato Lane Change maneuver [2], were among the compared DLC maneuvers. In this work, the maximum forward speed of the vehicle during a DLC maneuver is investigated. The vehicle must not cross the DLC maneuver road limits and tires must not lose contact with the road. It was concluded that vehicle length has more impact on DLC performance than vehicle width.

Kutluay and Winner [14] investigated the assessment of vehicle simulations with experimental data, mainly DLC maneuvers. Proposed methods can be used to achieve higher model confidence and simulation efficiency. The steering input for the DLC maneuver consists of two parts, the first part is responsible for vehicle to leave its original lane and arrive on the second lane and the second part of the input is responsible for leaving the second line and arriving on the original line again. If the vehicle velocity is held constant during the maneuver, the two parts of the steering input mirror each other very similarly. This information about the DLC steering input is used while creating the general open loop steering input function.

As mentioned before in this chapter, tires are one of the most challenging parts of the vehicle to model. Due to the elasticity of the many components and the air pressure inside the tire, a basic mathematical equation could not describe the behavior of the tire. Finite element models of the tires are studied, but due to the computational resource needed to run these models, they are not well suited to be used in vehicle dynamics simulations. Pacejka and Bakker [15] proposed a mathematical model to calculate the forces and the moments acting on the tire. The magic formula tire model calculates the lateral and longitudinal forces and overturning moments using the same trigonometric function with different coefficients in each case. These coefficients represent tire characteristics, and they can be found by conducting tests on the tires. Since its introduction in 1989, the magic formula tire model has been the industry standard for vehicle dynamics simulations. In this work, an improved version of the magic formula tire model that has been developed by MSC Software called PAC2002 is used.

Tremlett and Limebeer [16] proposed a methodology to optimize vehicle parameters to reduce tire wear. In this research about thermodynamical and wear tire models were created to investigate tire wear for Formula 1 vehicles. Although Formula 1 is an extreme case, tire wear mechanisms are still the same. There are several factors that effects tire wear like adhesion between the tire and the road surface and the road. However, in this thesis, the main criteria for tire wear is taken as the lateral slip as other parameters affecting tire wear are not in this works scope.

2.3 Multi-Axle Vehicle Dynamics

To be able to carry large payloads, high movement capabilities in off-road and stability while firing, multi-axle vehicles are commonly used in the defense industry. Due to the multiple axles, these vehicles tend to be much larger than commercial and passenger vehicles. To be able to use these vehicles on existing roads, their handling capabilities becomes crucial. Handling behavior of multi-axle vehicles must be investigated early in the design process. To be able to investigate the handling behavior of multi-axle vehicles, there has been much research done. Williams [17] proposed a generalized model to investigate the handling behavior of multi-axle vehicles. Primary vehicle handling concepts are reviewed using a two-axle bicycle model with minor changes to the convention that is mainly used in similar research. After the two-axle bicycle model, a three-axle bicycle model was created to investigate the effect of multiple axles. After these models, a generalized bicycle model was created that can be applied to vehicles with an arbitrary number of steerable and non-steerable axles. Finally, mathematical equations that are used to investigate vehicle understeer and wheelbase was derived.

Winkler [18] investigated the handling of the behavior of complex vehicles. For a vehicle modeled with a simple vehicle model, the tire slip angles only depend on the lateral acceleration, but for complex vehicles like heavy vehicles with multiple axles, the tire slip angles also depend on another motion variable like velocity, path curvature or any other combination as well. A new concept for complex vehicles called equivalent wheelbase is presented. Equivalent wheelbase can be used to investigate the understeer and oversteer quality of all vehicles. The effects that make the vehicle model complex can be given as;

- Aerodynamics effects,
- Load transfer effects,
- Wet surfaces,
- Multiple non-steerable axles and dual tires,
- AWS.

The equivalent wheelbase for a simple model of a four-wheel vehicle is equal to the actual wheelbase. However, for a multi-axle vehicle with multiple non-steerable axles, the equivalent wheelbase is longer than the real wheelbase. Another conclusion of the paper was, the understeer influence of the rear axle is greater than the front axle.

Winkler and Gillespie [19] investigated directional response characteristics of multi-axle vehicles. Although the physical rules that apply to the passenger cars apply to the heavy trucks, the rules-of-thumb gathered by the automobile industry does not apply to the heavy trucks in many cases. To evaluate the steady-state turning performance of the heavy trucks that have multiple non-steering axles and dual tires, generalized equations were developed. From the equation derived, it was concluded that both dual tires and tandem axles affect the vehicle handling similarly. The understeer gradient that can be calculated for passenger cars yields an error when applied to the vehicles with tandem axles or double tires, and the errors are caused by the method that the Ackerman angle is calculated. In the Ackerman angle calculation for heavy vehicles, equivalent wheelbase must be used. The research showed that both tandem axles and double tires reduce the yaw rate and increase the critical or characteristics velocity.

Qu and Zu [20] proposed a method to optimize the design parameters of the steering system for a commercial three-axle vehicle. To enhance the handling of a three-axle vehicle rear tires can be steered as well as the front tires. The front and rear tire of the three-axle vehicle is steered with a controller which is trying to follow the path of the reference model. So the uncertain vehicle models and non-linear vehicle model can have the same steering characteristics as the nominal model on different roads. Some modifications were made to the preview follower driver model to improve compatibility with the 4WS. These driver models were used to optimize the steering system parameters. Uncertain vehicle model was taken as bases, and the model-following controller was constructed. Simulations and simulation results tested the proposed design method shows that the method is reasonable.

Bayar and Unlusoy [22] investigated the steering strategies used for multi-axle vehicles. In this work, steering strategies created for 4WS are applied to multi-axle vehicles. Multi-axle vehicle models were simulated using the steering strategies, and results were compared to observe which steering strategy is the best. The steering strategies used for 4WS in two-axle vehicles extended to be able to apply to multi-axle vehicles. For three and four-axle vehicles, low vehicle side slip angle, with a high yaw rate and lateral acceleration can be achieved by steering the intermediate axles, coupled with the application of yaw feedback. By steering the intermediate axles reasonably, the lateral acceleration reached by the FWS configuration can be reached without degrading the zero vehicle side slip angle.

Watanabe et al. [23] proposed a method to model multi-axle vehicles. This model takes steering angles and wheel velocity as inputs and turning radius and tire slip angles as outputs. The results of the mathematical model were compared with the test conducted with the full-scale multi-axle vehicle. Different turning and driving strategies of an 8x8 vehicle were investigated. From the investigation results, it was conducted that; steering of the rear wheels had a considerable effect on the turning behavior of the vehicle, but the effect of the location of the steering center did not affect the turning behavior significantly. Another conclusion was made from the investigation of different driving strategies, which were; rear wheel steering had a significant effect on the required power and the power loss of the vehicle was minimum with the rear four-wheel drive.

2.4 Optimization

Optimization is the process of finding the best solution to the problem at hand. Optimization can be found in every part of our lives. In their book "Numerical Optimization", Nocedal and Wrigh [24] explains the optimization as;

> "People optimize. Investors seek to create portfolios that avoid excessive risk while achieving a high rate of return. Manufacturers aim for maximum efficiency in the design and operation of their production processes. Engineers adjust parameters to optimize the performance of their designs.
>
> Nature optimizes. Physical systems tend to a state of minimum energy. The molecules in an isolated chemical system react with each other until the total potential energy of their electrons is minimized. Rays of light follow paths that minimize their travel time."

Nocedal and Wrigh [24] goes into detail about the different numerical optimization methods. In this book, methods used to solve constraint and unconstraint optimization problems as well as advanced methods like non-linear programming and sequential quadratic programming are discussed. Chapra [25] wrote a book on numerical methods in general, and there is a constructive section on optimization methods in this book.

Optimization is used in the automotive industry extensively to make the vehicles faster, lighter and more durable. Oh et. al. [26] investigated minimization of turning radius for a multi-axle crane. Steering wheel angle was optimized to achieve minimum turning radius using model predictive control (MPC). During the optimization process, multi-axle bicycle model and error dynamics models were used. Controller with an optimal objective function was designed to minimize the error for minimum turning radius. Simulations were conducted with various vehicle speeds. Simulation results showed that the vehicle did not affect the optimal steering angle as the vehicle speed increased turning radius increased.

Felzein and Cronin [27] studied optimizing the steering error for a MacPherson strut suspension. Non-linear kinematic MacPherson strut suspension model with rack and pinion steering system was developed. The model was used to simulate steady-state turning maneuvers. The parameters of steering systems and the suspension were optimized using this model and steady-state turning maneuver, which represents the turning radius and body roll angles typical of highway driving. From the simulation results represented, it can be seen that steering performance of a vehicle can be improved significantly by steering system and suspension optimization.

Angelis et al. [28] investigated the maximization of lane change entry speed. DLC maneuver according to the ISO3888 part-2 was simulated using a two-track vehicle model with non-linear tire and suspension behavior and Dynamic Stability and Traction Control (DSTC) system. Using a direct transcription method, the optimal control problem is converted into an optimization problem. The simulation results and the optimization process were validated with real vehicle tests.

Arvind [29] investigated optimizing the turning radius of a vehicle by four-wheel steering (4WS). To achieve 4WS, the vehicle investigated has mechanical linkages between the front and rear axles and rack and pinion steering systems at both front and rear axles. The mechanical systems were studied by conducting kinematic analysis for both the vehicle that employs the proposed symmetric four-wheel steering (4WS) system and for a vehicle that does not have 4WS. The results showed that the proposed 4WS system decreased the turning radius of the vehicle considerably.

Aydın and Unlusoy [30] investigated suspension parameter optimization to improve impact harshness (IH) for road vehicles. To be used for the optimization multi-body dynamics model of a small commercial vehicle was created in Adams. For the optimization process, the methodology was using the design of the experiment together with response surface. Screening experiments were conducted to identify the parameters that significantly affect the IH. From the simulation results, it can be seen that selected suspension parameters for optimization process are capable of improving the IH performance of the vehicle. The results of the optimization process also show that considerable improvement in IH performance of the vehicle is possible.

2.5 Genetic Algorithms

John Holland [31] first proposed genetic algorithms in1975. Since their introduction genetic algorithms gained popularity due to their power of solving multi-objective optimization problems which other optimization methods usually get stuck at the local minima and their implementation in machine learning. Randy and Sue Haupt [32] published a beneficial book on genetic algorithms. In this book, basic genetic algorithm principles are given as well as continuous genetic algorithms and advanced applications of genetic algorithms. Kramer [33] also published a book in which the general principles of genetic algorithms are discussed. In this book, genetic algorithm applications and use cases are given as well as genetic algorithm theory. All of these books provide excellent insight into the genetic algorithms. In the remaining part of this section, research on the genetic algorithm used in vehicle dynamics area is given.

Hui [34] proposed a new configuration of the hydraulic hybrid vehicle (HHV) which is optimized by an adaptive simulated annealing genetic algorithm (ASAGA) to achieve better fuel economy and driving performance. In the proposed configuration the internal combustion engine drives the hydraulic pumps which provide high-pressure oil flow for the hydraulic pump/motors. Using this method the engine is decoupled from the vehicle operation and can be operated in the high-efficiency region. Two pumps/motors are connected to the front and rear axles through the differentials. These pump/motors act like motors and drive the tires in normal operations and act as pumps and capture energy in braking slowing the vehicle. The vehicle using this configuration has 4WD. To optimize the parameters of the HHV, genetic algorithms were used. When using genetic algorithms cost of the solutions improves rapidly at the beginning of the optimization process, but further along the optimization process, further improvements become challenging to obtain. To overcome the shortcomings of the genetic algorithms, a mixed method called ASAGA was used. This algorithm uses adaptive genetic operations to improve the improvement of the cost of solutions. Various optimization processes were conducted using different weighting factors in the objective function to achieve different objectives. The results of the optimization process showed that the crucial components of HHV could be optimized for different objectives.

Schuller et al. [36] investigated genetic algorithms to be used for multi-criteria, multi-scenario optimization of vehicle handling behavior. Due to the developments in computer technology optimizing vehicles in the design phase using simulation is possible. A two-track model of a BMW vehicle is created and integrated with an optimization algorithm to optimize dynamic performance. The dynamic performance of the vehicle is tested in three different maneuvers. The first maneuver is a steady-state turning maneuver which is used to evaluate the vehicles cornering abilities. The second maneuver is a J-turn maneuver which is used for investigating transient vehicle behavior. Finally, the third maneuver is riding in a rough road, which is used for investigating the comfort and line-holding performance of the vehicle. Design spaces for all of the design parameters were defined to reduce the search space and to implement physical limits. The genetic algorithm used in this study is a binary genetic algorithm, meaning all of the parameters are transformed into bites. The genetic algorithm used in this study turned out to be very efficient for multi-objective optimization with multiple scenarios. From the results of the optimization process, significant improvements in vehicle handling behavior were achieved when compared to the reference vehicle.

Huang and Ren [35] proposed a method for automated vehicle control. In this method, several controllers are tasked with guidance and a fuzzy neural network (NFN) is used to build the controllers. A hybrid two-phased learning algorithm was used to determine the weightings of the NFN. The study aims to improve the performance of vehicle control and at the same time, improve the learning convergence. The proposed guidance system takes control of the vehicle after the driver gets on to the highway and inputs the optimal vehicle speed. There are three main objectives of the proposed guidance system. The first objective of the guidance system is to follow the vehicle that is in front of the vehicle in the same lane at a safe distance. The second task of the proposed guidance system is to track an optimal vehicle as much as possible without violating any safety criterion. The third task of the proposed vehicle guidance system is to execute particular tasks such as lane changing. Computational simulations showed the effectiveness of the proposed controllers. The results of the study showed that NFN construction could be determined by a mix of genetic and gradient parameter algorithms.

Datoussaïd et al. [38] proposed an optimal control method for multi-body systems which contains closed loops. The multi-body systems that are being considered are also submitted to kinematic or dynamic time-dependent criteria. The desired kinematic or dynamic performance determines the design parameters that are being optimized. In this work, a stochastic swarm method based on genetic algorithms was chosen as the optimization method. Using this optimization method, a general method for optimizing the dynamic behavior of multi-body systems was proposed. For the kinematic optimization case, a double wishbone suspension was investigated. In this kinematic optimization, the aim is to keep the steering angle change due to vertical motion of the suspension to the minimum. For this optimization problem, six design parameters were chosen, which represents the lengths of the arms. For the dynamic optimization example, suspension of an urban railway vehicle was taken into consideration. In this dynamic optimization study, the suspension parameters were optimized to minimize the lateral acceleration when crossing tight curves to improve comfort. The study combines the computer-aided dynamic analysis and genetic algorithm optimization method. This method allows the optimization process that does not require the value or the assumption of derivatives.

2.6 Conclusion

In this section, research conducted on vehicle dynamics, multi-axle vehicle dynamics, optimization, and genetic algorithms are given. The subject of this work involves these fields, and these research will be used as a reference. This study aims to find the optimal tire maximum wheel steering angles that result in minimum turning radius and minimum tire lateral slip angles. Research conducted on the vehicle handling behavior, specially multi-axle vehicle handling behavior was investigated. The steering angle optimization is a multi-variable constrained optimization problem. After investigating different optimization methods and their implementations in vehicle dynamics area, genetic algorithms were selected as the method for optimization. Genetic algorithm is a vast subject with many use cases. The primary use cases investigated in this work is the use of genetic algorithms in vehicle dynamics area.

3 THEORETICAL BACKGROUND

3.1 Introduction

This chapter presents the theoretical background of the thesis. The section begins with the vehicle dynamics background of the work and moves on to the software used and to the genetic algorithm used for the optimization process.

The first section introduces the bicycle model, which is used to investigate the handling behavior of vehicles. One of the most popular models used to investigate the handling behavior of the vehicles is the bicycle model. The popularity of this model comes from its simplicity. The number of parameters needed to use the bicycle model is low, so it is an ideal model to be used in the early stages of vehicles design while most of the vehicle parameters are still unknown. The basic principles of the bicycle model can be used to create one tracked multi-axle model. After the equations of the one-track vehicle models are given, capabilities and the limitations of these models are discussed.

In this work, the multi-body dynamics vehicle model is created in the Adams Car software. The basic modeling and simulation principles of Adams Car is presented. The subsystems used in this work and their roles are explained as well. As mentioned before, tires are one of the hardest parts of the vehicle to model. The tire models that are used in the literature are presented in this section. The use of Adams command language and Python programming language in Adams will be a subject as well.

Finally, in this section, the basic principles of genetic algorithms are given. In both turning radius optimization and Nato Lane Change maneuvers, genetic algorithms were used to find the optimal solution. In both cases, the genetic algorithm that is suited to the problem was created. In this section, mathematical formulations that are used for both genetic algorithm are given. Moreover, different genetic operations and different methods used in these genetic operations are discussed as well.

3.2 Bicycle Model

One track models with varying degrees of complexities are widely used for investigating vehicle handling behavior. These models are beneficial for getting answers with minimum information about the real system. Due to linearity and ease of computation, they are suited for controller and observer design as well. Figure 3.1 shows a two-axle vehicle model.

Although there are many positive aspects to these one-track models, they have some drawbacks as well. While creating these models, many assumptions are made. The most obvious assumption is there is only one equivalent tire in each axle instead of two, this assumption allows this models to be used without suspension systems, but as a result, these models ignore load transfer that occurs during cornering. Due to not being able to compute load transfer bicycle model does not give accurate results for high lateral accelerations. Small angle approximations and modeling tire behavior with a single parameter are other significant simplifications of this model.

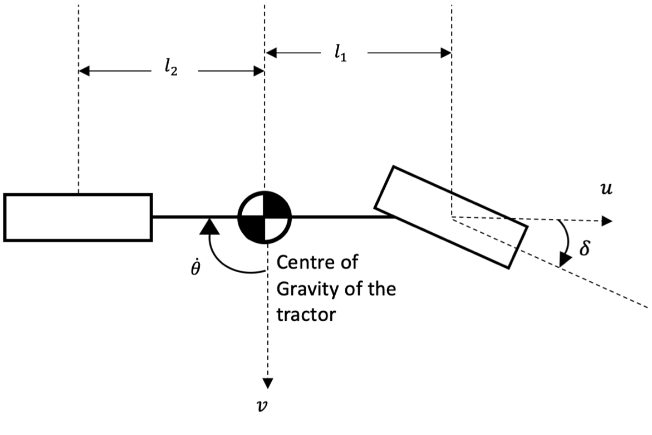


Figure 3.1: Bicycle model.

Equations for the bicycle model can be written as;

$$m(u + v\dot{\theta}) = F_{yr} + F_{yf} \tag{3.1}$$

$$I\ddot{\theta} = l_r F_{yr} + l_f F_{yf} \tag{3.2}$$

Where F_{yr} is the lateral force of the rear axle and F_{yf} is the lateral force of the front axle, and l_i is the distance from CoG to the corresponding axle, I is the momentum of inertia, u is the longitudinal velocity of the vehicle, v is lateral velocity of the vehicle, and m is the mass of the vehicle. Lateral forces can be written as;

$$F_{yi} = c_i \alpha_i \tag{3.3}$$

Where F_{yi} is cornering force of the corresponding axle, c_i is the cornering, and α is the lateral slip of the corresponding axle. Thus, the lateral slip angle can be written as;

$$\alpha_i = \frac{v - l_i\dot{\theta}}{u} - \delta_i \tag{3.4}$$

Where δ_i is the steering angle of the corresponding axle.

The cornering stiffness (c_i) calculation is straightforward but not an easy process. The cornering stiffness (c_i) is the tangent of the slope of the low lateral slip angles in the lateral force vs. lateral slip plot, where the behavior can be considered linear. Figure 3.2 shows the difference between the actual tire lateral behavior and linear cornering stiffness assumption comparison.

As seen from Figure 3.2, cornering stiffness approach is valid for low slip angles, but in the high slip angle region, the behavior becomes non-linear, and cornering stiffness approach does not give accurate answers.

Lateral force vs. laterals slips angle of the tire can be obtained by testing the tire or using simulations of a validated tire model. Lateral force vs. lateral slip behavior of the tire is not

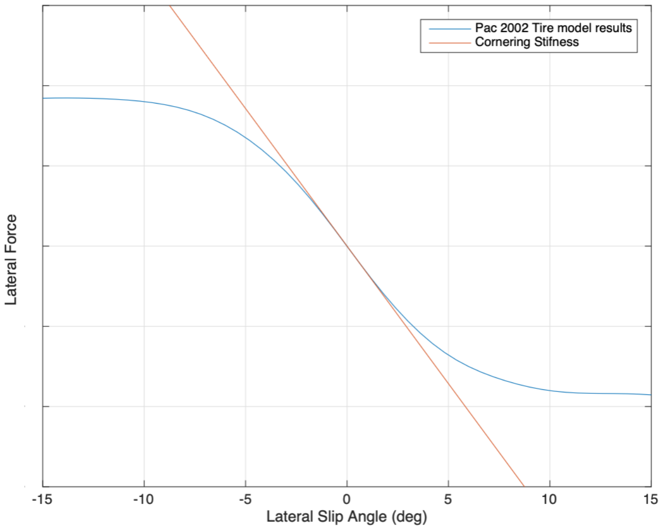


Figure 3.2: Lateral force vs lateral slip.

a static behavior and is affected by a lot of parameters like tire pressure, normal load on the tire, and temperature. So cornering stiffness changes depending on many parameters. The surface of lateral force vs. lateral slip angle obtained by simulating a tire model under different conditions is given in Figure 3.3.

So to summarize this model, the bicycle model is handy for investigating the vehicle behavior early in the design process because it does not require many parameters. The model also gives acceptable results for the linear range. However, for high lateral slip angles, the model overestimates the tire lateral force and diverges from the actual tire behavior. In the high lateral acceleration, region bicycle model does not give acceptable results either because it does not take the load transfer into account. Due to load transfer, the outer tire in each axle will have higher normal forces that the inner tires and usually the outer wheel steering angles are lower than the inner wheel steering angles to the overall lateral forces would be lower than bicycle model predicts. The bicycle model can be further improved by introducing non-linear behaviors for the systems and getting rid of small angle approximation, but the complexity of the model and computational power demand to run the model will increase. Improving the bicycle model would also mean that the parameters needed for the model would increase.

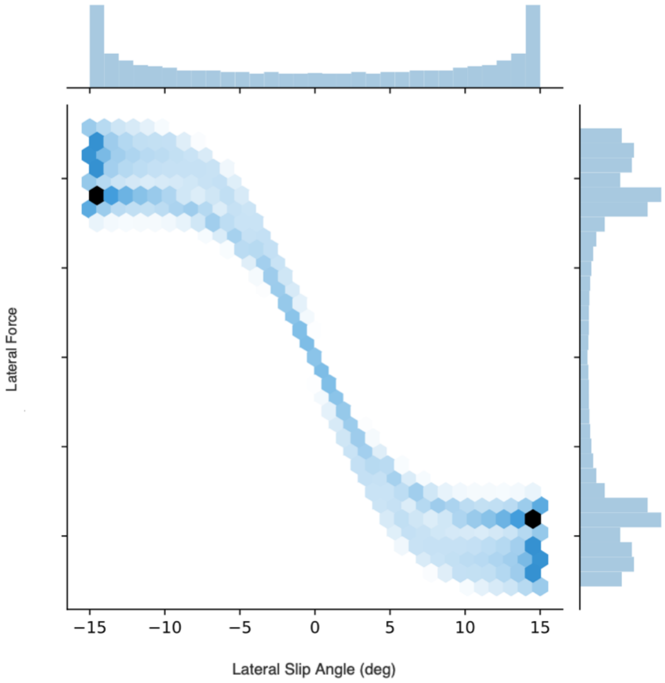


Figure 3.3: Lateral force vs lateral slip.

3.3 Multi-Axle Bicycle Model

Conventional bicycle model can be modified to have more than two axles. Also, the steering of auxiliary axles can be introduced to this model to achieve a multi-axle vehicle model with AWS. In this section, the multi-axle vehicle-bicycle model formulations for an 8x8 vehicle with AWS is given. Figure 3.4 shows the multi-axle vehicle model with AWS.

The multi-axle bicycle model equations for a vehicle with four axles can be written as;

$$m(u + v\dot{\theta}) = F_{y1} + F_{y2} + F_{y3} + F_{y4} \tag{3.5}$$

$$I\ddot{\theta} = l_1 F_{y1} + l_2 F_{y2} + l_3 F_{y3} + l_4 F_{y4} \tag{3.6}$$

In the equations given above tire lateral force (F_y) is calculated by multiplying tire lateral slip angle with the cornering stiffness.

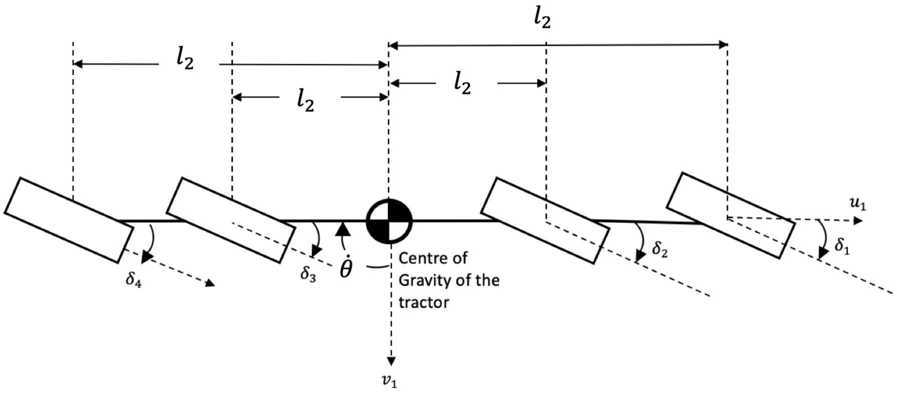


Figure 3.4: Four-axle bicycle model.

$$F_{yi} = c_i \alpha_i \tag{3.7}$$

Also, α_i is the tire lateral slip angle, which can be written for all of the axles as;

$$\alpha_i = \frac{v - l_i \dot{\theta}}{u} - \delta_i \tag{3.8}$$

The equations for the multi-axle bicycle model are almost identical to the two-axle bicycle model with only added terms for the additional axles. Cornering force and lateral slip angle equations are the same for all of the tires so the additional axle can be introduced by just adding the necessary terms to the equation. The slip angle calculation equation is the same for all of the tires. The difference between the axle located in front of the CoG and the tires located after the CoG comes from the distance term (l_i). The distance term (l_i) is positive if the axle is located before the CoG and negative if the axle is located after the CoG. The steering angle input (δ_i) notation is the same for all of the axles as well, so the vehicle can be steered by any combination of steering angles. For the non-steering axles, the steering input can be taken as zero.

The same assumptions made while creating the bicycle model were also made while creating the multi-axle bicycle model. Thus the multi-axle bicycle model has the same drawbacks as the bicycle model.

For the simulations, the multi-axle bicycle model was created in Matlab/Simulink environment. The Simulink model figure is given in Appendix A. This model was used to simulate the Nato Lane Change maneuver. The path of the vehicle is given in Figure 3.5.

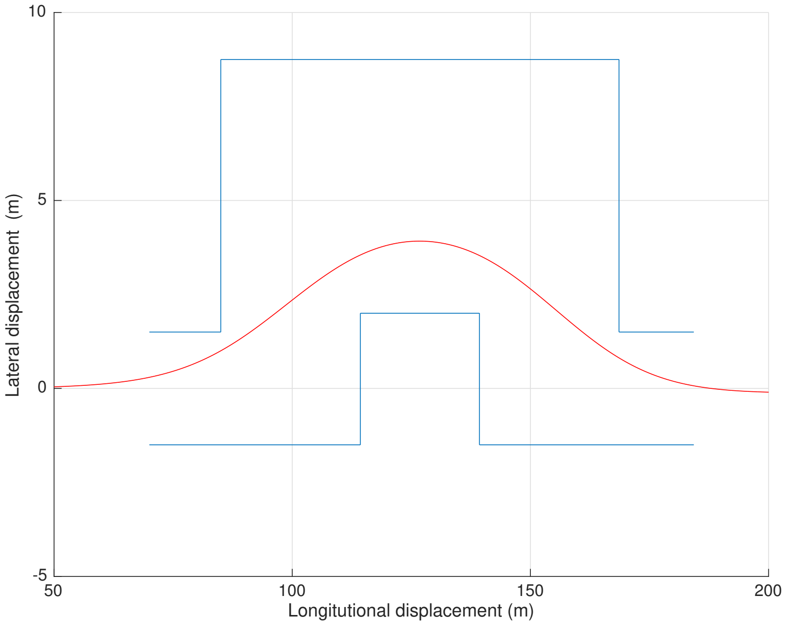


Figure 3.5: Vehicle path during DLC maneuver.

The steering input given to the model was created using trigonometric functions, and the equation was explicitly created for DLC maneuvers. The steering input function created for the DLC maneuver will be discussed in greater detail in the upcoming chapters. Figure 3.6 shows the steering input used in the simulation.

The bicycle model was created using linear cornering stiffness assumption. This assumption gives acceptable results for low lateral slip angles where the behavior can be considered linear, but for high slip angle region, this assumption overestimates the lateral force of the tire. During the DLC maneuver, the lateral slip angle of the front tire can be seen in Figure 3.7.

As seen in Figure 3.7, the tire lateral slip angle exceeds 3 degrees and as seen from Figure

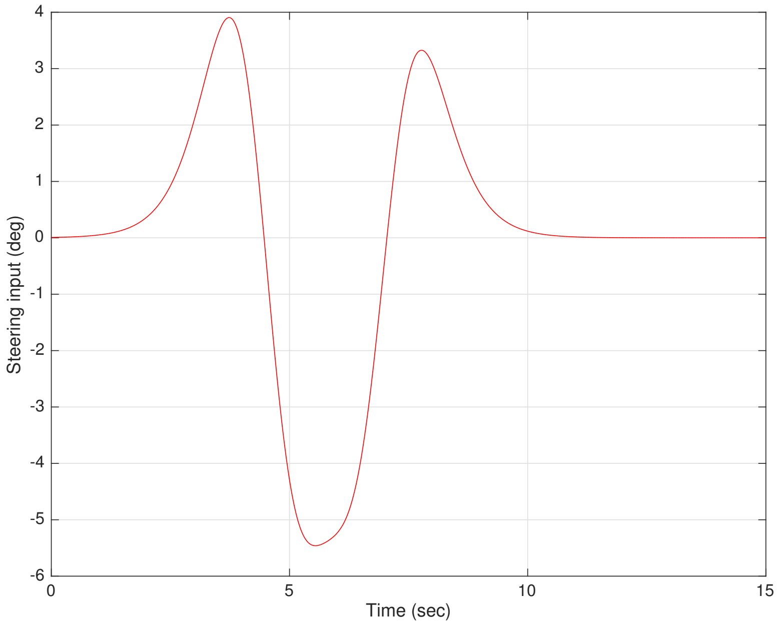


Figure 3.6: Steering angle input during the DLC maneuver.

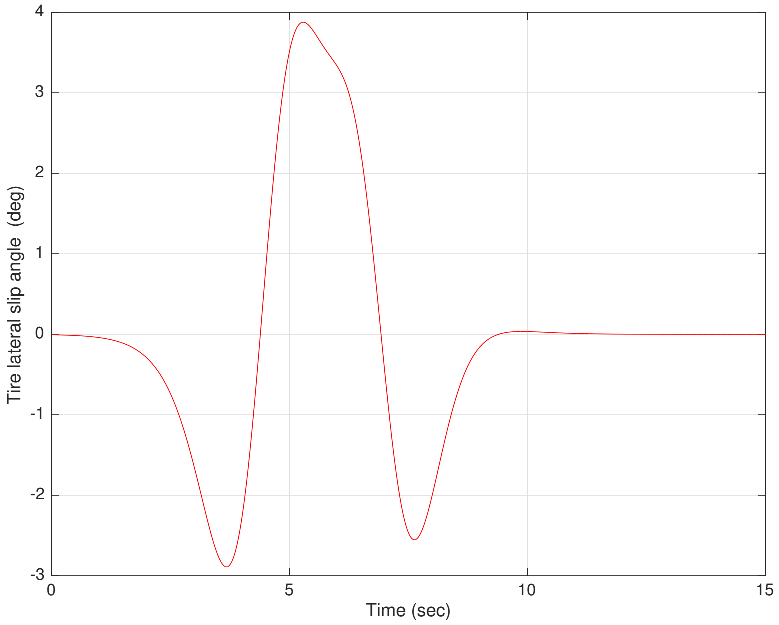


Figure 3.7: Lateral slip angle during the DLC maneuver.

3.2, cornering stiffness assumption starts to overestimate. The lack of load transfer is another drawback of the bicycle model. For maneuver in which the lateral accelerations are low, this is not a big problem, but for maneuvers with high lateral acceleration, the lack of load transfer affects the results significantly. The lateral acceleration of the vehicle during the maneuver is given in Figure 3.8.

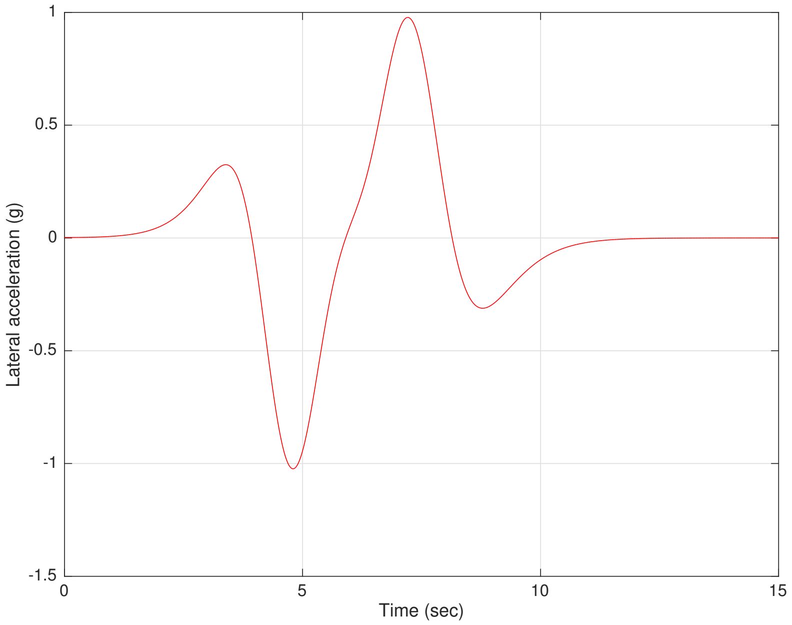


Figure 3.8: Lateral acceleration during the DLC maneuver.

From the simulation results, it can be seen that both lateral slip angle and lateral acceleration is high and can not be accurately modeled using a linear multi-axle bicycle model. Due to the drawbacks of the bicycle model, in this work, a multi-body dynamics model of the vehicle was used.

3.4 Multi-body Dynamics Vehicle Model In Adams Car

Due to the shortcomings of the bicycle model, which were discussed previously in this chapter, multi-body dynamics model of the 8x8 vehicle was created in Adams Car software. Adams is a multi-body dynamics simulation software created by MSC software. Adams Car is a specialized environment of Adams software which allows the creation of virtual vehicle prototypes. Using Adams Car vehicle tests can be performed in a virtual environment. In Adams Car, the simulated model is called an assembly; this assembly can be the full vehicle or part of the vehicle like suspension assemblies. The multi-axle vehicle model used in this study can be seen in Figure 3.9.

