ISSN (Print) : 2319-8613
ISSN (Online) : 0975-4024

Development of Kinematic Suspension Simulator for Double A-Arm suspension

Rishabh Bhatia [#1]

[#]Mechanical Engineering Department, VIT University
Near Katpadi Road, Vellore, Tamil Nadu 632014, India
[1] rishabh.bhatia2013@vit.ac.in

***Abstract*—The paper focuses on mathematical modelling of Double-A-Arm suspension and then using this model to obtain verifiable suspension characteristics as outputs. The program is made entirely from C++ and uses 3D vector modelling to calculate the desired parameters. Currently, software exists which can perform the same task but their algorithm is copyrighted hence secluded. The paper attempts to describe a source code which performs the same tasks as the software**

Keyword- Double A-Arm suspension, Suspension simulator, Kinematic simulator, Algorithm development.

I. INTRODUCTION

A suspension is nothing but a set of rigid links which at one end is connected to the body of the vehicle and at the other end at upright. An upright is a rigid body which is connected to the wheel indirectly (via a hub) and is responsible for passing all the motions and forces from the wheel to the links and vice-versa. The main aim of any suspension is to keep the tire upright at all times and maintain tire contact patch as effectively as possible. As the suspension moves up and down, the relative orientation of the links changes and the tire no longer remains perpendicular to the ground which leads to a change in toe, camber, caster, scrub and many other parameters. These parameters need to be minimized in most cases and optimized in some. These parameters behave in a nonlinear interdependent fashion when subjected to steering input and wheel travel and hence a balance needs to be achieved between these to obtain desirable kinematic characteristics of the system. To do this a software or simulator is required which performs the computation and gives these parameters as output. Currently existing soft wares like M.S.C Adams Car, Optimum K, Lotus perform these tasks but are either very expensive or inaccessible easily hence are often out of reach of many.

Apart from soft wares, strides have been made to calculate and understand these characteristics [3],[4].These references thoroughly explain modelling and analysis of the suspension on MATLAB[3]but one still requires MATLAB and the know how to use it. Rakholia [4] uses 3d vectors and closed loop concepts to arrive at a model for McPherson strut suspension but how these 3D vector formulations are plotted isn't clearly disclosed.

Fig.1-Top left upper view of Double A-Arm suspension

Thus there remains a gap or absence of tools in this domain. The proposed simulator aims to fill this gap. This simulator is based entirely on C++ which is free and open source software. Moreover the concepts used here give an in depth methodology of how to model 3D vectors into a high level programming language which can be used to model other types of suspension as well. Fig1 here shows double A-Arm suspension for better visualization of the title and author block, the appearance of section headings, document margins, column width, column spacing and other features.

II. Background

The Double-A-Arm suspension as shown in Fig.2 consists of two triangular rigid bodies. Both these A-Arms are connected at two ends to the body and at one end to the upright. All the joints are ball socket joints which allow three degrees of freedom of rotation. Both the upper and lower triangles (from now on called upper A-arm and lower A-arm) are connected the other end by upright. Another link connected to the upright which is called tie rod, is responsible for steering. This tie rod is indirectly connected to the body via rack. Rack has single translational degree of freedom and can slide in and out of its casing depending on input from steering wheel. Thus now the wheel can travel only up and down and rotate about Kingpin Axis (K.P.I.) but only on steering input, hence it has two degrees of freedom. Fig.2 shows all the hard points of the double A-Arms.

Hard points are the key points which according to their relative orientation decide suspension characteristics. They can either be joints (like UBJ, LBJ, tie rod) or any point of interest (like wheel centre) .They are basically input points for the simulation.

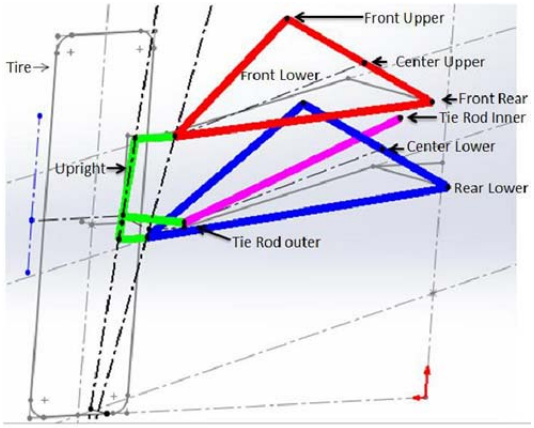

Fig 2 Top left upper view of double A-Arm Red designates Upper A-Arm, Blue designates Lower A-Arm, Green designates Upright, Black designates tire, Pink designates tie rod

III. Body

A. *About The simulator*

This is a kinematic suspension simulator for double A-Arm suspension which calculates the arcs traced by individual suspension components like ball joints, tie rod end, wheel centre etc. using mathematically modelled 3D vectors. The program is capable of calculating and displaying each and every characteristic of the suspension, from camber, toe and caster to Cartesian co-ordinates of ball joints, wheel centres, instantaneous centres and rack displacement.

