Department of Mechanical and Mechatronics Engineering
University of Waterloo

ME 303 – Advanced Engineering Mathematics

Vehicle Dynamics Numerical Models Report

Prepared by:
Ali Muizz
Yasir Ahmed
Vikesh Mistry
Krypton Purnama
Saleem Mohammed Ali

Prepared for:
ME 303 Teaching Team

Submitted on:
Monday, March 24, 2025

Table of Contents

Table of Figures

1.0 Methods for Solving ODEs

There are numerous methods of solving Ordinary Differential Equations (ODEs). Analytical methods provide exact mathematical solutions and are common for simple problems. However, as complexity increases, finding closed-form solutions becomes challenging or even impossible. In such cases, numerical methods are employed to approximate solutions.

This report explores different methods for solving ODEs, with a strong emphasis on numerical approximations and their application in simulating vehicle dynamics. Further background on this topic is provided in following sections.

1.1 Analytical Approach

For single order separable ODEs, the method of separation of variables (SOV) can be utilized to find the analytical solution. SOV will be utilized to solve the initial value ODE problem below.

$$y'(x) = y(x), \qquad y(x = 0) = 1$$

For simplicity, the function notation will be dropped, $y'(x) \rightarrow y'$

$$y' = y$$

$$\frac{dy}{dx} = y$$

$$\frac{1}{y}\, dy = dx$$

$$\int \frac{1}{y}\, dy = \int dx$$

$$\ln(y) = x + C$$

$$y = e^{x+C}$$

Let $A = e^C$

$$y = Ae^x$$

Apply the initial condition $y(0) = 1$

$$1 = Ae^0, \qquad A = 1$$

Therefore, the solution to the ODE is:

$$y = e^x$$

1.2 Power Series Approach

Instead of utilizing separation of variables, another common approach for solving simple ODEs is to assume a form of the solution, and to substitute it into the equation. For example, the previous ODE can be solved by assuming a power series form of the solution:

$$y = \sum_{n=0}^{\infty} c_n x^n$$

Taking the derivative of the form and substituting it into the ODE

$$y' = y$$

$$\sum_{n=1}^{\infty} n c_n x^{n-1} = \sum_{n=0}^{\infty} c_n x^n$$

Shift the index of the series starting at $n = 1$ to $n = 0$

$$\sum_{n=0}^{\infty} (n+1) c_{n+1} x^n = \sum_{n=0}^{\infty} c_n x^n$$

$$(n+1)c_{n+1} = c_n \qquad c_{n+1} = \frac{c_n}{n+1}$$

Expanding terms, starting at $n = 0$

$$c_1 = c_0, \qquad c_2 = \frac{c_1}{2} = \frac{c_0}{2}, \qquad c_3 = \frac{c_2}{3} = \frac{c_0}{6}, \qquad c_4 = \frac{c_3}{4} = \frac{c_0}{24}, \qquad c_5 = \frac{c_4}{5} = \frac{c_0}{120}$$

Therefore

$$y = c_0 \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

Apply the initial condition $y(0) = 1$

$$c_0 = 0$$

Therefore

$$y = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

This is the Maclaurin series for e^x, which matches the previous analytical solution. Thus, the numerical approach to solving this ODE is identical. Since infinite values cannot be summed, the accuracy of the solution will be compared to the number of terms being summed.

1.3 Plotting the ODE Solutions

The exact function was plotted using the exp $= (x)$ function within MATLAB. For the power series solution, N terms were summed up for each corresponding estimation to produce the plot in Figure 1. The code for this graph, along with the MATLAB code for all other generated graphs in this report, can be found in Appendix A.

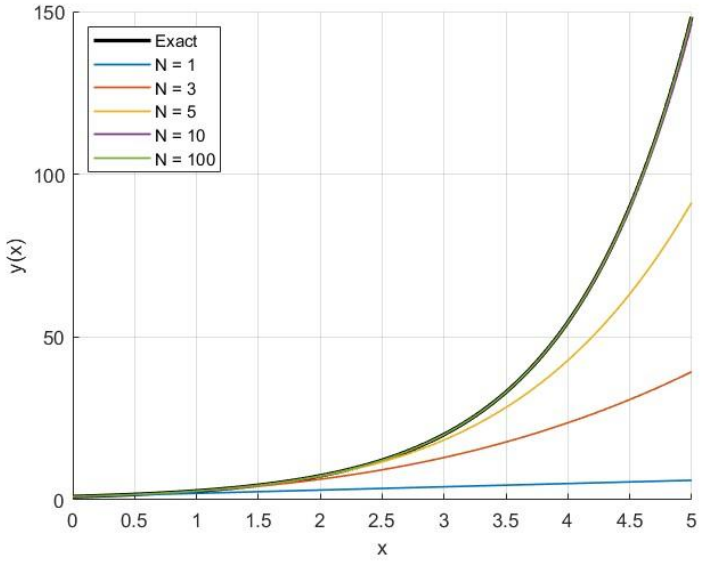


Figure 1: The solution for y'(x) = y(x) plotted using the exact function, and power series

The accuracy of the solution increases with the number of terms summed, N, for the power series approximation. The exact series is an infinite series, so when a finite number of terms are summed, it truncates the solution. When $N = 1$, only the leading term is considered, giving a large amount of truncation, while as N increases, the approximation becomes more accurate, as seen in Figure 1. For example, when $N = 100$, the power series is visually the same as the exact solution. For the solutions to be the exact same, $N \to \infty$.

1.4 Solving via Numerical Methods

For an ODE of the form $y' = f(t)$, the Forward Euler, or Explicit method, says that

$$y_{i+1} \approx y_i + f(t_i)\Delta t$$

Since the given ODE is

$$y' = y$$

This means

$$y_{i+1} = y_i + y_i\Delta t, \qquad y(0) = 1$$

Implementing this advancement scheme in MATLAB, Figure 2 compares the exact solution from the previous section with the Forward Euler estimate. The graphic shows the Forward Euler approximation using three different grid spacings.

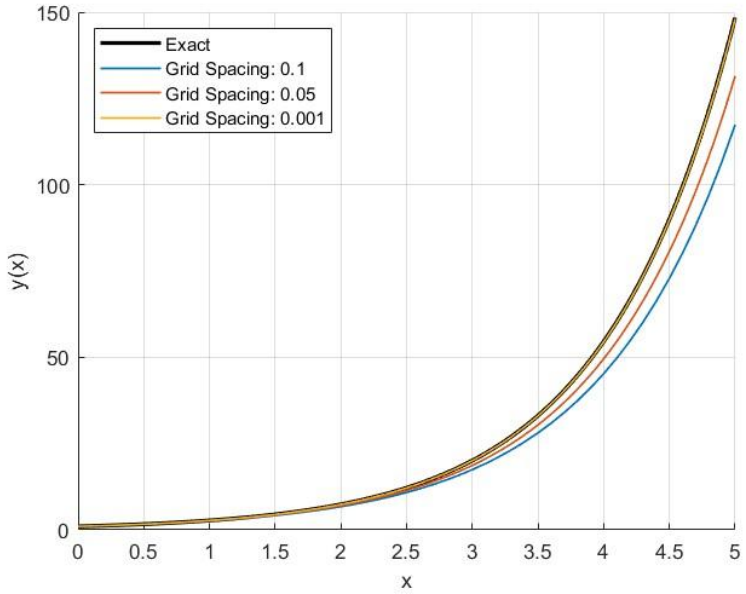


Figure 2: The solution for y'(x) = y(x) plotted using the exact function, and Forward Euler's

The accuracy of the Forward Euler numerical solution increases with decreased grid spacing, Δx. This is because the Forward Euler method is developed using the Forward Euler finite difference method and is truncated to have an error order of accuracy that scales with Δx. Therefore, since error is $O(\Delta x)$, larger Δx, as seen in Figure 2, result in a less accurate solution.

As grid spacing is reduced the truncation error does not accumulate as much, providing a solution close to the exact solution, as seen with $\Delta x = 0.001$.

2.0 Studying Vehicles Dynamics Through ODEs

2.1 Background

The study of vehicle dynamics involves analyzing the motion and behavior of vehicles under different conditions. This report focuses on utilizing ODEs to model and analyze vehicle handling characteristics through the bicycle model, a simple yet representative mathematical model of a four-wheel vehicle in the context of handling and motion analysis [1].