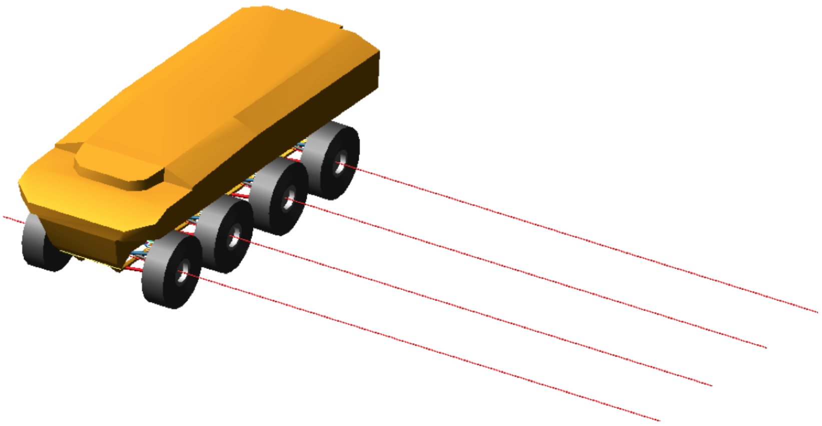


Figure 3.9: Multi-body dynamics vehicle model in Adams Car

Assemblies consist of subsystems. In this work, an 8x8 full vehicle assembly that consists of four suspension systems, four pairs of tires for each axle, a powertrain, brake system, and the body subsystems is used. There is no steering system in this model; instead, each tire is steered individually by the actuators in the suspension subsystems. This process allows the wheel steering angles to be modified quickly without the need for a steering mechanism.

Each subsystem is created using a template. Templates are building blocks of the subsystems. Templates hold essential information like the purpose of the subsystem, the number of parts and connection of the subsystem as well as the communicators which allow different subsystems to communicate with each other. The dimensions and basic behaviors like spring and damper characteristics can be modified directly from the Adams Car standard interface. However, to change significant aspects of the subsystem like adding a part or a joint, the template of the subsystems must be changed from the Adams Car template builder.

3.4.1 Body

Body subsystem acts as a base for other subsystems to attach to. This subsystem dominates the mass and the inertia of the vehicle as it is the heaviest subsystem. The markers which are used to calculate the turning radius and the ground clearance of the vehicle are located in this subsystem. Figure 3.10 shows the body subsystem used in this study.

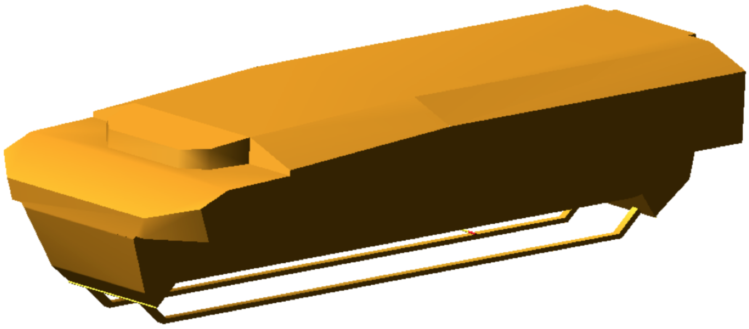


Figure 3.10: Body subsystem.

Turning radius of the center of gravity of the vehicle is calculated from the CoG marker's displacement. Also, curb to curb turning radius is calculated from the displacement of the marker located at the front right corner. There are four corner markers in each corner of the vehicle. These markers are used to determine the vehicle path and vehicle limits in the DLC maneuver.

3.4.2 Suspension

Suspension subsystem is the part of the model which connects the tires to the body, steering system, powertrain, and the braking system. This subsystem also provides wheel travel degree of freedom to the vehicle model. There are many suspension types used in the automotive industry. The structure of Adams with templates and subsystems allow quickly changing and testing different subsystems in vehicle assemblies. The suspension subsystem used in this model is double wishbone suspension. In Figure 3.11, a standard double wishbone suspension modeled in Adams Car can be seen.

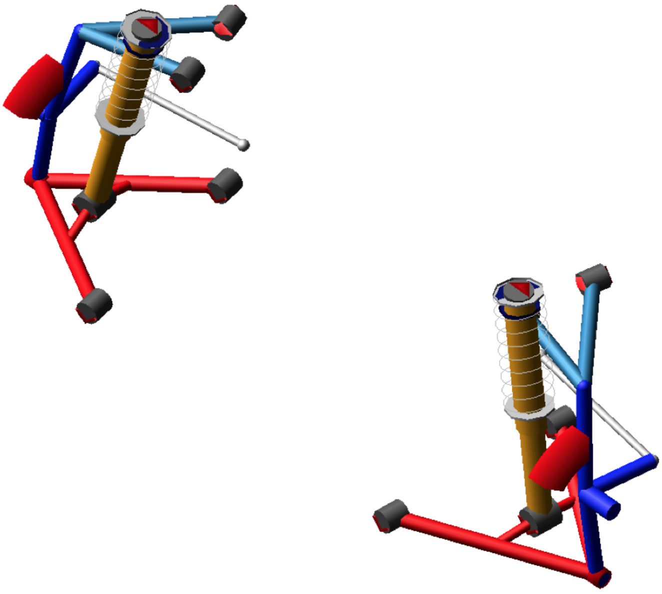


Figure 3.11: Double wishbone suspension subsystem in Adams Car.

The double wishbone suspension consists of two control arms, wheel end, driveshaft, and a tie rod. The tie rods are used to connect the wheel end to the steering system. In this model, tie rods are excluded from the model, and the suspension systems of each axle are modified so the steering input can be directly given to the kingpin axle. A dummy part is introduced between the upper control arm and the wheel end. The spherical joint that is used to connect the upper control arm to wheel end is replaced by two revolute joints between the upper control arm and the dummy part and wheel end and the dummy part. The revolute joint between the upper control arm and the dummy part allows the suspension to be moved upwards and downwards. The revolute joint between the dummy part and the wheel end allows the wheel end to be steered freely. Wheel steering is achieved by an actuator driving the revolute joint between the wheel end and the dummy part. The modified double wishbone suspension system can be seen in Figure 3.12.

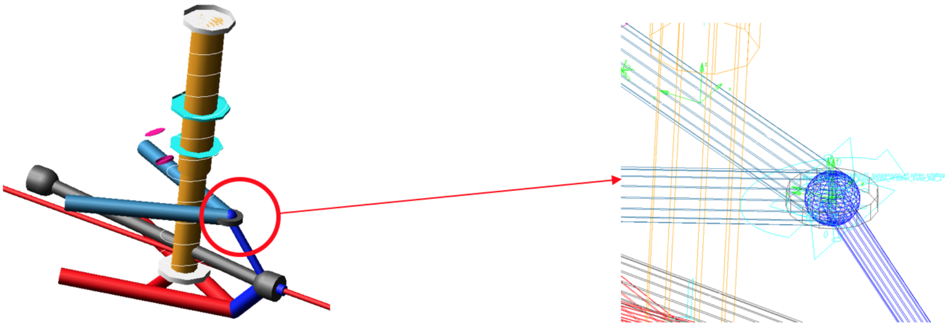


Figure 3.12: Modified double wishbone suspension system.

3.4.3 Powertrain System

The powertrain subsystem is connected to the body and the suspension subsystems. The drive torque that is used to propel the vehicle forward is created in this subsystem and delivered to the differentials that split the drive torque and delivers them to the drive shafts located in the suspension subsystem. The drive torque is then delivered to the tires by the drive shafts. Adams powertrain model used in this work can be seen in Figure 3.14.

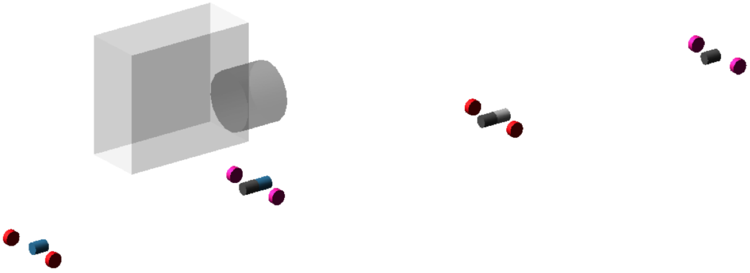


Figure 3.13: Powertrain subsystem.

3.4.4 Brake System

Braking subsystem is modeled as a torque acting on each tire. The model takes in the brake demand input and outputs the brake torque by multiplying the brake demand with maximum brake torque.

$$T_{Brake} = BD * T_{BrakeMax} \tag{3.9}$$

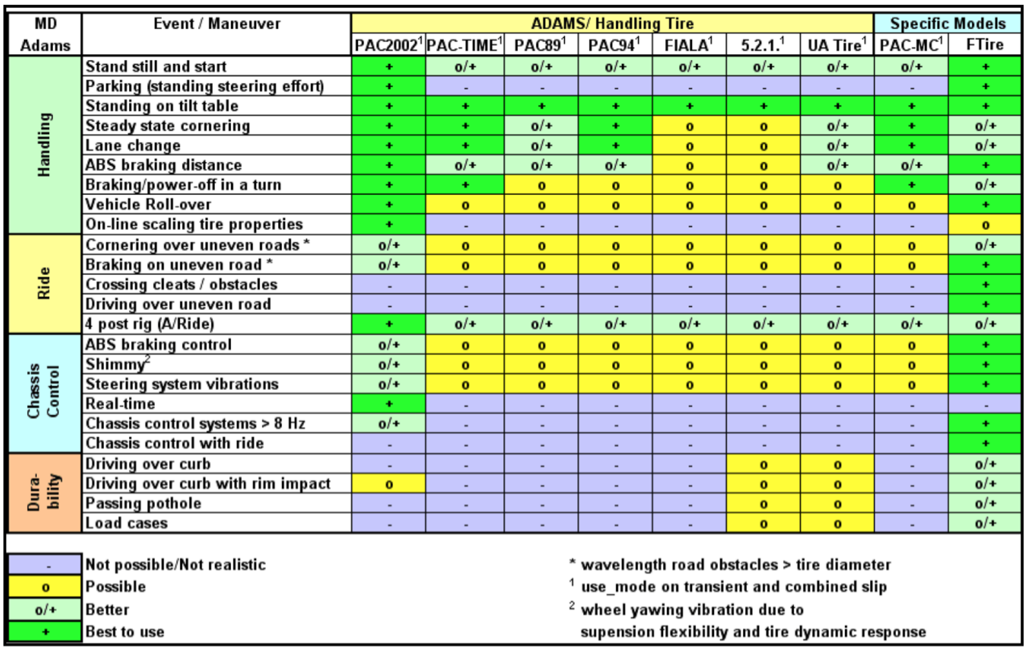
Where the T_{Brake} is the brake torque output of the braking subsystem, BD is the percentage of the brake force demanded this can be interpreted as the percentage brake pedal is pushed and $T_{BrakeMax}$ is the maximum brake torque that can be created by the braking subsystem. The velocity controllers in Adams Car controls the throttle demand and brake demand. In this work for both of the maneuvers, the vehicle velocity is constant.

3.4.5 Tire

Tires are one of the most challenging elements of the model. The behavior of rubber, the air pressure inside the tire, and the non-linear nature of the tire makes modeling the tire realistically, very difficult. There are many methods for modeling tires. Realistic finite element models usually give good results, but they require a lot of computational resources and therefore took a long time to simulate. In vehicle dynamics simulations, less computationally demanding tire models are needed. In Adams number of tire models can be used. Figure 3.14 shows the different tire models and their use cases.

As seen from Figure 3.14, PAC 2002 and PAC 2002 with belt dynamics are best to use for every handling simulation. PAC 2002 tire model is a magic formula tire model created by MSC Software based on research done by Pacejka [15]. Since its conception, the Magic Formula has been adopted as the industry standard tire model for vehicle handling simulations. The original magic formula is expressed as;

$$y(x) = Dsin(Carctan(Bx - E(Bx - arctan(Bx)))). \tag{3.10}$$



MD Adams	Event / Maneuver	ADAMS/ Handling Tire							Specific Models	
		PAC2002[1]	PAC-TIME[1]	PAC89[1]	PAC94[1]	FIALA[1]	5.2.1.[1]	UA Tire[1]	PAC-MC[1]	FTire
Handling	Stand still and start	+	o/+	o/+	o/+	o/+	o/+	o/+	o/+	+
	Parking (standing steering effort)	+	-	-	-	-	-	-	-	+
	Standing on tilt table	+	+	+	+	+	+	+	+	+
	Steady state cornering	+	+	o/+	+	o	o	o/+	+	o/+
	Lane change	+	+	o/+	+	o	o	o/+	+	o/+
	ABS braking distance	+	o/+	o/+	o/+	o	o	o/+	o/+	+
	Braking/power-off in a turn	+	+	o	o	o	o	o	+	o/+
	Vehicle Roll-over	+	o	o	o	o	o	o	o	+
	On-line scaling tire properties	+	-	-	-	-	-	-	-	o
Ride	Cornering over uneven roads *	o/+	o	o	o	o	o	o	o	o/+
	Braking on uneven road *	o/+	o	o	o	o	o	o	o	+
	Crossing cleats / obstacles	-	-	-	-	-	-	-	-	+
	Driving over uneven road	-	-	-	-	-	-	-	-	+
	4 post rig (A/Ride)	+	o/+	o/+	o/+	o/+	o/+	o/+	o/+	o/+
Chassis Control	ABS braking control	o/+	o	o	o	o	o	o	o	+
	Shimmy[2]	o/+	o	o	o	o	o	o	o	+
	Steering system vibrations	o/+	o	o	o	o	o	o	o	+
	Real-time	+	-	-	-	-	-	-	-	-
	Chassis control systems > 8 Hz	o/+	-	-	-	-	-	-	-	+
	Chassis control with ride	-	-	-	-	-	-	-	-	+
Dura-bility	Driving over curb	-	-	-	-	-	o	o	-	o/+
	Driving over curb with rim impact	o	-	-	-	-	o	o	-	o/+
	Passing pothole	-	-	-	-	-	o	o	-	o/+
	Load cases	-	-	-	-	-	o	o	-	o/+

-	Not possible/Not realistic
o	Possible
o/+	Better
+	Best to use

* wavelength road obstacles > tire diameter

[1] use_mode on transient and combined slip

[2] wheel yawing vibration due to supension flexibility and tire dynamic response

Figure 3.14: Adams help tire use cases.[40]

This equation can be used to calculate longitudinal and lateral forces as well as self-aligning moments. The D, C, and B parameters represent the tire properties and change from the tire to tire.

In Adams, tire models are imported from tire property files. After the tire models are imported into Adams environment, the road profile that is going to use in the simulation is selected. The workflow of using tires in Adams can be seen in Figure 3.15.

There are four tire road contact models available for in Adams. Except for FTire tire model which models a complete contact patch. The tire road contact models in Adams are;

- One point contact
- Equivalent Volume contact
- Cross Section contact
- 3D Enveloping contact

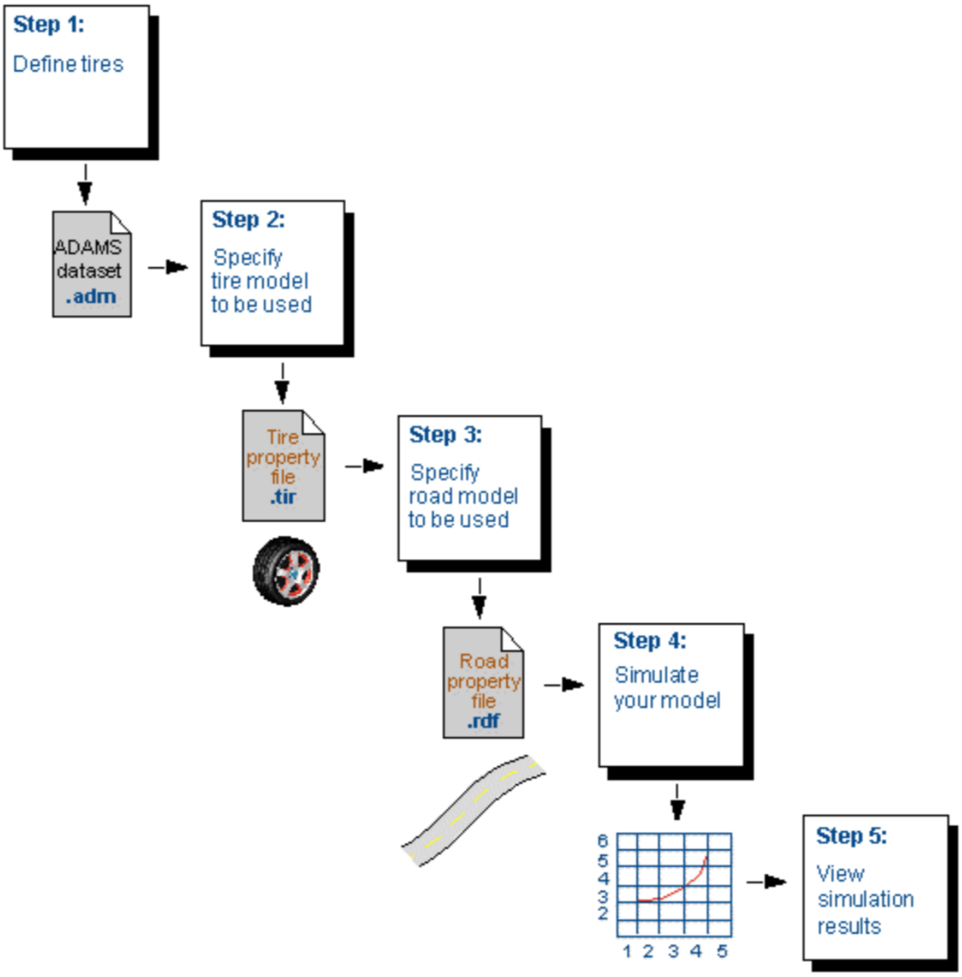


Figure 3.15: Adams tire workflow.[40]

When using the one point contact model, the tire forces are applied at one point that is below the wheel center. This model is useful for flat road simulations. At the roads that are not flat, the one point contact model does not yield good results due to the location of the contact point. The tire model does not react to the changes in the road profile until the contact point is reached to the change in the road profile. Equivalent volume contact model computes the volume of intersection between the road and the tire. Cross section contact model can only be used for PAC-MC tire model. This model shifts the contact point of the tire-road contact according to the cross-section shape of the tire. 3D enveloping contact model uses ellipse that surrounds the contact patch of a tire road intersection and calculates the movement of these ellipses. This contact model is highly effective in uneven roads.

In this work, the vehicle turning radius and the handling behavior is investigated by simulations conducted on a flat surface. PAC 2002 tire model is used as it is the suggested tire model for handling simulations. Because of the flat road, one point contact model was used.

3.5 Using Python In Adams

Python is an object-oriented, open source and high-level scripting language, created by Guido van Rossum in 1991. The main advantage of the Python programming language is its readability. The popularity of Python language has been increasing in the past years due to its use in machine learning and data science. The libraries such as Numpy and Pandas improved the languages popularity and data handling capabilities.

Recent versions of Adams supports Python language interface as an alternative to Adams View command language. This feature allows users to interact with Adams software using Python code. Adams creates a virtual Python environment and run integrated python version in this environment. This integrated Python environment comes with some of the most popular Python libraries like NumPy. Using Python in Adams, modeling objects can be created and modified. This feature gives the user a lot of freedom in model creation and automating processes in Adams. There are a few different ways of using Python language in Adams. Different ways of using Python language in Adams can be given as;

- Importing Python scripts in Adams
- A set of Python, commands in Adams command window
- Run Python scripts in the batch mode

By importing Python files in Adams, complicated tasks can be achieved. In this work, the optimization processes were done by importing Python files into Adams. Also, Python commands can be executed directly from the Adams command window. Running Python lines in Adams command window is a very convenient way of modifying the model. By running the Python command line by line can be time-consuming and prone to error for complicated tasks.

3.6 Genetic Algorithm

Genetic algorithm is an optimization technique that was inspired by nature. In genetic algorithms, a population of potential solutions is selected, paired, and mutated to find an optimal solution. Randomness plays a big part in genetic algorithms. The first population of solutions is usually created randomly. Some of the selection process used in genetic algorithms chooses individuals randomly, and random mutations occur while creating the next generation. Due to these operations, genetic algorithms search the working space constantly, and because of this property, genetic algorithms usually do not get stuck in local minima as often as other optimizations algorithms. Genetic algorithms are powerful optimization algorithms for multi-variable constraint problems. Genetic algorithms are usually are used with binary numbers; these genetic algorithms are called binary genetic algorithms. However, genetic algorithms can be used with any data type in theory. In this section, the genetic algorithm examples given use integers as genes. A flow chart that describes the working principle of the genetic algorithms is given in Figure 3.16.

Optimization process using genetic algorithm starts with creating the initial population. The fitness values for each individual in the initial population must be calculated. After the fitness values are calculated, the selection process is applied to the first generation. In the selection process, the individuals that are going to be in the next generation and will be used for the mating process are selected. The individuals that are not selected get eliminated. After the individuals are selected for the next generation, the pairing process is applied to the selected individuals. The pairing process groups the selected individuals for the mating process. The mating process creates offsprings from the selected and paired individuals, which are called parents. After the mating process, the number of individuals that are selected, and the offsprings is equal to the number of individuals in the first generation. At this point, random mutations are introduced to the population. Random mutations occur for both the parents and the offspring to increase the searching area. After the random mutations occur the fitness values of the individuals are calculated. Selecting, pairing,

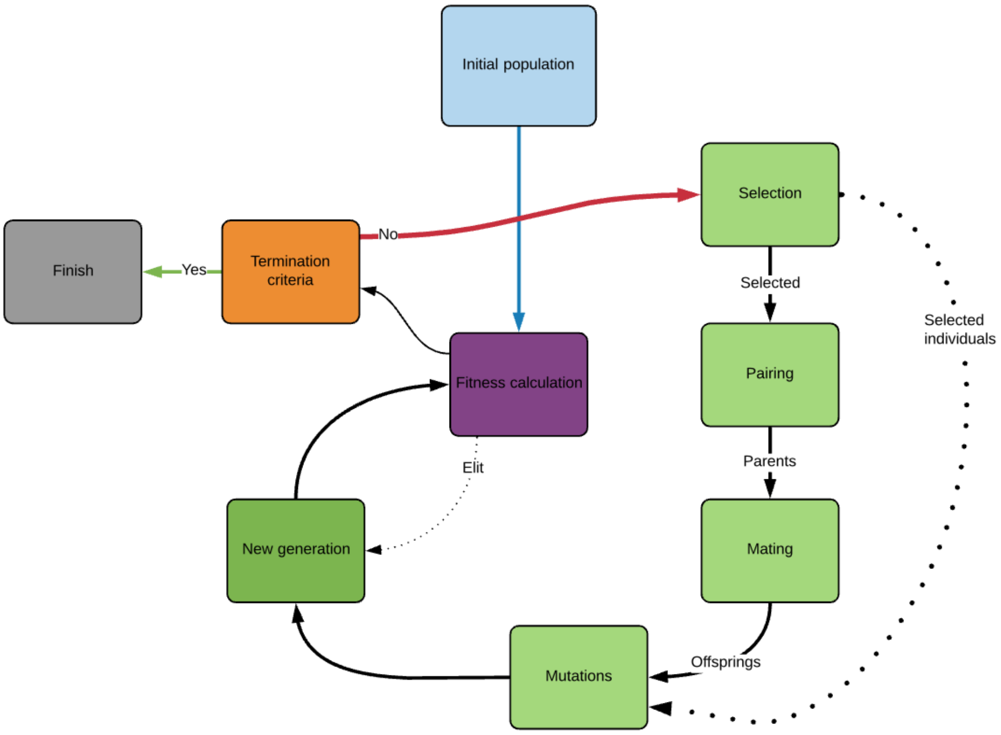
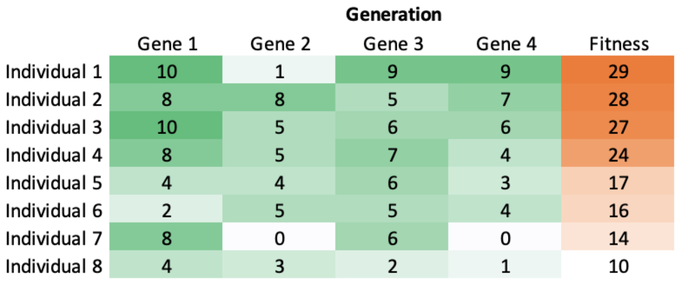


Figure 3.16: Genetic algorithm flow chart.

mating, and mutation operations are conducted repeatedly until the termination criterion is met, and the next generation is created. In the remaining part of this section, the properties of the genetic algorithm are discussed. In the selection process, the individuals that are going to be in the next generation and will be used for the mating process are selected. The individuals that are not selected get eliminated. After the individuals are selected for the next generation, the pairing process is applied to the selected individuals. The pairing process groups the selected individuals for the mating process. The mating process creates offsprings from the selected and paired individuals, which are called parents. After the mating process, the number of individuals that are selected, and the offsprings is equal to the number of individuals in the first generation. At this point, random mutations are introduced to the population. Random mutations occur for about the parents and the offspring to increase the searching area. After the random mutations occur the fitness values of the individuals are calculated. The process of creating the next generation using selecting, pairing, mating, and mutating repeats until the termination criterion is met. In the remaining part of this section, the properties of the genetic algorithm are discussed.

3.6.1 Initial population

A population consists of a predetermined number of candidate solutions called individuals. The number of individuals in the generations is determined before the start of the optimization process. The genes in the individuals are randomly generated. For binary genetic algorithms, each gene is randomly given the value 0 or 1. Moreover, for continuous genetic algorithms, the values of genes are randomly generated in the given interval. The interval that is selected for the genes provides constraints for the genes. Figure 3.17 shows a generation that consists of eight individuals that have four genes each and their corresponding fitness values.



Generation

	Gene 1	Gene 2	Gene 3	Gene 4	Fitness
Individual 1	10	1	9	9	29
Individual 2	8	8	5	7	28
Individual 3	10	5	6	6	27
Individual 4	8	5	7	4	24
Individual 5	4	4	6	3	17
Individual 6	2	5	5	4	16
Individual 7	8	0	6	0	14
Individual 8	4	3	2	1	10

Figure 3.17: Example of a generation.

3.6.2 Fitness

The fitness value of an individual determines how good of a solution that individual is. Fitness value can be thought of as the cost function in an optimization problem. In each generation fitness values for each, each is calculated. Just like any optimization problem effectiveness of the genetic algorithm depends on the fitness function selected. For the optimization problems that have more than one optimization parameter, bias is given to each parameter. Generally, the sum of these biases is chosen to be 1. The higher the bias of the parameter more important that parameter is to the optimization algorithm. Usually, when creating a multi-parameter fitness function for a genetic algorithm, the optimization parameters are normalized, so the fitness function is not affected by the magnitude of the parameters. There are a few normalization functions that can be used for the task. Some of these function can be given as;

- Rescaling
- Mean normalization
- Standardization
- Scaling to unit length

The rescaling method, also known as the min-max normalization is a method that scales the input between the range of 0 and 1. The rescaling function can be written as;

$$m_s = \frac{m_o - min(m_o)}{max(m_o) - min(m_o)} \tag{3.11}$$

Where m_o is the original value of the input and m_s is the scaled value of the input. Mean normalization method is similar to the rescaling method; instead of subtracting the minimum value of the input, the mean value of the input is subtracted from the input data.

$$m_s = \frac{m_o - mean(m_o)}{max(m_o) - min(m_o)} \tag{3.12}$$

Standardization method uses standard deviation to normalize the input value.

$$m_s = \frac{m_o - mean(m_o)}{\sigma} \tag{3.13}$$

Where σ is the standard deviation, scaling to unit length is a method used to find the sign of the input value. This method is useful in some cases, but it does not take the magnitude of the input value, so it is not very suitable for many optimization problems.

$$m_s = \frac{m_o}{|m_o|} \tag{3.14}$$

3.6.3 Selection

The selection operation in the genetic algorithm represents the natural selection in nature. The selection genetic operation picks some of the individuals in a generation to be moved to the next generation and to be parents of the remaining individuals in the next generation. There are some different selection algorithms. In some of the selection algorithms, the fitness of the individuals is used. In the remaining part of this section, different selection algorithms are discussed.

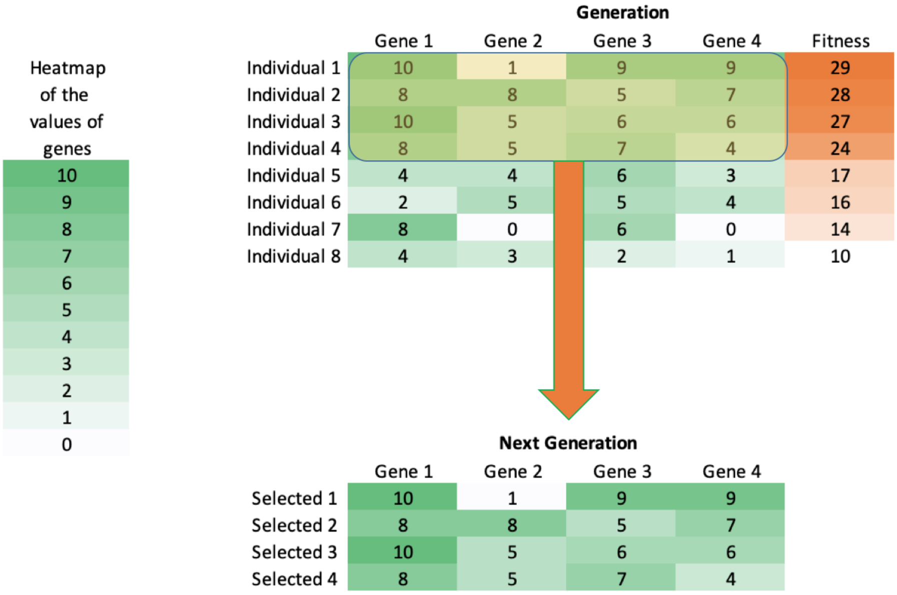
3.6.3.1 Basic Selection

In this selection strategy, half of the population with the highest fitness values are selected. This algorithm represents the survival of the fittest and very easy to implement into code. However, this algorithm lacks the randomness, and so it may be challenging to search the whole solution space and find the global minimum. Basic selection operation performed on the example generation given in Figure 3.17 can be seen in Figure 3.18.

3.6.3.2 Random Selection

The random selection strategy randomly selects individuals that are going to be moved up to the next generation and will be the parents of the remaining individuals in the next generation. Random selection operation visualization can be seen in Figure 3.19.

This selection strategy is beneficial for searching for the search space. The possibility of



Heatmap of the values of genes

10
9
8
7
6
5
4
3
2
1
0

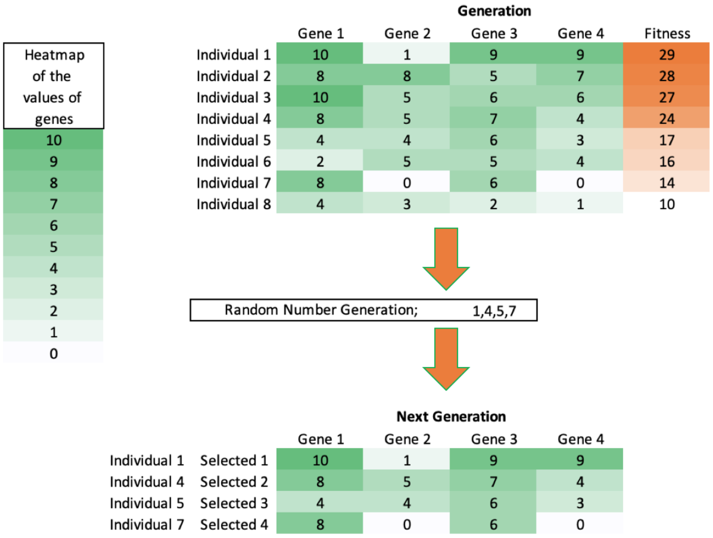
Generation

	Gene 1	Gene 2	Gene 3	Gene 4	Fitness
Individual 1	10	1	9	9	29
Individual 2	8	8	5	7	28
Individual 3	10	5	6	6	27
Individual 4	8	5	7	4	24
Individual 5	4	4	6	3	17
Individual 6	2	5	5	4	16
Individual 7	8	0	6	0	14
Individual 8	4	3	2	1	10

Next Generation

	Gene 1	Gene 2	Gene 3	Gene 4
Selected 1	10	1	9	9
Selected 2	8	8	5	7
Selected 3	10	5	6	6
Selected 4	8	5	7	4

Figure 3.18: Basic selection example.



Heatmap of the values of genes

10
9
8
7
6
5
4
3
2
1
0

Generation

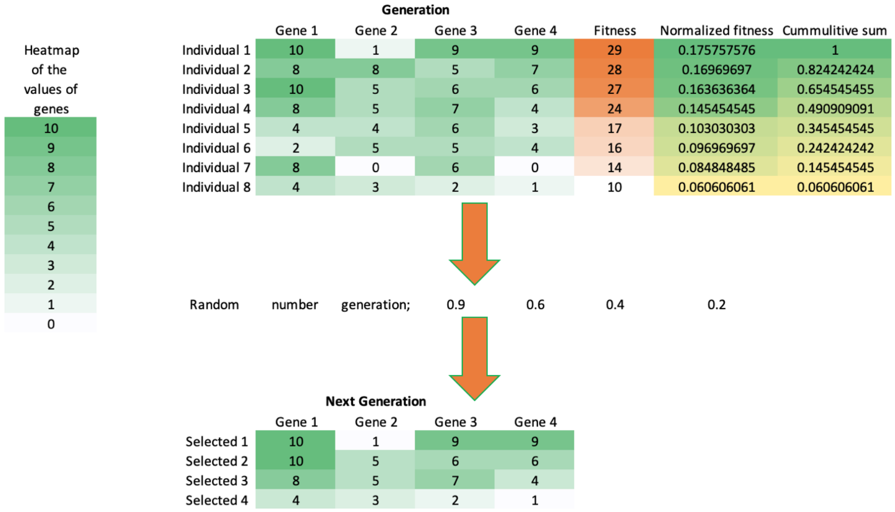
	Gene 1	Gene 2	Gene 3	Gene 4	Fitness
Individual 1	10	1	9	9	29
Individual 2	8	8	5	7	28
Individual 3	10	5	6	6	27
Individual 4	8	5	7	4	24
Individual 5	4	4	6	3	17
Individual 6	2	5	5	4	16
Individual 7	8	0	6	0	14
Individual 8	4	3	2	1	10

Random Number Generation; 1,4,5,7

Next Generation

		Gene 1	Gene 2	Gene 3	Gene 4
Individual 1	Selected 1	10	1	9	9
Individual 4	Selected 2	8	5	7	4
Individual 5	Selected 3	4	4	6	3
Individual 7	Selected 4	8	0	6	0

Figure 3.19: Random selection example.