B. *Input*

The program first asks for coordinates of hard points (in mm) and then action choice- Wheel travel (bump) or steer or both simultaneously. Then it asks for values of either rack travel or wheel travel (in mm) depending on user's previous choice.

C. *Output*

The standard outputs are coordinates of UBJ, LBJ, tie rod outer, wheel centre, U_X ,U_Y, U_Z ,L_X, L_Y , L_Z, camber ,caster, toe but the program can be modified to display any other suspension characteristics.

D. *Environmental details*

The simulator computes all the characteristics for left hand side geometry with origin at geometric centre of tire contact patch. The code keeps the body of the vehicle at rest at all times and moves the suspension

components up and down as per the input. All the inputs are supposed to be given in mm. Y is the vertical axis, X goes to the left and Z axis follows along vehicle centreline. Fig.2 here shows X and Y axis going left and up respectively. Z axis is going into the page.

E. *Basic Methodology*

The program basically uses the inputs from the user, divides it into small amounts and then adds this to current parameters and makes the changes. For example when the user gives wheel travel input as 50mm, the program divides this amount into 100 parts (.5mm). This amount is added to the designated variable (here UBJ_Y coordinate) now since UBJ has a particular locus, UBJ_X and UBJ_Z coordinate are also modified according to its directional parameters (U_X, U_Y, U_Z) so as to keep UBJ within that locus. In each iteration it calculates the changes to be brought to the corresponding parameters attached to the variable (here U_X, Direction of propagation, UBJ_X, UBJ_Y etc.). Theses changes are reflected in the next iteration and the process continues until the wheel has travelled a height of 50mm.

F. *Flow of control*

First the program asks for input of hard points i.e. initial coordinates of wheel centre, UBJ, LBJ, tie rod inner etc. Then the program asks for operation type i.e. parallel wheel travel or parallel wheel travel with steer. The first step to model the suspension is to find out the angles of the pivot direction of the A-Arms with respect to the axes. These angles are called as U_X, U_Y, U_Z for upper and L_X, L_Y, L_Z for the lower A-Arm. U_X is the angle essentially between the two lines lying in a plane parallel to XY. Line I is a line parallel to X-Axis through the point centre upper. centre upper lies at the intersection of vector joining front upper and front lower with a plane parallel to XY containing UBJ in it. Line II is a line parallel to X-axis containing center upper. The angle between these two lines is U_X (changes w.r.t. travel, steer). U_Y (remains constant w.r.t travel, steer) is the angle between the line projection of the line joining front upper and front lower on YZ plane with another line parallel to Z-axis containing front lower. U_Z (remains constant w.r.t travel, steer) is the angle between the line joining front upper and front lower with it's projection on YZ plane. These angles are indispensable for calculation as they are responsible for determining trajectories of A-Arms. Fig.3 explains the process.

The first thing that happens in the code is increment of UBJ_Y coordinate as per the wheel travel and then subsequent changes are made in the parameters related to UBJ (like U_X, UBJ_X , UBJ_Z etc.) ,then in lower ball joint ,then in tie rod outer and then later in the wheel centre. This order is necessary to maintain as the input in this program (wheel travel) is given to UBJ. It could have been given to LBJ also, but since UBJ gets the input, it is manipulated first. Now among LBJ, tie rod outer and wheel canter, LBJ is selected for further manipulation because both UBJ and LBJ have a fully defined motion (single degree of freedom i.e. rotation about their axes) as in their motion is independent of tie rod movement. Please note that all these hard points never have an independent motion with each other for e.g.- if even without wheel travel, tie rod outer points are changed, UBJ and LBJ will change. However, here UBJ and LBJ are taken independent of tie rod movement because this change is later incorporated in the code.

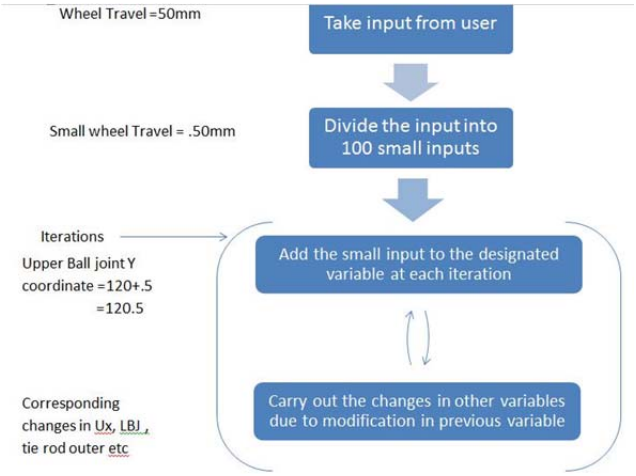

Fig 3-Process flowchart

Since it is well established now that the changes brought in UBJ, LBJ due to tie rod at this stage are irrelevant hence LBJ can be manipulated. Now between wheel centre and tie rod outer, tie rod outer is selected next for manipulation. This is for the simple reason that a rigid body's motion can be defined by 3 points maintaining a constant distance with each other in space. Here the three points can be taken as UBJ, LBJ and tie rod outer which act as input for upright motion and the wheel centre acts as output.