The bicycle model reduces a four-wheel vehicle to an equivalent two-wheel system by assuming the wheels of the front and rear axles are lumped together into one front and one rear wheel. This approach allows the reduction of system's Degrees of Freedom to two allowing a more manageable mathematical representation while preserving key dynamic behaviors. The two degrees of freedom include the lateral motion and yaw (rotation about vertical axis). Both of which are determined by the fundamental laws of motion [1].

To mathematically describe the bicycle model, key physical parameters must be defined to capture the forces and motions governing the vehicle's behavior. These parameters include the car's mass, geometry, inertia, inputs, and external forces, all of which influence lateral motion and yaw dynamics. Along with these variables, the vehicle's movement is assumed to be restricted in the X-Y plane analyzed relative to its center of mass, allowing the derivation of equations of motions to predict the vehicles behavior under different conditions. The diagram in Figure 3 illustrates the bicycle model [1].

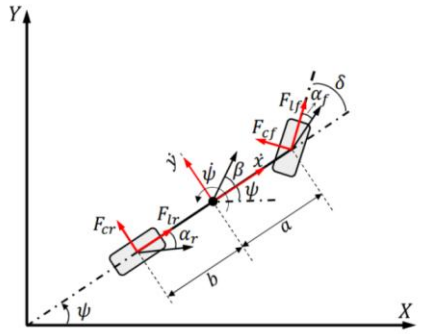


Figure 3:Bicycle representation of a two-axle, four-wheel, car [1]

Key parameters that are utilized throughout this model and report are listed below.

$\boldsymbol{F_l}$ **and** $\boldsymbol{F_c}$**:** The longitudinal and lateral tire forces
$\boldsymbol{I_z}$**:** The moment of inertia around the vertical axis
$\boldsymbol{X}$ **and** $\boldsymbol{Y}$**:** Absolute car position inertial coordinates
$\boldsymbol{a}$ **and** $\boldsymbol{b}$**:** Car geometry, the distance of front and rear wheel from center of mass
$\boldsymbol{x}$ **and** $\boldsymbol{y}$**:** The local longitudinal and lateral coordinates fixed on the car body
$\dot{\boldsymbol{x}}$ **and** $\dot{\boldsymbol{y}}$**:** The vehicle longitudinal and lateral speeds
Subscripts $[\cdot]_f$ **and** $[\cdot]_r$: Denotes a variable at the front wheel and the rear wheel
$\boldsymbol{\psi}$**:** The heading angle (yaw angle, the rotation around the z-axis)
$\boldsymbol{m}$**:** Mass of the car
$\boldsymbol{s}$**:** Slip ratio
$\boldsymbol{\alpha}$**:** The tire slip angle
$\boldsymbol{\delta}$**:** The wheel steering angle
$\dot{\boldsymbol{\psi}}$**:** The yaw rate
$\boldsymbol{\beta}$**:** The vehicle side slip angle.
C.M: Center of Mass

The primary contributor to external forces acting on the vehicle are tire-road interactions. Further, the bicycle model assumes only lateral tire forces. Therefore, through a linear tire model, the bicycle model includes lateral tire forces as a multiplication of tire cornering stiffness, a function of friction coefficient and normal force, and slip angle as seen in Equation 1. This model assumes the left and right tire slip angles as equal, allowing both to experience the same lateral tire force [1].

$$F_c = C_\alpha(\mu, F_z)\alpha \tag{1}$$

In addition to the assumptions made previously, several other assumptions are made to simplify the mathematical model while maintaining its accuracy and reality. The effect of the steering system is neglected, and the front wheel steering angle δ is taken as the direct input, rather than being derived from the steering wheel angle. Further, δ is assumed to be small. Thus, small-angle approximations ($\sin\delta \approx \tan\delta \approx \delta$ and $\cos\delta \approx 1$) are applied, simplifying trigonometric expressions in the process of deriving equations of motion [2]

The vehicle is also treated as a rigid body with negligible flexing or deformation, and lateral acceleration is assumed to remain small. While high performance vehicles experience large lateral accelerations, vehicles for daily driving typically experience a significantly lower lateral acceleration making this assumption reasonable. This assumption ensures linear car movement. Additionally, aerodynamic effects and load transfer between the left and right tires are neglected, focusing the external force analysis solely on tire-road interactions [2]

By implementing Newton's 2nd law of motion and isolating for the second derivative of y and yaw, the model equation can be derived with Equations 2 and 3.

$$\ddot{y} = -u\dot{\psi} - \frac{C_{\alpha f}\left(\frac{\dot{y} + a\dot{\psi}}{u} - \delta\right)}{m} - \frac{C_{\alpha r}\left(\frac{\dot{y} - b\dot{\psi}}{u}\right)}{m} \tag{2}$$

$$\ddot{\psi} = -\frac{1}{I_z}\left(aC_{\alpha f}\left(\frac{\dot{y} + a\dot{\psi}}{u} - \delta\right) - bC_{\alpha r}\left(\frac{\dot{y} - b\dot{\psi}}{u}\right)\right) \tag{3}$$

These equations can also be represented in the state-space form as seen in Equation 4.

$$\begin{bmatrix}\ddot{y}\\ \ddot{\psi}\end{bmatrix} = \begin{bmatrix} -\frac{C_{\alpha f} + C_{\alpha r}}{mu} & -\frac{aC_{\alpha f} - bC_{\alpha r}}{mu} - u \\ -\frac{aC_{\alpha f} - bC_{\alpha r}}{I_z u} & -\frac{a^2C_{\alpha f} + b^2C_{\alpha r}}{I_z u}\end{bmatrix}\begin{bmatrix}\dot{y}\\ \dot{\psi}\end{bmatrix} + \begin{bmatrix}\frac{C_{\alpha f}}{m}\\ \frac{aC_{\alpha f}}{I_z}\end{bmatrix}\delta \tag{4}$$

Equation 4 can be represented with a simpler notation as seen in Equation 5.

$$\dot{x} = \mathrm{A}x + \mathrm{B}\delta \tag{5}$$

where $\dot{x} = \begin{bmatrix}\ddot{y}\\ \ddot{\psi}\end{bmatrix}$, $\mathrm{A} = \begin{bmatrix} -\frac{C_{\alpha f}+C_{\alpha r}}{mu} & -\frac{aC_{\alpha f}-bC_{\alpha r}}{mu} - u \\ -\frac{aC_{\alpha f}-bC_{\alpha r}}{I_z u} & -\frac{a^2C_{\alpha f}+b^2C_{\alpha r}}{I_z u}\end{bmatrix}$, $\mathrm{B} = \begin{bmatrix}\frac{C_{\alpha f}}{m}\\ \frac{aC_{\alpha f}}{I_z}\end{bmatrix}$

Furthermore, the coordinates of the vehicle on the X-Y plane are governed by the ODEs depicted by Equations 6 and 7:

$$\dot{\mathrm{X}} = u\cos(\psi) - \left(\dot{\mathrm{y}} - a\dot{\psi}\right)\sin(\psi) \tag{6}$$

$$\dot{\mathrm{Y}} = \left(\dot{y} - a\dot{\psi}\right)\cos(\psi) - u\sin(\psi) \tag{7}$$

Lastly, the parameters for the vehicle of interest are listed in Table 1.

Table 1: Control Car Parameters

Symbol	Parameter description	Value
m	mass	1400 (kg)
a	Distance of centre of mass from front axle	1.14 (m)
b	Distance of centre of mass from rear axle	1.33 (m)
$C_{\alpha f}$	Front tire cornering stiffness	25000 (N/rad)
$C_{\alpha r}$	Rear tire cornering stiffness	21000 (N/rad)

I_z	Yaw inertia	2420 $(kg \cdot m^2)$
u	Velocity in x direction	75 (km/h)

2.2 Developing Solution Models

To establish a reliable simulation platform, the given ODE below will be solved in terms of lateral velocity ($\dot{y}$) and yaw rate ($\dot{\psi}$) using both Euler's Method and RK4.

$$\begin{bmatrix} \ddot{y} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} -\frac{C_{\alpha f} + C_{\alpha_r}}{mu} & -\frac{\alpha C_{\alpha f} - bC_{\alpha r}}{mu} - u \\ -\frac{\alpha C_{\alpha f} - bC_{\alpha r}}{I_z u} & -\frac{a^2 C_{\alpha f} + b^2 C_{\alpha r}}{I_z u} \end{bmatrix} \begin{bmatrix} \dot{y} \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} \frac{C_{\alpha f}}{m} \\ \frac{aC_{\alpha f}}{I_z} \end{bmatrix} \delta$$

$$\dot{X} = AX + B\delta$$

$$X = \begin{bmatrix} \dot{y} \\ \dot{\psi} \end{bmatrix}$$

$$\begin{bmatrix} \ddot{y} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} \dot{y}A_{11} + \dot{\psi}A_{12} + B_{11}\delta \\ \dot{y}A_{21} + \dot{\psi}A_{22} + B_{21}\delta \end{bmatrix}$$

This is a second order coupled ODE which will require numerical methods to approximate a solution.