Heatmap of the values of genes
10
9
8
7
6
5
4
3
2
1
0

Generation

	Gene 1	Gene 2	Gene 3	Gene 4	Fitness	Normalized fitness	Cummulitive sum
Individual 1	10	1	9	9	29	0.175757576	1
Individual 2	8	8	5	7	28	0.16969697	0.824242424
Individual 3	10	5	6	6	27	0.163636364	0.654545455
Individual 4	8	5	7	4	24	0.145454545	0.490909091
Individual 5	4	4	6	3	17	0.103030303	0.345454545
Individual 6	2	5	5	4	16	0.096969697	0.242424242
Individual 7	8	0	6	0	14	0.084848485	0.145454545
Individual 8	4	3	2	1	10	0.060606061	0.060606061

Random number generation; 0.9 0.6 0.4 0.2

Next Generation

	Gene 1	Gene 2	Gene 3	Gene 4
Selected 1	10	1	9	9
Selected 2	10	5	6	6
Selected 3	8	5	7	4
Selected 4	4	3	2	1

Figure 3.20: Roulette wheel selection example.

getting stuck in a local minimum is reduced when using this strategy. However, due to the randomness of this strategy, some of the individuals with functional fitness will be eliminated, and some of the individuals with low fitness will move on to the next generation. Due to this property of random selection strategy converging to an optimized solution can take many iterations.

3.6.3.3 Roulette Wheel Selection

The final selection strategy discussed is the roulette wheel selection. In this method, each individual in a population has a chance to be selected. The chance of an individual to be selected is based on the fitness value of the individual. Fitter individuals have a higher chance to be selected. A commonly used method for implementing this selecting strategy is calculating the cumulative sum of the individuals in the population and generating a random number and select the individual which has the cumulative sum nearest to the random number generated. Using this method, fitter individuals have a higher chance, and unfit individuals are closer to each other, so they have less chance to be selected. Roulette wheel selection algorithm example can be seen in Figure 3.20.

Using roulette wheel selection, every individual has a chance to be selected to move up to the next generation and for pairing. There is no point in choosing an unfit individual to move up to the next generation, but the individuals selected to go through mutations before moving up to the next generation, this gives individuals a chance to be more fit in the next generation. Also, pairing fit individuals with unfit individuals may result in a very fit offspring. So using roulette wheel selection, genetic algorithm acts more randomly, and because of this has a higher chance to find the global optimum and not to get stuck in a local optimum. Both selecting strategies were implemented in Python to observe the effectiveness of each strategy.

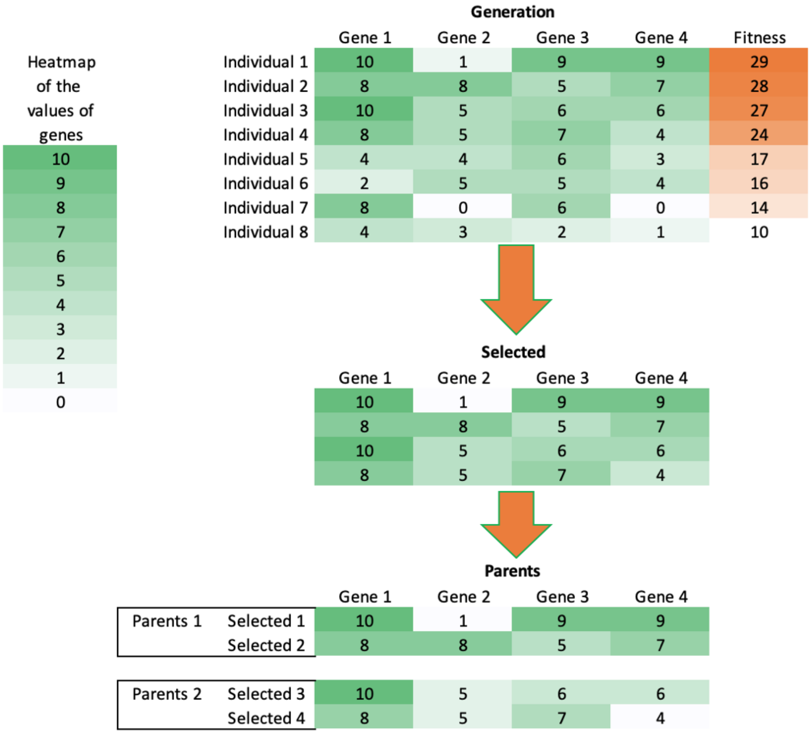
3.6.4 Pairing

Pairing genetic operation is the process in which the selected individuals are grouped for the mating process — deciding which individual will be paired with which individual effects the offsprings. There are a few pairing strategies that can be used in the genetic algorithm optimization process. In genetic algorithms, offsprings can have more than two parents, but often two parents are used to create offsprings. Examples given at the remaining part of this section are for two-parent pairing.

3.6.4.1 Fittest Individual Pairing

In the fittest individual pairing strategy, the fittest individuals that are selected are paired with each other. The individuals are grouped starting from the fittest individual and groups the unfit individuals together. In Figure 3.21, the fittest individual pairing example can be seen.

In the example given in the figure above the individuals of the previous generation are selected using the basic selection strategy and are grouped two by two. This method is handy for grouping together the fittest individuals to obtain even better offsprings. However, the strategy also groups together the unfit individuals, and this may result in unfit offsprings.



Heatmap of the values of genes

Heatmap of the values of genes
10
9
8
7
6
5
4
3
2
1
0

Generation

	Gene 1	Gene 2	Gene 3	Gene 4	Fitness
Individual 1	10	1	9	9	29
Individual 2	8	8	5	7	28
Individual 3	10	5	6	6	27
Individual 4	8	5	7	4	24
Individual 5	4	4	6	3	17
Individual 6	2	5	5	4	16
Individual 7	8	0	6	0	14
Individual 8	4	3	2	1	10

Selected

Gene 1	Gene 2	Gene 3	Gene 4
10	1	9	9
8	8	5	7
10	5	6	6
8	5	7	4

Parents

		Gene 1	Gene 2	Gene 3	Gene 4
Parents 1	Selected 1	10	1	9	9
	Selected 2	8	8	5	7
Parents 2	Selected 3	10	5	6	6
	Selected 4	8	5	7	4

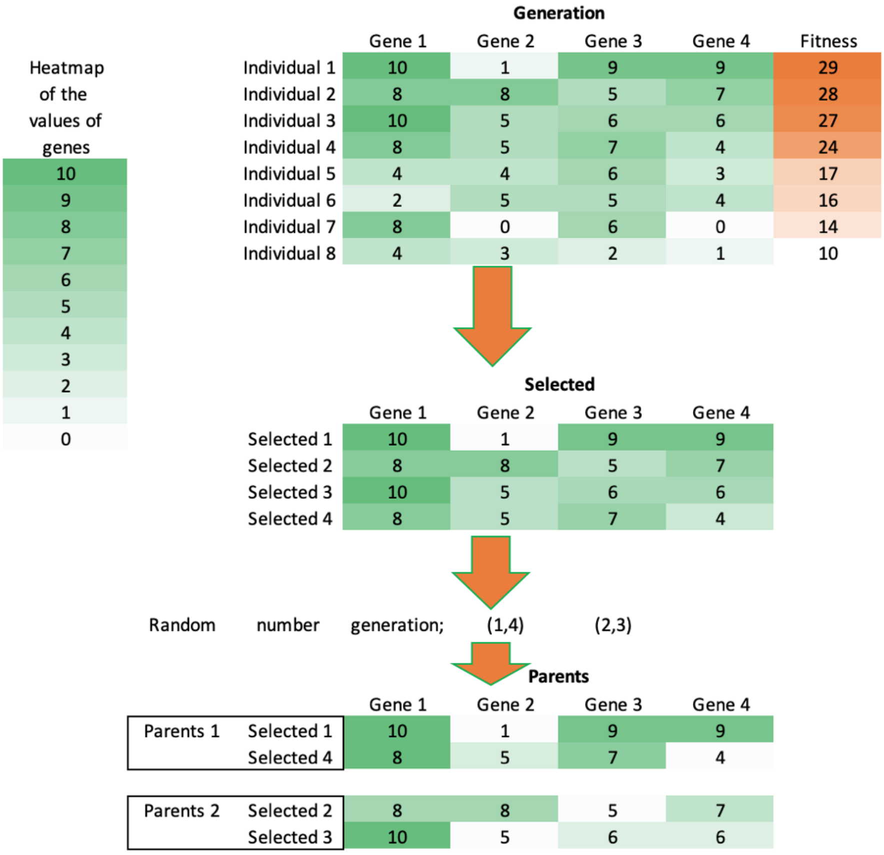
Figure 3.21: Fittest individual pairing example.

3.6.4.2 Random Pairing

In a random pairing strategy, the selected individuals are randomly grouped. This method increases the randomness of the optimization process, which ultimately increases the search space. Due to the randomness of the process, unfit individuals may be grouped and produce unfit offsprings. Figure 3.22 shows a random pairing example.

3.6.4.3 Weighted Random Pairing

Weighted random pairing strategy uses summative sums of the normalized fitnesses of the selected individuals. In this strategy, the random number generation is used to select individuals for grouping together. The method works like the roulette wheel selecting a strategy. Weighted random pairing example is given in Figure 3.23.



Heatmap of the values of genes
10
9
8
7
6
5
4
3
2
1
0

Generation

	Gene 1	Gene 2	Gene 3	Gene 4	Fitness
Individual 1	10	1	9	9	29
Individual 2	8	8	5	7	28
Individual 3	10	5	6	6	27
Individual 4	8	5	7	4	24
Individual 5	4	4	6	3	17
Individual 6	2	5	5	4	16
Individual 7	8	0	6	0	14
Individual 8	4	3	2	1	10

Selected

	Gene 1	Gene 2	Gene 3	Gene 4
Selected 1	10	1	9	9
Selected 2	8	8	5	7
Selected 3	10	5	6	6
Selected 4	8	5	7	4

Random number generation; (1,4) (2,3)

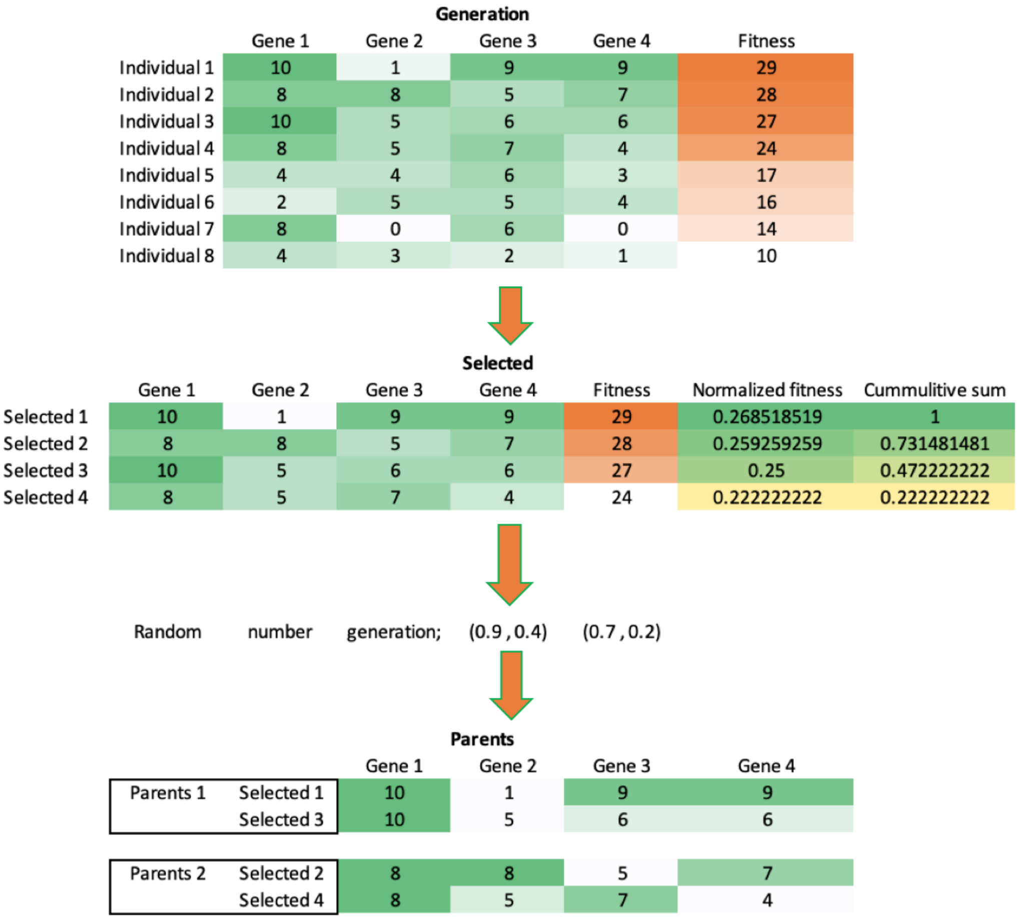
Parents

		Gene 1	Gene 2	Gene 3	Gene 4
Parents 1	Selected 1	10	1	9	9
	Selected 4	8	5	7	4
Parents 2	Selected 2	8	8	5	7
	Selected 3	10	5	6	6

Figure 3.22: Random pairing example.

3.6.5 Mating

The mating process is the genetic operation where the parent individuals produce offsprings. Usually, in the mating process, two-parent individuals produce two offsprings. This process of two parents creating two offsprings makes the controlling of the population size of the next generation easier. After the initial population is randomly created, the mating process is one of the primary sources of searching the search space for the genetic algorithm. The mating process creates half of the individuals in the next, so the impact of the mating process in the effectiveness of the genetic algorithm is very high. There are a few mating methods that can be used for genetic algorithms. The choice of the mating process is crucial for the effectiveness of the genetic algorithm. The mating method choice is dependent on the problem.



Generation

	Gene 1	Gene 2	Gene 3	Gene 4	Fitness
Individual 1	10	1	9	9	29
Individual 2	8	8	5	7	28
Individual 3	10	5	6	6	27
Individual 4	8	5	7	4	24
Individual 5	4	4	6	3	17
Individual 6	2	5	5	4	16
Individual 7	8	0	6	0	14
Individual 8	4	3	2	1	10

Selected

	Gene 1	Gene 2	Gene 3	Gene 4	Fitness	Normalized fitness	Cummulitive sum
Selected 1	10	1	9	9	29	0.268518519	1
Selected 2	8	8	5	7	28	0.259259259	0.731481481
Selected 3	10	5	6	6	27	0.25	0.472222222
Selected 4	8	5	7	4	24	0.222222222	0.222222222

Random number generation; (0.9 , 0.4) (0.7 , 0.2)

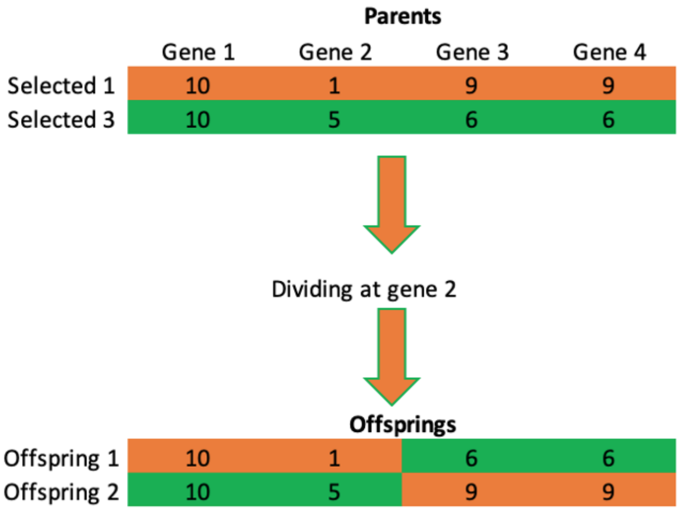
Parents

		Gene 1	Gene 2	Gene 3	Gene 4
Parents 1	Selected 1	10	1	9	9
	Selected 3	10	5	6	6
Parents 2	Selected 2	8	8	5	7
	Selected 4	8	5	7	4

Figure 3.23: Weighted random pairing example.

3.6.5.1 Single Point Mating

In the sign point mating process, the two parent individuals are divided from the designated dividing point, and the parts are combined with the other part of the other parent. Dividing point of the algorithm can be predetermined before the optimization process if a particular point in the individual will make more sense to divide the individuals at. The dividing point can be decided randomly or selecting a random point in a predetermined point set. Figure 3.24 shows a single point mating example.



Parents

	Gene 1	Gene 2	Gene 3	Gene 4
Selected 1	10	1	9	9
Selected 3	10	5	6	6

Dividing at gene 2

Offsprings

Offspring 1	10	1	6	6
Offspring 2	10	5	9	9

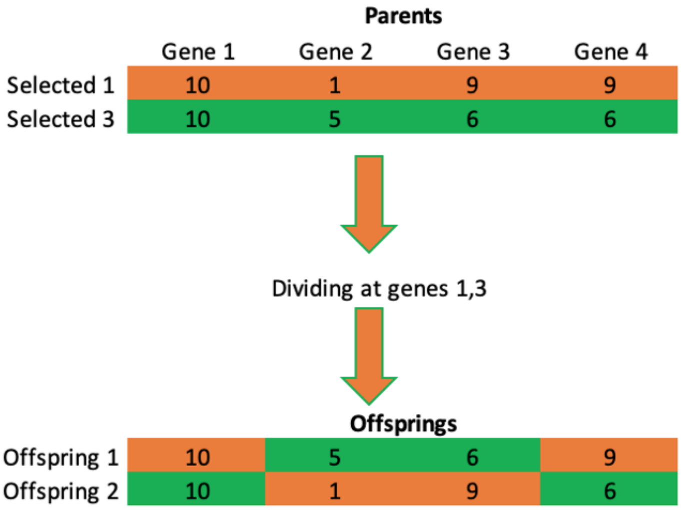
Figure 3.24: Single point mating example.

3.6.5.2 Two Points Mating

Two points mating strategy is similar to the single point mating strategy, but instead of dividing the parent individuals, the parent individuals are divided at two points. Dividing parent individuals in two places creates three parts from each parent. These parts are combined so each offspring would have two parts from one parent and one part from the other parent. Two points mating example is given in Figure 3.25.

3.6.5.3 Single Station Mating

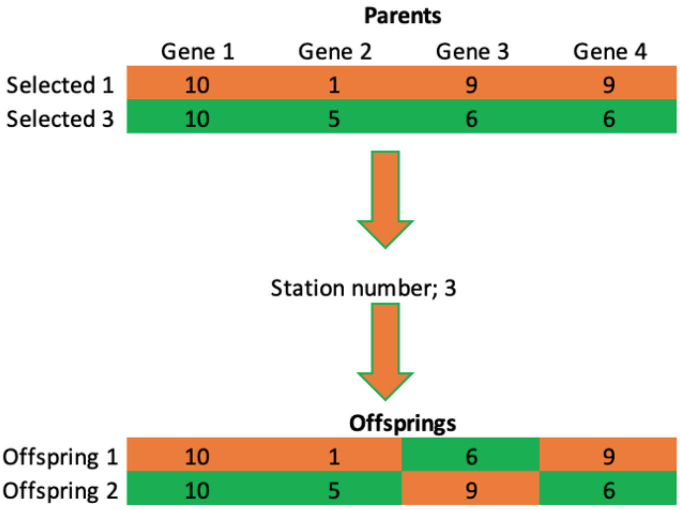
Single station mating method is a specialized version of the two points mating method. Instead of dividing the parent individuals at random paints, in this method, one dividing point is selected, and the other dividing point is calculated by summing the gene number of the first dividing point and the predetermined station length. This method can give good results in some cases, but its effectiveness is very dependent on the problem. In Figure 3.26, single station mating example can be seen.



	Gene 1	Gene 2	Gene 3	Gene 4
Selected 1	10	1	9	9
Selected 3	10	5	6	6

Offspring 1	10	5	6	9
Offspring 2	10	1	9	6

Figure 3.25: Two points mating example.



	Gene 1	Gene 2	Gene 3	Gene 4
Selected 1	10	1	9	9
Selected 3	10	5	6	6

Offspring 1	10	1	6	9
Offspring 2	10	5	9	6

Figure 3.26: Single station mating example.

3.6.6 Mutation

Mutation operation changes some aspects of the individuals so that already known individuals do not move up to the next generation. Mutation operation is one of the most critical operations of the genetic algorithm to traverse the searching space and not to get stuck in local minima. Different mutation strategies can be used as genetic algorithms.

3.6.6.1 Reset Mutation

In the reset mutation method, the selected gene of the individual is replaced by randomly generating a number according to the constraints that apply to that gene. This method increases the ability of the genetic algorithm to search the search space but can lead to longer convergence times. The genes that are going to be mutated are selected randomly. The number of genes that are going to mutated is called the mutation rate. The mutation rate can be selected randomly, which will further increase the searching ability of the genetic algorithm, or a specific number can be chosen as the mutation rate. The mutation rate dramatically impacts the effectiveness of the genetic algorithm. Figure 3.27 shows reset mutation example.

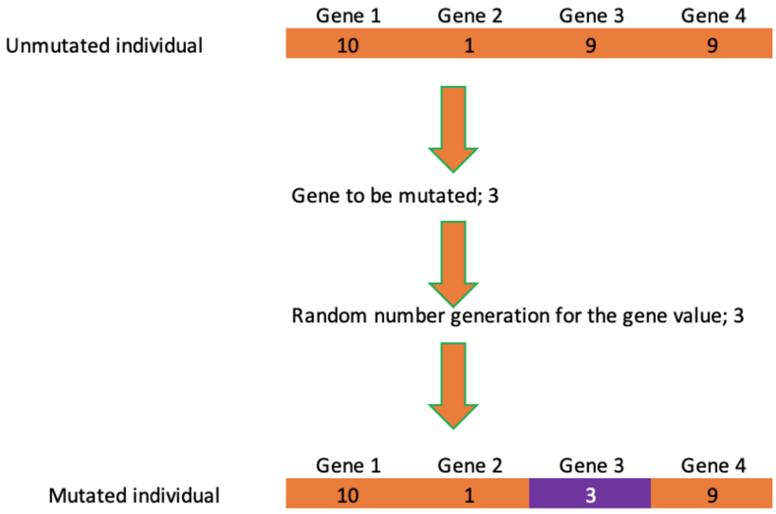


Figure 3.27: Reset mutation example.

3.6.6.2 Gauss Distribution Mutation

In the Gauss distribution mutation method, the selected individual for mutation is replaced by a random number that has been randomly generated according to Gauss distribution. Using this method random number generation can be made more predictable. This method is particularly useful for fit individuals as it will most likely change the genes by a small margin; this may cause the individuals to be fitter. However, for unfit individuals changing the genes by a small margin may restrict their fitness. A gauss distribution example can be seen in Figure 3.28.

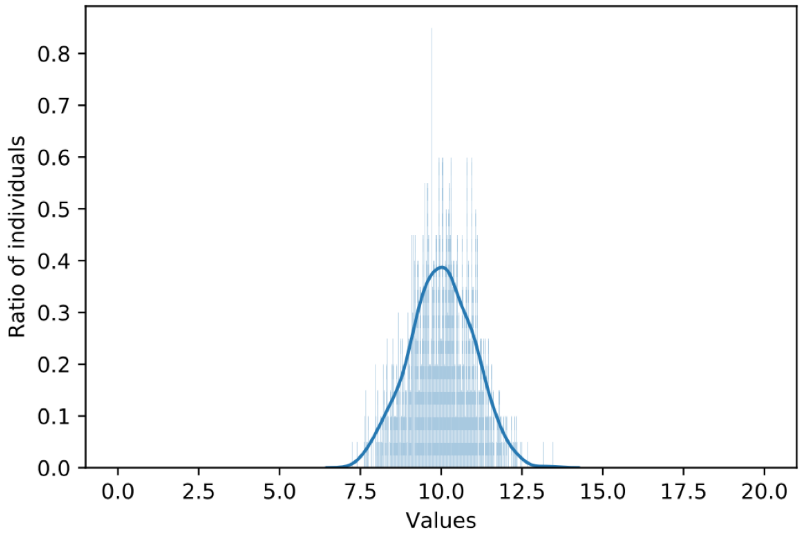


Figure 3.28: Gauss distribution example.

3.6.7 Next Generation

Now that all of the genetic operations are presented, the creation of the next generation can take place. After the first generation, all of the generations are created using the same method. After the fitness values of the individuals in a generation are calculated, the selection process begins. A concept called elitism can be introduced to the genetic algorithm in this part. Elitism is the process of automatically selecting the fittest individual in a generation to be in the next generation and to produce offsprings. The elite does not go through random mutations before moving to the next generation. The intent of this concept is saving the best solution. Selected individuals and the elite are paired for the mating process. The mating process creates the offsprings that will be in the next generation. Offsprings and selected individuals go through random mutations before moving up to the next generation. After the mutations, the fitness values of the elite, selected individuals and the offsprings are calculated, and the next generation is completed. This process is repeated until the termination criteria are met.

3.6.8 Termination of the Genetic Algorithm

The optimization process using a genetic algorithm begins with creating the first generation and continuous by creating the next generations after that. The algorithm must stop when the desired results are obtained to complete the optimization process. Many termination criteria can be used with genetic algorithms. The termination criterion that is discussed in this chapter can be used together in the genetic algorithm, and the algorithm will stop whichever termination criteria are met first.

- Limiting number of generations
- Limiting maximum fitness
- Limiting average fitness
- Limiting maximum fitness repetition

One of the most popular termination criteria is stopping the algorithm when a certain number of generations are created. This method is straightforward to implement is very useful to ensure that the genetic algorithm will stop at a particular time. Usually, the optimization algorithms are used with a while loop, which means if the termination criteria are not met the genetic algorithm will go on working. Stopping the genetic algorithm at a certain number of generations works as a safety measure that guarantees the while loop will not go on forever.

Another termination criteria are setting a maximum fitness limit. The maximum fitness of each generation is calculated in this method, and the algorithm is stopped when the maximum fitness limit is reached. This is one of the most useful termination criteria there is. In many optimization problems, there is a specific goal. This criterion ensures that the genetic algorithm is finished after reaching the goal of the optimization process. In the figure below, it can be seen that there are four minima. Two of the local minima and the global minima are below the maximum fitness limit, meaning if the optimization algorithm can reach one of these points, the termination criteria will be triggered. One of the local minima is higher than the maximum fitness limit and reaching this point will not trigger the termination criteria. Optimization algorithms can get stuck in local minima like that. An example function is given in Figure 3.29; in this function, it can be seen that some of the local minima satisfy the termination criteria.

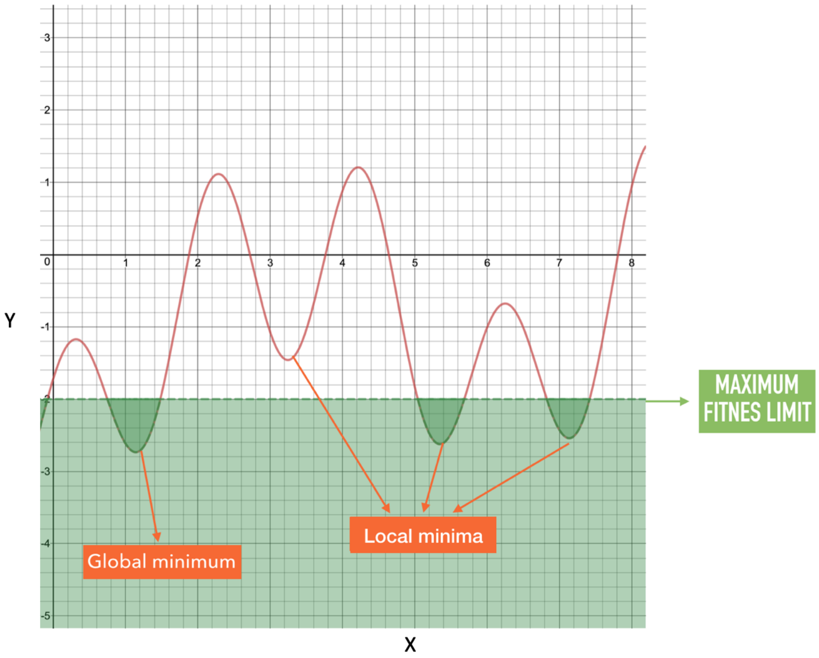


Figure 3.29: 2D fitness value visualization.

Limiting the average fitness value of the generation is a very useful termination criterion in some cases. In this method, the average fitness value of each generation is calculated, and the genetic algorithm is terminated if the average fitness value of a generation reaches a certain level. This method is; the goal of the genetic algorithm is to find a set of suitable solutions.

The last termination criteria discussed is limiting the repetition of maximum fitness value.

If the genetic algorithm has elitism, the individual with the maximum fitness value goes on to the next generation without random mutations occurring, if the genetic algorithm cannot find a better solution the best individual will be the best solution again. In this case, the maximum fitness value will be the same. Limiting the number of the same maximum fitness number can help terminate genetic algorithms that are stuck in a local minimum. The method also helps as a safety measure that stops the algorithm if there is no progress.

3.7 Conclusion

Due to the limitations of the multi-axle bicycle model, a multi-axle multi-body dynamics model of the vehicle was used in this thesis. The modeling process of the vehicle model used in the optimization study is very similar to the modeling process reviewed in this chapter. There are, however, some differences to make the model more suited to the task at hand. The vehicle model and the changes to the convention will be discussed in detail in the upcoming chapter. The genetic algorithm that is used for the optimization process is written in the Python language. The theory of the genetic algorithm code written in Python language is based on the theory presented in this chapter.

4 TURNING RADIUS OPTIMIZATION

4.1 Introduction

In this chapter, optimal wheel steering angles that cause the minimum turning radius and at the same time cause the minimum tire lateral slip angles are obtained. For the optimization study, multi-body dynamics model of the investigated 8x8 vehicle was created in Adams Car. An Adams command file was designed to run the simulations and gather the results that are going to be used by the optimization algorithm. A continuous genetic algorithm was used to conducts the optimization process. The genetic algorithm was implemented into the Python code. The Python code runs the simulations and gathers the results by calling upon the Adams command file created and performs the optimization process using the genetic algorithm code. The workflow of the optimization process can be seen in Figure 4.1.

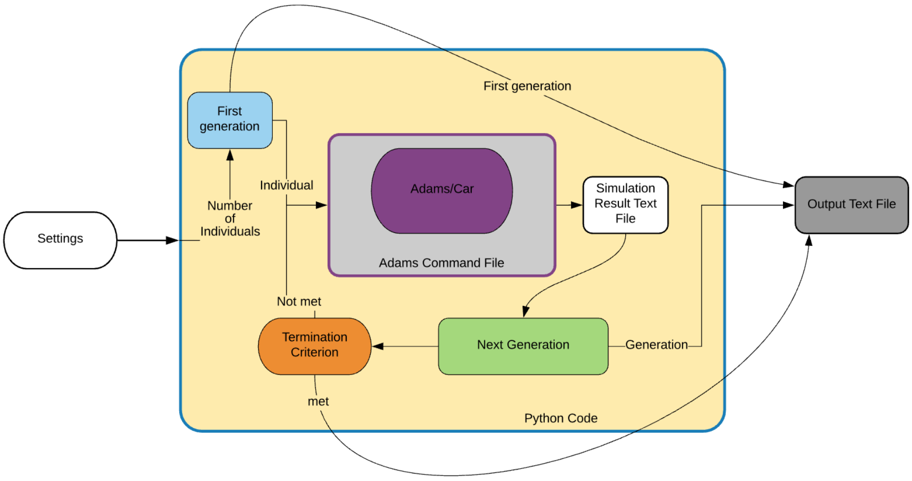


Figure 4.1: Turning radius optimization flow chart.

The chapter begins with the methodology part; in this section, the working principle of the optimization process is given. Multi-body dynamics model used in the simulations is presented after this section. The Adams command file that runs the simulations and outputs the results is presented after the model. The Python code that connects the different parts of the process is given in this section. Moreover, the chapter is concluded with the discussion of the optimization results.

4.2 Maneuver

The maneuver used in the turning radius optimization process is a low-speed steady-state turning maneuver. The figures shown in this section are the images and simulation results of the vehicle with the original configuration. The vehicle path during the maneuver can be seen in Figure 4.2. During the maneuver, the longitudinal velocity of the vehicle is controlled by the velocity controller of Adams Car. The aim of the controller during the maneuver is to keep the longitudinal speed of the vehicle constant. The steering inputs for all of the wheels are inputted separately as step inputs. The turning radius calculation is done after the vehicle reaches steady-state. The examples given in the rest of this section are from the simulations that were conducted using the originally proposed steering angles for the prototype vehicle. The optimized vehicle wheel steering angles will be compared with the these originally proposed steering angle configuration.

The longitudinal velocity of the vehicle during the maneuver is given in the below figure. The velocity controller was trying to keep the longitudinal vehicle velocity close to the desired velocity, which is 10 km/h in this maneuver. The vehicle velocity fluctuates between 10km/h at the beginning of the simulation, but after some time passes, the vehicle velocity reaches steady-state at 10 km/h. The vehicle velocity change with respect to time can be seen in Figure 4.3. In the vehicle turning radius calculation, the part of the simulation where the vehicle velocity is not constant is not taken into consideration.

The steering inputs of the wheels are given as step inputs. The step function in Adams command language uses hyperbolic tangent function to smooth the sharp edges of the step function. All of the steering angle inputs starts and reaches to the maximum values at the

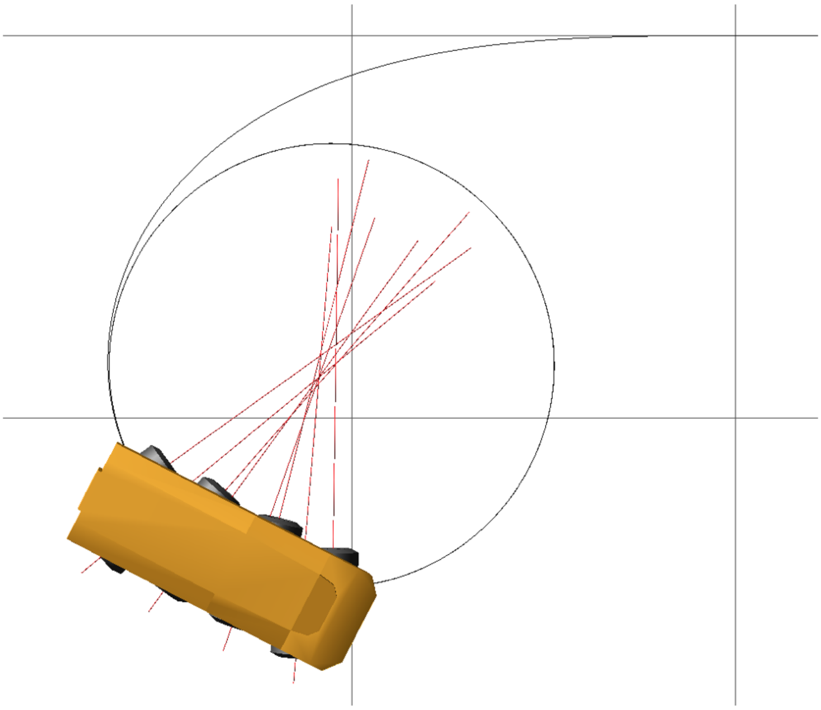


Figure 4.2: Topdown look at the vehicle during the low-speed steady-state turning maneuver.