The manipulation done to all the three coordinates of UBJ and LBJ happen at same time. As in UBJ_X, UBJ_Y, UBJ_Z all get manipulated in the same function in the same iteration. Since all axes' values are being determined at once it can be said that the manipulations are handled simultaneously. But with tie rod outer this isn't the case. The coordinates of tie rod outer points are changed two at a time. For example in the program changes are made first to Y and Z coordinates then to X and Y and then then Y and Z coordinates. Reason for this is explained later. First changes are made in caster then camber and then toe. Thus 3 separate snippets of code do these modifications individually and all these snippets modify the coordinates in 3D space in the order mentioned.

During the developmental stages of the simulator it was observed that often changes made in the coordinates due to toe and camber changed the coordinates enough to disturb caster values. For example when tie rod outer points are modified for caster and then camber and then toe, the points just modified after toe adjustment deviated from the conditions that it satisfied previously for camber. Hence approximation algorithms are used so that the previous conditions are also maintained.

During the developmental stages of the simulator it was observed that often changes made in the coordinates due to toe and camber changed the coordinates enough to disturb caster values. For example when tie rod outer points are modified for caster and then camber and then toe, the points just modified after toe adjustment deviated from the conditions that it satisfied previously for camber. Hence approximation algorithms are used so that the previous conditions are also maintained.

IV. WORKING

A. Trajectory of upper ball joint

The concept used here is that the circle is traced but in small linear steps as shown in Fig.4 .The first objective is to find out the vector tangent to the circle on UBJ. Any vector requires direction as well as magnitude to completely define it. The direction again requires two more parameters (straight line in 3D). One is direction of propagation which is nothing but a vector perpendicular to the plane of the A-Arm and the second is coordinates of U.B.J. The magnitude is defined according to the wheel travel given as input by the user. This wheel travel is divided by 100 and then stored in another variable with some trigonometric manipulations. Now the direction and magnitude of propagation is given and UBJ is modified and the modified value is stored in another variable. As the upper A-Arm moves up U_X changes. If UBJ is below mid point upper then U_X value decreases with positive wheel travel and if UBJ is above then U_X value increases with positive wheel travel. These changes in the angle due to small wheel travel need to be accommodated in the angle hence it too is modified.

B. Trajectory of lower ball joint

Just like in UBJ, spatial orientation angles for LBJ are also calculated however calculation of its trajectory relies on a more complex relation as explained below.

UBJ and LBJ are two ends of upright which is a rigid body. If they were not held together by a rigid body they had the independence of moving in their separate arc freely but since that is not the case the lower ball joint has to maintain a specific constant distance between itself and the upper ball joint so as to be in accordance with the rigid body concept. When the first leg of the code is completed (when the UBJ is given upward travel) the distance between current UBJ and LBJ is measured. This new distance is then compared the original distance. If this new distance is larger a while loop is run in which LBJ is given very small wheel travel in its trajectory until this distance becomes equal to original distance. Thus with this condition the two points maintain a constant distance between them .This small change also brings a small change in Lx which is accommodated. A mathematical model is made so that this condition is always satisfied and hence the trajectory of lower a arm can be traced.

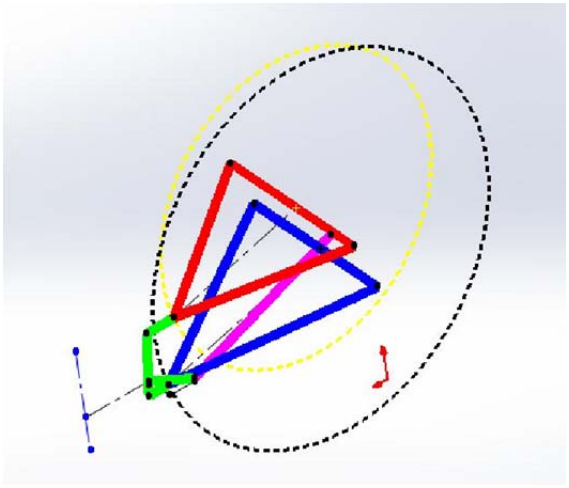

Fig.4-Trajectories of UBJ (yellow dashed), Trajectories of LBJ (Black dashed)

C. Caster angle determination

Since the final position of the new UBJ and LBJ have been determined, change in caster angle can be calculated. Original Caster angle value is stored beforehand from initial UBJ and LBJ and the small caster value can be added or subtracted for each iteration depending on its direction. Small caster angle is the angle calculated between the projections of lines drawn by connecting old UBJ and old LBJ and new UBJ and new LBJ on YZ plane.

D. Changes in tie rod outer due to caster

Since tie rod outer, UBJ, LBJ and wheel centre are a single rigid body hence rotation induced in the part due to movement of UBJ and LBJ must reflect in both tie rod outer and wheel centre. To determine the changes in tie rod outer point due to caster the following things need to be done.