2.2.1 Euler's Method

The first approach to solve this equation will use Euler's Method with t ∈ [0,5] seconds and a grid spacing of 0.001s. These parameters were selected according to Section 1.0. The general formula for Euler's Method is:

$$X_{i+1} = X_i + \Delta t F_i$$

Since the equation is a second order ODE, new variables 'v' and 'w' will be introduced to create a first order problem that can be approximated with Euler's Method.

$$let\ v = \dot{y}, \dot{v} = \ddot{y} \mid let\ w = \dot{\psi}, \dot{w} = \ddot{\psi}$$

Now the equation can be rewritten as:

$$\begin{bmatrix} \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} vA_{11} + wA_{12} + B_{11}\delta \\ vA_{21} + wA_{22} + B_{21}\delta \end{bmatrix}$$

With this first order ODE, Euler's Method can be applied. Note, including 'y' and 'ψ' is not necessary in solving for, but was included in the code to provide more information about the nature of the vehicle and for future kinematic use.

$$X_i = \begin{bmatrix} y \\ \psi \\ \dot{y} \\ \dot{\psi} \end{bmatrix}_i = \begin{bmatrix} y \\ \psi \\ v \\ w \end{bmatrix}_i \quad F_i = \begin{bmatrix} v \\ w \\ vA_{11} + wA_{12} + B_{11}\delta \\ vA_{21} + wA_{22} + B_{21}\delta \end{bmatrix}_i \quad \Delta t = 0.001s$$

The following equation can be used to determine the lateral velocity and yaw rate following Euler's Method.

$$X_{i+1} = \begin{bmatrix} y \\ \psi \\ v \\ w \end{bmatrix}_i + \Delta t \begin{bmatrix} v \\ w \\ vA_{11} + wA_{12} + B_{11}\delta \\ vA_{21} + wA_{22} + B_{21}\delta \end{bmatrix}$$

Using the values given in Table 1, matrices A and B can be calculated. A turn angle of $\delta = 0.1$ rad will be used to test the behaviour when the car has a steep turning angle. To solve this pair of second order ODEs, four initial conditions are required: y(0), $\psi(0)$, $\dot{y}(0)$, $\dot{\psi}(0)$.

Given initial conditions for the car at t = 0s:

$$\begin{bmatrix} y \\ \psi \\ \dot{y} \\ \dot{\psi} \end{bmatrix}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The solutions of X (lateral acceleration and yaw rate) are plotted in Figure 4 using Euler's method and initial conditions previously mentioned.

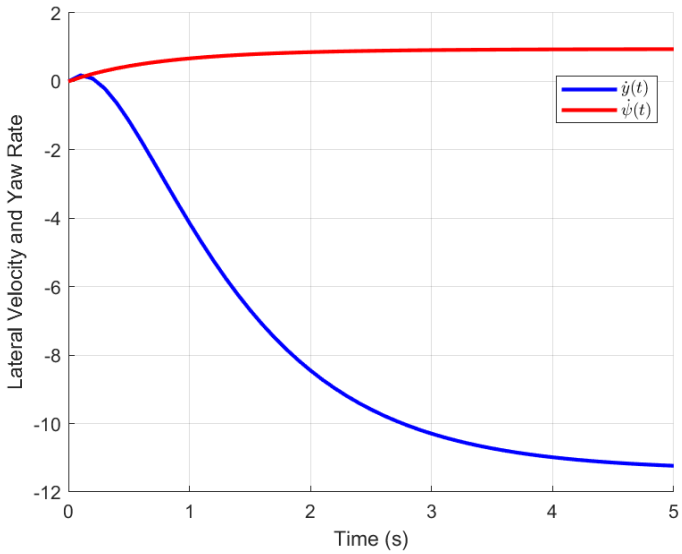


Figure 4: Solution to Equation (3) using Euler's Method and I.C from Table 1.

Analyzing Figure 4, the lateral velocity starts positive but at around 0.15 seconds switches to negative while the yaw rate stays a positive value. The convergence of both these solutions are important to take note of and will be further explored in the following sections.

2.2.2 Runge-Kutta Method

Another way to solve the second order, coupled ODEs is by the Runge-Kutta (RK4) Method. This is a more computationally demanding version of Euler's Method that averages multiple slopes at different time steps to give a fourth degree of accuracy guess on the next value of X. The process used to complete this is shown below.

Find the initial slope:

$$F_i = \begin{bmatrix} v \\ w \\ vA_{11} + wA_{12} + B_{11}\delta \\ vA_{21} + wA_{22} + B_{21}\delta \end{bmatrix}_i$$

Estimate $X\left(\frac{\Delta t}{2}\right)$ to find the slope:

$$X_{i+0.5}{}^{*} = X_i + \frac{\Delta t}{2} F_i$$

$$F_{i+0.5}{}^{*} = \begin{bmatrix} v \\ w \\ vA_{11} + wA_{12} + B_{11}\delta \\ vA_{21} + wA_{22} + B_{21}\delta \end{bmatrix}_{i+0.5}$$

Find the new guess for $\mathrm{X}\left(\frac{\Delta t}{2}\right)$ and find the slope:

$$X_{i+0.5}{}^{**} = X_i + \frac{\Delta t}{2} F_{i+0.5}{}^{*}$$

$$F_{i+0.5}{}^{**} = \begin{bmatrix} v \\ w \\ vA_{11} + wA_{12} + B_{11}\delta \\ vA_{21} + wA_{22} + B_{21}\delta \end{bmatrix}_{i+0.5}^{*}$$

Approximate $\mathrm{X}(\Delta t)$ and find the slope:

$$X_{i+1} = X_i + \Delta t F_{i+0.5}{}^{**}$$

$$F_{i+1} = \begin{bmatrix} v \\ w \\ vA_{11} + wA_{12} + B_{11}\delta \\ vA_{21} + wA_{22} + B_{21}\delta \end{bmatrix}_{i+1}$$

Average all the slopes and find X_{i+1}:

$$\bar{F} = \frac{F_i}{6} + \frac{F_{i+0.5}{}^{*}}{3} + \frac{F_{i+0.5}{}^{*}}{3} + \frac{F_{i+1}}{6}$$

$$X_{i+1} = X_i + \Delta t \bar{F}$$

This is the general method used to find the next numerically approximated point. Theoretically, the RK4 solution is more accurate than the Euler's Method, as RK4 is a fourth order approximation, while Forward Euler's method is a first order approximation. Using the same parameters and initial conditions as the previous method, a plotted solution for X is shown in Figure 5.

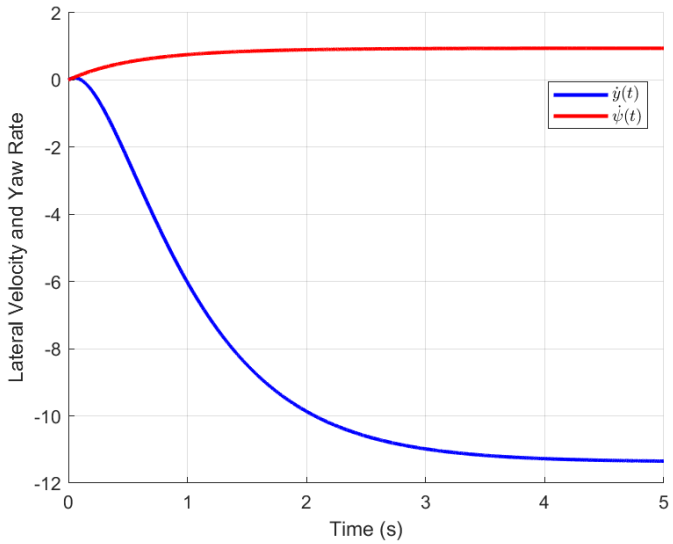


Figure 5: Solution to Equation (3) using RK4

This solution in Figure 5 demonstrates the same phenomenon as Figure 4, where both the lateral velocity and yaw rate converge to values after a length of time.