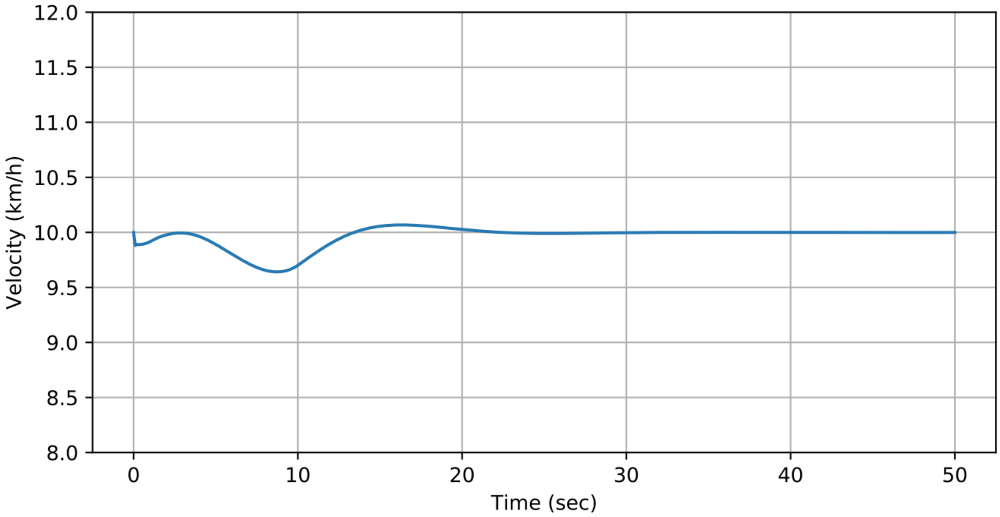


Figure 4.3: Vehicle velocity change during the maneuver.

same time. Wheel steering angle change with respect to time can be seen in Figure 4.4.

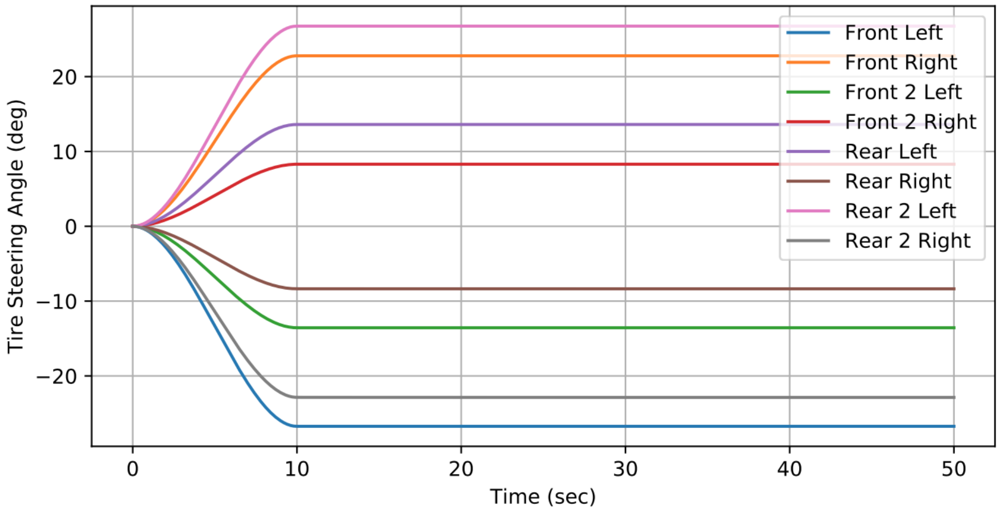


Figure 4.4: Steering inputs given to the tires during the maneuver.

4.3 Adams Command File

The Adams command file is the file that runs Adams Car simulations, gathers the required results, and outputs them. Adams command file is a file that contains Adams command that is written in Adams command language. After functions of the motions that turn the tires are changed using Python code, Adams command file is called by the Python code to run the simulations and output the results. The Adams command file starts the simulation by calling upon the Adams event file. Adams event file contains all of the information about the simulation that is going to be run. The event file determines the duration and the step size of the simulation as well as the desired vehicle behavior. The desired vehicle behavior, in this case, means the longitudinal velocity of the vehicle should be kept constant as well as the gear ratio. There is no steering or braking demand. The steering of the wheels is achieved by the motions in the suspension system these motions has time-dependent functions and are not affected by the simulation parameters as long as the simulation time is sufficient. Wheel steering angle change with respect to time is given in Figure 4.5.

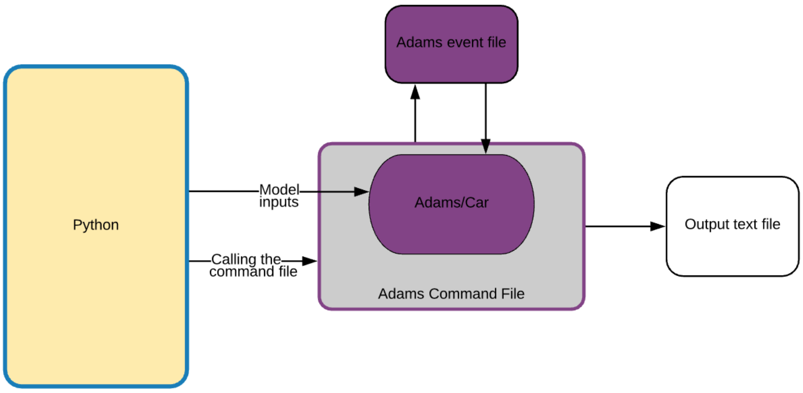


Figure 4.5: Adams command file work flow.

After the simulation is finished, the Adams command file outputs the desired results as text files. In this case, the longitudinal and lateral position of the marker, which is located at the outside of the front left tire is outputted. Also, tire lateral slip angles of all the tires are outputted as well. After the desired results are outputted the Adams command file deletes the simulations results from Adams Car. After the simulations results are deleted, Adams Car is ready to run another simulation. The Adams command file that runs the turning radius optimization simulations, as well as the Adams event file which contains the simulation details, are given in Appendix B.

4.4 Python Code

The Python code is the part of the method which connects all of the elements of the study. After the multi-body dynamics vehicle model is opened in Adams Car, the turning radius optimization Python file is imported to Adams Car. The Python code then conducts the optimization process by running Adams Car simulations and investigating the results of these simulations. Figure 4.6 shows the workflow of the Python code.

The Python code starts by creating the initial population. The fitness values of the individuals in the first population are found next. The first generation is then written to the output text file. After the first generation is completed, generations that come after are completed by using the next generation function. This function takes in the old generation

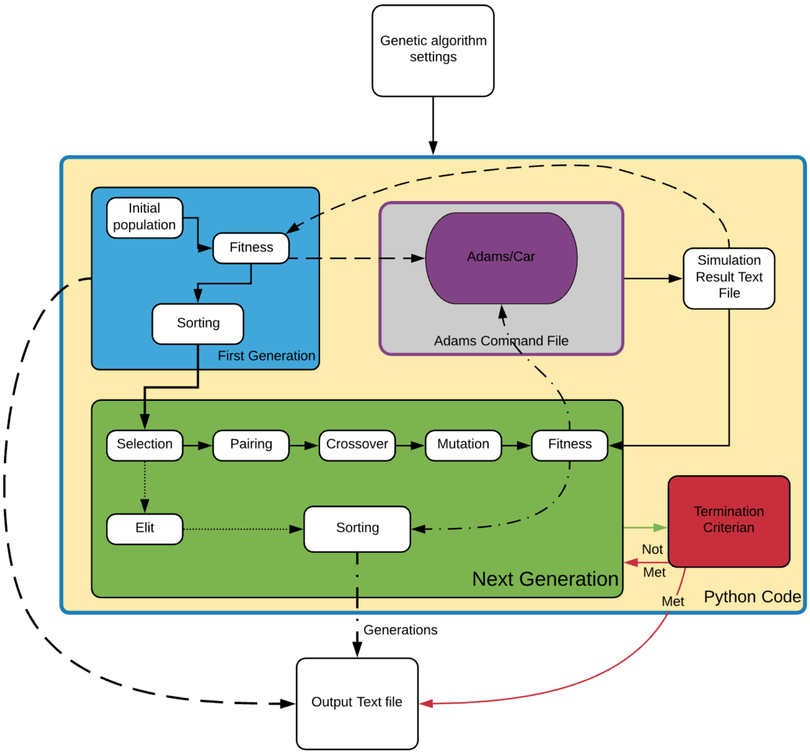


Figure 4.6: Work flow of the Python code.

and outputs the next generation. After each generation, the fitness values of the individuals in that function are given to the termination criterion function. If the termination criterion is met with the fitness values of the individuals in that generation, the optimization process is completed. The generation, maximum, and average fitness values are written to the output text file. If the termination criterion is not met the optimization process repeats until the termination criterion is met. The Python code that performs the optimization process is presented in Appendix C.

4.4.1 Genetic Algorithm

In this work, a continuous genetic algorithm was used. The reason for using continuous genetic algorithm instead of using a binary genetic algorithm is, there are eight genes in each individual, and floating numbers represent these genes, the binary representation of a floating number is very long, and it would make the individuals very hard to interpret. An example of a binary representation of an integer and the floating number is given in Table

4.1.

Table 4.1: Binary vs floating number representation.

	Number	Binary
Integer	29	11101
Floating Number	29.9	11101.1110011001100110011

In this study, each tire has a set of maximum steering angles created with 0.1-degree increments between two constraints. A 0.1-degree increase in a tire maximum steering angle in one of the tires causes about 1 mm decrease in the vehicle turning radius. Effect of any lower increment in tire maximum steering angle would be ignorable and it would be challenging to design a steering mechanism with such sensitivity. Any higher increment in maximum wheel steering angle could cause the optimization algorithm to miss the global minima.

In this work, each candidate solution is called an individual. Each individual is made up of eight genes. These genes represent steering angles of the tires. In genetic algorithm optimization, individuals have a chance to be selected for moving up to the next generation and pairing to create offsprings for the next generation. Fitness values for each individual must be calculated using a fitness function. Individuals with higher fitness values are better solutions to the problem at hand.

The functions that perform the genetic algorithm optimization process were written in Python language. The genetic operations were discussed in the Theoretical Background chapter. The genetic algorithm code was created using the principles reviewed in the Theoretical Background chapter. Creating the initial population is a problem specific operation and will be presented in this chapter. The selection genetic operation is a general operation, and the principles were reviewed in the Theoretical Background chapter. The selection function in Python was written in a way that it would take in the fitness values and the individuals from the previous generation independent of the input size. Two different selection strategies were implemented to the selection function, which are roulette wheel selection and basic selection. The selection function takes two inputs the previous generation without the elite and the selection type. The pairing process is also a general operation. There are three different pairing strategies implemented to the pairing function, which are the fittest pairing, random pairing, and weighted random pairing. The pairing function takes three inputs the fitness and the individuals of the selected individuals, fitness, and the individual of the elite and the pairing strategy. The mating function was written according to the theory discussed in the Theoretical Background chapter. Both single point and single station codes were implemented into the Python code. The mating function takes two inputs parents and the mating type. Finally, the mutation function was created with gauss and reset mutation strategies. The mutation function takes three inputs, which are individuals to be mutated, mutation rate, and mutation method. The fitness function that is used to calculate the fitness values of the individuals is also problem specific and will be presented in the upcoming sections.

4.4.2 Initial Population

The initial population consists of a predetermined number of individuals, the number of individuals for each optimization process is given in Table 4.4. Each individual consists of eight genes which represent the wheel steering angles. The function that creates the individuals, creates each gene by generating a random number between the given constraints. The upper and lower limit used in each axle are given in Table 4.2.

Table 4.2: Gene limits.

	Upper Limit (deg)	Lower Limit (deg)
L1	-20	-30
L2	-5	-15
L3	15	5
L4	30	20
R1	-18	-28
R2	-3	-13
R3	13	3
R4	28	18

These limits represent the constraints of the system. As was discussed in the introduction chapter, the search space is vast, and it can take a lot of time for optimization algorithms to search all the search space. Some additional constraints were introduced to the system to limit the search to a section of the search space that is more suited to the desired properties. Limits of the right side of the vehicle as well as the second and third axles are lower than the left front limits. These limitations helps the genetic algorithm to keep the lateral slip angles low. The limits given in Table 4.2 does not guarantee the right wheel steering angles to be lower than the left wheel in the same axle, so to obtain lower wheel steering angles in right side of the axle compared to the left side, the random number generation process for the right side of the axle takes place in a while loop after the steering angle of the left side wheel is determined. The while loop continuously generates number until a lower number than the left wheel steering angle is reached.

4.4.3 Fitness Function

The fitness values of the individuals are calculated using the fitness function. This function inputs the steering angles to Adams Car and calls the Adams command file to run the simulations. After the Adams command file runs the simulations, the simulations results which are going to be used for fitness calculations are outputted as text files. The Python code reads the text files that contain the simulation results and converts them into Python objects. The fitness calculation is conducted using these parameters. The workflow of the fitness function is given in Figure 4.7.

The slip angles of the tires are directly outputted from Adams Car. The fitness function reads the result text files that contain the tire lateral slip angles and creates a Python object. These Python objects only include the part of the simulations results where the vehicle reaches steady-state. The duration of the simulations for this problem is chosen as 50 seconds. The steering inputs of the tires reach their maximum values after 10 seconds. The results that are going to be used for the fitness calculation starts 25 seconds into the simulation and gathers the results until the end of the simulation time. This process gives the vehicle 15 seconds to reach steady-state. From the simulation results provided in this section, it can be seen that is enough time for the vehicle to reach steady-state, and there is some safety margin as well. After the Python object that contains the vehicle lateral slip angles is created, the absolute value of the maximum of the tire lateral slip angles is calculated. Due to the nature of the maneuver even if the vehicle does not reach steady

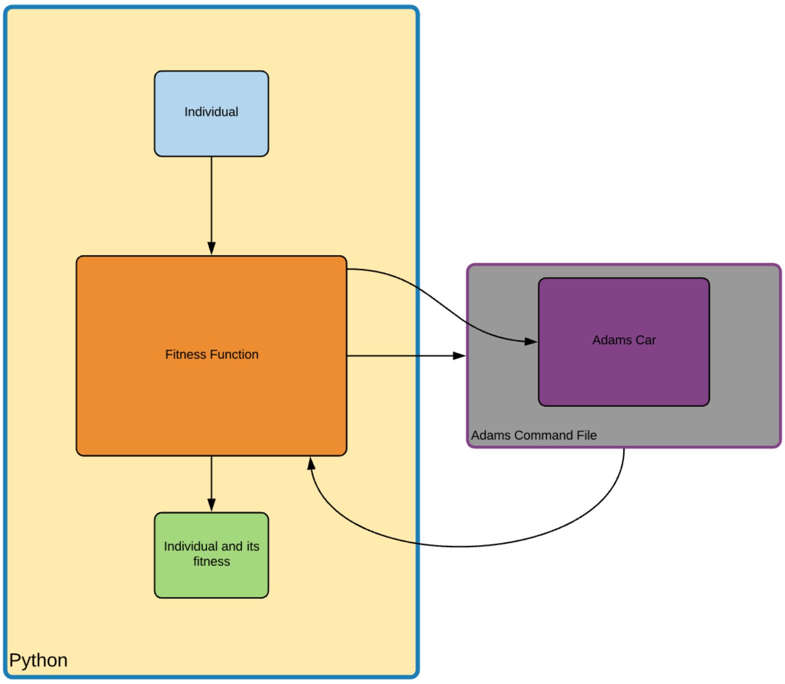


Figure 4.7: Fitness function flow chart.

state 15 seconds after the steering inputs reach their maximum values, the minimum lateral slip angle will still occur after the vehicle reaches steady-state.

The turning radius calculation is a little bit trickier than the slip angles. Adams Car does not directly output the curb-to-curb turning radius of the vehicle. The Python code must calculate the curb-to-curb turning radius of the vehicle. For this task, lateral and longitudinal positions of the marker located at the outside of the right wheel are outputted from the Adams Car as text files by the Adams command file. These text files are then red by the Python code, and Python objects are created from the simulation results like the lateral slip angles. The Python objects are created from the part of the simulation results in which the vehicle is performing a steady state turn. During this maneuver, the marker located at the outside of the front right tire completes a perfect circle. The radius of this circle is the vehicles curb-to-curb turning radius. To calculate the curb-to-curb turning radius of the vehicle, three points are selected from the Python objects which contain the lateral and longitudinal displacement of the marker located at the outside of the front right wheel. Only one circle connect three-point and radius of this circle can be calculated using the positions of these three points.

$$r = \sqrt{\frac{B^2 + C^2 - 4AD}{4A^2}} \tag{4.1}$$

Where A, B, C, and D can be calculated as;

$$A = x_1(y_2 - y_3) - y_1(x_2 - x_3) + x_2y_3 - x_3y_2 \tag{4.2}$$

$$B = (x_1^2 + y_1^2)(y_3 - y_2) + (x_2^2 + y_2^2)(y_1 - y_3) + (x_3^2 + y_3^2)(y_2 - y_1) \tag{4.3}$$

$$C = (x_1^2 + y_1^2)(x_3 - x_2) + (x_2^2 + y_2^2)(x_1 - x_3) + (x_3^2 + y_3^2)(x_2 - x_1) \tag{4.4}$$

$$D = (x_1^2 + y_1^2)(x_3y_2 - x_2y_3) + (x_2^2 + y_2^2)(x_1y_3 - x_3y_1) + (x_3^2 + y_3^2)(x_2y_1 - x_1y_2) \tag{4.5}$$

Where x_i and y_i parameters are the x and y coordinates of the corresponding points, when using the equations given above no matter which three points are chosen on the circle the result will be the same. In this work, the points are chosen with a constant interval between them. The curb-to-curb turning radius of the vehicle is calculated by the Python code using the equations given above.

After Python objects are created from the turning radius of the vehicle, and the tire lateral slip angles, the fitness of the individual can be calculated. In this work, a few different fitness functions are considered for the optimization process. The first fitness function considered is giving very high bias for the turning radius and giving tire lateral slip angles very low bias. In this cost function a limit slip angle is chosen and if the lateral slip angle is lower than the limit slip angle value is taken, with a very low bias in this case effect of the lateral slip angle is minimum on the fitness function. However, if the lateral tire slip angle is higher than the limit, a huge number is taken instead. Using this method, tire lateral slip angles can be kept below a specified limit. The fitness function in this method can be found by taking one over the described function, so the higher fitness value means a fitter individual.

$$F = \frac{1}{0.92r + 0.01\sum_{i=1}^{8} a_i}, \tag{4.6}$$

Where,

$$a_i = \begin{cases} a_i = a_i & a_i < 3 \\ a_i = 1000000 & a_i > 3 \end{cases} \tag{4.7}$$

This fitness function favors the individuals that have lower tire slip angles than the limit. However, the search space is vast, and finding a set of genes that will result in lower than limit tire lateral slip angles randomly is highly unlikely. Using this fitness function, controlling the genetic algorithm is not possible. Slip angle limits in the cost function act as additional constraints, and the only objective is to minimize the turning radius.

To be able to control the fitness function to obtain the desired results, another fitness function was created. In this fitness function, the vehicle turning radius and the maximum of the tire lateral slip angles were normalized, and each was given a bias. The value of the bias allows the fitness function to be manipulated to provide the desired results. To normalize the vehicle turning radius and the maximum tire lateral slip angle rescaling method was used. For the normalization of the vehicle turning radius the equation can be written as;

$$\dot{r}_n = \frac{7000 - r}{1000}. \tag{4.8}$$

Where r_n is the normalized vehicle turning radius, and r is the turning radius of the vehicle. The unit of the turning radius in this equation is millimeters. The turning radius of the vehicle is expected to be between 7000 and 8000 millimeters, the normalized turning radius value changes between 0 and -1 in this region. If the vehicle turning radius is lower than 7000 millimeters, the normalized vehicle turning radius value becomes positive. The positive normalized value allows the genetic algorithm to achieve the desired fitness value with higher tire lateral slip angles. The normalized tire lateral slip angles can be calculated using the equation;

$$\alpha_{maxn} = \frac{3 - \alpha_{max}}{2} \tag{4.9}$$

Where α_{maxn} is the normalized tire lateral slip angle and α_{max} is the maximum tire lateral slip angle. The maximum tire lateral slip angle is expected to be between 3 and 5 degrees. The value of the normalized maximum tire lateral slip angle (α_{maxn}) changes between 0 and -1 while maximum tire lateral slip angle changes between 3 and 5 degrees. If the maximum tire lateral slip angle is lower than 3 degrees, the value of the normalized maximum tire lateral slip angle (α_{maxn}) becomes positive.

The fitness function can be created using the normalized vehicle turning radius r_n and normalized maximum tire lateral slip angle (α_{maxn}). The fitness function can be written as;

$$J = b_1 r_n + b_2 \alpha_{maxn} \tag{4.10}$$

Where b_1 is the bias of the normalized vehicle turning radius (r_n) and b_2 is the bias of the normalized maximum tire lateral slip angle (α_{maxn}). The sum of the values of these biases is one. The values of these biases determine how significant the corresponding parameters are to the genetic algorithm, and this means minimizing the parameters with higher bias has more reward than decreasing the parameter with low bias, by the same amount. In the Optimization Results section of this chapter, the results of different optimization processes that are conducted with different biases will be given.

4.4.4 Termination Criterion

The optimization process runs in a while loop; this means if the process is not terminated, it will go on forever. Usually, there are a few termination criteria used together for algorithm termination. Some of these methods were discussed in the theoretical background chapter. In this study, four termination criteria are used together. If any of these criteria are met, the optimization process is terminated. The termination criterion can be listed as;

- Maximum generation number
- Maximum fitness value repetition
- Maximum average fitness value
- Maximum fitness value.

In this work, the genetic algorithm is terminated if the number of generations reaches 104. Maximum fitness value repetition termination criteria is activated if the maximum fitness value does not improve for 50 generations. If the average fitness value reaches -0.1, before the specified maximum fitness number is reached, the genetic algorithm is terminated. The maximum fitness number termination criteria is activated if the maximum fitness value reaches 0.

4.5 Current Configuration Results

The original maximum wheel steering angles of the considered 8x8 vehicle was found by a trial and error process. As mentioned before, the search space of the optimization problem is vast, and finding a suitable solution using the brute force method is highly unlikely. The method used for the trial and error method is outside of the scope of this thesis. The original maximum wheel steering angles of the 8x8 vehicle are known. These maximum set of maximum wheel steering angles will be referred to as the original configuration. Using the original configuration, vehicle simulations were conducted for comparison. The maximum wheel steering angles are given in Table 4.3.

Table 4.3: Maximum wheel steering angles of the original configuration.

	Inner Tire Angle(deg)	Outer Tire Angle(deg)
1st axle	-27	-23
2nd axle	-13.7	-8.39
3rd axle	13.7	8.39
4th axle	27	23

Vehicle velocity during the steady-state cornering maneuver is kept constant at 10 km/h. The vehicle velocity change with respect to time can be seen in Figure 4.3. The path of the vehicle during the steady-state cornering maneuver was given in Figure 4.2. The wheel steering angles inputted can be seen in Figure 4.4. Tire lateral slip angles during the maneuver can be seen in Figure 4.5.

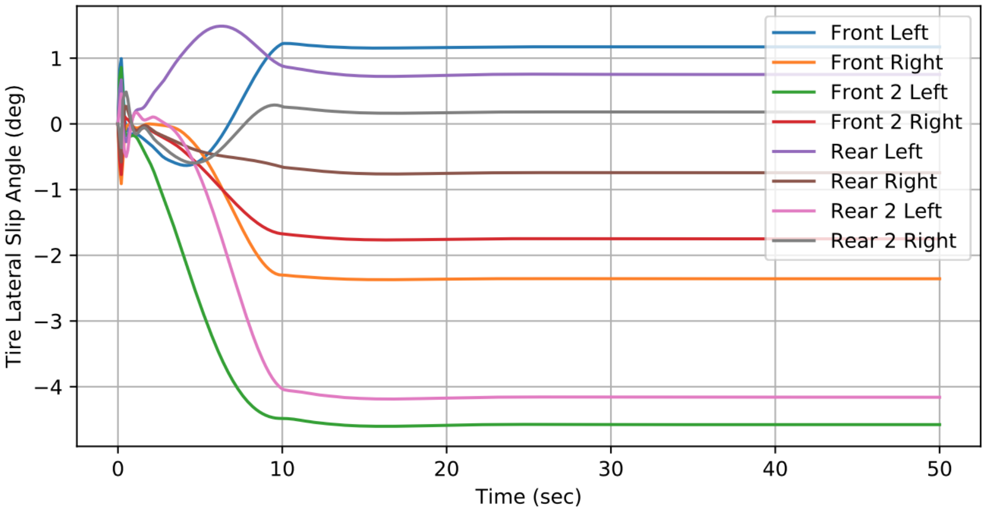


Figure 4.8: Tire lateral slip angles during the maneuver.

As seen from Figure 4.5, maximum tire lateral slip angle appears at the front left tire, and the steady-state value of the maximum slip angle is 4.6 degrees. During the steady-state portion of the maneuver, the lateral acceleration of the vehicle is 0.202 g’s. The curb-to-curb turning radius of the vehicle is 7.43 meters. At the steady-state portion of the maneuver, the vehicle roll angle is 3.66 degrees.

Table 4.4: Properties of the optimization processes.

	Individuals in a Generation	Selection Type	Pairing Type	Mating Type	Mutation Type	Fitness Biases
Run 1	20	Fittest Half	Random	Reset	Single Point	0.5 - 0.5
Run 2	40	Fittest Half	Random	Reset	Single Point	0.5 - 0.5
Run 3	20	Fittest Half	Random	Reset	Single Point	0.6 - 0.4
Run 4	20	Fittest Half	Random	Reset	Single Point	0.7 - 0.3
Run 5	20	Roulette Wheel	Weighted Random	Gauss	Single Station	0.5 - 0.5
Run 6	20	Roulette Wheel	Weighted Random	Gauss	Single Station	0.7 - 0.3

4.6 Optimization Results

In this study, a few different optimization processes were conducted to observe the effects of the fitness function, genetic operations, and generation size. There is a total of six optimization processes presented in this section. The optimization processes are referred to as runs in this section. In Table 4.4, properties of the optimization processes are given.

In Table 4.4, individuals in a generation represent the number of simulations needed to create a generation. Selection type, pairing type, mating type, and mutation type are parameters of the genetic algorithm. Moreover, fitness biases are the coefficients of the turning radius and the slip angle terms in the fitness function, respectively. As seen in Table 4.4, run 1 and run 2 are identical except for the number of individuals in a generation, these two runs will be compared to investigate the effects of the number of individuals in a generation. Run 5 and run 6 have different genetic algorithm configurations than other runs. There are three sets of fitness function biases in the optimization processes. In Figure 4.9, the maximum fitness value change with respect to the generation number for all of the runs. The maximum fitness value is the fitness value of the best individual in each generation.

It can be seen from Figure 4.9 that, all of the runs except the run 4 have reached the

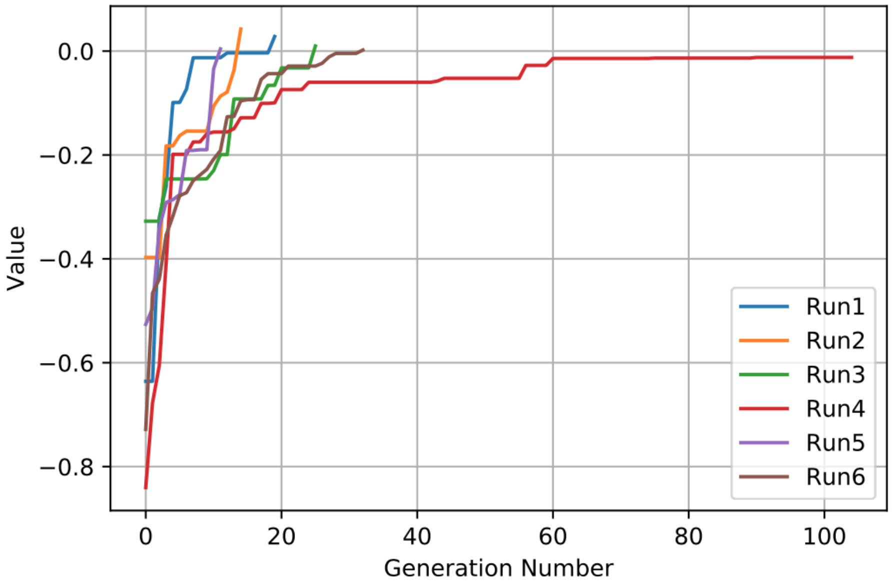


Figure 4.9: Maximum fitness change with respect to generation for all of the runs

maximum fitness limit, which in this case is 0. The run 4 did not reach the maximum fitness limit, and the genetic algorithm was terminated because it reached the maximum generation number, which is 100. The regions where the maximum fitness value is constant for a few generations are the regions where the optimization algorithm could not found a better solution, and the best solution from the previous generation becomes the best solution in the current generation due to elitism. The average fitness value change with respect to generation for all of the optimization processes can be seen in Figure 4.10.

The value of the average fitness may go up or down from generation to generation. Increase in the average fitness value indicates that a genetic algorithm is closing on an optimal solution, but rapidly increasing average fitness value may lead to getting stuck on a local optimum. Decreasing average fitness value indicates the genetic algorithm finds worst results overall, but to find the global minimum, the genetic algorithm must search the search space and some times that means the average fitness value will decrease. Usually, during a genetic algorithm optimization process, the average fitness value would increase rapidly in the first generations, and then the increase slows and finding better solutions may become harder in the later generations. To investigate the effectiveness of the genetic algorithm investigating the trend of the average fitnesses instead of the average fitness

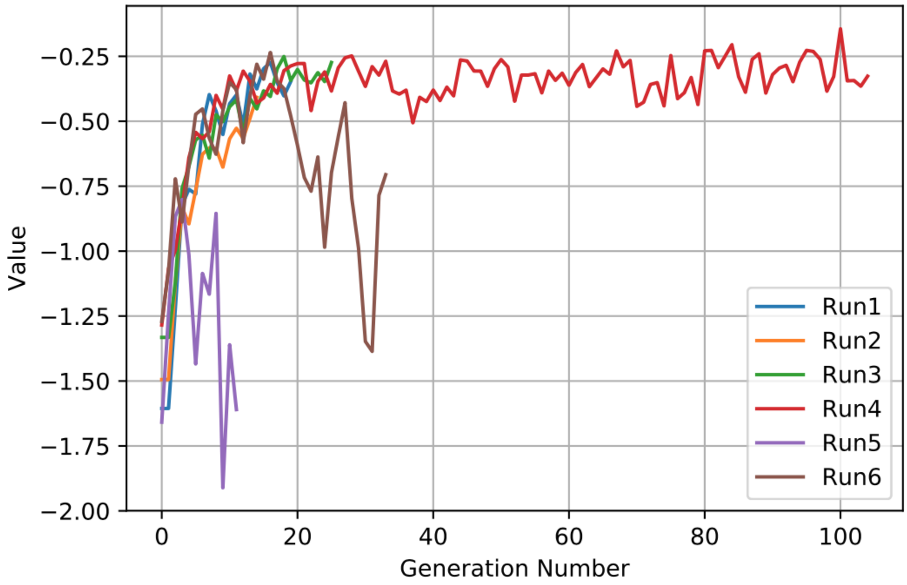


Figure 4.10: Average fitness change with respect to generation for all of the optimizations processes.

values themselves results in a better understanding of the genetic algorithm. From Figure 4.10, it can be seen that run 1, run 2, run3, and run 4 have similar behavior, and the trend of the average fitness values is increasing. Run 5 and run 6 have similar behavior and decreasing trends. The difference between the trends is a result of the genetic algorithm settings. From Table 4.4, it can be seen that the genetic algorithm settings for run 5 and run 6 are different from the other runs.

To investigate the effect of the number of individuals in a generation on the effectiveness of the genetic algorithm, run 1 and run 2 is compared. As seen in Table 4.4, the only difference between run 1 and run 2 is the number of individuals in a generation. In Figure 4.11, the maximum fitness values and average fitness values of run 1 and run 2 are given as well as the regression lines of the average fitness values. The regression line describes the trend of average fitness values.

As seen from Figure 4.11, both run 1 and run 2 have reached the maximum fitness limit. The optimization process with a higher number of individuals in each generation (Run2) satisfied the termination criterion faster. The number of simulations done in both optimization processes can be calculated as;

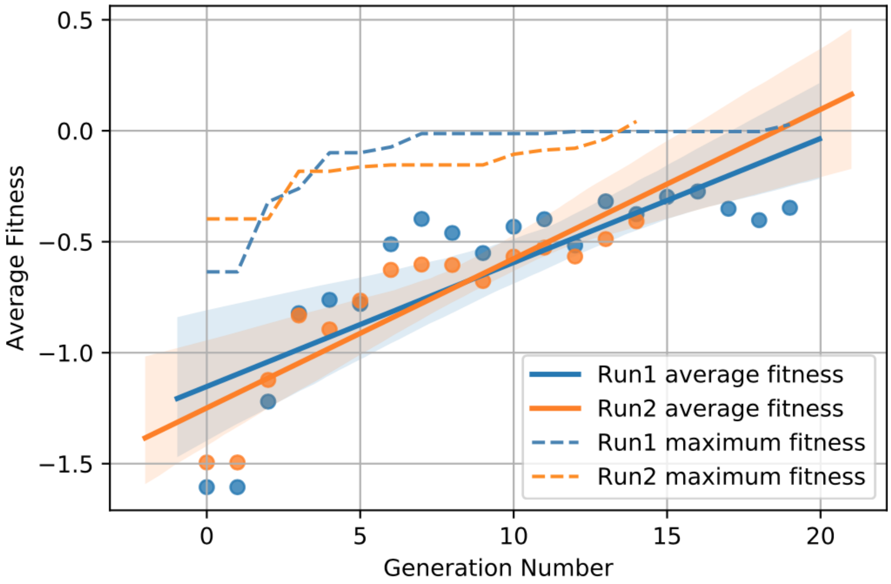


Figure 4.11: Comparison of Run 1 and Run 2.

$$n_{run1} = 20 + (20 - 1) * 19 = 381 \tag{4.11}$$

and

$$n_{run2} = 40 + (40 - 1) * 14 = 586. \tag{4.12}$$

So the number of generations created to reach to the termination criteria in run 2 is lower than run 1, but due to the size of the populations, the number of simulations done for run 1 is less than run 2.

To observe the effect of different genetic optimization settings, run 1 and run 5 were compared. Run 1 and run 5 has the same number of individuals in a generation, and the biases of the fitness function are the same; however, the genetic algorithm settings are different. Maximum and average fitness value change with respect to the generation number as well as the regression lines of the average fitnesses can be seen in Figure 4.12.