I. *I.C. (Finding out instantaneous centre):* In case of change in caster angle, the upright rotates parallel to YZ plane about an instantaneous centre. If a rigid body is rotated in a plane then by law all the points rotate about a common centre called instantaneous centre[5]. According to this law all the points of a rigid body revolve around this point in the form of circle with different radii but same angular displacement. This instantaneous centre is found out in form of Cartesian coordinates and changes in every iteration as per orientation of the links.

II. To find coordinates of IC the old UBJ and LBJ positions are first copied to another set of variables called UBJ caster and LBJ caster. But z axis value of UBJ caster and LBJ caster is equated to that of UBJ new and LBJ new. This is done so as to translate the upright in Z direction as shown in Fig.5. Since now the upright is rotated relative to its previous orientation IC can be found out.

III. Next the angular displacement needs to be found out i.e. the angle by which the upright has rotated by about the IC. This is pretty straight forward. A line is drawn from UBJ old to I.C. and from I.C. to UBJ new. Angle between these two lines is found. out and is essentially the angular displacement. Since angular displacement for a rigid body about an IC remains the same hence this angle is used for further calculations here it is called small caster as this is also equal to the small caster change.

E. Camber angle determination

Old and new camber angle is found out with the help of old and new UBJ and LBJ positions. The difference between the angles is found out and is stored as small camber. The rest of the procedure is very much the same as for caster except that all the vector projections are considered on the YZ plane.

F. Changes in tie rod outer due to camber angle

The influence of camber on tie rod outer very much the same as caster . The only difference is that all the projections are carried out XY plane and initial camber angle is zero.

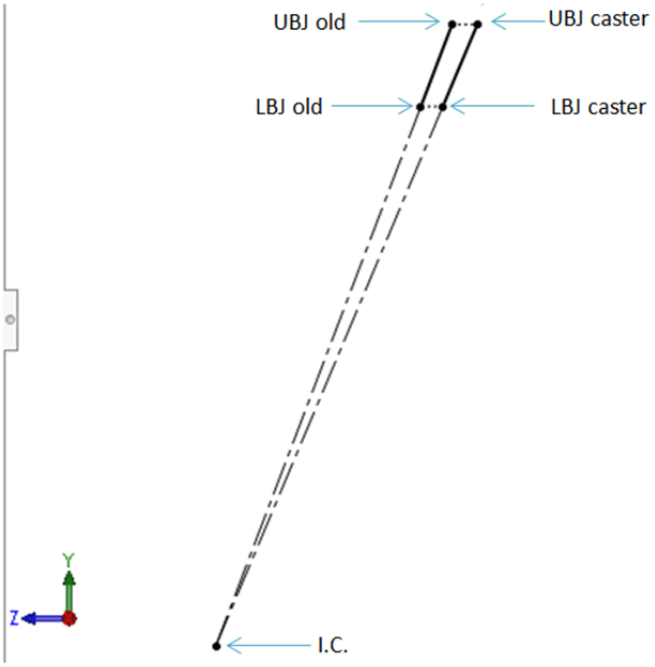

Fig.5- Projection of UBJ and LBJ on YZ plane

G. Toe angle variation

To determine toe angle, first of all direction of propagation of tie rod outer needs to be given. The whole wheel assembly as a rigid body can rotate only about KPI hence the locus of tie rod outer is essentially a circle in 3D space lying in a plane perpendicular to KPI and containing the point tie rod outer. These two constraints can fully define the circle. However to calculate variation in toe still two things need to be determined first the magnitude of toe change and second the direction of it inwards or outwards.

To compute these two remaining parameters rigid body concept is used. At the start of first iteration the tie rod length is calculated and stored. It is calculated as the distance between tie rod outer point and tie rod inner point. This distance remains same in accordance with rigid body concept that two points on a rigid body will maintain a constant distance no matter what the circumstance (except for mechanical fracture). As the upright along with the wheel assembly moves this distance changes. This new distance is stored as new tie rod length and if it is greater than the original tie rod length the upright is moved about KPI inwards (making toe out). This is achieved by reducing the tie rod length in small amounts and adjusting wheel toe until the new tie rod length becomes equal to original tie rod length. If it is lesser, the upright is moved outwards (making toe in). This is achieved by increasing the tie rod length in small amounts and adjusting wheel toe due to this small orientation until the tie rod length becomes equal to original tie rod length.

H. Change in camber due to toe adjustment

As explained before due to change in toe, the tie rod outer points get shifted. It was observed that the value of tie rod outer point computed with this program differed significantly from that obtained by commercially available soft wares. It was concluded that after modification of the tie rod outer point due to toe, its distances from UBJ and LBJ changed which is in violation of rigid body concept. To rectify this problem first of all direction vectors were established from UBJ to tie rod outer and LBJ to tie rod outer. Initial distance between tie rod outer and UBJ and tie rod outer with LBJ was stored and later compared to the new distance. If the new distance was found out to be lesser than original for e.g.- If the distance between tie rod outer and UBJ was found out to be more, then tie rod outer is moved in direction of the vector joining them until that distance is reached .