2.2.3 Grid Independence Check

To ensure these numerical solvers are reliable, a grid independence check will be conducted. Maintaining a bound of t ∈ [0,5], the number of steps within this frame will be adjusted to test the rigidity of these methods. As shown in Figure 6, RK4 is more accurate than Euler's Method with a large time step of 1s.

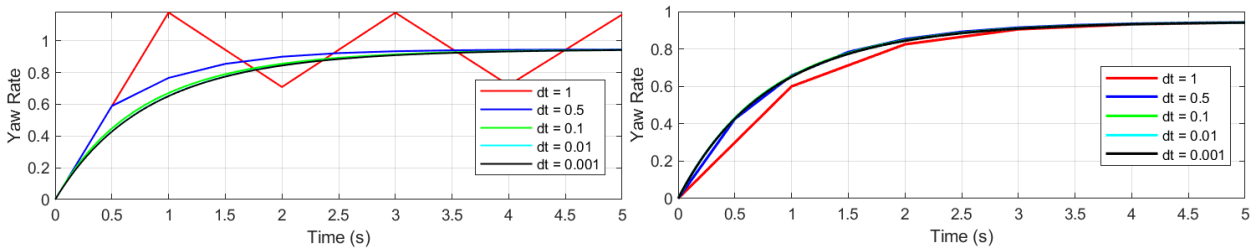


Figure 6: Euler's Method (left) and RK4 (right) grid independence check

To complete the grid independence check, the log of error is plotted against the log of the various time steps in Figure 7. This error is calculated as the second norm between the values at each time step and those from the $\Delta t = 0.001$s time step. This assumes that $\Delta t = 0.001s$ represents the ground truth. This method of error calculation compares a slightly coarser time step to the more refined one to find the error value, a common technique in numerical analysis.

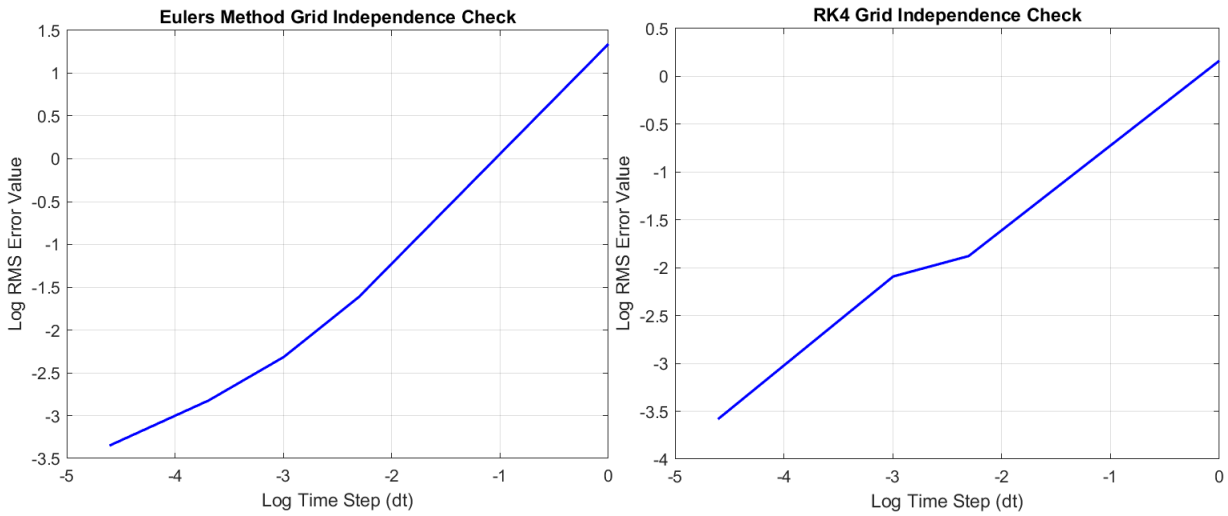


Figure 7: Grid independence check (lateral acceleration)

These graphs demonstrate that as the time step decreases, the accuracy of the model will increase. As expected, RK4 has less error so it will be used for the following experiments. Thus, a $\Delta t = 0.001s$ and RK4 will be used in the rest of the simulations.

2.3 Testing the Performance of the Car

2.3.1 Analyzing Different Tangential Speeds

Using the RK4 model, the vehicle's performance can be analyzed with a fixed steering angle at 0.1 radians. The effect different tangential speeds have on the vehicle's lateral acceleration and yaw rate are plotted in Figure 8.

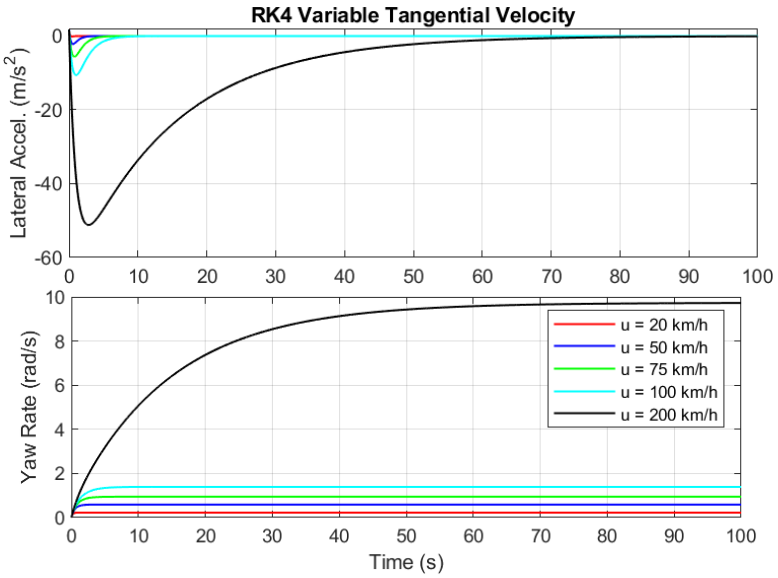


Figure 8: Behaviour of the vehicle at various speeds

The general trend in Figure 8, is that a higher tangential velocity results in a larger magnitude of both lateral acceleration and yaw rate. So, the angle of the car will change at a higher rate and the force on the car is much higher with a greater constant tangential velocity. An important characteristic of these graphs is that the lateral acceleration converges to zero and the yaw rate values converges to a value based on the car's speed. This convergence is important as it means the car will eventually drive is stable circles.

2.3.2 Determining Maximum Stable Speed

2.3.2.1 Setting Stable Conditions

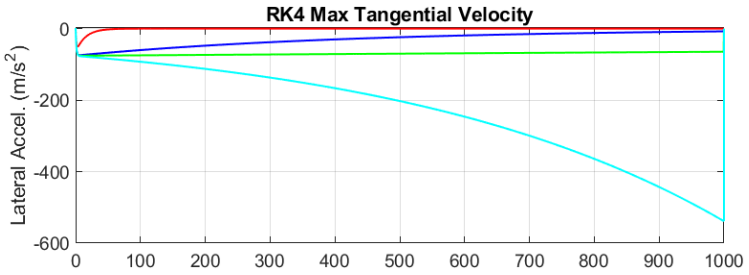
As shown previously, Figure 8 proves the system is stable at 200 km/h since the lateral acceleration converges to zero after some time. However, during this turn the driver will experience around 50 m/s^2 or 5g of force which very high, on par with the forces felt by Formula One drivers [3]. It is important to consider the driver when determining a safe turn. Thus, if the car is driving at a stable safe speed, the following conditions for the behaviour of the car must be met:

1) The lateral acceleration and yaw rate must converge
2) The driver must experience less than 1g (~10 m/s^2) of lateral force

The first condition tests the maximum capability of the car, while the second condition is set for the safety and comfort of the average driver based on research [3].

2.3.2.2 Modelling the Behaviour

As mentioned previously, it is important to have a stable system so that the car turns at a constant rate without losing control. Figure 9 demonstrates what may happen if the car has too high of a speed and loses control while keeping the steering wheel at 0.1 radians.