The average fitness value of run 5 has a downward trend, but the algorithm finds an optimal

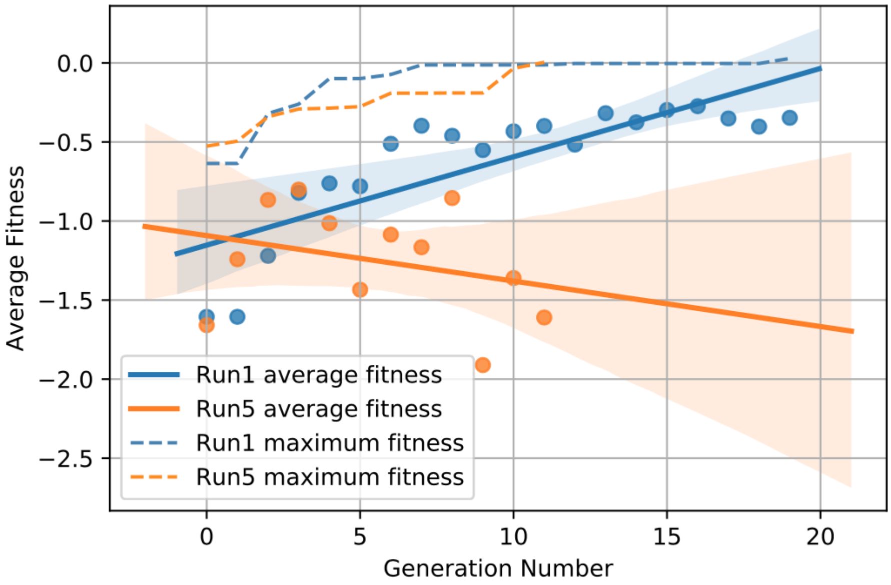


Figure 4.12: Comparison of Run 1 and Run 5.

solution quicker than run 1. Genetic algorithm settings of run 5 allow the algorithm to search the search space more efficiently, while run 1 got stuck on a local optimum before finding the optimal solution while run 5 continuous to search the search space and find the optimal solution. Due to elitism introduced to the genetic algorithm, a downward trend of the average fitness does not affect the maximum fitness value. The same phenomenon can be observed when run 4 and run 6 are considered. The comparison between run 4 and run 6 can be seen in Figure 4.13.

Run 4 and run 6 has the same fitness function biases, so the objective of the two optimization problem is the same. However, the two runs have different genetic algorithm settings like run 1 and run 5. Figure 4.13 supports the conclusion that the genetic algorithm with run 5 and run 6's settings search the search space more efficiently and finds the optimal solution faster than other genetic algorithm settings.

The biases in the fitness function of the genetic algorithm determine the objective of the optimization process. To observe the effect of the biases in the fitness function run 1, run 3, and run 4 can be compared. The maximum and average fitness value changes for these runs can be seen in Figure 4.14.

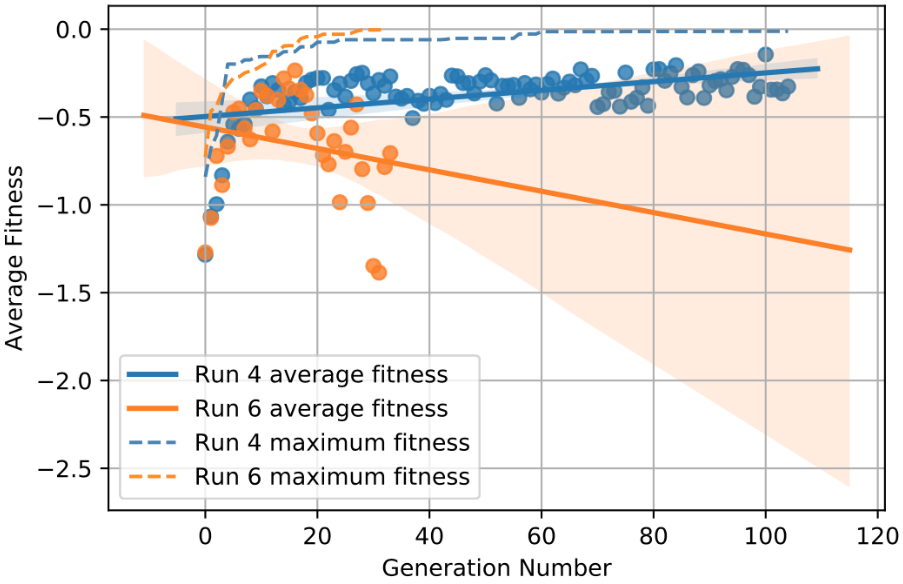


Figure 4.13: Comparison of Run4 and Run6.

The biases in the fitness function allow manipulating the genetic algorithm to obtain desired results. The results of the optimization runs are given in Table 4.5.

Table 4.5: Optimization results.

	Tuning Radius (mm)	Maximum Tire Lateral Slip Angle (deg)	Lateral Acceleration (g)	Vehicle Roll Angle (deg)
Original Configuration	7432.84	4.6	0.202	3.658
Run 1	7621.46	1.38	0.194	3.491
Run 2	7477.91	1.87	0.203	3.777
Run 3	7398.53	1.75	0.207	3.862
Run 4	7108.9	2.51	0.216	3.96
Run 5	7236.82	2.5	0.207	3.721
Run 6	6940.16	3.26	0.227	4.259

Table 4.5 shows that the runs with higher turning radii bias have the lower turning radiuses like run 4 and run 6. While the runs that have the same biases, have higher turning radiuses compared to the other optimization results but have lower tire lateral slip angles like Run1.

The optimized maximum wheel steering angles can be seen in Table 4.6.

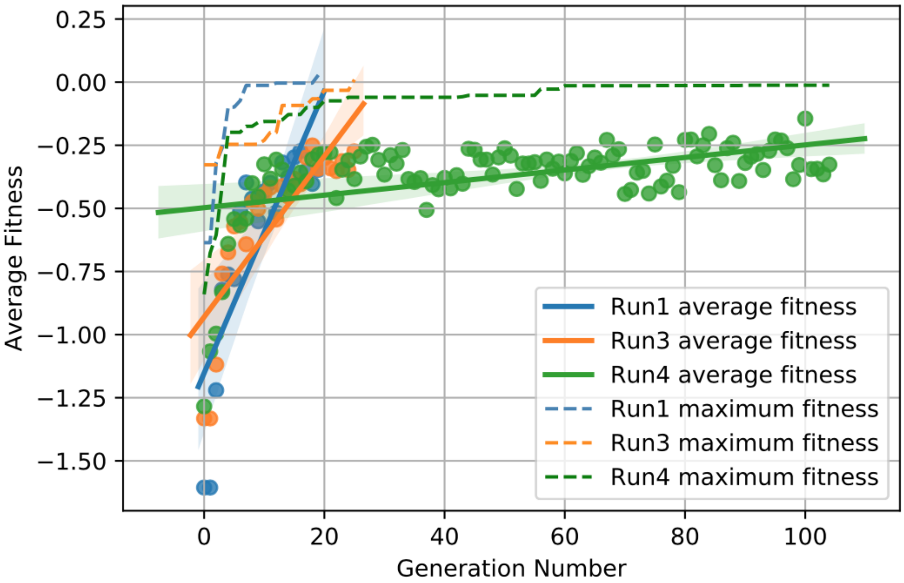


Figure 4.14: Comparison of Run1, Run3 and Run4.

Table 4.6: Maximum wheel steering angles.

	1st axle (deg)		2nd axle (deg)		3rd axle (deg)		4th axle (deg)	
	Left	Right	Left	Right	Left	Right	Left	Right
Run 1	-27.6	-20.9	-9.4	-7	11.9	7.9	28.8	22.2
Run 2	-29.8	-29.2	-10.6	-8.9	12.3	7.8	29.2	20.5
Run 3	-29.9	-23	-11.9	-9.2	11.9	6.8	29.5	21.2
Run 4	-30.0	-24.7	-13	-10.2	10.7	7.1	29.9	24.7
Run 5	-30	-23.3	-10.3	-8.1	12.2	11.8	30	21.4
Run 6	-30	-26.6	-15.5	-11.7	11,2	6.3	30	25.9

To visualize the results, the path of vehicle CoG during the best results of the optimization processes can be seen in Figure 4.15.

Maximum tire lateral slip angle change with respect to time for all of the runs can be seen in Figure 4.16.

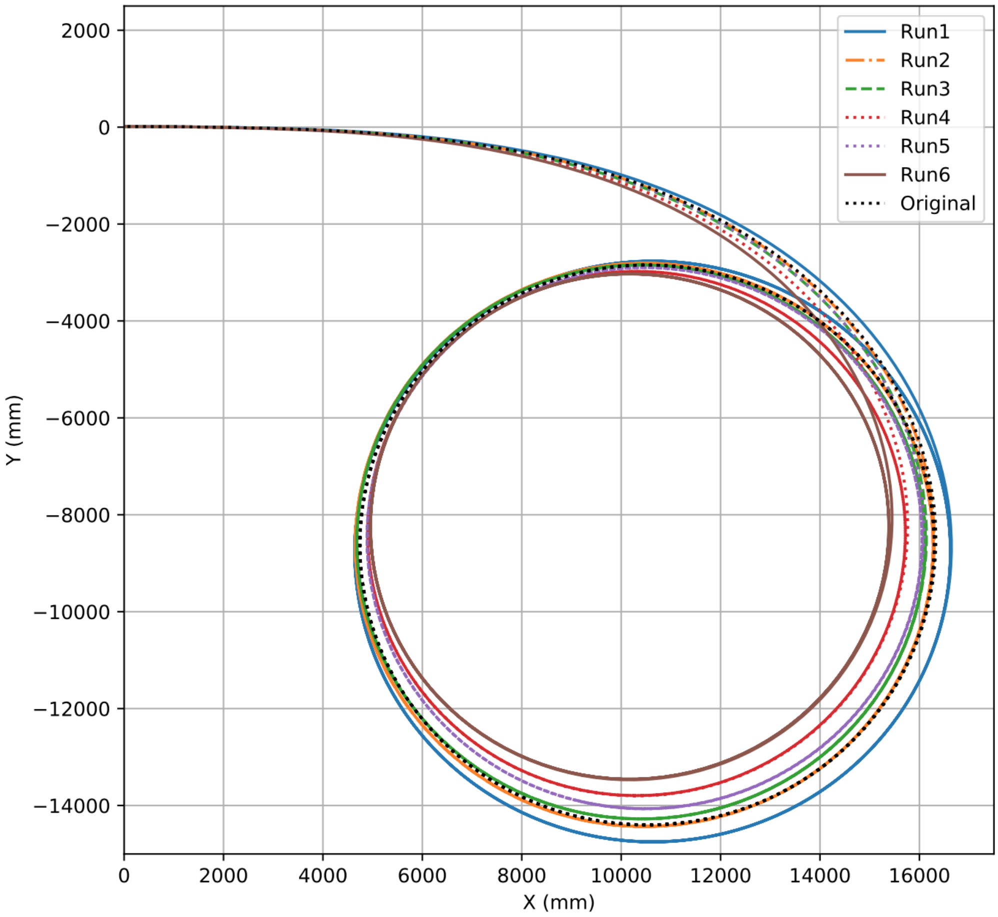


Figure 4.15: Vehicle paths during the steady-state cornering maneuver.

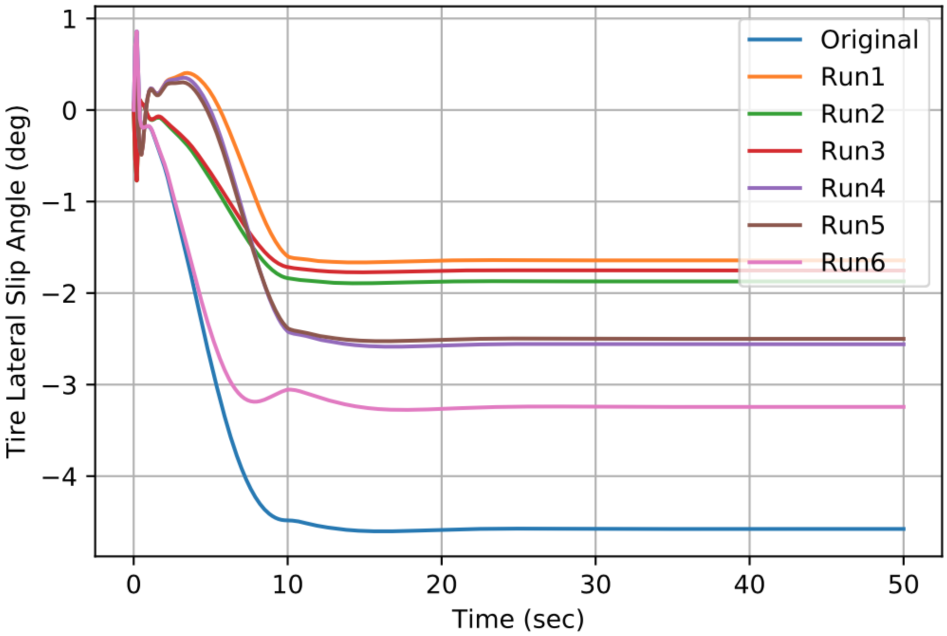


Figure 4.16: Maximum slip angles of all the runs.

From Figure 4.16, it can be seen that all of the maximum slip angles of the optimization processes are considerably lower than the maximum slip angle of the original configuration. The optimization process results will be discussed in greater detail in the discussion and conclusion chapter.

4.7 Conclusion

In this chapter, the method used for the turning radius optimization study has been given, as well as the study results. The structure and use case of the python code used for the optimization process was discussed. Use of Adams command file used to run the simulation was also discussed.

For the turning radius optimization study, six different optimization processes were conducted. These optimization processes are referred to as runs. The properties of the fittest individuals in the last generations of these runs were given in Table 4.5. Except for run 4, the individuals given in Table 4.5 are the individuals that reached the maximum fitness value of zero and caused the genetic algorithms to terminate. For run 4, the genetic algorithm got stuck on local minima and was terminated because the genetic algorithm reached the maximum allowed generation number. The effect of the number of individuals was investigated by comparing run 1 and run 2, 20 individuals in each generation were more efficient than 40 individuals in each generation in terms of the number of generations created before the genetic algorithm was terminated. The effects of genetic algorithm settings were also investigated, run 5 and run 6 have the same genetic algorithm settings and the average fitness values of these runs have a decreasing trend, however due to the searching capabilities and elitism introduced to the genetic algorithm both runs find the optimal solution quicker than other runs that have the same fitness biases. Runs 1, 2, 3 and 4 have the same genetic algorithm settings, and it was observed that for all of these runs the trend of the average fitness values is increasing, meaning the genetic algorithm is improving even if the maximum fitness does not change. Runs with this genetic algorithm settings quickly converge around an optimum. For runs 1, 2 and 3 this was not a problem as the local minima they initially converged was sufficient enough to trigger the termination criteria, however run 4 got stuck in a local optima and was not able to find a solution that was good enough to trigger a termination criteria before the genetic algorithm reached the maximum generation limit. Lastly, the effect of fitness function biases was investigated. As expected, the runs with higher turning radius bias produced results with lower turning radiuses and runs with higher slip angle bias produced results with lower tire lateral slip angles.

All of the results obtained from the optimization processes yield lower tire lateral slip angles than the original configuration. The only run 1 and run 3 have higher turning radiuses than the original configuration, but these runs yield much lower tire lateral slip angles than the original configuration. These results show that the proposed method consistently finds optimal results for the given properties. As shown in table 4.5, the results obtained depend on the inputted fitness function biases. Using this method, maximum wheel steering angles that cause desired vehicle behavior can be obtained. In this chapter, the optimal maximum wheel steering angles for different objectives were obtained. In the next chapter, the high speed handling characteristics of these wheel steering angle configurations will be investigated.

5 NATO LANE CHANGE SIMULATIONS

5.1 Introduction

This chapter discusses the methodology, and the results of the Nato Lane Change simulations conducted using the wheel steering angle configurations obtained in the turning radius optimization section. The DLC study aims to investigate the stability and high speed handling characteristics of the vehicle with the maximum wheel steering angle configurations. A general function that represents the steering input for any DLC maneuver was derived. The function is a trigonometric function with parameters that determines the behavior of the input signal. Parameters of this function must be optimized for the specific maneuver, vehicle, and longitudinal vehicle velocity. In this study, the same multi-body dynamics model of the vehicle that was used for the turning radius optimization was used in this part as well. The maximum steering angle sets that were obtained by turning radius optimization study were used to create steering angle constraints between the wheels. Each axle is assumed to achieve perfect Ackermann steering with no steering error. The DLC maneuver considered is a specific maneuver called Nato Lane Change maneuver. The optimization process is very similar to the optimization process of the turning radius optimization discussed in the last chapter. The genetic algorithm used in the turning radius optimization process was revised for the lane change maneuver.

The chapter begins with discussing the derivation of the general steering input for the DLC maneuver. The vehicle model used for this study is the same as the vehicle model used in the turning radius optimization study. In the Turning Radius Optimization chapter, the full vehicle model and its subsystems were discussed. The DLC maneuver used in this study is called Nato Lane Change maneuver. Details of this maneuver are discussed in the corresponding section. Adams command file used in this study is very similar to the Adams command file used in the turning radius optimization study. In this chapter, the inputs and outputs of the Adams command file will be discussed without going into detail. The Python code used for the DLC study is more complicated than the Python code used for the turning radius optimization study, but the basic principles are similar.

5.2 General Open Loop Steering Input For Lane Change Maneuvers

In this work lane change maneuver for a multi-axle vehicle is conducted using an open loop control. The steering inputs of the tires are calculated in advance of the simulation and given to Adams software simulation to conduct the lane change maneuver. The general steering input for the lane change maneuver can be found using the formula:

$$\begin{aligned}\delta(t) = (a1 * cos(\pi * tanh(t - b1)) - c1 * cos(\pi * tanh(t - d1)) \\ + (a1 - c1)) - (a2 * cos(\pi * tanh(t - b2)) \\ - c2 * cos(\pi * tanh(t - d2)) + (a2 - c2))\end{aligned} \tag{5.1}$$

Where a_1, b_1, c_1, d_1, a_2, b_2, c_2 and d_2 are the parameters which changes the shape of the input signal, t is time. A general shape of the input signal is shown in Figure 5.1.

This equation can be separated into two parts for a better explanation. The first part of the equation is;

$$\delta_1(t) = (a1 * cos(\pi * tanh(t - b1)) - c1 * cos(\pi * tanh(t - d1)) + (a1 - c1)) \tag{5.2}$$

For the first part of this equation, a_1 parameter corresponds to the amplitude of the first peak, b_1 parameter determines when the first peak will appear, c_1 corresponds to the magnitude of the first drop, d_1 determines when the peak of the drop will occur. The first part of this equation is responsible for the single lane change. This function can be used alone to simulate single-lane change maneuvers.

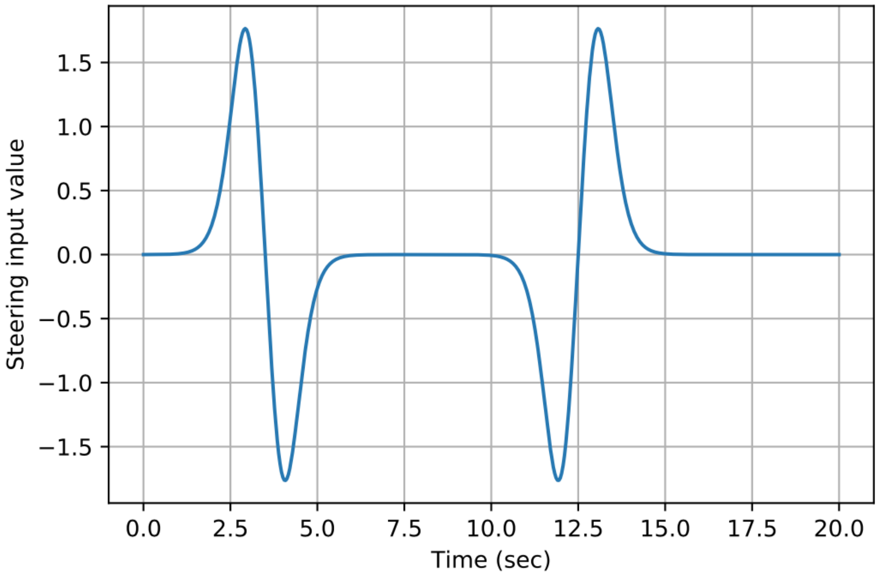


Figure 5.1: Steering angle input for the lane change maneuver

The second part of the equation is;

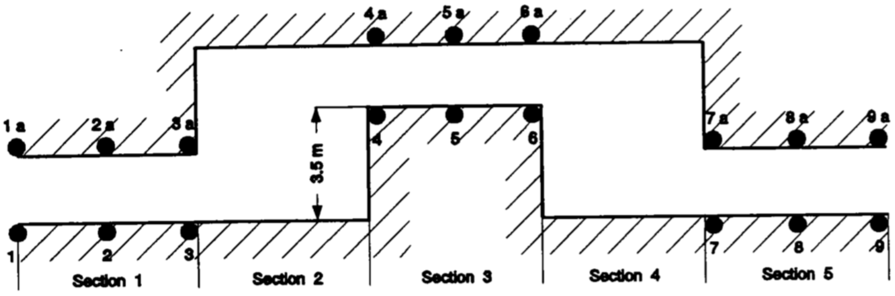
$$\delta_2(t) = -(a2 * cos(\pi * tanh(t - b2)) - c2 * cos(\pi * tanh(t - d2)) + (a2 - c2)) \quad (5.3)$$

The second part of the equation is the same as the first part in terms of its construction, and the main difference is the negative sign in the second part of the equation. The negative sign ensures the second part of the equation to behave opposite of the first equation. This behavior allows the vehicle to change the lane in the opposite direction of the first part of the equation and return to its original lane completing the DLC maneuver. For the second part of the equation, a_2 parameter corresponds to the amplitude of the first peak, b_2 parameter determines when the first peak will appear, c_2 corresponds to the magnitude of the first drop, d_2 determines when the peak of the drop will occur.

By changing the timing and the amplitudes of the peaks and drops in this equation, an infinite number of DLC maneuvers can be obtained, as long as the vehicle dynamics allows the maneuver. The same conclusion can be made for single lane change maneuvers as well using the first part of the equation. This equation can be used for every wheeled vehicle independent of the number of axles and the number of steerable axles.

5.3 Nato Lane Change Maneuver

In this study, Nato Lane Change maneuver [2] was simulated to investigate the high-speed handling and dynamic stability of the vehicle. The vehicle goes through a narrow path that has two lane change events, the dimensions of the road changes according to the dimensions of the vehicle. The longitudinal vehicle velocity is kept constant during the maneuver. In this work, the vehicle speed is kept steady at 80 km/h. The test track is described as; flat road with a slope of less than 2 percent, good grip with friction coefficient higher than 0.7 and uniform paved road. For the real world test, devices with supporting wheels on each side of the vehicle are required to be mounted to the vehicle. During this maneuver, the vehicle velocity must be kept as steady as possible. The road, the vehicle must follow during NATO Lane Change maneuver is given in Figure 5.2.



Lane – change track dimensions

Section 1 : Length = 15 m
Width = 1.1 . vehicle width + 0.25 m
Section 2 : Length = Overall length of vehicle*) + 24 m
Section 3 : Length = 25 m
Width = 1.2 vehicle width + 0.25 m
Section 4 : Length = Overall length of vehicle + 24 m
Section 5 : Length = 15 m
Width = 1.1 . vehicle width + 0.25 m

*) Overall length of vehicle, measured at 0.50 m from the ground.

Figure 5.2: Nato lane change maneuver specifications. [2]

The Python code determines the dimensions of the road. After the Python code is imported, the Python code measures the dimensions of the vehicle and calculates the dimensions of the road.

5.4 Adams Command File

The Adams command file used in this study is very similar to the Adams command file used in the turning radius optimization study. The Adams command file used in this study also runs the simulations and outputs the desired results. In this study, the outputs that are used in the fitness calculation are the positions of the markers that are located at the four corners of the vehicle. The Adams command file creates four text files for each of the corners. These text files contain the x and y locations of the markers.

Adams command file calls an Adams event file created for this study. In this event file, the duration of the simulation, the time step of the simulation, the longitudinal velocity of the vehicle and controller objectives are given. The controller in this study is trying to keep vehicle longitudinal velocity as steady as possible. There are no other objectives, the gear ratio is constant, and there is no steering.

5.5 Python Code

The optimization process starts with importing the Python file into Adams. The Python code starts by evaluating the vehicle dimensions. The vehicle dimensions are used to create the DLC path the vehicle will follow; also, the functions used to calculate the wheel steering angles uses the vehicle dimensions as well. The maximum steering angles of the tire are inputted to the Python code. The maximum wheel steering angles are also used in wheel steering angle calculation functions.

In this study, each tire is steered individually. It is assumed that the tires are steered according to Ackermann steering in each axle, without steering error, this means the two lines drown normal to the tires in each axle will cross each other somewhere in a straight line. In the figure above the normal lines from the front tires of the vehicle crosses each other at the straight line at the third axle. An example of the Ackermann steering for one axle is given in Figure 5.3.

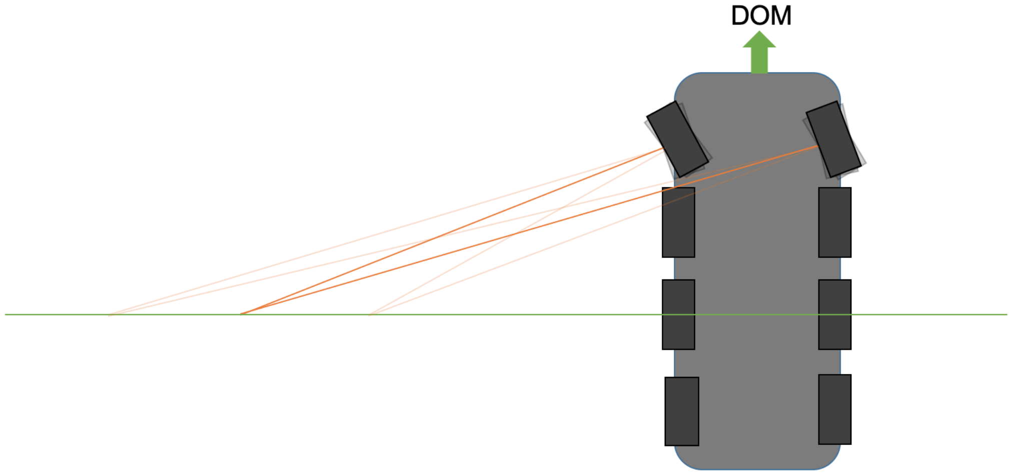


Figure 5.3: Ackermann steering example for one axle.

Vehicles have steering errors due to the mechanism used in the steering system. When designing a mechanism for the steering systems, the goal is to minimize the steering error and correctly follow Ackermann steering. The intended use case for this method is early in the design process, where the maximum steering angles of the tries are known, but the steering system is not designed yet. Using this method maximum wheel steering angle configurations can be examined without having to develop a steering system mechanism for all of the configurations. The Python code that was used in the DLC study is given in Appendix C.

5.5.1 Genetic Algorithm

The genetic algorithm used for the DLC maneuver is created by modifying the genetic algorithm used for the turning radius optimization study. The aim of the genetic algorithm used for the DLC study is to find the set of parameters of the general DLC steering input function, which will cause the vehicle to satisfies the Nato Lane Change maneuver. There are eight parameters in the general DLC steering input function, but the parameter that decides the start of the maneuver does not change. So there are seven parameters to be optimized.

Creating the individuals and the mutation genetic operations are where the genetic algorithm used in this study differs from the genetic algorithm used in the turning radius optimization section. Bout of these parts are problem specific, and constraints of the problem must be applied individually. Other parts of the genetic algorithm were created to be used with any problem and can be directly applied to this problem.

5.5.2 Fitness Function

The steering input function for the front left tire created by the general steering input function is inputted to the fitness function. Other wheel steering angles are calculated by the fitness function and then inputted to the vehicle model in Adams. Simulation is then run by calling upon the Adams command file. Adams command file outputs the x and y coordinates of the markers located at the corners of the vehicle as text files. The Python code reads the text files containing the coordinates and created Python objects containing those coordinates. The workflow of the fitness function is given in Figure 5.4.

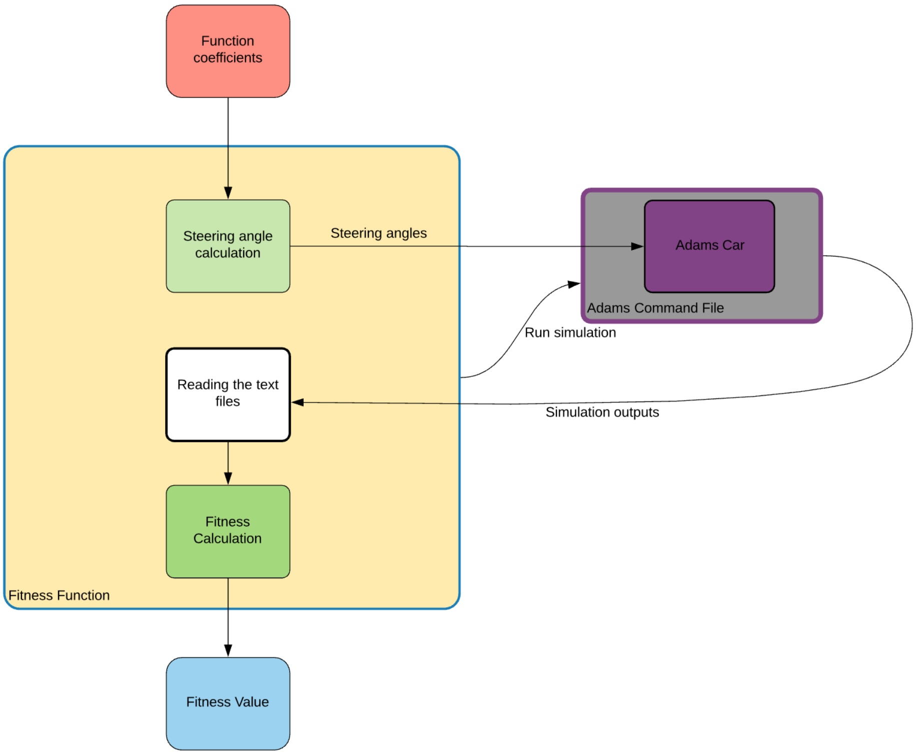


Figure 5.4: Fitness function flow chart for the DLC maneuver.

The steering angle of the front left tire is calculated using the general DLC steering input function, which was discussed earlier in this chapter. For an axle that satisfies the Ackermann steering, outer wheel steering angle for a given inner wheel steering angle can be calculated as;

$$\delta_o = arctan(\frac{1}{\frac{w}{l} + \frac{1}{tan(\delta_i)}}) \tag{5.4}$$

Where δ_o is the outer wheel steering angle, δ_i is the steering angle of the inner tire, w is the vehicle width, and l_i is the vertical distance between the axle and the line which the normal lines from the tires cross, where i is the axle number. For a regular four-wheeled vehicle with Ackermann steering, l is the distance between the front and the rear axles. For an axle which the maximum inner and outer wheel steering angles are known l_i can be calculated as;

$$l_i = \frac{w}{\frac{1}{tan(\delta_{omax})} - \frac{1}{tan(\delta_{imax})}}. \tag{5.5}$$

Where δ_{omax} is the maximum angle of the outer tire and δ_{imax} is the maximum angle of the inner tire. Using the functions given in above steering angle inputs for the front four tires are calculated. After the Python code calculates the wheel steering angles, they are inputted to the Adams model. Simulation can be conducted after this process. To perform the simulation Python code calls the Adams command file and reads the output text files. Python code creates Python objects from the simulation results.

The cost function can then be calculated using the Python objects containing the coordinates of the corner markers. The function takes one corner at a time and looks at the y coordinates according to the x coordinates if the y coordinate value is not in the specified range for that x coordinate than the function increases the cost. After the cost of all the corners are calculated the functions adds the cost and outputs the negative of the cost. This way if the cost is zero, that means the vehicle completed the maneuver without crossing the road limits. The lower the cost of an individual, the worst the solution is.

5.5.3 Termination Criteria

The objective of this study differs from the turning radius optimization study, in the turning radius optimization study the aim is to find an optimal solution, but in this study, the objective is to find a solution that satisfies the criteria of the maneuver. Three of the four termination criterion used in the turning radius study can be used for this study. Which are;

- Maximum generation number
- Maximum fitness value repetition
- Maximum fitness value.

Maximum average fitness value as termination criteria cannot be used in this situation as we are not interested in the conversion of the genetic algorithm, we are only interested in finding a specific solution. The maximum fitness termination criteria ensure that when the desired solution is reached the genetic algorithm stops. Limiting the maximum number of generations and maximum fitness repetition are safeguards against the while loop.

5.6 DLC Study Results

In this section, results of the DLC studies are given. There are three DLC studies conducted. The first DLC study conducted is for the vehicle with the original maximum wheel steering angle configuration. For the second DLC study, run 1 configuration was used. The run 1 configuration provides the lowest tire lateral slips and highest turning radius. In the last DLC study conducted, the vehicle with run 6 configurations were used. The vehicle with run 6 configuration achieved the lowest turning radius and second highest tire lateral slip angles. In the vehicle design phase, either the turning radius is going to be prioritized, or the tire lateral slip angles are going to be prioritized for minimal tire wear. Run 1 and run 6 provides these priorities, and one of them is likely to be chosen.

The genetic algorithm configurations used for the three different DLC studies are the same and can be seen in Table 4.1.

Table 5.1: DLC genetic algorithm configurations.

	Original Configuration	Run 1 Configuration	Run 2 Configuration
Population Size	20	20	20
Selection Type	Fittest Half	Fittest Half	Fittest Half
Pairing Type	Random	Random	Random
Crossover Type	Single Point	Single Point	Single Point
Mutation Type	Reset	Reset	Reset

For the original configuration, the optimization process can be summarized with the maximum and average fitnesses with regression line given in Figure 5.5.

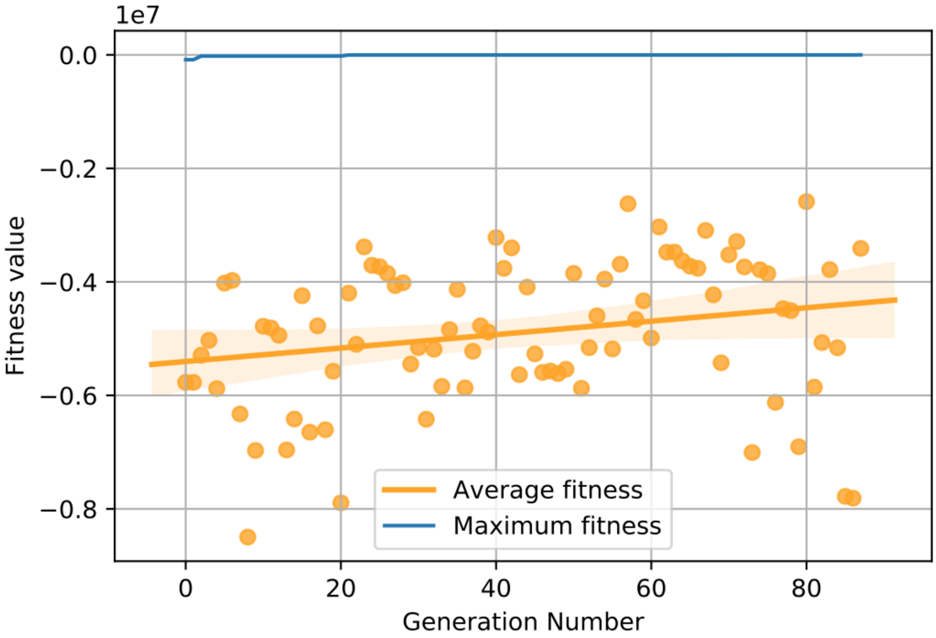


Figure 5.5: Maximum and average fitness of the original configuration optimization process.