I. Determination of wheel centre

As established before any rigid body requires 3 non collinear revolute joints to constraint it completely. This has been achieved with the help of LBJ,UBJ and tie rod outer. Thus the upright is fully constrained in its

motion. Since wheel centre is also a part of this rigid body, its distances from the 3 points- UBJ, LBJ and tie rod outer should remain constant. This simple principle is used and the algorithm is developed.

First of all the original distance of wheel centre with the three points is stored in 3 different variables. These 3 variables are of vector type meaning that they have X,Y and Z components. Each of those components are then modified to make a unit direction vector with each one pointing from base point (UBJ or LBJ or tie rod outer) to wheel centre. At the end of each iteration these distances are checked and if the new distances are not in accordance with original distances then the distance is either shortened or lengthened in the direction of the vectors calculated before. This is done in an iterative process adding or subtracting very small amounts to the distance in the direction of the vector until the distance condition is satisfied for all 3 points.

V. Results And Discussions

Here the results of the program are validated against a well known commercial software –Optimum K. The program prints the results specifically Camber, Toe, Caster, Coordinates of hard points and many other parameters to a text file. These parameters along with their 100 iterative results are used to form the graphs in excel as shown below. These graphs are essentially set Cartesian coordinates joined together. Although any parameter could be selected , plotted and compared here three primary parameters- camber ,caster ,toe are selected here. Fig.6 here shows the suspension arrangement as seen in OptimumK.

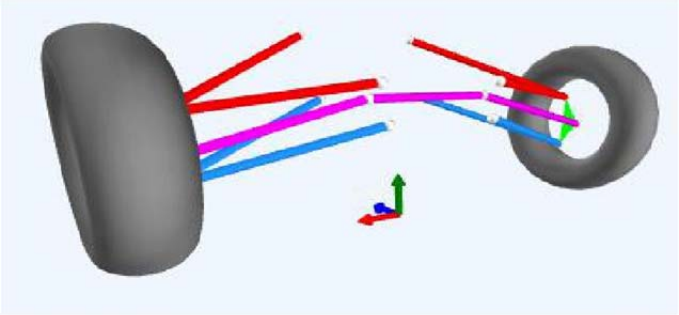

Figure 6-Suspension setup in Optimum K

The corresponding graphs are compared below for parallel wheel travel. The Y axis corresponds to the parameter (in degrees) and the X axis shows wheel travel (50mm) divided into steps.

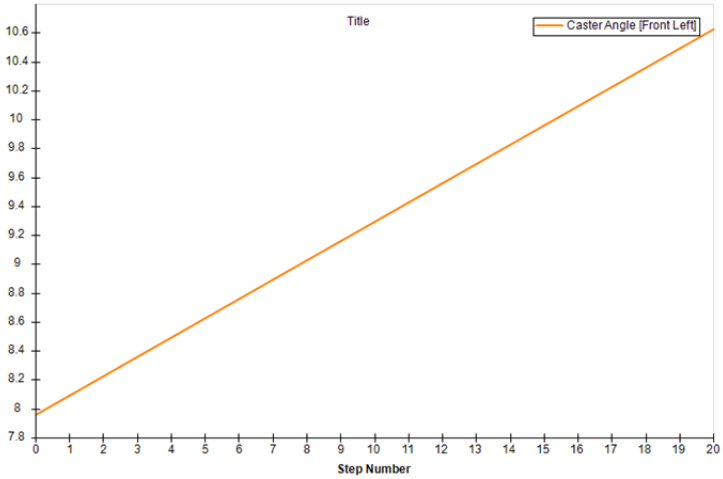

Figure 7-Caster angle vs wheel travel (OptimumK)

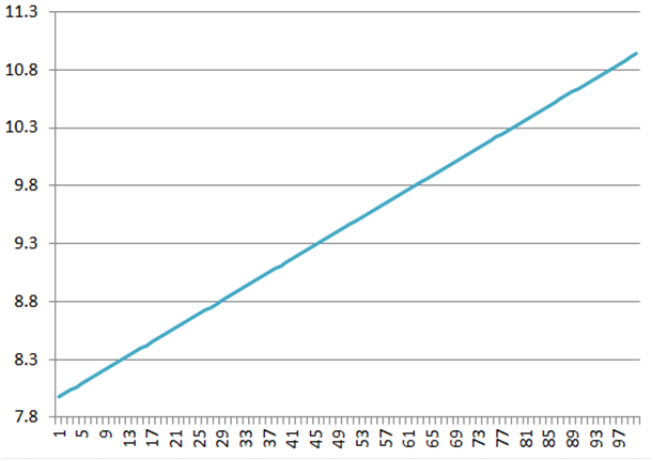

Figure 8-Caster angle vs wheel travel (simulator)