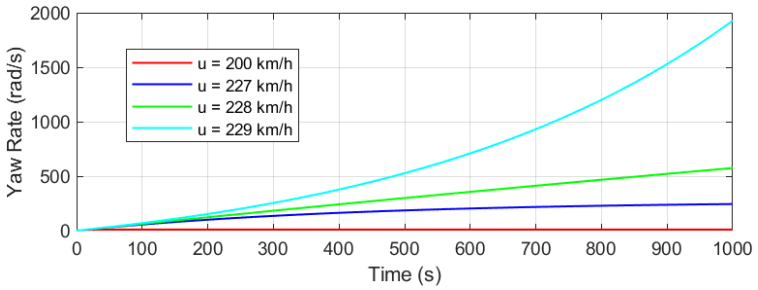


Figure 9: Computing the highest stable speed for the car

As seen above, the lateral acceleration and yaw rate both diverge with a speed of 229 km/h. This means that the car will spin out of control as spin rate increases with time. So, the highest stable speed of this car as seen from Figure 9 is approximately 228 km/h. Another way to determine the maximum stable speed is by calculating the eigenvalue of the A matrix. The mathematical explanation as to why this represents instability can be found in Appendix B - Solving the Eigenvalue problem. As seen in Figure 10, this approach gives a more accurate highest stable speed of around 228.1 km/h.

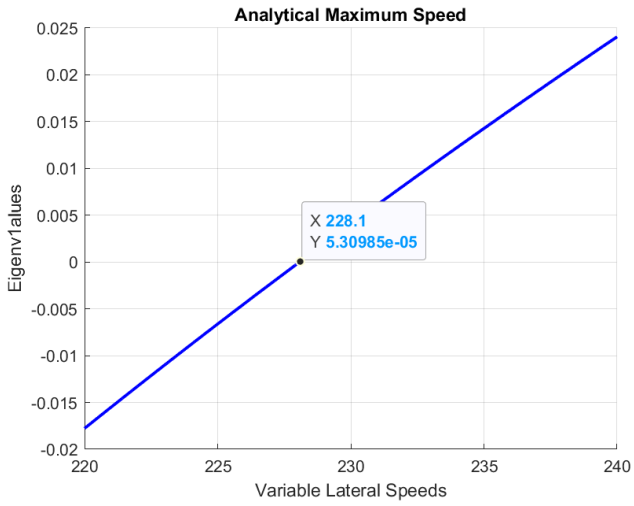


Figure 10: Analytical approach to highest stable speed

Despite being stable at 228 km/h, this is not the safest speed, as the driver will experience more than 1g of force as seen from Figure 9. To determine the highest safest speed, a limit of 1g for the lateral acceleration was set as shown in Figure 11. At 96 km/h, while turning at 0.1 radians, the driver will experience just less than 1g. Though this is considered an aggressive turn, it will be utilized as the threshold of the safety metric.

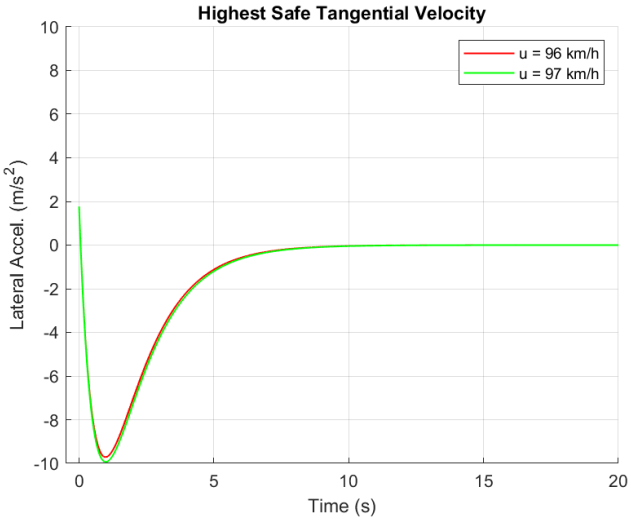


Figure 11: Maximum speed for a safe turn

Thus, after concluding that the vehicle can reach a speed of around 228 km/h without losing control, the maximum stable, safe speed is 96 km/h since the driver will experience less than 1g of force at this steering angle.

2.4 Plotting the Kinematics of the Car

Given the functions for $\dot{X}$ and $\dot{Y}$, the kinematic behaviour of the car can be plotted using the solutions calculated with the RK4 method. Figure 12 plots the path of the car with a constant 100 km/h tangential speed and a turn angle of 0.1 rad.

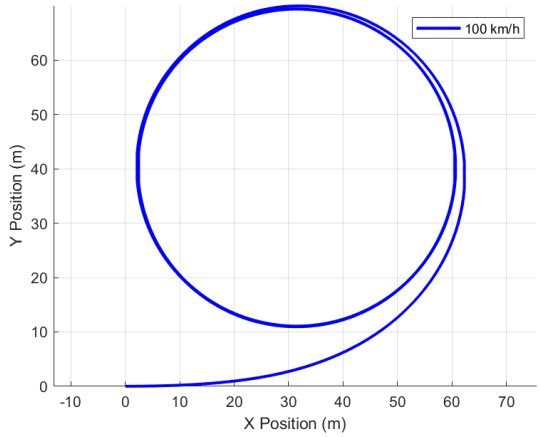


Figure 12: Kinematics of the car at 100 km/h for 100 seconds

At this speed and steering angle, the simulation indicates that the car will experience oversteering. This means, that despite maintaining a constant steering angle, the radius of the car's path will decrease. The car is turning sharper than expected, which is characterized as oversteered handling. To determine the radius of the turn, the tangential speed is divided by the yaw rate and plotted in Figure 13.

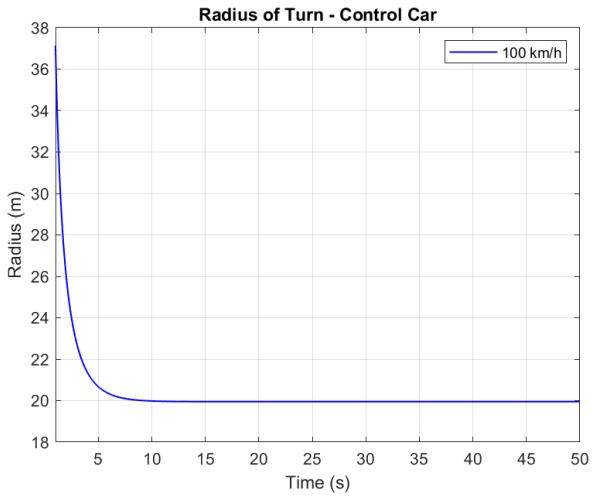


Figure 13: Plotting the turn radius at 100km/h

The above figure shows that at 100km/h the car's turning radius will decrease until it converges to around 20m.

2.5 Testing the Performance and Handling with Additional Weight

2.5.1 Preliminary Assumptions

Looking at a scenario where a 50kg weight is added to the rear or front trunk of the car, the following assumptions will be made to be able to use the previously derived model.

1) The 50kg is added inline with the center of the rear or front axis as a point mass
2) The Yaw Inertia (I_z) and Tire Cornering Stiffnesses (C_f, C_r) are unaffected

With these assumptions, the new total mass (m) of the system can be calculated. Similarly, the new distance of center of mass from the front axle (a) and rear axle (b) is calculated directly in the code as seen from Appendix A – MATLAB Code. Furthermore, all other variables except the tangential velocity will be held constant. The resulting lateral acceleration and yaw rates were plotted using the RK4 numerical solver that was coded for the previous section.

2.5.2 Additional Weight in the Trunk

The stability of the car with this added weight in the trunk will be analyzed first. Similarly to the previous section, RK4 will be used to plot the function at different tangential speeds to determine the highest stable speed of the car. Figure 14 shows that going above 103 km/h causes both the lateral acceleration and yaw rate to diverge. This means the car is instable

and will spiral out of control. Note, since the car itself is being tested here, the forces felt on the driver will not be considered.