As discussed earlier, if the fitness value reaches zero, that means the vehicle is moving through the road without crossing the borders. From Figure 5.5, it can be seen that the fitness value reached zero at the last generation, and that triggered the termination criteria. The vehicle path during the DLC maneuver can be seen in Figure 5.6.

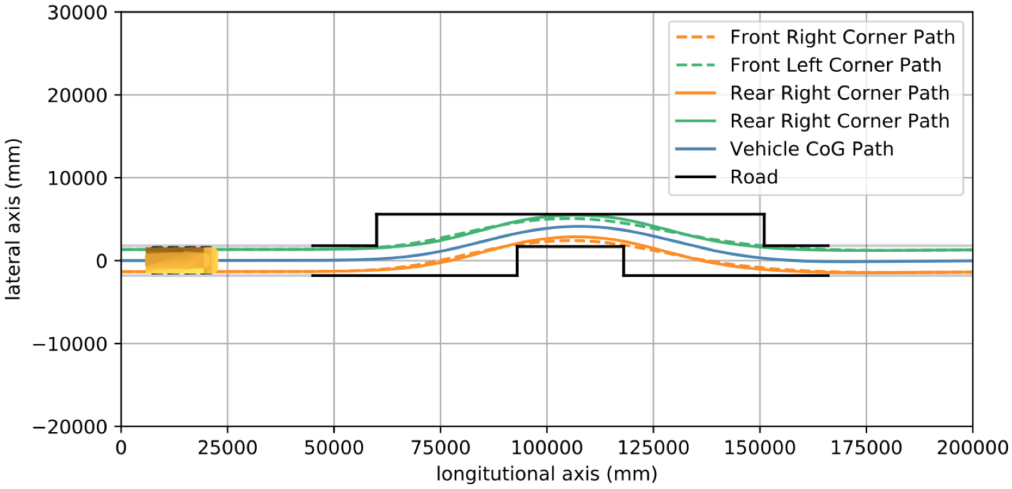


Figure 5.6: Path of the vehicle with the original steering angle configuration.

The DLC study with the run 1 maximum steering angle configuration is discussed next. The maximum and average fitnesses of the genetic algorithm, as well as the regression line of the average fitness values, is given in Figure 5.7.

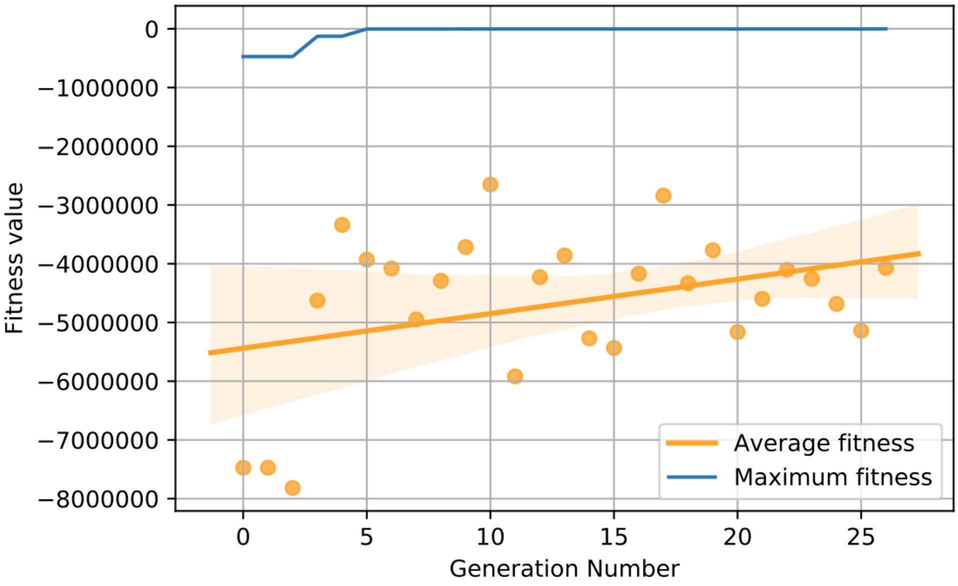


Figure 5.7: Maximum and average fitness of the run 1 configuration

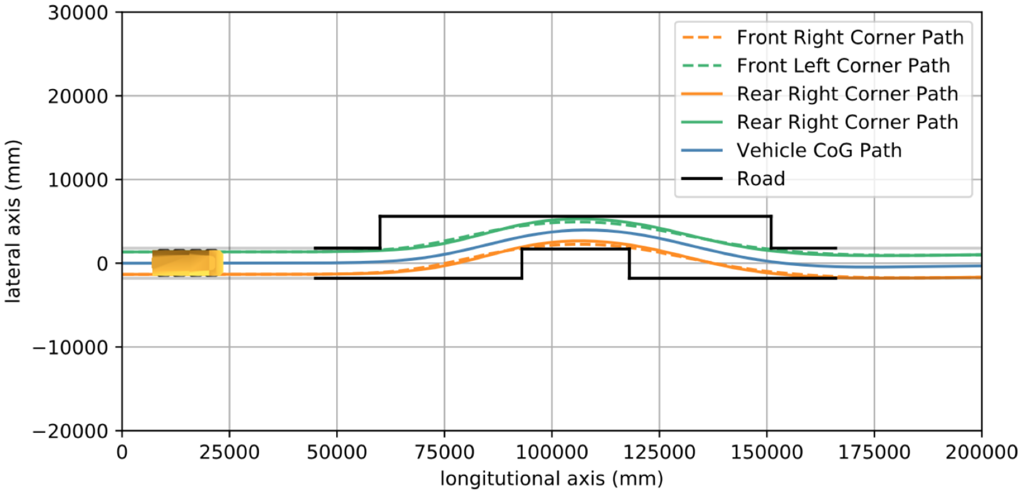


Figure 5.8: Path of the vehicle with the run 1 steering angle configuration.

From Figure 5.7, it can be seen that the genetic algorithm was terminated because the maximum fitness value reached zero. With the best solution found in the last generation of the genetic algorithm, the vehicle completed the DLC maneuver without crossing the road boundaries with the run 1 steering angle configuration. The path of the vehicle during this maneuver can be seen in Figure 5.8.

The maximum and average fitness values of the genetic algorithm used for the vehicle with run 6 configuration can be seen in Figure 5.9.

The path of the vehicle CoG, as well as the paths of the four corners of the vehicle with the run 6 steering angle configuration, can be seen in Figure 5.10.

Although the vehicle has completed the maneuver with all three of the steering angle configurations, the inputs and the vehicle behavior differ from configuration to configuration. The torque the driver must apply to the steering wheel can not be calculated without the steering system and hydraulic actuator models. However, how much does the driver has to turn the steering wheel to complete the maneuver successfully can be calculated with some assumptions. To compare the steering inputs necessary to complete the maneuver, the front left wheel steering input can be compared. However, all of the maximum steering angles are different from each other for the different steering angle configurations. So to compare the steering input the steering angle of the front left tire is divided by the

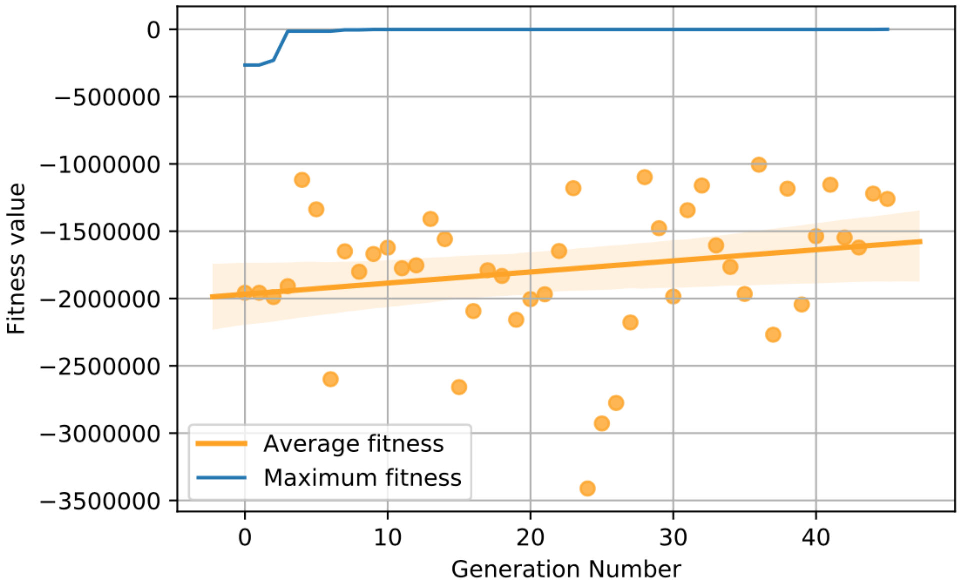


Figure 5.9: Maximum and average fitness of the run 6 configuration

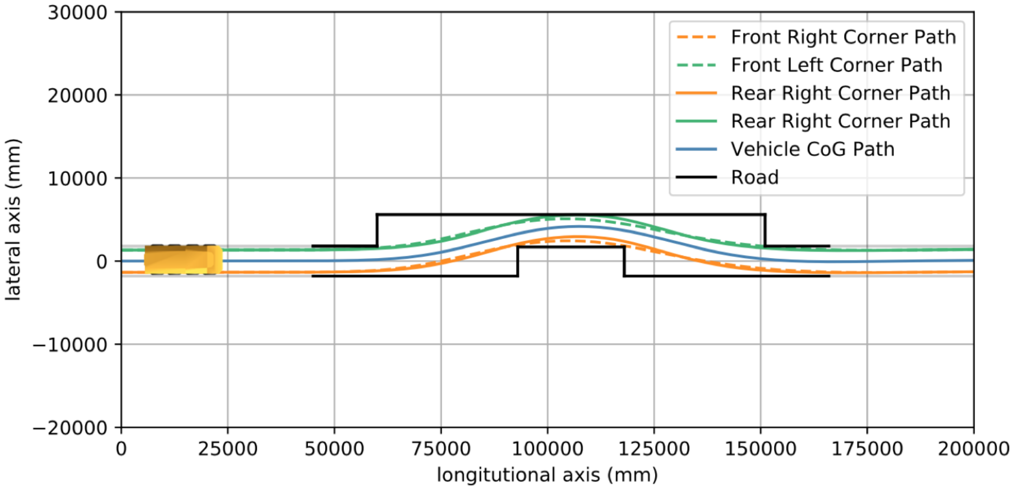


Figure 5.10: Path of the vehicle with the run 6 steering angle configuration.

maximum steering angle of the corresponding configuration. Using this method and the assumption that the wheel steering angle and steering wheel angle changes linearly, the percentage of the maximum steering wheel angle the driver must input to the system can be obtained. The driver input for the vehicle with different configurations can be seen in Figure 5.11.

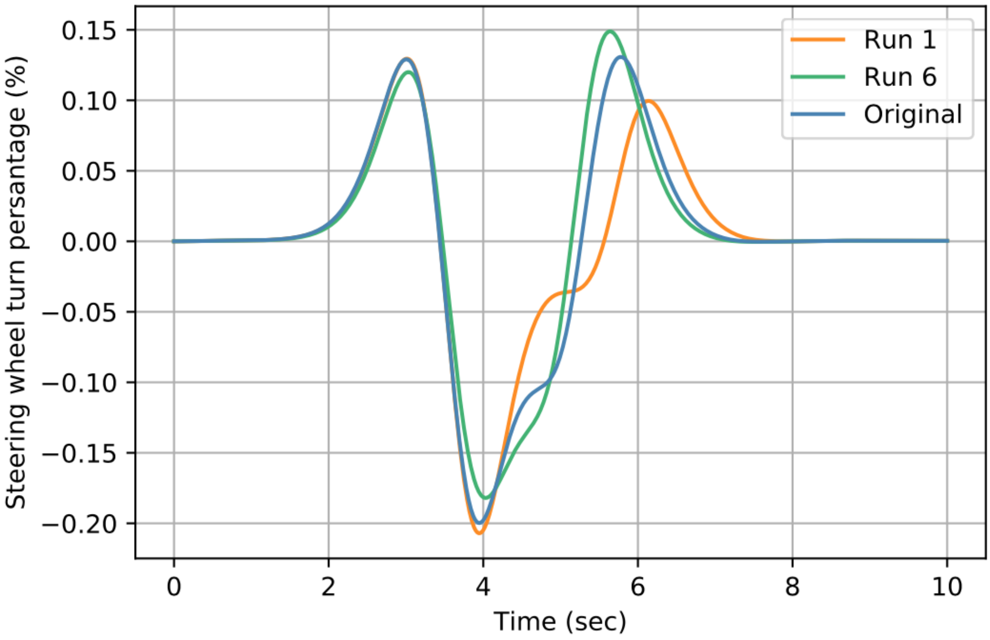


Figure 5.11: Steering wheel turn percentage for the vehicle with different configurations.

As discussed previously in this chapter, the velocity of the vehicle during the DLC maneuver must be kept as steady as possible. The vehicle longitudinal velocity of the vehicles with three different steering angle configurations during the DLC maneuver can be seen in Figure 5.12.

The Nato Lane Change maneuver specifies that error of less than one percent in the vehicle longitudinal velocity. As seen from Figure 5.12, vehicle velocity with all of the configurations are in the allowable range. The longitudinal velocity controller inputs throttle and brake to the systems to keep the velocity study. The controller does not input and brake demand in this study. Investigating the throttle input to the system can give insight into the system behavior. In Figure 5.13, throttle demands for the different

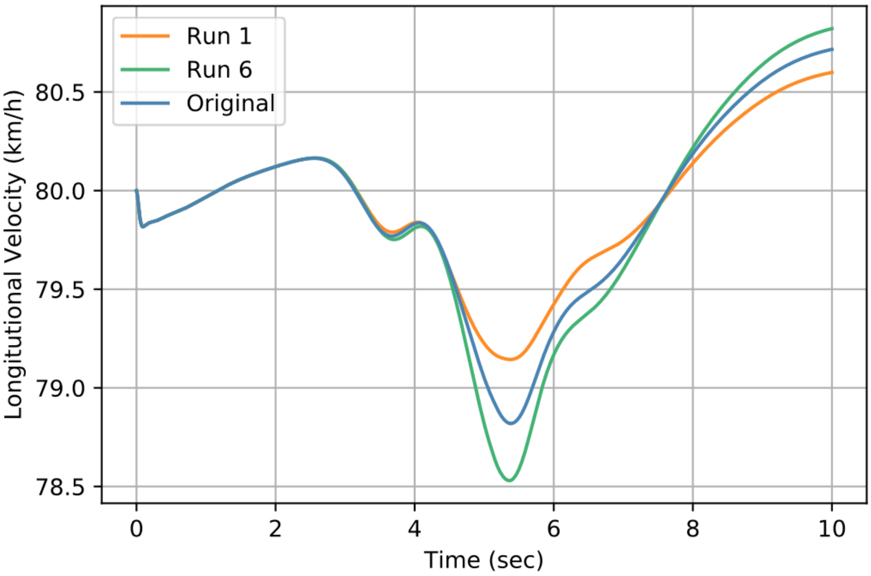


Figure 5.12: Velocities of the vehicle with different steering angle configurations during the DLC maneuver.

configurations can be seen.

As seen from Figure 5.13, the vehicle with run 1 configuration had the lowest throttle demand during the DLC maneuver. Although the different configurations of the same vehicle are simulated in the same road with very similar longitudinal velocities, the vehicle behavior during the maneuver is very different. Because there are two steerable axles in the vehicle, the handling behavior is more complicated than that of a vehicle with only one steerable axle. In this case, the relationship between the first and second axle steering angles affects the vehicle's tendency of yaw gain and lateral acceleration gain. The lateral accelerations for the vehicle with different steering angle configurations can be seen in Figure 5.13.

The yaw rates of the vehicle with different steering angle configurations can be seen in Figure 5.14.

Vehicle roll angle during the maneuver for all of the configurations can be seen in Figure 5.15.

The Nato Lane Change maneuver conducted at 80 km/h for an 8x8 vehicle is a very

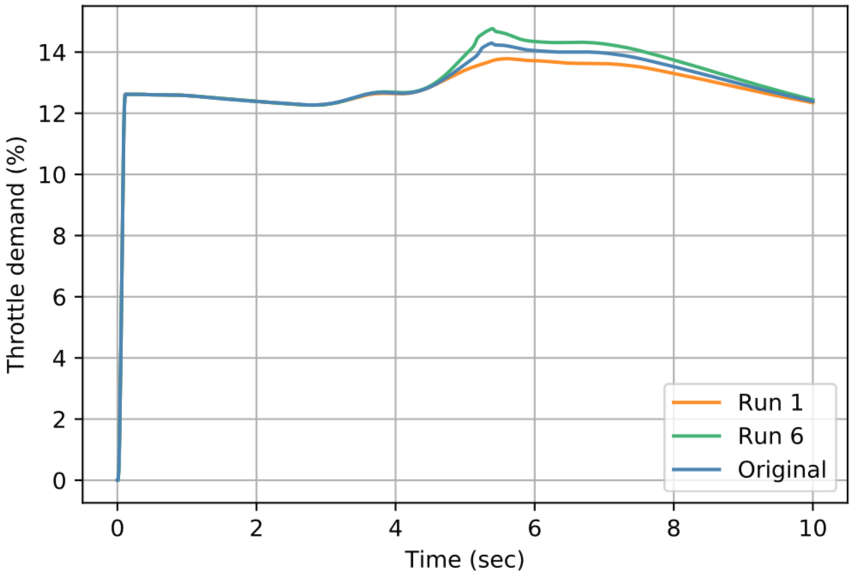


Figure 5.13: Throttle demands for all of the configurations.

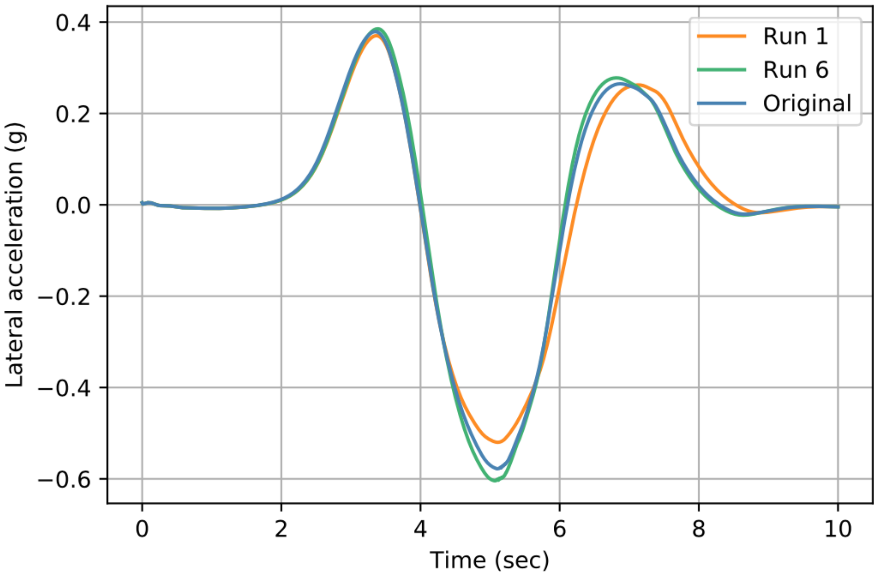


Figure 5.14: Lateral acceleration change with respect to time.

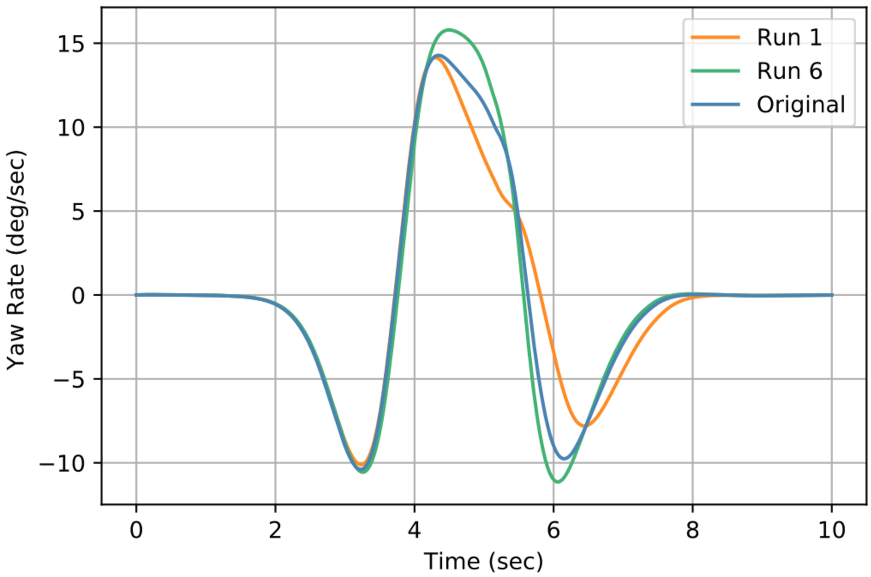


Figure 5.15: Yaw rates of the vehicle with different steering angle configurations during the DLC maneuver.

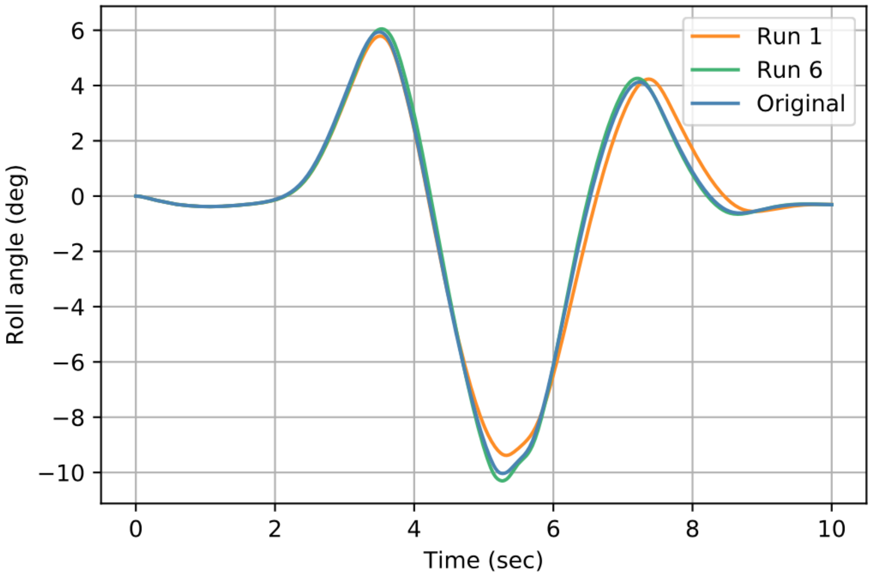


Figure 5.16: Roll angles of the vehicle with different steering angle configurations during the DLC maneuver.

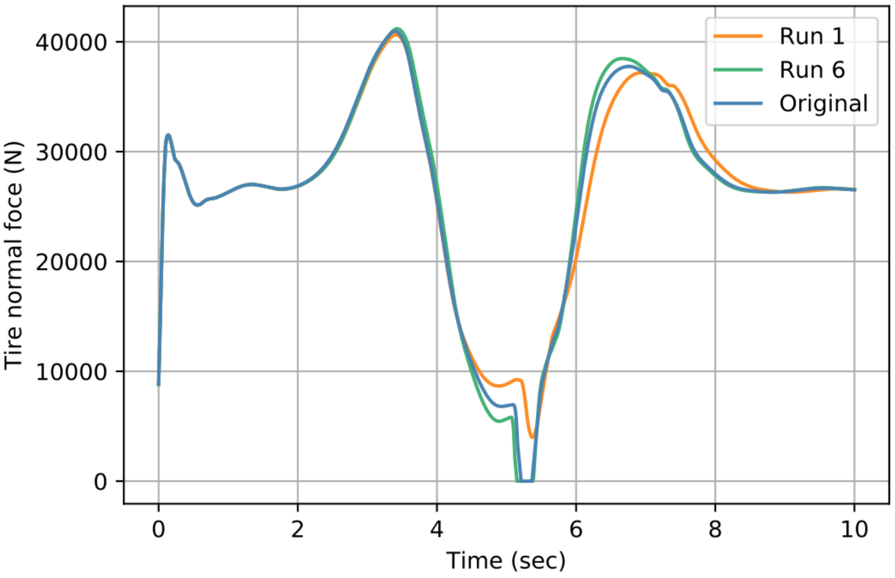


Figure 5.17: Normal rear left tire forces with different steering angle configurations during the DLC maneuver.

demanding maneuver and the vehicle is pushed to its limits. The roll angles observed in the original configuration and run 6 configurations are very high and coupled with the yaw rate of the vehicle the rear left tire of the vehicle loses contact with the road. The normal load on the rear left tire of the vehicle can be seen in Figure 5.17 for all of the configurations.

As seen from Figure 5.17, rear left tire of the run 6 and original configuration loses contact with the road around 5 seconds. The vehicle with the run 1 configuration does not lose contact with the road, but the normal force on the tire is lower than the unsprung mass, meaning the tire is standing on the rebound stops and is about to lose contact. The moment the tire normal forces reaches a minimum for all the configurations is the point where the vehicle roll angle, yaw rate, and the lateral acceleration is at its maximum. In Figure 5.18, slip angle change with respect to time of the left tires of the fourth axle of the vehicle with different steering angle configurations can be seen. The left tire of the fourth axle is given because that is the axle with the highest slip angles.

A summary of the DLC study results can be seen in table 5.2.

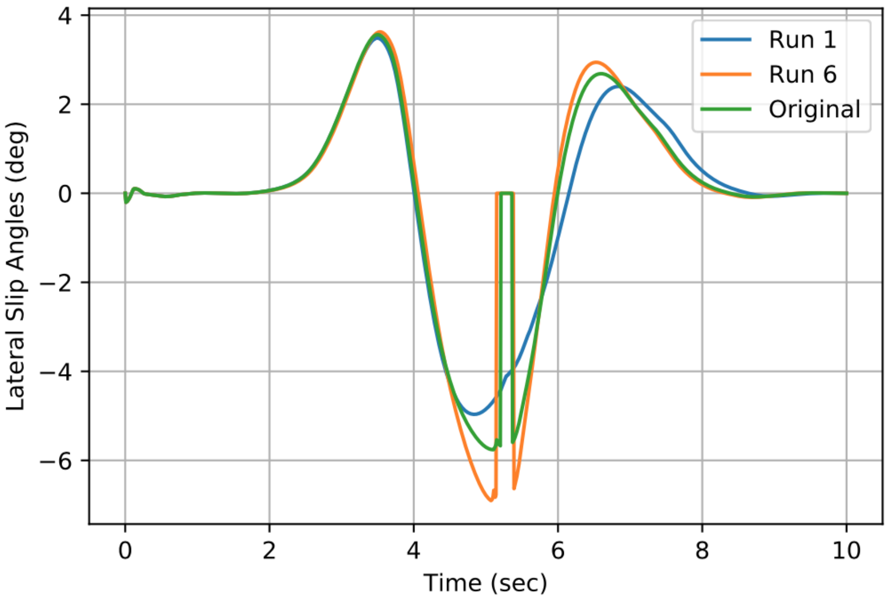


Figure 5.18: Slip angles rear left tire forces with different steering angle configurations during the DLC maneuver.

Table 5.2: DLC study results summary.

	Original Configuration	Run 1 Configuration	Run 6 Configuration
Max Steering Percentage (%)	19.9	20.7	18.2
Min Longitudinal Velocity (km/h)	78.8	79.14	78.5
Max Throttle Demand (%)	14.3	13.8	14.8
Max Lateral Acceleration (g)	0.379	0.370	0.385
Max Yaw Rate (deg/sec)	14.3	14.1	15.8
Max Roll Angle (deg)	10.0	9.39	10.3
Min Tire Normal Force (N)	0	3978	0
Max Tire Slip Angle (deg)	5.76	4.97	6.9

5.7 Conclusion

In this chapter, the proposed method for simulating a DLC maneuver for a vehicle without a steering system was presented. A generalized open loop function was derived to create steering input for a double lane change maneuver. The input can be manipulated by changing the parameters of the generalized function. A method similar to the method used in turning radius optimization study was used to optimize the parameters of the general DLC input function.

Three different DLC maneuvers were conducted. The first DLC maneuver conducted was for the original configuration. The second optimization process was conducted for the vehicle with run 1 configuration, which was the configuration with the highest turning radius and lowest lateral slip angles. Also, the final optimization study was conducted for the vehicle with run 6 configuration, which is the configuration with the lowest turning radius and highest tire lateral slip angles out of all the runs. The vehicle completed the maneuver with all three configurations. From Figure 5.12, it can be seen that the longitudinal velocity of the vehicle with run 1 configuration had the least deviation from the target velocity. Also, as seen from Figure 5.13, the vehicle with the run 1 configuration had the lowest throttle demand. From these results, it can be said that the vehicle with run 1 configuration encountered the least resistance. As seen from Figure 5.14 and Figure 5.15, the vehicle with run 1 configuration yields the lowest results in terms of lateral acceleration and yaw rate. One of the reasons the yaw rate and lateral acceleration of the vehicle with run 1 configuration are lower than other configurations is the tire lift off occurring in the original and run 6 configurations, which can be seen in Figure 5.17. Yaw rate of the vehicle coupled with the roll angle seen in Figure 5.16, results in tire lift off for the original and run 6 configurations. From Figure 5.11, it can be seen in the first part of the maneuver the run 1 configuration input demand is higher than other configuration but in the second part of the maneuver where the tire loft off occurs the steering demand of the original and run 6 configurations are higher than the run 1 configuration.

Using the proposed method, DLC maneuvers for the same vehicle with three different wheel steering angle configurations were successfully conducted. These results show that the proposed method can be used to conduct high-speed DLC maneuvers for vehicles without a steering system. The method is particularly useful for the early design phase of the vehicle. Usually, the maximum wheel steering angles of the vehicle is restricted by the body or the suspension system of the vehicle. With this information, before designing the steering system of the vehicle, the high-speed capabilities of the vehicle can be investigated using this method.

6 DISCUSSION AND CONCLUSIONS

6.1 Turing Radius Optimization Discussion

The first method discussed in this study is the turning radius optimization study. In this study, the maximum steering angles of the prototype 8x8 AWS vehicle were optimized to obtain a small turning radius and low tire lateral slip angles. To be able to calculate the vehicle turning radius and to measure the tire lateral slip angles, multi-body dynamics model of the 8x8 vehicle was created in Adams Car. The multi-body dynamics model of the vehicle was used to conduct low-speed steady-state turning simulations, and the simulation results were used in the optimization process. For the optimization process, a genetic algorithm was written in Python. Using the 8x8 multi-body vehicle model and the Python code optimization processes were conducted.

In this study, several different optimization processes with different genetic algorithm settings, various number of individuals in a generation, and different biases in the fitness function were conducted. Results of optimization processes were used to conduct comparisons. A summary of the optimization processes is given in Table 6.1.

As seen in Table 6.1, by changing the biases in the fitness function, the results that are obtained from the optimization process can be tailored as desired. The turning radius of the vehicle can be lowered by increasing the value of the bias that is multiplied by the normalized turning radius in the fitness function, and the same method can be used to reduce the tire lateral slip angles.

Due to the number of parameters and the number of different values each parameter can take, the search space of the optimization algorithm is enormous. Before this study, the

Table 6.1: Turning radius optimization results.

	Tuning Radius (mm)	Maximum Tire Lateral Slip Angle (deg)	Number of Generations Created	Fitness Function Biases	Termination Reason
Original Configuration	7432.84	4.6	N/A	N/A	N/A
Run 1	7621.46	1.38	19	0.5 - 0.5	Maximum Fitness Reached
Run 2	7477.91	1.87	14	0.5 - 0.5	Maximum Fitness Reached
Run 3	7398.53	1.75	25	0.6 - 0.4	Maximum Fitness Reached
Run 4	7108.9	2.51	100	0.7 - 0.3	Maximum Iteration Number Reached
Run 5	7236.82	2.5	12	0.5 - 0.5	Maximum Fitness Reached
Run 6	6940.16	3.26	34	0.7 - 0.3	Maximum Fitness Reached

optimal tire maximum steering angles were found by trial and error, but for such an ample search space finding the best possible solution by hand is very unlikely.

By comparing the original configuration and Run 2, it can be seen that a similar turning radius to the original configuration and much lower tire lateral slip angles can be found by genetic algorithm optimization. In this case, a turning radius that is very similar to the original configuration was reached with 70% lower maximum tire lateral slip angle was reached. Also, in Run 6 much smaller vehicle turning radius than original configuration was achieved with 30% lower tire lateral slip angle.

From the optimization process results, it can be seen that the genetic algorithm consistently found acceptable results regardless of the genetic algorithm configuration or the population size. Optimization process results showed that the optimized tire maximum steering angles cause lower tire lateral slip angels and according to the fitness function biases lower turning radiuses than the original configuration are possible.

6.2 Nato Lane Change Results Discussion

The low-speed handling capabilities and the turning radius of the vehicle were optimized in the turning radius optimization study. The maximum wheel steering angles obtained in this study guarantees the low turning radius and low tire lateral slip angles, but the high-speed handling characteristics of the vehicle which is using the optimized maximum wheel steering angles are still unknown. Usually, the low-speed and high-speed steering strategies differ in an AWS vehicle. In this study, the control strategy used in the prototype vehicle is used for the optimized wheel steering angle configurations. In this strategy, the vehicle is steered in all of the wheels for low vehicle longitudinal velocities, and when the longitudinal vehicle velocity is higher than a specific limit, the rear four wheels are kept at zero steering angle and the vehicle is steered only by the front four tires. In the NATO Lane Change maneuver, the longitudinal vehicle velocity was kept constant at 80 km/h, which is considered high speed considering 8x8 military vehicles usually have top speeds around 100 km/h. Due to the high-speed nature of the maneuver, the rear four tires are not steered during the maneuver.

Three steering angle configurations were compared in this study. The first configuration tested is the original configuration, this is the configuration which the optimized wheel steering angle configurations will be compared against. The second configuration is the run 1 configuration, and this configuration provided the lowest tire lateral slip angles and the highest turning radius. The final configuration is the run 6, this configuration provided the smallest turning radius, but the tire lateral slip angles were higher than any of the other optimization results. However, the tire lateral slip angles were still lower than the original configuration.

All of the steering angle configurations successfully completed the DLC maneuver. However, there were some apparent differences between the configurations. From the study results, it was observed that both the original configuration and the run 6 configuration diverged from the intended 80 km/h longitudinal velocity that the run 1 configuration.