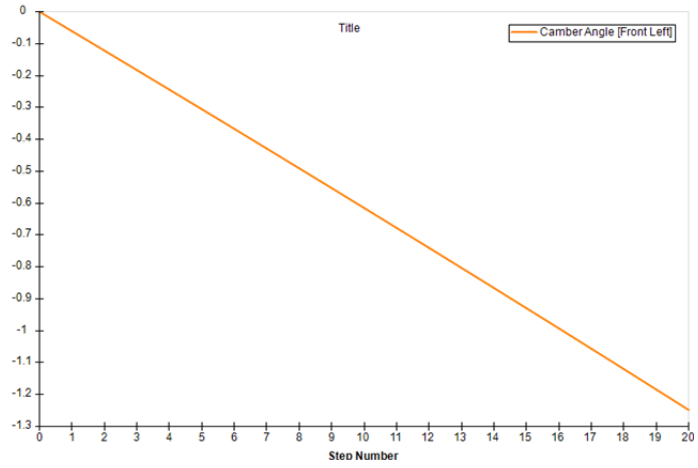

Figure 9-Camber angle vs wheel travel (OptimumK)

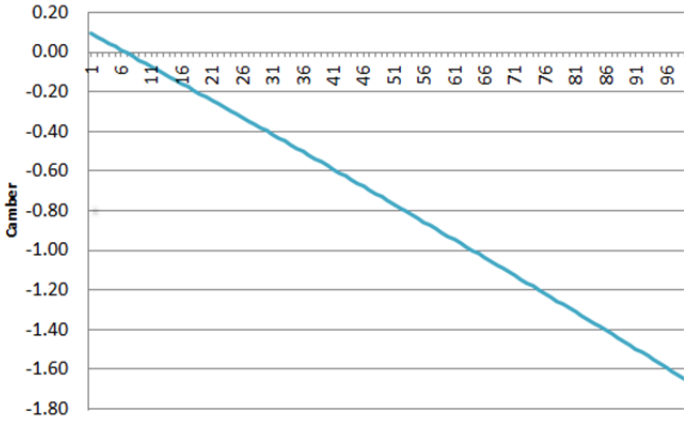

Figure 10-Camber angle vs wheel travel (Simulator)

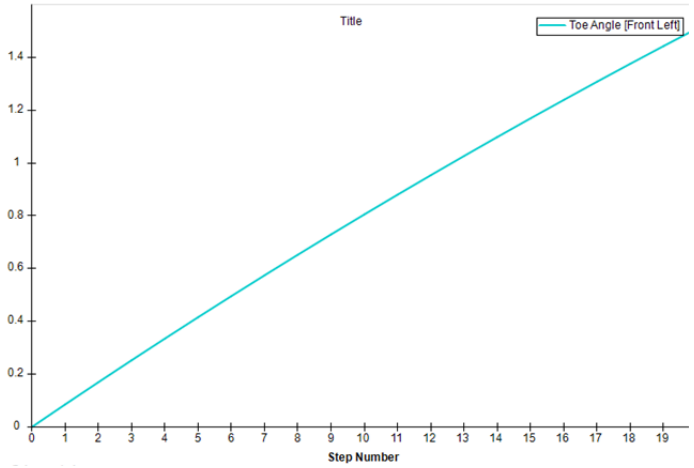

Figure 11-Toe angle vs wheel travel (OptimumK)

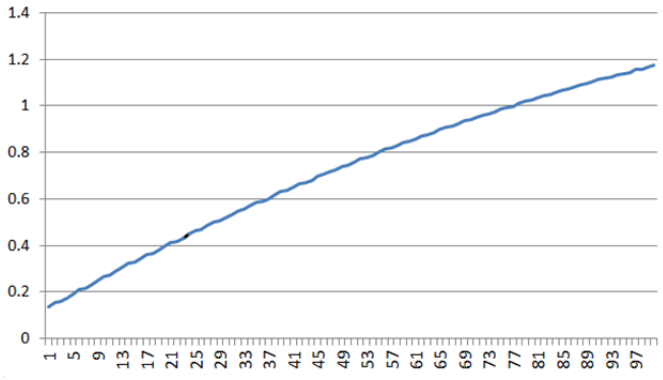

Figure 12-Toe angle vs wheel travel (Simulator)

Table 1

Characteristics comparison between simulator output and Optimum K output

Characteristics	Optimum K (degrees)	Simulator (degrees)
Caster	8 to 10.7	8 to 10.8
Camber	0 to -1.2	.1 to -1.45
Toe	0 to 1.35	.2 to 1.2

The following graphs compare the characteristics for steering input. Here the rack was displaced 20mm towards the left.