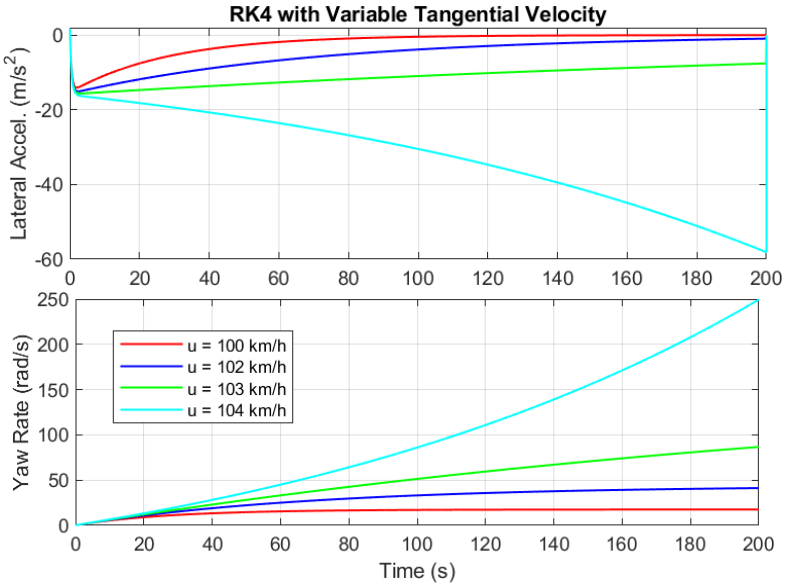


Figure 14: RK4 approach for the highest stable speed with 50kg in trunk

This solution can be compared analytically to the eigenvalues of the A matrix as seen below.

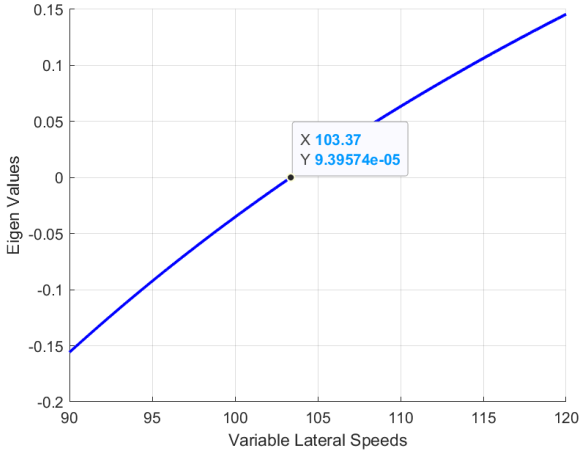


Figure 15: Analytical approach for the highest stable speed with 50kg in trunk

Approximately 103.3 km/h is the maximum speed one can go before losing control of the vehicle with the additional 50kg weight added onto the rear axle.

The next step is to determine the effect this added weight has on the handling of the vehicle. Previous simulations at 100 km/h with no additional weight showed that oversteering was the case. Using the previous simulation for $\dot{X}$ and $\dot{Y}$, the 'a', 'b' and 'm' values will be modified to account for an additional 50kg on the rear axles. The added weight in the trunk of the car results in a relatively smaller steady state radius as seen in Figure 17.

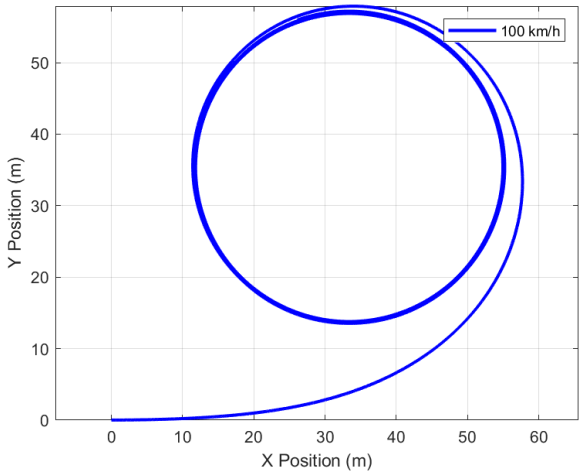


Figure 16: Kinematics of the car at 100 km/h for 10 seconds, with 50kg in trunk

This new radius is plotted with the same equation derived in the previous section to determine the radius of the turn with the added 50kg mass.

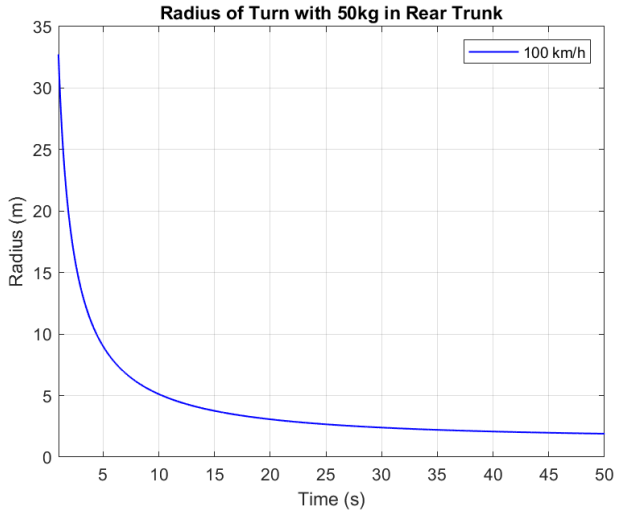


Figure 17: Turn radius at 100km/h with 50kg in the trunk

From gathered research, oversteering occurs when the rear wheels experience a greater lateral acceleration in comparison to the front wheels [4]. In a turn, the lateral force is applied to the centre of mass of the system, as the centrifugal turning force**.** As weight is added to the trunk and the centre of mass is shifted closer to the rear axle and this means that the rear wheels will lose traction faster then the front tires. This increases the oversteer and results in a sharper turn radius with this additional weight in the trunk [4].

2.5.3 Additional Weight in the Front Trunk

The same approach will be used to determine the effect of adding 50kg point mass to the front of the vehicle. However, with the new calculated parameters for ‘a’, ‘b’ and ‘m’, the

numerical solver creates a sinusoidal decaying function that will always converge as seen in Figure 18. Based on the previously mentioned stable conditions and the new weight distribution, the car would technically meet the first condition of its lateral acceleration converging at any speed.

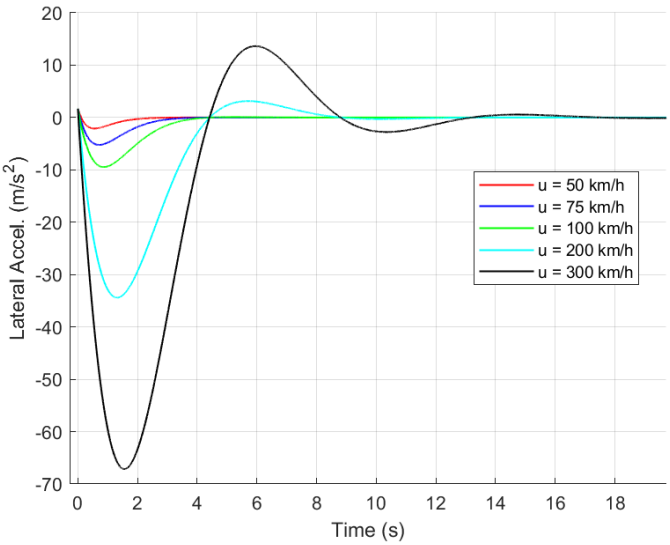


Figure 18: Stability with 50kg in the front trunk

The kinematic graph can be plotted with this additional weight to see the effect on the vehicle's handling at various speeds. Figure 19 shows that the car will experience understeering with an additional weight placed on the front axles.

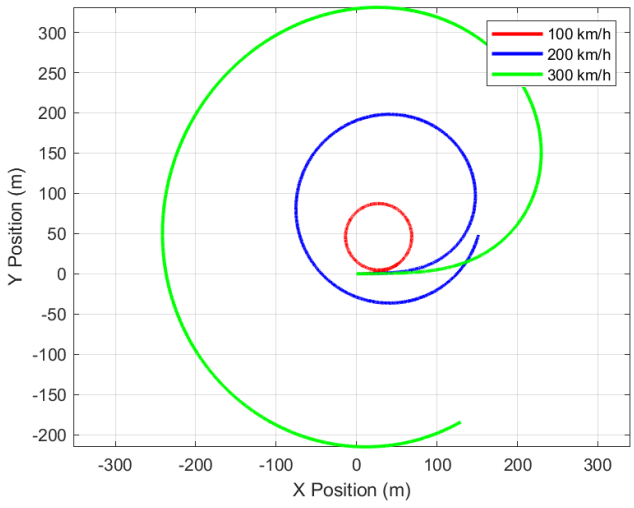


Figure 19: Kinematics of the car with 50kg in the front trunk

The radius of the turn can be plotted to demonstrate the increased understeering, seen in Figure 20.