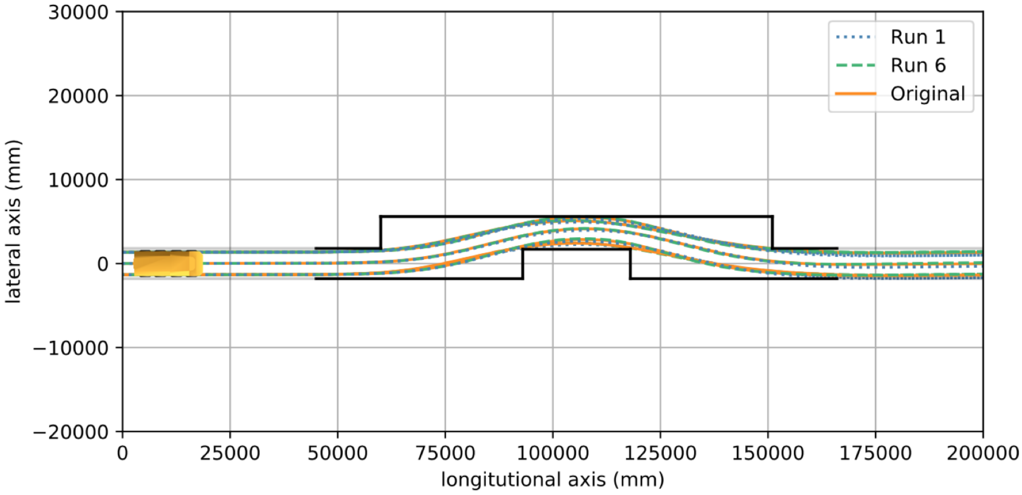


Figure 6.1: Paths of all the configurations during the DLC maneuver.

From the throttle inputs, it was observed that the run 6 required the most throttle while diverging from the intended velocity more than the other two configurations. The original configuration was better in terms of deviating from the expected velocity and the throttle input needed, but the run 1 configuration was better than both configurations in both areas. It can be said that the vehicle with run 1 configuration encountered the least resistance in the longitudinal direction.

Another big difference between the configurations is the left tire in the fourth axle losing contact with the road in the original and run 6 configurations. High roll angles of the vehicles for the original and the run 6 configuration coupled with high yaw rates caused the left tire in the fourth axle to lose contact with the road. This issue did not occur in the run 1 configuration. There are no specifications about tire lift off in the Nato Lane Change maneuver. However, some of the DLC studies specify that the tire lift off is not acceptable [13].

As discussed before, the steering wheel input percentage was calculated by dividing the front left wheel steering input with the corresponding maximum steering angle. The initial steering demand for run 1 is higher than the other configurations. However, after the initial lane change steering demand for run 1 is lower than the other two configurations. From these results, it can be concluded that the steering effort for run 1 is slightly higher than the other configurations, but due to high roll angles and tire lift off the steering effort for the run 6 and original configurations are higher in the second part of the maneuver.

As seen from the DLC study results, this method can be used to conduct DLC maneuver with a vehicle model that does not have a steering system. This process consistently finds the steering inputs needed to perform the DLC maneuver for different wheel steering angle configuration.

6.3 Conclusion

In this work, a method for optimizing the set of tire maximum steering angles for multi-axle vehicles with AWS was proposed. The optimization process is done using Adams Car for simulations and a Python code for the genetic algorithm optimization. Results of the study show that the proposed method finds steering angle sets that caused lower tire lateral slip angle independent of the genetic algorithm settings and the biases of the fitness function. Depending on the biases of the fitness function lower turning radiuses that the original configuration with lower tire lateral slip angles was obtained. Considering the number of possible options for the problem, finding an optimal solution with trial and error method which is used currently, is highly unlikely. Design specifications of the wheel steering angles can be found reliably with the proposed method.

Another method was developed to evaluate the high-speed handling behavior of the vehicle fitted with the optimized set of tire maximum wheel steering angles. A function was derived to produce the generalized open loop steering input for DLC maneuvers. The parameters of the generalized open loop steering angle for the DLC maneuvers are optimized for each maximum steering angle set simulated so that the vehicle could complete the DLC maneuver without crossing the road limits. The simulation results showed that the proposed method has successfully optimized the parameters of the generalized open loop steering input for DLC maneuvers equation, for all of the maximum steering angle sets tested. The method can be used for investigating the high-speed vehicle handling behavior for vehicles that have no steering system.

The two proposed methods can be used together to obtain an optimized set of maximum wheel steering angles that caused the vehicle to have a small turning radius as well as low tire lateral slip angles and the high-speed handling capabilities of the vehicle can be investigated before the steering system is designed for the optimized set of maximum steering angles. The two methods can be used separately as well. The DLC maneuver method can be used to investigate the high-speed vehicle handling early in the design process. The effects of wheelbase distance between the axles and different AWS strategies can be investigated using this method.

6.4 Future Work

As shown in the sections above, each of the proposed methods works as intended. However, some improvements can be made to enhance the usefulness of the methods together and on their own. One of these improvements is to automatize the turning radius optimization study to perform multiple optimization processes. Using this method, the user would only need to run the optimization process once and different optimization processes would run with different fitness function biases and output the results. Using these results user would choose the best one knowing the tradeoff.

Similar changes can be made to the DLC study so that all of the possible steering configurations can be tested automatically. The DLC study can be used to investigate the effects of other parameters as well. For example, at the beginning of the design phase, the effects of the axle locations on the high-speed handling behavior of the vehicle can be investigated using this method.

Another improvement can be made to the generalized DLC steering input function. If the parameters in the function can be approximated with vehicle properties like the understeer coefficient of the vehicle, the optimization process would be simplified or maybe made become obsolete altogether. The generalized DLC input can be used for a single lane change, as explained in the previous chapter. This function can be implemented into autonomous driving, and simple lane change or emergency maneuvers can be performed using this function. This method would include the effects of vehicle dynamics in a simple, easy to compute function.

REFERENCES

[1] Harrer M, Pfeffer P, editors. Steering handbook. Cham: Springer; 2017.

[2] North Atlantic Treaty Organization (NATO) Allied Vehicle Testing Publication (AVTP) 03-160W

[3] Gillespie TD. Fundamentals of vehicle dynamics: a professional development technical video tutorial series: workbook & educational support material. Warrendale, PA: Society of Automotive Engineers; 1992.

[4] Jazar RN. Vehicle dynamics: theory and applications. 3. ed. New York, NY: Springer; 2014.

[5] Wong JY. Theory of Ground Vehicles. John Wiley & Sons; 2001.

[6] Pacejka HB. Tyre and vehicle dynamics. Amsterdam: Elsevier Butterworth-Heinemann; 2009.

[7] Marzbani H, Vo D, Khazaei A, Fard M, Jazar RN. Transient and steady-state rotation centre of vehicle dynamics. International Journal of Nonlinear Dynamics and Control. 2017;1(1):97–113.

[8] Menhour L, Lechner D, Charara A. Design and experimental validation of linear and nonlinear vehicle steering control strategies. Vehicle System Dynamics. 2012;50(6):903–38.

[9] Furukawa Y, Yuhara N, Sano S, Takeda H, Matsushita Y. A Review of Four-Wheel Steering Studies from the Viewpoint of Vehicle Dynamics and Control. Vehicle System Dynamics. 1989;18(1-3):151–86.

[10] Sano S, Furukawa Y, Shiraishi S. Four Wheel Steering System with Rear Wheel Steer Angle Controlled as a Function of Steering Wheel Angle. SAE Technical Paper Series. 1986.

[11] Takiguchi T., Yasuda, N., Furutani, S., Kazanawa, H., and Inoue, H., “Improvement of Vehicle dynamics by Vehicle-Speed-Sensing Four Wheel Steering System”, SAE Transactions, Vol. 95, No. 860624, pp. 870-879, 1986.

[12] Fukada Y. Slip-Angle Estimation for Vehicle Stability Control. Vehicle System Dynamics. 1999;32(4-5):375–88.

[13] Peng Y, Yang X. Comparison of various double-lane change maneuver specifications. Vehicle System Dynamics. 2012;50(7):1157–71.

[14] Emir Kutluay and Hermann Winner, 2012, Proceedings of the 2012 Winter Simulation Conference

[15] Pacejka HB, Bakker E. The Magic Formula Tyre Model. Vehicle System Dynamics. 2004;21(1):1–18.

[16] Tremlett AJ, Limebeer DJN. Optimal tyre usage for a Formula One car. Vehicle System Dynamics. 2016;54(10):1448–73.

[17] Williams DE. Generalised multi-axle vehicle handling. Vehicle System Dynamics. 2012;50(1):149–66.

[18] Winkler CB. Simplified Analysis of the Steady-State Turning of Complex Vehicles. Vehicle System Dynamics. 1998;29(3):141–80.

[19] Winkler CB, Gillespie TD. On the Directional Response Characteristics of Heavy Trucks. Vehicle System Dynamics. 1977;6(2-3):120–3.

[20] Qu Q, Zu JW. On Steering Control of Commercial Three-Axle Vehicle. Journal of Dynamic Systems, Measurement, and Control. 2008;130(2):021010.

[21] Ellis JR. The Steering Characteristics of Multiple Axle Bogie Systems. Vehicle System Dynamics. 1976;5(4):221–38.

[22] Bayar K, Unlusoy YS. Steering strategies for multi-axle vehicles. International Journal of Heavy Vehicle Systems. 2008;15(2/3/4):208–36.

[23] Watanabe K, Yamakawa J, Tanaka M, Sasaki T. Turning characteristics of multi-axle vehicles. Journal of Terramechanics. 2007;44(1):81–7.

[24] Nocedal J, Wrigh SJ. Numerical Optimization. Springer Science & Business Media; 2006.

[25] hapra SC, Canale RP. Numerical methods for engineers. 6th ed. New York, NY: McGraw-Hill Education; 2010.

[26] Oh K, Seo J, Kim J, Yi K. An investigation on steering optimization for minimum turning radius of multi-axle crane based on MPC algorithm. 2015 15th International Conference on Control, Automation and Systems (ICCAS). 2015.

[27] Felzien M, Cronin D. Steering error optimization of the Macpherson strut automotive front suspension. Mechanism and Machine Theory. 1985;20(1):17–26.

[28] Angelis S, Tidlund M, Lidberg M, Leledakis A, Katzourakis DI, Nybacka M. 12th International Symposium on Advanced Vehicle Control , AVEC '14. Tokyo, Japan; 2014.

[29] Arvind V. Optimizing the turning radius of a vehicle using symmetric four wheel steering system,International. International Journal of Scientific & Engineering Research. 2013;4(12):2177–84.

[30] Aydın M, Ünlüsoy YS. Optimization of suspension parameters to improve impact harshness of road vehicles. The International Journal of Advanced Manufacturing Technology. 2011;60(5-8):743–54.

[31] John H. Holland. Adaptation in Natural and Artificial Systems. Ann Arbor: University of Michigan Press; 1975.

[32] Haupt RL, Haupt SE. Practical genetic algorithms. Hoboken, NJ: John Wiley; 2004.

[33] Kramer O. Genetic Algorithm Essentials. Cham: Springer International Publishing; 2018.

[34] Hui S. Multi-objective optimization for hydraulic hybrid vehicle based on adaptive simulated annealing genetic algorithm. Engineering Applications of Artificial Intelligence. 2010;23(1):27–33.

[35] Huang S, Ren W. Use of neural fuzzy networks with mixed genetic/gradient algorithm in automated vehicle control. IEEE Transactions on Industrial Electronics. 1999;46(6):1090–102.

[36] Schuller J, Haque I, Eckel M. An Approach for Optimisation of Vehicle Handling Behaviour in Simulation. Vehicle System Dynamics. 2002;37(sup1):24–37.

[37] Milliken WF, Milliken DL, Metz LD. Race car vehicle dynamics. Warrendale, PA.: Society of Automative Engineers; 1995.

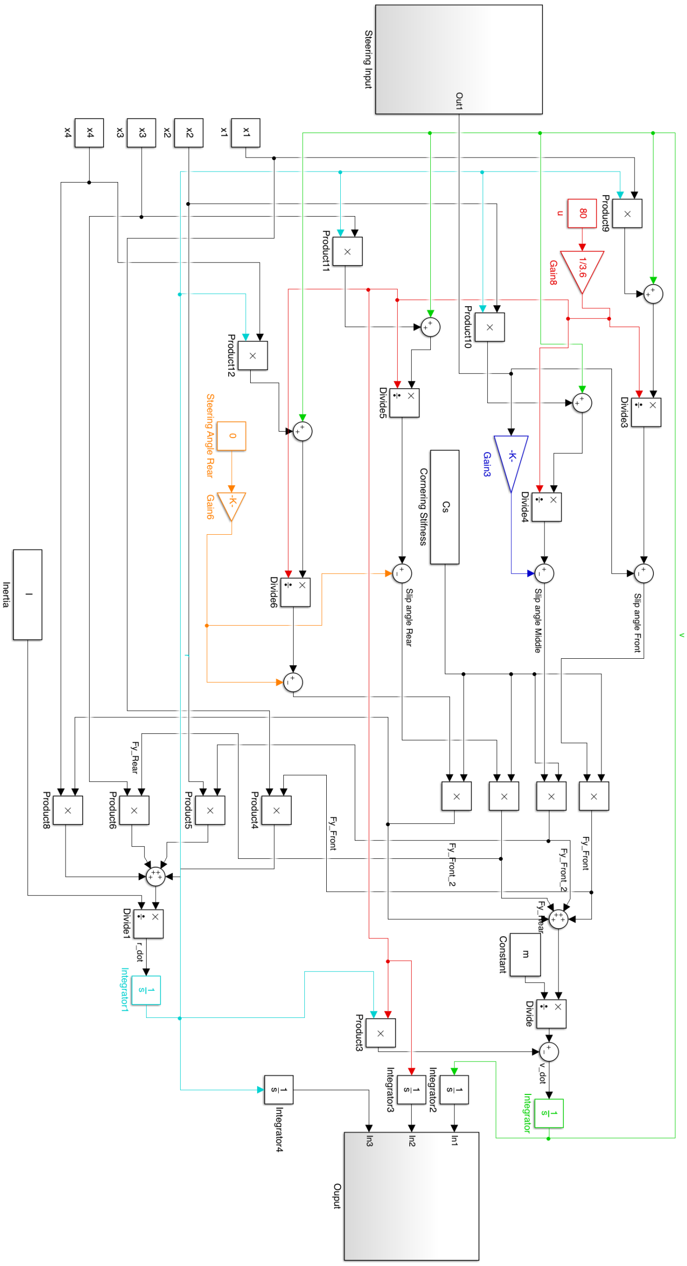
[38] Datoussaïd S, Hadjit R, Verlinden O, Conti C. Optimization design of multibody systems by using genetic algorithms. Vehicle System Dynamics. 1998;29(sup1):704–10.

[39] Ellis JR. The Steering Characteristics of Multiple Axle Bogie Systems. Vehicle System Dynamics. 1976;5(4):221–38.

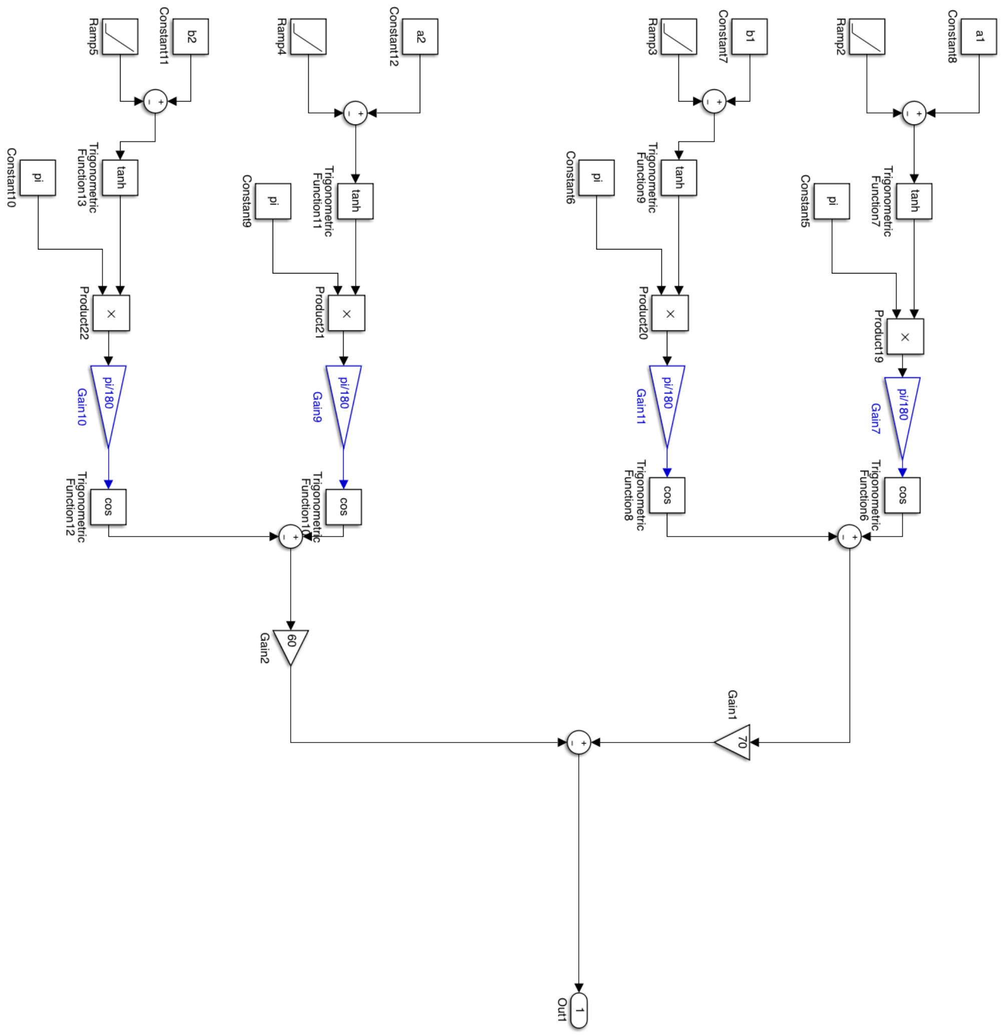
[40] Adams Help, MSC Software, 2018.

[41] https://www.python.org/dev/peps/pep-0008/

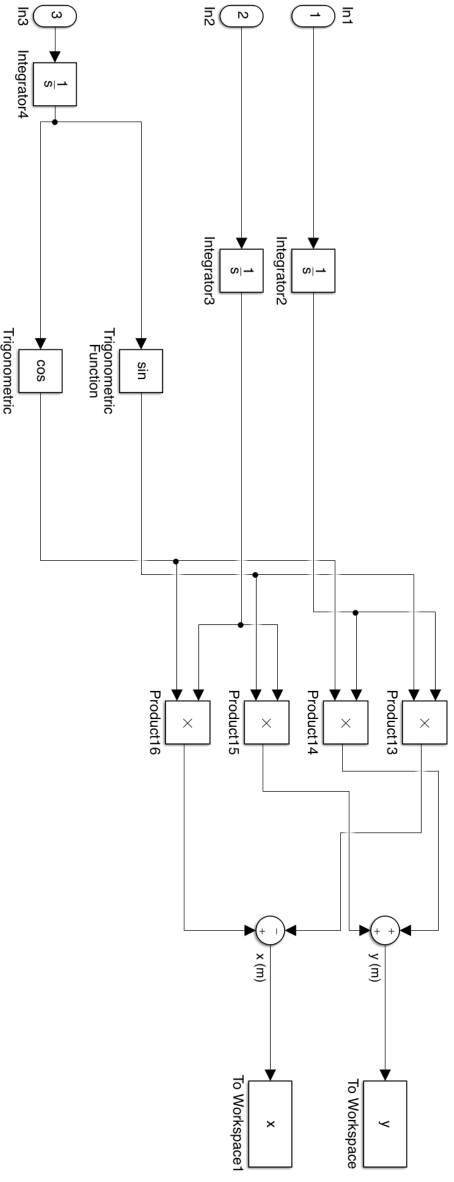
APPENDIX A


Steering Input
Out1
x1
x2
x3
x4
Product9
80
u
1/3.6
Gain8
Product11
Product10
Product12
Divide5
Divide3
0
Steering Angle Rear
-K-
Gain6
Cs
Cornering Stiffness
-K-
Gain3
Divide4
Divide6
Slip angle Rear
Slip angle Middle
Slip angle Front
I
Inertia
v
Fy_Rear
Product8
Product6
Product5
Product4
Fy_Front
Fy_Front_2
Fy_Front
Fy_Rear
m
Constant
Divide1
r_dot
Integrator1
Divide
v_dot
Integrator
Product3
Integrator3
Integrator2
Integrator4
In3
In2
In1
Ouput

Steering input block



Output block



APPENDIX B

```
macro modify macro=.ACAR.macros.mac_ana_sub &
 user="acar␣analysis␣submit" &
 commands="acar␣analysis␣full_vehicle␣sdi␣submit␣&", &
          "␣assembly=.Vehicle␣&", &
          "␣variant=default␣&", &
    "␣output_prefix=\"Pars3\"␣&", &
          "␣analysis_mode=interactive␣&", &
    "␣road_data_file=\"mdids://acar_shared/roads.tbl/2d_flat.rdf\"
&", &
    "
dcf_file=\"file://F:/Thesis/steer.xml\"␣&", &
          "␣log_file=yes␣&", &
    "␣comment=\"\""
variable delete variable=.ACAR.dboxes.dbox_ana_ful_sdi_sub.prefix,
.ACAR.dboxes.dbox_ana_ful_sdi_sub."tmp*"
acar analysis submit

numeric_results write &
   result_set_component_name =
.Vehicle.Simulation.Body.Curb_To_Curb.
        Curb_to_Curb_x,
.Vehicle.Simulation.Body.Curb_To_Curb.
        Curb_to_Curb_y &
   sort_by = by_time &
   order = ascending &
   write_to_terminal = off &
   file_name = "disp.txt" &
     &

numeric_results write &
   result_set_component_name =
.Vehicle.Simulation.
        til_wheel_tire_kinematics.lateral_slip_front,
.Vehicle.Simulation.
        til_wheel_tire_kinematics.lateral_slip_front2,
.Vehicle.Simulation.
        tir_wheel_tire_kinematics.lateral_slip_front,
.Vehicle.Simulation.
        tir_wheel_tire_kinematics.lateral_slip_front2 &
   sort_by = by_time  &
   order = ascending  &
   write_to_terminal = off  &
   file_name = "slip_front.txt"  &
     &

numeric_results write  &
   result_set_component_name =
.Vehicle.Simulation.
til_wheel_tire_kinematics.lateral_slip_rear,
.Vehicle.Simulation.
        til_wheel_tire_kinematics.lateral_slip_rear2,
.Vehicle.Simulation.
        tir_wheel_tire_kinematics.lateral_slip_rear,
.Vehicle.Simulation.
        tir_wheel_tire_kinematics.lateral_slip_rear2  &
   sort_by = by_time  &
   order = ascending  &
   write_to_terminal = off  &
   file_name = "slip_rear.txt"  &
     &

analysis delete&
        analysis_name = Simulation
```

APPENDIX C

```
#Turning Radius Optimization File
#
import Adams
from os import path, mkdir, chdir
from math import atan, sqrt, exp, tan
from random import random as rnd
import numpy as np
from random import randint,gauss,randrange

#Output text file name
Result_file = 'Turning_Radius_Optimization_Study.txt'

#Selection
#selection_type = 'Fittest Half'
selection_type = 'Roulette␣Wheel'

#Pairing
#pairing_type = 'Random'
pairing_type = 'Fittest'

#Crossover
#crossover_type = 'Single Point'
crossover_type = 'Single␣Station'

#Mutation
mutation_type = 'Reset'
#Omutation_type = 'Gauss'
#
#Fitness Biases
turn_radii_bias = 0.5
slip_bias = 1 - turn_radii_bias

#Duration of the steering manuver (seconds)
Ts = 10

#Command files
command_file  = 'F:\\Thesis\\steer.cmd'
```

```
#Path to the main directory
MainFolderPath = path.abspath(r'F:\Thesis\Optimization')


def individual(number_of_genes = 8):
#first axle
    individual = [round(rnd()*-10-20,1)]
    individual.append(round(rnd()*-10-18,1))
    while individual[1] < individual[0]:
        individual[1] = round(rnd()*-10-18,1)
#second axle
    individual.append(round(rnd()*-10-15,1))
    individual.append(round(rnd()*-10-13,1))
    while individual[3] < individual[2]:
        individual[3] = round(rnd()*-10-13,1)
#third axle
    individual.append(round(rnd()*10+15,1))
    individual.append(round(rnd()*10+13,1))
    while individual[5] > individual[4]:
        individual[5] = round(rnd()*10+13,1)
#fourth axle
    individual.append(round(rnd()*10+20,1))
    individual.append(round(rnd()*10+18,1))
    while individual[7] > individual[6]:
        individual[7] = round(rnd()*10+18,1)
    return [individual[0],individual[2],individual[4],
        individual[6],individual[1],individual[3],
        individual[5],individual[7]]


def population(number_of_individuals):
    return ([individual() for x in range(number_of_individuals)])


def max_of(x):
    return max(abs(max(x)),abs(min(x)))


def fit_test(x):
#Changing the motions
    L1_motion.function = 'STEP(time,0,0,10,'+str(x[0]*0.01745)+')'
    L2_motion.function = 'STEP(time,0,0,10,'+str(x[1]*0.01745)+')'
    L3_motion.function = 'STEP(time,0,0,10,'+str(x[2]*0.01745)+')'
    L4_motion.function = 'STEP(time,0,0,10,'+str(x[3]*0.01745)+')'
    R1_motion.function = 'STEP(time,0,0,10,'+str(x[4]*0.01745)+')'
    R2_motion.function = 'STEP(time,0,0,10,'+str(x[5]*0.01745)+')'
    R3_motion.function = 'STEP(time,0,0,10,'+str(x[6]*0.01745)+')'
    R4_motion.function = 'STEP(time,0,0,10,'+str(x[7]*0.01745)+')'
#Run the simulation
    Adams.read_command_file(command_file)
#Reading simulation results
    results_disp = open('disp.txt','r').readlines()
    results_slip_front = open('slip_front.txt','r').readlines()
    results_slip_rear = open('slip_rear.txt','r').readlines()
#Result parameters
    path_x = []
    path_y = []
    sa1 = []
    sa2 = []
    sa3 = []
    sa4 = []
    sa5 = []
    sa6 = []
    sa7 = []
    sa8 = []
#Creating lists of the results
    for x in range(209,507):
        path_x.append(float(results_disp[x].split()[0]))
        path_y.append(float(results_disp[x].split()[1]))
        sa1.append(float(results_slip_front[x].split()[0]))
        sa2.append(float(results_slip_front[x].split()[1]))
        sa3.append(float(results_slip_front[x].split()[2]))
        sa4.append(float(results_slip_front[x].split()[3]))
        sa5.append(float(results_slip_rear[x].split()[0]))
        sa6.append(float(results_slip_rear[x].split()[1]))
        sa7.append(float(results_slip_rear[x].split()[2]))
        sa8.append(float(results_slip_rear[x].split()[3]))
    p1 = {'x':path_x[50],'y':path_y[50]}
    p2 = {'x':path_x[100],'y':path_y[100]}
    p3 = {'x':path_x[150],'y':path_y[150]}
    A = p1['x']*(p2['y']-p3['y'])-p1['y']*(p2['x']-p3['x'])
    +p2['x']*p3['y']-p3['x']*p2['y']
    B = (p1['x']**2+p1['y']**2)*(p3['y']-p2['y'])
        +(p2['x']**2+p2['y']**2)*(p1['y']-p3['y'])
        +(p3['x']**2+p3['y']**2)*(p2['y']-p1['y'])
    C = (p1['x']**2+p1['y']**2)*(p3['x']-p2['x'])
        +(p2['x']**2+p2['y']**2)*(p1['x']-p3['x'])
        +(p3['x']**2+p3['y']**2)*(p2['x']-p1['x'])
    D = (p1['x']**2+p1['y']**2)*(p3['x']*p2['y']
        -p2['x']*p3['y'])+(p2['x']**2+p2['y']**2)*
        (p1['x']*p3['y']-p3['x']*p1['y'])+(p3['x']
        **2+p3['y']**2)*(p2['x']*p1['y']-p1['x']*p2['y'])
#Calculating the turning radius
```

```
    r = sqrt(float(B**2+C**2-4*A*D)/(4*A**2))
#Slip angles
    sa = [sa1,sa2,sa3,sa4,sa5,sa6,sa7,sa8]
#Cost function output
    sa_max = (max([max_of(sa[x]) for x in range(8)]))
    return
{'Fitness':[turn_radii_bias*((7000-r)/1000)+slip_bias*((3-sa_max)/2
)],
        'Turn␣Radii␣(mm)':[r],'Tire␣Lateral␣Slips␣(deg)':
        [max_of(sa1),max_of(sa2),max_of(sa3),max_of(sa4)
                ,max_of(sa5),max_of(sa6),max_of(sa7),max_of(sa8)]}


def first_generation(pop):
    fitness = [fit_test(pop[x]) for x in range(len(pop))]
    sorted_fitness = sorted([[pop[x], fitness[x]['Fitness'],
        fitness[x]['Turn␣Radii␣(mm)'],fitness[x]
                ['Tire␣Lateral␣Slips␣(deg)']]
        for x in range(len(pop))], key=lambda x: x[1])
    population = [sorted_fitness[x][0] for x in
        range(len(sorted_fitness))]
    radius = [sorted_fitness[x][2][0] for x in
        range(len(sorted_fitness))]
    slips = [sorted_fitness[x][-1] for x in
        range(len(sorted_fitness))]
    fitness = [sorted_fitness[x][1][0] for x in
        range(len(sorted_fitness))]
    return {'Individuals': population
            , 'Fitness': sorted(fitness)
            , 'Turn␣Radii␣(mm)': radius
            , 'Tire␣Lateral␣Slips␣(deg)': slips}


def roulette(cum_sum,chance):
    veriable = list(cum_sum.copy())
    veriable.append(chance)
    veriable = sorted(veriable)
    return veriable.index(chance)


def selection(generation, method='Roulette␣Wheel'):
    generation['Normalized␣Fitness'] = sorted(
        [generation['Fitness'][x] /sum(generation['Fitness'])
                for x in range(len(generation['Fitness']))]
                        ,reverse = True)
    generation['Cumulative␣Sum'] = np.array(
        generation['Normalized␣Fitness']).cumsum()
    if method == 'Roulette␣Wheel':
        selected = []
        for x in range(len(generation['Individuals'])//2):
            selected.append(roulette(
                generation['Cumulative␣Sum'], rnd()))
            while len(set(selected)) != len(selected):
                selected[x] = (roulette(
                        generation['Cumulative␣Sum'], rnd()))
        selected = {'Individuals':
                [generation['Individuals'][int(selected[x])]
        for x in range(len(generation['Individuals'])//2)]
            , 'Fitness': [generation['Fitness'][int(selected[x])]
        for x in range(len(generation['Individuals'])//2)]
            , 'Turn␣Radii␣(mm)': [generation['Turn␣Radii␣(mm)']
                [int(selected[x])]
        for x in range(len(generation['Individuals'])//2)]
            , 'Tire␣Lateral␣Slips␣(deg)':
            [generation['Tire␣Lateral␣Slips␣(deg)']
            [int(selected[x])] for x in range(len(
                generation['Individuals'])//2)]}
    elif method == 'Fittest␣Half':
        selected_individuals = [generation['Individuals'][-x-1]
        for x in range(int(len(generation['Individuals'])//2))]
                        selected_fitnesses = [generation['Fitness'][-x-1]
        for x in range(int(len(generation['Individuals'])//2))]
                        selected_radius = [generation['Turn␣Radii␣(mm)'][-x-1]
        for x in
                        range(int(len(generation['Individuals'])//2))]
                        selected_slips =
                                [generation['Tire␣Lateral␣Slips␣(deg)'][-x-1]
        for x in range(int(len(generation['Individuals'])//2))]
                        selected = {'Individuals': selected_individuals
                    , 'Fitness': selected_fitnesses
                    , 'Turn␣Radii␣(mm)': selected_radius
                    , 'Tire␣Lateral␣Slips␣(deg)': selected_slips}
    return selected


def pairing(elit, selected, method = 'Random'):
    individuals = [elit['Individuals']]+selected['Individuals']
    fitness = [elit['Fitness']]+selected['Fitness']
    if method == 'Fittest':
        parents = [[individuals[x],individuals[x+1]]
                for x in range(len(individuals)//2)]
    if method == 'Random':
        parents = []
        for x in range(len(individuals)//2):
            parents.append([individuals[randint(0,
                (len(individuals)-1))],individuals[randint
                        (0,(len(individuals)-1))]])
            while parents[x][0] == parents[x][1]:
                parents[x][1] = individuals[randint(0,
                        (len(individuals)-1))]
    if method == 'Weighted␣Random':
        normalized_fitness = sorted([[fitness[x] /sum(fitness)
                for x in range(len(individuals)//2)]],reverse = True)
        cummulitive_sum = np.array(normalized_fitness).cumsum()
        parents = []
        for x in range(len(individuals)//2):
            parents.append([individuals[roulette(
                cummulitive_sum,rnd())],
                individuals[roulette(cummulitive_sum,rnd())]])
            while parents[x][0] == parents[x][1]:
                parents[x][1] = individuals[roulette
                        (cummulitive_sum,rnd())]
    return parents


def crossover(parents, method='Single␣Point'):
    if method == 'Single␣Point':
        pivot_point = randint(1, 3)*2
        offsprings = [parents[0][0:pivot_point]+parents[1]
                [pivot_point:]]
        offsprings.append(parents[1][0:pivot_point]+parents[0]
                [pivot_point:])
    if method == 'Single␣Station':
        section = randint(0, 3)
        offsprings = [parents[0].copy()]
        offsprings[0][section] = parents[1][section]
        offsprings[0][section+4] = parents[1][section+4]
        offsprings.append(parents[1])
        offsprings[1][section] = parents[0][section]
        offsprings[1][section+4] = parents[0][section+4]
    return offsprings


def mutation(individual, muatation_rate=1, method='Reset'):
    gene = [randint(0, 7)]
    for x in range(muatation_rate-1):
        gene.append(randint(0, 7))
        while len(set(gene)) < len(gene):
            gene[x] = randint(0, 7)
    mutated_individual = individual.copy()
    if method == 'Gauss':
        for n in range(len(gene)):
            if gene[n] < 4:
                mutated_individual[gene[n]] = round(gauss
                        (mutated_individual[gene[n]], abs(
                    mutated_individual[gene[n]] -
                        mutated_individual[gene[n]+4])), 1)
            else:
                mutated_individual[gene[n]] = round
                        (gauss(mutated_individual[gene[n]], abs(
                    mutated_individual[gene[n]] -
                        mutated_individual[gene[n]-4])), 1)
        for x in range(muatation_rate):
            if mutated_individual[gene[x]] > 27:
                mutated_individual[gene[x]] = 27
            if mutated_individual[gene[x]] < -27:
                mutated_individual[gene[x]] = -27
    if method == 'Reset':
        for x in range(len(gene)):
#First axle
            if gene[x] == 0:
                mutated_individual[gene[x]] = round(rnd()*-10-17,1)
            elif gene[x] == 4:
                mutated_individual[4] = round(rnd()*-10-16,1)
                while mutated_individual[4] <
mutated_individual[0]:
                    mutated_individual[4] = round(rnd()*-10-16,1)
#Second axle
            elif gene[x] == 1:
                mutated_individual[gene[x]] = round(rnd()*-10-14,1)
            elif gene[x] == 5:
                mutated_individual[5] = round(rnd()*-10-12,1)
                while mutated_individual[5] <
mutated_individual[1]:
                    mutated_individual[5] = round(rnd()*-10-12,1)
#Third axle
            elif gene[x] == 2:
                mutated_individual[gene[x]] = round(rnd()*10+14,1)
            elif gene[x] == 6:
                mutated_individual[6] = round(rnd()*10+12,1)
                while mutated_individual[6] >
mutated_individual[2]:
                    mutated_individual[6] = round(rnd()*10+12,1)
#Fourth axle
            elif gene[x] == 3:
                mutated_individual[gene[x]] = round(rnd()*10+17,1)
            elif gene[x] == 7:
                mutated_individual[7] = round(rnd()*10+16,1)
                while mutated_individual[7] >
mutated_individual[3]:
                    mutated_individual[7] = round(rnd()*10+16,1)
    return mutated_individual



def next_generation(gen):
    elit = {}
    next_gen = {}
    elit['Individuals'] = gen['Individuals'].pop(-1)
    elit['Fitness'] = gen['Fitness'].pop(-1)
    elit['Turn␣Radii␣(mm)'] = gen['Turn␣Radii␣(mm)'].pop(-1)
    elit['Tire␣Lateral␣Slips␣(deg)'] = gen['Tire␣Lateral␣Slips
(deg)'].pop(-1)
    selected = selection(gen, method = selection_type)
    parents = pairing(elit, selected, method = pairing_type)
    offsprings = [[[crossover(parents[x], method = crossover_type)
                    for x in range(len(parents))]
                   [y][z] for z in range(2)] for y in
range(len(parents))]
    offsprings1 = [offsprings[x][0] for x in range(len(parents))]
    offsprings2 = [offsprings[x][1] for x in range(len(parents))]
    unmutated = selected['Individuals']+offsprings1+offsprings2
    mutated = [mutation(unmutated[x], method = mutation_type) for x
                in range(len(gen['Individuals']))]
    unsorted_individuals = mutated + [elit['Individuals']]
    unsorted_next_gen = [fit_test(mutated[x]) for x in
range(len(mutated))]
    unsorted_fitness = [unsorted_next_gen[x]['Fitness']
        for x in range(len(gen['Individuals']))] +[[elit['Fitness']]]
    unsorted_turn_radii = [unsorted_next_gen[x]['Turn␣Radii␣(mm)']
        for x in range(len(gen['Individuals']))] +
                [[elit['Turn␣Radii␣(mm)']]]
    unsorted_slip = [unsorted_next_gen[x]['Tire␣Lateral␣Slips
(deg)'] for x in range(
        len(gen['Individuals']))] + [elit['Tire␣Lateral␣Slips
(deg)']]
    sorted_next_gen = sorted([[unsorted_individuals[x],
unsorted_fitness[x], unsorted_turn_radii[x],
                               unsorted_slip[x]] for x in
range(len(unsorted_individuals))], key=lambda x: x[1])
    next_gen['Individuals'] = [sorted_next_gen[x][0]
                               for x in
range(len(sorted_next_gen))]
    next_gen['Fitness'] = [sorted_next_gen[x][1][0]
                           for x in range(len(sorted_next_gen))]
    next_gen['Turn␣Radii␣(mm)'] = [sorted_next_gen[x][2][0]
```

```
                                        for x in
range(len(sorted_next_gen))]
    next_gen['Tire␣Lateral␣Slips␣(deg)'] = [sorted_next_gen[x][3]
                                              for x in
range(len(sorted_next_gen))]
    gen['Individuals'].append(elit['Individuals'])
    gen['Fitness'].append(elit['Fitness'])
    gen['Turn␣Radii␣(mm)'].append(elit['Turn␣Radii␣(mm)'])
    gen['Tire␣Lateral␣Slips␣(deg)'].append(elit['Tire␣Lateral␣Slips
(deg)'])
    return next_gen


#Motions
L1_motion =
Adams.stoo('.PARS3_LOWERARM_FULLVEHICLE_V4.
FNSS_PARS3_front_suspension_v1.MOTION_2')
L2_motion =
Adams.stoo('.PARS3_LOWERARM_FULLVEHICLE_V4.
FNSS_PARS3_front2_suspension_v1.MOTION_2')
L3_motion =
Adams.stoo('.PARS3_LOWERARM_FULLVEHICLE_V4.
FNSS_PARS3_rear_suspension_v1.MOTION_2')
L4_motion =
Adams.stoo('.PARS3_LOWERARM_FULLVEHICLE_V4.
FNSS_PARS3_rear2_suspension_v1.MOTION_2')
R1_motion =
Adams.stoo('.PARS3_LOWERARM_FULLVEHICLE_V4.
FNSS_PARS3_front_suspension_v1.MOTION_1')
R2_motion =
Adams.stoo('.PARS3_LOWERARM_FULLVEHICLE_V4.
FNSS_PARS3_front2_suspension_v1.MOTION_1')
R3_motion =
Adams.stoo('.PARS3_LOWERARM_FULLVEHICLE_V4.
FNSS_PARS3_rear_suspension_v1.MOTION_1')
R4_motion =
Adams.stoo('.PARS3_LOWERARM_FULLVEHICLE_V4.
FNSS_PARS3_rear2_suspension_v1.MOTION_1')

#Seting the working directory
chdir(MainFolderPath+'\\Turning_Radius_Results')

pop = population(32)
gen = [first_generation(pop)]
fitness_avg = np.array([sum(
        gen[0]['Fitness'])/len(gen[0]['Fitness'])])
fitness_max = np.array([max(gen[0]['Fitness'])])
```

```
res = open(Result_file ,'a')
res.write('\n'+str(gen)+'\n')
res.close()

def fitness_similarity_chech(max_fitness ,number_of_similarity):
    result = False
    similarity = 0
    for n in range(len(max_fitness)-1):
        if max_fitness[n] == max_fitness[n+1]:
            similarity += 1
        else:
            similarity = 0
    if similarity == number_of_similarity -1:
        result = True
    return result


for x in range(3):
    gen.append(next_generation(gen[x]))
    fitness_avg = np.append(fitness_avg ,sum
        gen[x]['Fitness'])/len(gen[x]['Fitness']))
    fitness_max = np.append(fitness_max ,
        max(gen[x]['Fitness']))
    res = open(Result_file ,'a')
    res.write('\n'+str(gen[x+1])+'\n')
    res.close()


finish = False
while finish == False:
    if max(fitness_max) > 0:
        break
    if max(fitness_avg) > -0.1:
        break
    if fitness_similarity_chech(fitness_max ,50) == True:
        break
    if len(gen) > 104:
        break
    gen.append(next_generation(gen[-1]))
    fitness_avg = np.append(fitness_avg ,
        sum(gen[-1]['Fitness'])/len(gen[-1]['Fitness']))
    fitness_max = np.append(fitness_max ,
        max(gen[-1]['Fitness']))
    res = open(Result_file ,'a')
    res.write('\n'+str(gen[-1])+'\n')
    res.close()
```

```
res = open(Result_file,'a')
res.write('Maximum␣Fitness␣=␣'+str(fitness_max)+
          ',␣Avarage␣Fitness'+str(fitness_avg))
res.close()
```