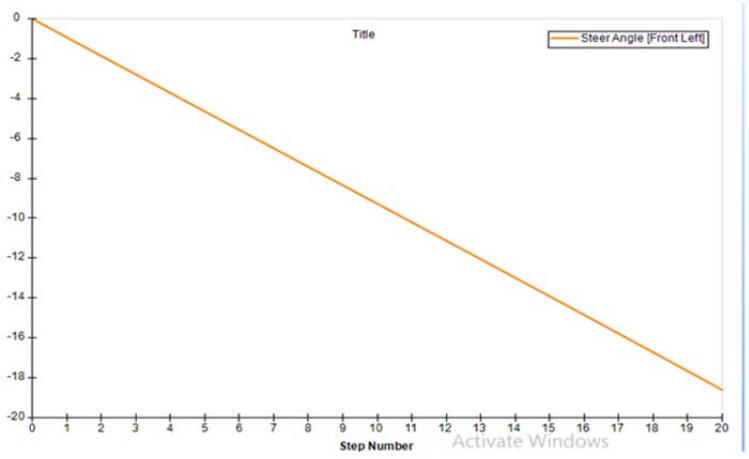

Figure 13-Steer angle vs rack travel(Optimum K)

ISSN (Print) : 2319-8613
ISSN (Online) : 0975-4024

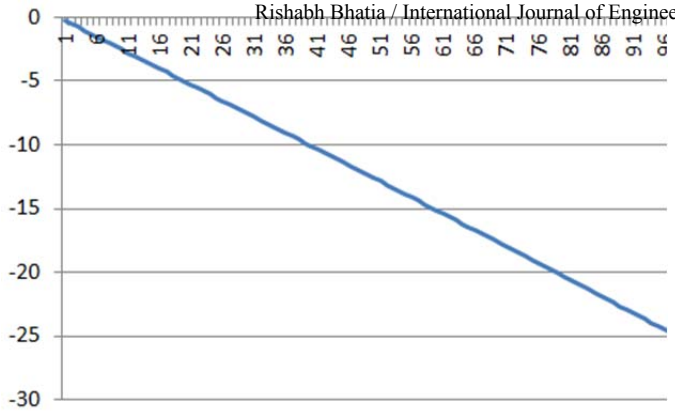

Figure 14-Steer angle vs rack travel(Simulator)

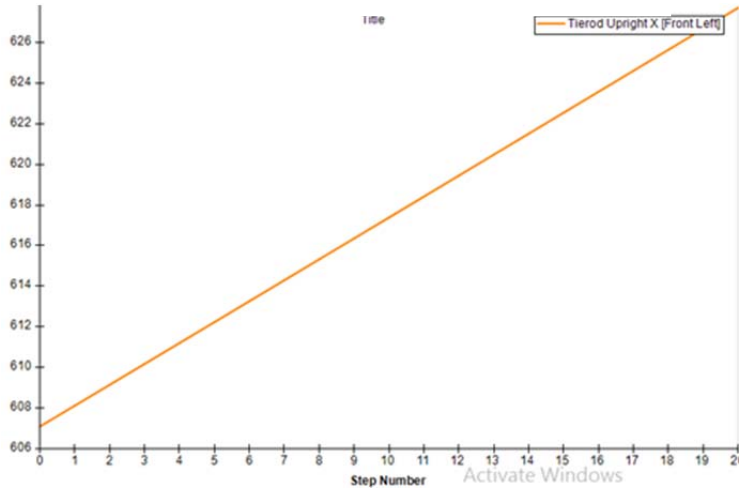

Figure 15-Tie rod outer X vs rack travel(Optimum K)

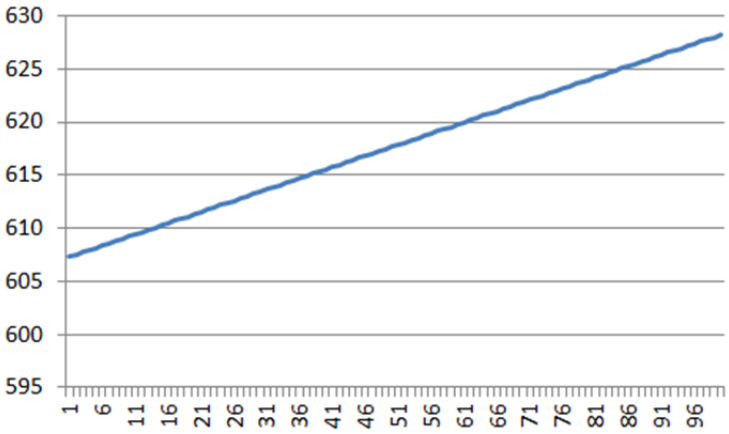

Figure 16-Tie rod outer X vs rack travel(Simulator)

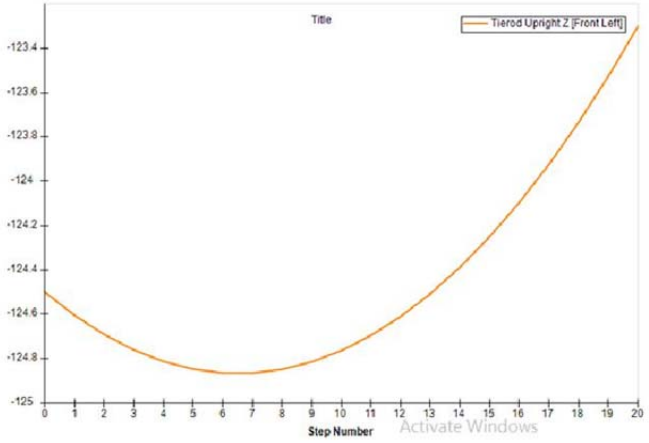

Figure 17-Tie rod outer Z vs rack travel(Optiumu K)