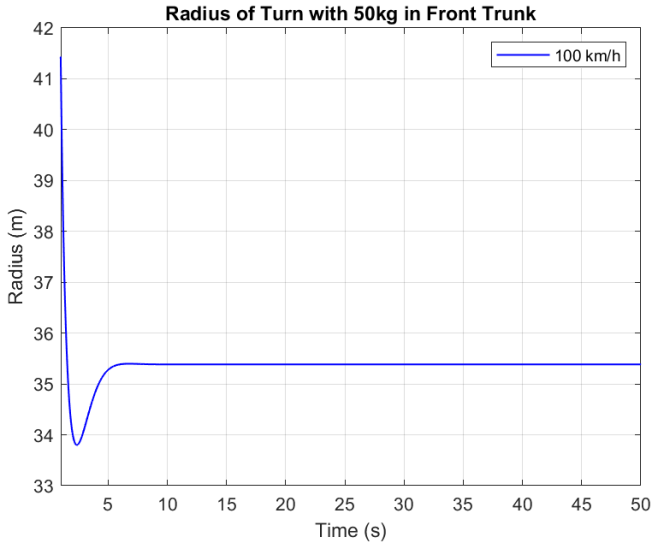


Figure 20: Turn radius with 50kg in front trunk

Understeering is amplified as the center of mass is shifted towards the front of the car. Understeering happens when the front tires lose grip before the rear tires, this effect is amplified when the centre of mass of the car is shifted more towards the front by placing weight in the front trunk. With more weight at the front, the front tires reach their grip limit earlier due to the closer point of application of centrifugal force. As speed increases, the pressure on the front tires is increased due to more lateral force acting upon the vehicle, this amplifies the effect of understeering and results in a larger turn radius as seen from Figure 19 [5].

2.6 Driving in Winter Conditions

2.6.1 Parameter Setup and Stability Condition

To simulate the dangers of driving fast and/or steering hard during winter conditions a ‘mu’ variable will be introduced. This represents the friction coefficient and will be set to 0.3 to simulate ice or snow. This value will be multiplied by the original Cf (25000) and Cr (21000) values to obtain the vehicle’s behaviour in winter conditions with the following parameters:

1) A low steering angle of $\delta = 0.05$
2) A steep steering angle of $\delta = 0.1$
3) A sharp steering angle of $\delta = 0.5$

To define a controlled turn, the kinematics of the vehicle will be plotted using the maximum safe speed in summer conditions (mu = 1). This will give the largest turn that the car can theoretically perform without the driver experiencing more than 1g of force. For example, at a steer angle of 0.1 radians the maximum stable safe speed was simulated to be 96 km/h. This speed is used to plot the trajectory of the car and set a standard for the expected vehicle’s path.

2.6.2 Simulating Summer Trajectory

The maximum safe speed will be determined through simulations for these new turn angles as completed in previous sections considering the safety and comfort constraint of 1g. On the right graph of Figure 21, the idealized vehicle dynamics is illustrated, where the car follows a controlled turning behavior at 96 km/h given a steering angle of 0.1 radians and a friction coefficient of 1. Similarly, the controlled turning speeds for 0.05 and 0.5 rad steering angles were found to be 132 km/h and 48 km/h respectively in the left graph of Figure 21. These three speeds are set as benchmarks to compare with controlled turning speeds in winter conditions.

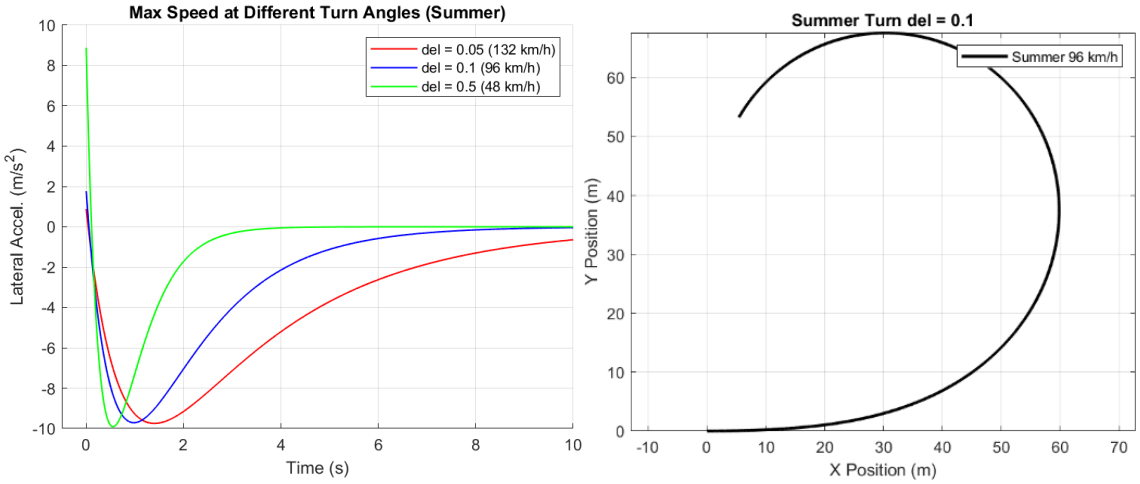


Figure 21: Maximum safe speed and controlled trajectory after 5 seconds

2.6.3 Simulating Winter Conditions

With the discovered safe speeds at each respective turn angle that yields a controlled turn, the friction coefficient is set to 0.3 to simulate winter conditions and the new trajectories at steering angles of 0.05, 0.1, and 0.5 are plotted in Figure 22.

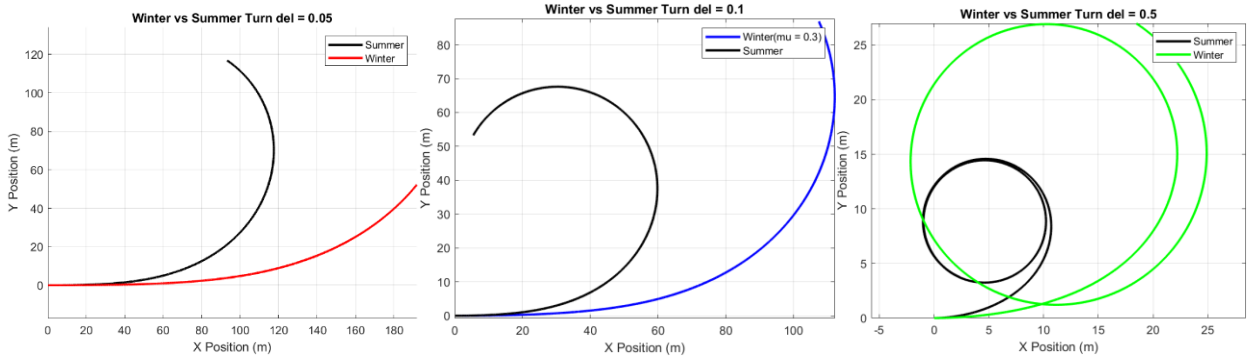


Figure 22: Expected (summer) vs actual (winter) vehicle behaviour

As seen in Figure 22, at a given speed and steering angle, the car takes a much wider turn compared to the controlled turn condition depicted by the black lines. Hence, the behavior shown by the red, blue, and green line are indicative of an uncontrolled turn. This behaviour is due to the reduced friction $(C_{\alpha f}, C_{\alpha r})$.

The speed of each turn in winter conditions was reduced until a near perfectly controlled turn was completed after 5 seconds. This condition occurs when the coloured lines overlap with the black lines, seen in Figure 23.