APPENDIX D

```
macro modify macro=.ACAR. macros . mac_ana_sub &
 user="acar␣analysis␣submit" &
 commands="acar␣analysis␣full_vehicle␣sdi␣submit␣&" , &
          "␣assembly=. Vehicle␣&" , &
          "␣variant=default␣&" , &
    "␣output_prefix =\" Vehicle \"␣&" , &
          "␣analysis_mode=interactive␣&" , &
    "␣road_data_file =\" mdids : // acar_shared / roads . tbl /2 d_flat . rdf \"
&" , &
    "
dcf_file =\" file : / / F : / Thesis / Lane_Change . xml \"␣&" , &
          "␣log_file=yes␣&" , &
    "␣comment =\"\" "
variable delete variable =.ACAR. dboxes . dbox_ana_ful_sdi_sub . prefix ,
.ACAR. dboxes . dbox_ana_ful_sdi_sub . "tmp*"
acar analysis submit

numeric_results write &
   result_set_component_name =
. Vehicle . DLC_Simulation . Front_Left_Corner .
TIME,
. Vehicle . DLC_Simulation . Front_Left_Corner .
Front_Left_x ,
. Vehicle . DLC_Simulation . Front_Left_Corner .
Front_Left_y &
   sort_by = by_time &
   order = ascending &
   write_to_terminal = off &
   file_name = "Front_Left_Corner . txt" &
     &

numeric_results write &
   result_set_component_name =
. Vehicle . DLC_Simulation . Front_Right_Corner
. TIME,
. Vehicle . DLC_Simulation . Front_Right_Corner
. Front_Right_x ,
.Vehicle.DLC_Simulation.Front_Right_Corner
.Front_Right_y &
   sort_by = by_time  &
   order = ascending  &
   write_to_terminal = off  &
   file_name = "Front_Right_Corner.txt"  &
     &

numeric_results write  &
   result_set_component_name =
.Vehicle.DLC_Simulation.Rear_Left_Corner.
TIME,
.Vehicle.DLC_Simulation.Rear_Left_Corner.
Rear_Left_x,
.Vehicle.DLC_Simulation.Rear_Left_Corner.
Rear_Left_y &
   sort_by = by_time  &
   order = ascending  &
   write_to_terminal = off  &
   file_name = "Rear_Left_Corner.txt"  &
     &

numeric_results write  &
   result_set_component_name =
.Vehicle.DLC_Simulation.Rear_Right_Corner.
TIME,
.Vehicle.DLC_Simulation.Rear_Right_Corner.
Rear_Right_x,
.Vehicle.DLC_Simulation.Rear_Right_Corner.
Rear_Right_y &
   sort_by = by_time  &
   order = ascending  &
   write_to_terminal = off  &
   file_name = "Rear_Right_Corner.txt"  &
     &

analysis delete&
        analysis_name = DLC_Simulation
```

APPENDIX E

```
# Double Lane Change Python File
import Adams
from os import path, mkdir, chdir
from math import atan, sqrt, exp, tan
import numpy as np
from numpy import sin, tanh, pi, cos, arctan
from numpy.random import randint
from random import random as rnd
from random import gauss, randrange

# Output text file name
Result_file = 'DLC_Result.txt'

# Selection
#selection_type = 'Fittest Half'
selection_type = 'Roulette␣Wheel'

# Pairing
#pairing_type = 'Random'
pairing_type = 'Fittest'

# Crossover
#crossover_type = 'Single Point'
crossover_type = 'Single␣Station'

# Mutation
mutation_type = 'Reset'
#Omutation_type = 'Gauss'

# Maximum steering angles
# Left Front, right front, left front 2, right front 2
delta_1_max = 29
delta_2_max = 26.5
delta_3_max = 16
delta_4_max = 13.6

# First and second axle steering ratio
```

```
Second_to_first = delta_3_max/delta_1_max

# Main Folder
MainFolderPath = path.abspath(r'F:\Thesis')

# Duration of the steering manuver (seconds)
Ts = 20

# Command files
command_file = 'F:\\Thesis\\Lane_change.cmd'
Steer_input_2 = 'F:\\Thesis\\Steer_input_2.cmd'
Steer_input_3 = 'F:\\Thesis\\Steer_input_3.cmd'
Steer_input_1 = 'F:\\Thesis\\Steer_input_1.cmd'
Steer_input_4 = 'F:\\Thesis\\Steer_input_4.cmd'
Steer_input_5 = 'F:\\Thesis\\Steer_input_5.cmd'
Steer_input_6 = 'F:\\Thesis\\Steer_input_6.cmd'
Steer_input_7 = 'F:\\Thesis\\Steer_input_7.cmd'
Steer_input_8 = 'F:\\Thesis\\Steer_input_8.cmd'
Input_list = [Steer_input_1, Steer_input_2,
        Steer_input_3, Steer_input_4, Steer_input_5,
        Steer_input_6, Steer_input_7, Steer_input_8]

# Files
file = 'F:\\Thesis\\'

# Input file location
Input_files = file + 'Steer_Inputs\\'

# Steering motions in Adams
L1_motion = Adams.stoo(
    '.DLC_Vehicle.Vehicle_front_suspension.MOTION_2')
L2_motion = Adams.stoo(
    '.DLC_Vehicle.Vehicle_front2_suspension.MOTION_2')
L3_motion = Adams.stoo(
    '.DLC_Vehicle.Vehicle_rear_suspension.MOTION_2')
L4_motion = Adams.stoo(
    '.DLC_Vehicle.Vehicle_rear2_suspension.MOTION_2')
R1_motion = Adams.stoo(
    '.DLC_Vehicle.Vehicle_front_suspension.MOTION_1')
R2_motion = Adams.stoo(
    '.DLC_Vehicle.Vehicle_front2_suspension.MOTION_1')
R3_motion = Adams.stoo(
    '.DLC_Vehicle.Vehicle_rear_suspension.MOTION_1')
R4_motion = Adams.stoo(
    '.DLC_Vehicle.Vehicle_rear2_suspension.MOTION_1')

# Locations
```

```
front_axle_loc = Adams.stoo
        ('.DLC_Vehicle.Body.ground.origo').location
front2_axle_loc = Adams.stoo(
    '''.DLC_Vehicle.Vehicle_front2_suspension.
        ground.hps_ground_clearance''').location
rear_axle_loc = Adams.stoo(
    '''.DLC_Vehicle.Vehicle_rear_suspension
        .ground.hps_ground_clearance''').location
rear2_tire_loc = Adams.stoo(
    '''.DLC_Vehicle.Vehicle_rear2_suspension
        .ground.hpr_wheel_center''').location
rear2_axle_loc = Adams.stoo(
    '''.DLC_Vehicle.Vehicle_rear2_suspension
        .ground.hps_ground_clearance''').location


# Length and width of the vehicle
veh_len_x = front_axle_loc[0]-rear2_axle_loc[0]
veh_len_y = rear2_tire_loc[1]*2


# Crossing point calculation
# Crossing point distance for the first axle
l1 = veh_len_y/(1/tan(delta_2_max)-1/tan(delta_1_max))
# Crossing point distance for the first axle
l2 = veh_len_y/(1/tan(delta_3_max)-1/tan(delta_4_max))

# Creating the splines
# Time of the curve -same for all of the spline files
t = np.linspace(0, 20, num=500, endpoint=True)


# Rear steering input (0 in this case)
Steering_angle_input_rear = np.zeros(500)


# Creating the steering inpur spline text files
chdir(MainFolderPath+'\\Steer_Inputs')


for x in range(1, 9):
    input_text = Input_files+'Steer_input_'+str(x)+'.txt'
    with open(input_text, 'w') as s_text:
        for y in range(len(t)):
            s_text.write(str(t[y])+'␣␣␣␣'+str(t[y])+'\n')
    Adams.read_command_file(Input_list[x-1])
```

```
# Modifying the Motions
Steer_Spline_1 = Adams.stoo('.DLC_Vehicle.SPLINE_1')
Steer_Spline_2 = Adams.stoo('.DLC_Vehicle.SPLINE_2')
Steer_Spline_3 = Adams.stoo('.DLC_Vehicle.SPLINE_3')
Steer_Spline_4 = Adams.stoo('.DLC_Vehicle.SPLINE_4')
Steer_Spline_5 = Adams.stoo('.DLC_Vehicle.SPLINE_5')
Steer_Spline_6 = Adams.stoo('.DLC_Vehicle.SPLINE_6')
Steer_Spline_7 = Adams.stoo('.DLC_Vehicle.SPLINE_7')
Steer_Spline_8 = Adams.stoo('.DLC_Vehicle.SPLINE_8')


# Inputing zero steering angle to the rear tires
Steer_Spline_3.y = Steering_angle_input_rear
Steer_Spline_4.y = Steering_angle_input_rear
Steer_Spline_7.y = Steering_angle_input_rear
Steer_Spline_8.y = Steering_angle_input_rear


# Using the splines in the Steering Motions in radians
L1_motion.function = 'CUBSPL(time,␣0,␣.DLC_Vehicle.SPLINE_1)*PI/180'
L2_motion.function = 'CUBSPL(time,␣0,␣.DLC_Vehicle.SPLINE_2)*PI/180'
L3_motion.function = 'CUBSPL(time,␣0,␣.DLC_Vehicle.SPLINE_3)*PI/180'
L4_motion.function = 'CUBSPL(time,␣0,␣.DLC_Vehicle.SPLINE_4)*PI/180'
R1_motion.function = 'CUBSPL(time,␣0,␣.DLC_Vehicle.SPLINE_5)*PI/180'
R2_motion.function = 'CUBSPL(time,␣0,␣.DLC_Vehicle.SPLINE_6)*PI/180'
R3_motion.function = 'CUBSPL(time,␣0,␣.DLC_Vehicle.SPLINE_7)*PI/180'
R4_motion.function = 'CUBSPL(time,␣0,␣.DLC_Vehicle.SPLINE_8)*PI/180'


def Second_axle_steering(First_axle_steering):
    second_axle_steering = []
    for x in range(len(First_axle_steering)):
        if First_axle_steering[x] == 0:
            second_axle_steering.append(0)
        else:
            second_axle_steering.append(
                arctan(1/(veh_len_y/l1+
                        (1/tan(First_axle_steering[x]*
                                pi/180))))*180/pi)
    return second_axle_steering


def Fourth_axle_steering(Third_axle_steering):
    fourth_axle_steering = []
    for x in range(len(Third_axle_steering)):
        if Third_axle_steering[x] == 0:
            fourth_axle_steering.append(0)
        else:
            fourth_axle_steering.append(
                arctan(1/(veh_len_y/l1+
                        (1/tan(Third_axle_steering[x]*
                                pi/180))))*180/pi)
    return fourth_axle_steering


def individual():
    a1 = round(rnd()*1.2+3,2)
    b1 = 3.25
    c1 = round(a1+(rnd()),2)
    d1 = round(b1+(rnd()*0.8+0.15),2)
    a2 = round(rnd()*1.2+3,2)
    b2 = round(d1+(rnd()*0.5+1.25),2)
    c2 = round(a2+(rnd()),2)
    d2 = round(b2+(rnd()*0.6+0.4),2)
    return [a1, b1, c1, d1, a2, b2, c2, d2]


def population(number_of_individuals):
    return [individual() for x in range(number_of_individuals)]


def cost_calculation(x, y, initial=42000):
    cost = 0
    W1 = 1.1*veh_len_y+250
    W2 = 1.2*veh_len_y+250
    for n in range(len(x)):
    # First Region
        if x[n] < 15000+initial:
            if y[n] > W1/2:
                cost += y[n]-W1/2
            if y[n] < -W1/2:
                cost += abs(y[n]+W1/2)
    # Second Region
        elif x[n] >= 15000+initial
                and x[n] < 15000+24000+veh_len_x+initial:
            if y[n] > -W1/2+3500+W2:
                cost += y[n]-(-W1/2+3500+W2)
            if y[n] < -W1/2:
                cost += abs(y[n]+W1/2)
    # Third Region
        elif x[n] >= 15000+24000+veh_len_x+initial
                and x[n] < 15000+24000+veh_len_x+25000+initial:
            if y[n] > -W1/2+3500+W2:
                cost += y[n]-(-W1/2+3500+W2)
            if y[n] < -W1/2+3500:
                cost += abs(y[n]-(-W1/2+3500))
    # Fourt Region
        elif x[n] >= 15000+24000+veh_len_x+25000+initial
                and x[n] < 15000+24000+veh_len_x+
                        25000+24000+veh_len_x+initial:
            if y[n] > -W1/2+3500+W2:
                cost += y[n]-(-W1/2+3500+W2)
            if  y[n] < -W1/2:
                cost += abs(y[n]+W1/2)
    # Fifth region
        elif x[n] >= 15000+24000+veh_len_x+
                25000+24000+veh_len_x+initial
                and x[n] < 15000+24000+veh_len_x+
                        25000+24000+veh_len_x+15000+initial:
            if y[n] > W1/2:
                cost += y[n]-W1/2
            if y[n] < -W1/2:
                cost += abs(y[n]+W1/2)
    return cost


def fit_test(Individual):

    Steering_angle_input = (Individual[0]*cos(pi*tanh(
        t-Individual[1]))-Individual[2]*
                cos(pi*tanh(t-Individual[3])) +
            (Individual[0]-Individual[2]))-
                (Individual[4]*cos(pi*tanh(t-Individual[5]))-
                        Individual[6]*cos(pi*tanh(
                                t-Individual[7]))+
                                        (Individual[4]-Individual[6]))

    Steer_Spline_5.y = Steering_angle_input
    Steer_Spline_1.y = Second_axle_steering(Steering_angle_input)
    Steer_Spline_6.y = Steering_angle_input*Second_to_first
    Steer_Spline_2.y = Fourth_axle_steering(
        Steering_angle_input*Second_to_first)

    Adams.read_command_file(command_file)

    # Reading simulation results
    Front_Left = open('Front_Left_Corner.txt', 'r').readlines()
    Front_Right = open('Front_Right_Corner.txt', 'r').readlines()
    Rear_Left = open('Rear_Left_Corner.txt', 'r').readlines()
    Rear_Right = open('Rear_Right_Corner.txt', 'r').readlines()
    # Result parameters
    Front_Left_x = []
    Front_Left_y = []
    Front_Right_x = []
    Front_Right_y = []
    Rear_Left_x = []
    Rear_Left_y = []
    Rear_Right_x = []
    Rear_Right_y = []

    # Creating lists of the results
    for x in range(9, len(Front_Left)):
        Front_Left_x.append(−float(Front_Left[x].split()[1]))
        Front_Left_y.append(float(Front_Left[x].split()[2]))
        Front_Right_x.append(−float(Front_Right[x].split()[1]))
        Front_Right_y.append(float(Front_Right[x].split()[2]))
        Rear_Left_x.append(−float(Rear_Left[x].split()[1]))
        Rear_Left_y.append(float(Rear_Left[x].split()[2]))
        Rear_Right_x.append(−float(Rear_Right[x].split()[1]))
        Rear_Right_y.append(float(Rear_Right[x].split()[2]))

    if len(Front_Left_x) < 140:
        cost = 1000000000
    else:
        cost = cost_calculation(Front_Left_x, Front_Left_y) +
                cost_calculation(Front_Right_x, Front_Right_y) + \
            cost_calculation(
            Rear_Left_x, Rear_Left_y) +
                cost_calculation(Rear_Right_x, Rear_Right_y)
    return −cost


def first_generation(pop):
    fitness = [fit_test(pop[x]) for x in range(len(pop))]
    sorted_fitness = sorted([[pop[x], fitness[x]]
        for x in range(len(pop))], key=lambda x: x[1])
    population = [sorted_fitness[x][0]
        for x in range(len(sorted_fitness))]
    fitness = [sorted_fitness[x][1]
        for x in range(len(sorted_fitness))]
    return {'Individuals': population, 'Fitness': sorted(fitness)}


def roulette(cum_sum, chance):
    veriable = list(cum_sum.copy())
    veriable.append(chance)
    veriable = sorted(veriable)
    return veriable.index(chance)


def selection(generation, method='Fittest␣Half'):
    generation['Normalized␣Fitness'] = sorted(
        [generation['Fitness'][x] /sum(generation['Fitness'])
                for x in range(len(generation['Fitness']))],
                        reverse = True)
    generation['Cumulative␣Sum'] = np.array(
        generation['Normalized␣Fitness']).cumsum()
    if method == 'Roulette␣Wheel':
        selected = []
        for x in range(len(generation['Individuals'])//2):
            selected.append(roulette(generation[
                'Cumulative␣Sum'], rnd()))
            while len(set(selected)) != len(selected):
                selected[x] = (roulette(generation[
                        'Cumulative␣Sum'], rnd()))
        selected = {'Individuals': [generation[
                'Individuals'][int(selected[x])]
                for x in range(len(generation[
                        'Individuals'])//2)],
                                'Fitness': [generation[
                                        'Fitness'][int(selected[x])]
                    for x in range(len(generation[
                        'Individuals'])//2)]
                    }
    elif method == 'Fittest␣Half':
        selected_individuals = [generation[
                'Individuals'][-x-1]
            for x in range(int(len(generation['Individuals'])//2))]
        selected_fitnesses = [generation['Fitness'][-x-1]
            for x in range(int(len(generation['Individuals'])//2))]
        selected = {'Individuals': selected_individuals,
                    'Fitness': selected_fitnesses}
    return selected


def pairing(elit, selected, method = 'Fittest'):
    individuals = [elit['Individuals']]+selected['Individuals']
    fitness = [elit['Fitness']]+selected['Fitness']
    if method == 'Fittest':
        parents = [[individuals[x],individuals[x+1]]
                for x in range(len(individuals)//2)]
    if method == 'Random':
        parents = []
        for x in range(len(individuals)//2):
            parents.append([individuals[randint(0,
                (len(individuals)-1))],
                        individuals[randint(0,(len(individuals)-1))]])
            while parents[x][0] == parents[x][1]:
                parents[x][1] = individuals[
                        randint(0,(len(individuals)-1))]
    if method == 'Weighted␣Random':
        normalized_fitness = sorted([fitness[x]/
                sum(fitness) for x in
                        range(len(individuals)//2)], reverse = True)
        cummulitive_sum = np.array(normalized_fitness).cumsum()
        parents = []
        for x in range(len(individuals)//2):
            parents.append([individuals[
                roulette(cummulitive_sum, rnd())],
                        individuals[roulette(
                                cummulitive_sum, rnd())]])
            while parents[x][0] == parents[x][1]:
                parents[x][1] = individuals[
                        roulette(cummulitive_sum, rnd())]
    return parents


def crossover(parents, method='Single␣Station'):
    if method == 'Single␣Point':
        pivot_point = randint(1, 6)
        offsprings = [parents[0][0:pivot_point]+
                parents[1][pivot_point:]]
        offsprings.append(parents[1]
                [0:pivot_point]+parents[0][pivot_point:])
    if method == 'Single␣Station':
        section = randrange(0, 4)
        offsprings = [parents[0].copy()]
        if section == 0:
            offsprings[0][0] = parents[1][0]
            offsprings[0][2] = parents[1][2]
            offsprings.append(parents[1])
            offsprings[1][0] = parents[0][0]
            offsprings[1][2] = parents[0][2]
        elif section == 1:
            offsprings[0][1] = parents[1][0]
            offsprings[0][3] = parents[1][3]
            offsprings.append(parents[1])
            offsprings[1][1] = parents[0][1]
            offsprings[1][3] = parents[0][3]
        elif section == 2:
            offsprings[0][4] = parents[1][4]
            offsprings[0][6] = parents[1][6]
            offsprings.append(parents[1])
            offsprings[1][4] = parents[0][4]
            offsprings[1][6] = parents[0][6]
        elif section == 3:
            offsprings[0][5] = parents[1][5]
            offsprings[0][7] = parents[1][7]
            offsprings.append(parents[1])
            offsprings[1][5] = parents[0][5]
            offsprings[1][7] = parents[0][7]
    return offsprings


def mutation(individual, muatation_rate=2, method='Reset'):
    gene = [randint(0, 7)]
    for x in range(muatation_rate -1):
        gene.append(randint(0, 7))
        while len(set(gene)) < len(gene):
            gene[x] = randint(0, 7)
    mutated_individual = individual.copy()
    if method == 'Gauss':
        for x in range(len(gene)):
        # a1 parameter
            if gene[x] == 0:
                mutated_individual[0] =
                        round(individual[0]+gauss(0, 0.001), 4)
        # c1 parameter
            elif gene[x] == 2:
                mutated_individual[2] =
                        round(individual[2]+gauss(0, 0.001), 4)
        # d1 parameter3
            elif gene[x] == 3:
                mutated_individual[3] =
                        round(individual[3]+gauss(0, 0.001), 4)
        # a2 parameter
            elif gene[x] == 4:
                mutated_individual[4] =
                        round(individual[4]+gauss(0, 0.001), 4)
        # b2 parameter
            elif gene[x] == 5:
                mutated_individual[5] =
                        round(individual[5]+gauss(0, 0.001), 4)
        # c2 parameter
            elif gene[x] == 6:
                mutated_individual[6] =
                        round(individual[6]+gauss(0, 0.001), 4)
        # d2 parameter
            elif gene[x] == 7:
                mutated_individual[7] =
                        round(individual[7]+gauss(0, 0.001), 4)
    if method == 'Reset':
        for x in range(len(gene)):
        # a1 parameter
            if gene[x] == 0:
                mutated_individual[0] =
                        round(rnd()*1.2+3,2)
        # c1 parameter
            elif gene[x] == 2:
                mutated_individual[2] =
                        round(mutated_individual[0]+(rnd()),2)
        # d1 parameter
            elif gene[x] == 3:
                mutated_individual[3] =
                        round(mutated_individual[1]+
                                (rnd()*0.8+0.15),2)
        # a2 parameter
            elif gene[x] == 4:
                mutated_individual[4] =
                        round(rnd()*1.2+3,2)
        # b2 parameter
            elif gene[x] == 5:
                mutated_individual[5] =
                        round(mutated_individual[2]+
                                (rnd()*0.5+1.25),2)
        # c2 parameter
            elif gene[x] == 6:
                mutated_individual[6] =
                        round(mutated_individual[4]+
                                (rnd()),2)
        # d2 parameter
            elif gene[x] == 7:
                mutated_individual[7] =
                        round(mutated_individual[5]+
                                (rnd()*0.6+0.4),2)
    return mutated_individual


def next_generation(gen):
    elit = {}
    next_gen = {}
    elit['Individuals'] = gen['Individuals'].pop(-1)
    elit['Fitness'] = gen['Fitness'].pop(-1)
    selected = selection(gen,selection_type)
    parents = pairing(elit, selected, pairing_type)
    offsprings = [[[crossover(parents[x])
        for x in range(len(parents))]
            [y][z] for z in range(2)]
                for y in range(len(parents))]
    offsprings1 = [offsprings[x][0]
                   for x in range(len(parents))]
    offsprings2 = [offsprings[x][1]
                   for x in range(len(parents))]
    unmutated = selected['Individuals']+
        offsprings1+offsprings2
    mutated = [mutation(unmutated[x], mutatoion_type)
        for x in range(len(gen['Individuals']))]
    unsorted_individuals = mutated +
        [elit['Individuals']]
    unsorted_next_gen = [fit_test(mutated[x])
        for x in range(len(mutated))]
    unsorted_fitness = [unsorted_next_gen[x]
         for x in range(len(gen['Fitness']))] +
                [elit['Fitness']]
    sorted_next_gen = sorted([[unsorted_individuals[x],
        unsorted_fitness[x]]
        for x in range(len(unsorted_individuals))],
                key=lambda x: x[1])
    next_gen['Individuals'] = [sorted_next_gen[x][0]
        for x in range(len(sorted_next_gen))]
    next_gen['Fitness'] = [sorted_next_gen[x][1]
        for x in range(len(sorted_next_gen))]
    gen['Individuals'].append(elit['Individuals'])
    gen['Fitness'].append(elit['Fitness'])
    return next_gen


# Seting the working directory
chdir(MainFolderPath+'\\Lane_Change_Results')




# Creating the First Generation
pop = population(16)
gen = []
gen.append(first_generation(pop))
fitness_avg = np.array([sum(gen[0]['Fitness'])/
        len(gen[0]['Fitness'])])
fitness_max = np.array([max(gen[0]['Fitness'])])
res = open(Result_file, 'a')
res.write('\n'+str(gen)+'\n')
res.close()


# The function that checks the fitness to find repetition
def fitness_similarity_chech(max_fitness, number_of_similarity):
    result = False
    similarity = 0
    for n in range(len(max_fitness)−1):
        if max_fitness[n] == max_fitness[n+1]:
            similarity += 1
        else:
            similarity = 0
    if similarity == number_of_similarity −1:
        result = True
    return result


for x in range(2):
    gen.append(next_generation(gen[x]))
    fitness_avg = np.append(fitness_avg, sum(
        gen[x]['Fitness'])/len(gen[x]['Fitness']))
    fitness_max = np.append(fitness_max,
        max(gen[x]['Fitness']))
    res = open(Result_file, 'a')
    res.write('\n'+str(gen[x+1])+'\n')
    res.close()


finish = False
while finish == False:
    if max(fitness_max) > 0:
        break
    if max(fitness_avg) > −20000:
        break
    if fitness_similarity_chech(fitness_max, 50) == True:
        break
    gen.append(next_generation(gen[−1]))
    fitness_avg = np.append(fitness_avg, sum(
        gen[−1]['Fitness'])/len(gen[−1]['Fitness']))
    fitness_max = np.append(fitness_max,
        max(gen[−1]['Fitness']))
    res = open(Result_file, 'a')
    res.write('\n'+str(gen[−1])+'\n')
    res.close()


res = open(Result_file, 'a')
```

```
res.write('Maximum Fitness = '+str(fitness_max) +
          ', Avarage Fitness'+str(fitness_avg))
res.close()
```

HACETTEPE UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING
THESIS/DISSERTATION ORIGINALITY REPORT

HACETTEPE UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING
TO THE DEPARTMENT OF MECHANICAL ENGINEERING

Date:09/07/2019

Thesis Title / Topic: STEERING OPTIMIZATION OF AN 8X8 VEHICLE

According to the originality report obtained by my thesis advisor by using the *Turnitin* plagiarism detection software and by applying the filtering options stated below on 09/07/2019 for the total of 115 pages including the a) Title Page, b) Introduction, c) Main Chapters, d) Conclusion sections of my thesis entitled as above, the similarity index of my thesis is 7%.

Filtering options applied:

1. Bibliography/Works Cited excluded
2. Quotes excluded
3. Match size up to 5 words excluded

I declare that I have carefully read Hacettepe University Graduate School of Sciene and Engineering Guidelines for Obtaining and Using Thesis Originality Reports; that according to the maximum similarity index values specified in the Guidelines, my thesis does not include any form of plagiarism; that in any future detection of possible infringement of the regulations I accept all legal responsibility; and that all the information I have provided is correct to the best of my knowledge.

I respectfully submit this for approval.

09/07/2019

Name Surname: Cahit Bartu YAZICI

Student No: N16222041

Department: Mechanical Engineering

Program: Mechanical Engineering

Status: ☒ Masters ☐ Ph.D. ☐ Integrated Ph.D.

ADVISOR APPROVAL

APPROVED.

Emir Kutluay

Asst.Prof.Dr. Emir Kutluay

CURRICULUM VITAE

Credentials

Name, Surname : Cahit Bartu YAZICI

Place of Birth : Ankara, TURKEY

Marital Status : Married

E-mail : cbyazici@gmail.com

Adress : Ankara, TURKEY

Education

BSc. : Gazi University

MSc. : –

PhD. : –

Foreign Language

English

Work Experience

Simulation Engineer, FNSS Defence Systems (2016 -)

Areas of Experiences

-

Projects and Budgets

-

Publications

-

Oral and Poster Presentation

-