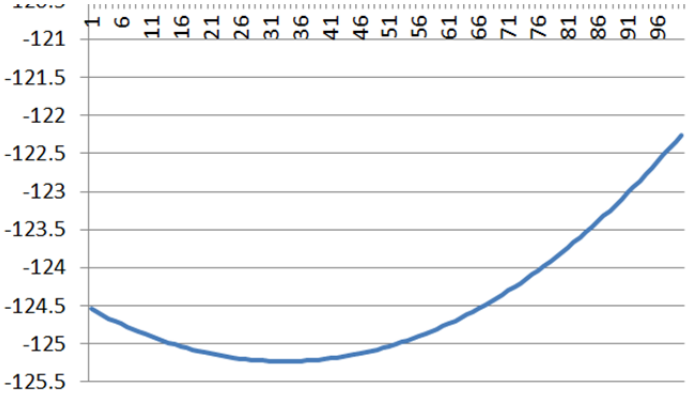

Figure 18-Tie rod outer X vs rack travel(Simulator)

Table 2

Characteristics Comparison Between Simulator Output And Optimum K Output

Characteristics	Optimum K	Simulator
Steer angle left	0 to -19(deg.)	0 to-24(deg.)
Tie od outer X	607 to 627(mm)	607-627 (mm)
Tie od outer Z	-124.5 to -122 (mm)	-124.5 to 121 (mm)

VI. Conclusion

In this paper the methodology for development of kinematic suspension simulator for double A-Arm suspension has been thoroughly. The results from the simulator stand validated hence confirm the method for algorithm development using rigid body principles.

Abbreviations

Front upper- front pivot point of upper A-Arm (chassis side)
Rear upper- rear pivot point of upper A-Arm (chassis side)
Front lower- front pivot point of lower A-Arm (chassis side)
Rear lower- rear pivot point of lower A-Arm (chassis side)

canter upper - centre of arc traced by upper A-Arm.
centre lower - Centre of arc traced by lower A-Arm. U.B.J.
(Upper Ball Joint)- Pivot point of upper A-Arm (upright side)
L.B.J.(Lower Ball Joint)- Pivot point of lower A-Arm (upright side)
Wheel centre- Geometric centre of wheel
Tie rod outer- Outer tie rod pivot point
Tie rod inner- Inner tie rod pivot point
U_X, U_Y, U_Z-orientation angle of upper wishbone
L_X, L_Y, L_Z- orientation angle of lower wishbone
Tie rod outer- Outer tie rod pivot point
Tie rod inner- Inner tie rod pivot point Wheel canter- Geometric centre of wheel
UBJ_X, UBJ_Y, UBJ_Z-denotes X, Y, Z coordinate of upper ball joint (same applies for all other hard points).
K.P.I (Kingpin Axis Inclination)- imaginary line joining UBJ and LBJ.

ACKNOWLEDGMENT

I would like to thank my parents for always supporting me in my endeavors. I would also like to thank my team for their constant patience and endless hours of toil to get in depth view of the subject and VIT University for providing such a platform. I am also very thankful to Prof. Om. P. Suthar for his imperative guidance throughout my college life.

REFERENCES

[1] [1] William F Milliken and Douglas L.Milliken, Race car vehicle dynamics, SAE, Warrandale, Pa, section 17.5,pp 627-629.
[2] [2] Reza N. Jazar, Vehicle Dynamics Theory and Application, Dept. of Mechanical Engineering Manhattan College Riverdale, NY 10471: Springer, ch. 8, pp.478-483
[3] [3] Rakholia Meet," Optimization of Double Wishbone Suspension using MATLAB" International Journal of Scientific & Engineering Research, Volume 5, Issue 8,August-2014 811 ISSN 2229-5518
[4] [4]Double-A-Arm simulation in MATLAB. Retrieved from https://in.mathworks.com/matlabcentral/fileexchange/12837-optimization-of-a-double-wishbone-suspension-system
[5] [5] William F Milliken and Douglas L.Milliken, Race car vehicle dynamics, SAE, Warrandale, Pa, section 17.5,pp 610-615.
[6] I used Optimum K v1.1 from Optimum G,6450 South Quebec St, Suite 5-28 Centennial, CO 80111, USA to complete my work.

AUTHOR PROFILE

Rishabh Bhatia, is a fourth year B.tech Mechanical student at VIT University, Vellore. His research interest is in the area of creating vehicle dynamics simulation soft wares and tools. At the University he was lead vehicle dynamics engineer in an international Baja team- KI Racing and represented the University at various international and national competitions.