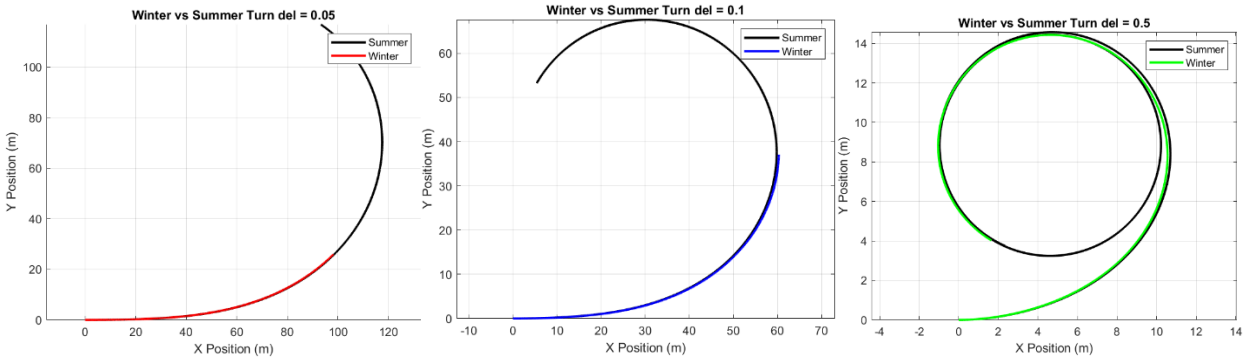


Figure 23: Controlled turn in winter (72, 53, 26 km/h, left to right)

The tangential speed was reduced to 72, 53, and 26 km/h for a 0.05, 0.1, and 0.5 radians steering angle respectively to achieve a controlled turn. Due to the reduced friction, the vehicle must reduce its speed before attempting any of these turns to make a controlled, predictable turn. Driving any faster, and/or steering harder in any of these scenarios would cause the vehicle to deviate from the predefined standard path and is considered an uncontrolled turn.

Therefore, the simulations prove the dangers of driving fast and/or turning sharply in winter conditions. Turning at 0.05 radians and 132 km/h is proven safe during the summer. However, if one does not consider the weather conditions and attempts to make this turn during the winter, it will cause them to lose control and be unable to complete the desired turn.

2.7 Analyzing Seasonal Tires

2.7.1 Parameter Setup

Previous simulations demonstrate the dangers of driving fast and steering hard in winter conditions. This section will explore the effect winter tires have on the vehicle's handling and determine if they are a good investment for driving safely in the winter.

With slippery conditions, the corning stiffness is variable depending on the turn angle and obviously reduced due to the lack of grip on ice and snow. The following conditions, provided within the instructions of the model, will be tested for all season and winter tires.

$$C_{\alpha f} = C_{\alpha_r} = C$$

For all-season tires:

$$C = \begin{cases} 0, & \delta = 0.3 \\ 100, & \delta = 0.1 \\ 20\,000, & \delta = 0.05 \end{cases}$$

For winter tires:

$$C = \begin{cases} 0, & \delta = 0.3 \\ 5\,000, & \delta = 0.1 \\ 20\,000, & \delta = 0.05 \end{cases}$$

2.7.2 Testing Summer Tires

The above conditions were plotted with the same kinematic equation given previously. The parameters were set to simulate a circular on-ramp used to merge onto a highway. The speed was set to 70 km/h with a duration of 30 seconds.

Figure 24 show that all steering at 0.1 radians or above at 70 km/h will cause the car tires to slip. This decreased friction will cause the car to have a much larger turning radius as denoted on the right graph in Figure 24. Thus, this car is very unstable and has the potential to lose control if the turning angle goes above 0.05 radians while merging onto the highway.

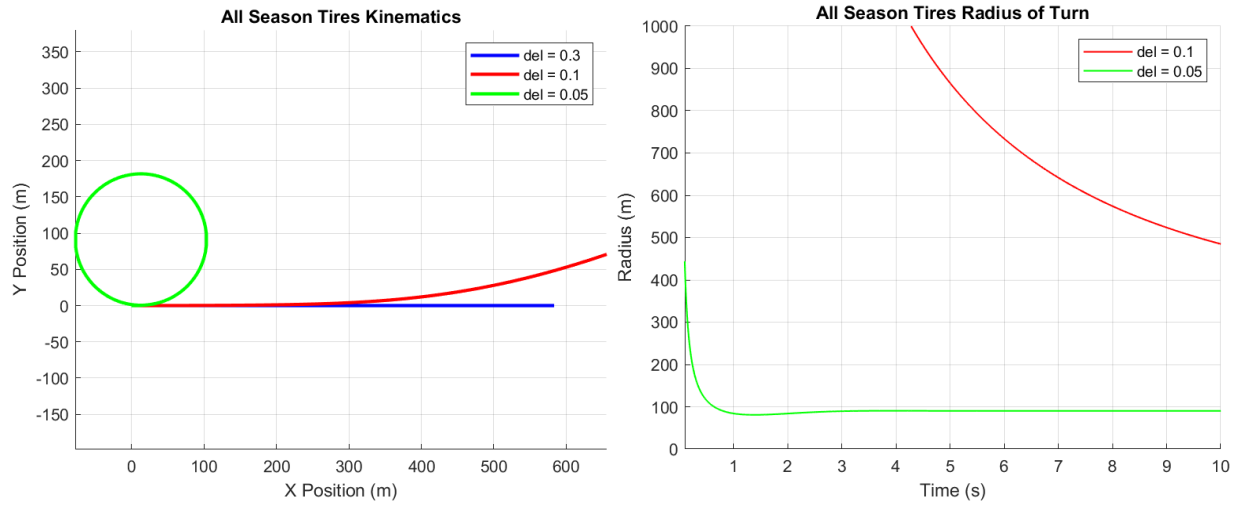


Figure 24: All Season Tires Turning at 70km/h for 30 seconds

2.7.3 Testing Winter Tires

The same driving conditions are simulated with the winter tire parameters above. The new simulations in Figure 25Figure 25 show that the car will be stable up to a steering angle of 0.1 radians and speed of 70km/h. With winter tires, the increased friction makes turning at higher

speeds, such as merging onto a highway, much safer with a forgiving range of steering angles. 0.05 radians to 0.2 radians (tested in simulation based on piecewise function given).

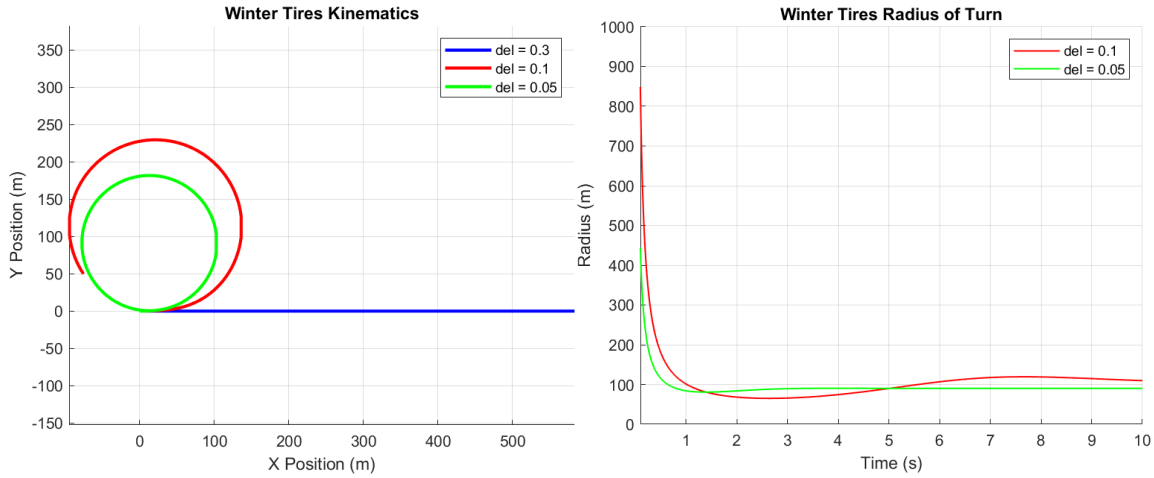


Figure 25: Winter tires turning at 70km/h for 30 seconds

2.7.4 Buy Winter Tires

The results from the previous sections prove that winter tires are crucial for winter conditions. They provide much more grip in snowy conditions and allow the car to be able to make higher speed turns with a steep turning angle. This is important for scenarios such as merging onto a high with a circular onramp. This logic also applies to everyday city driving. It is important to be able to make sharp turns in any situation and stay in control. Winter tires excel in this area and will give the driver much more control in these situations.

2.8 Tuning the Handling

Reverting to the original non-winter conditions, the control car will be modified to improve the handling as defined below.

2.8.1 Performance Cars

The ideal handling performance various per car. For example, a Formula 1 and high-end performance cars, would generally be designed to understeer. This avoids instability of oversteering and the potential to lose control while at high speeds of around 300 km/h.

2.8.2 Modifying the Tires

Wider tires will provide more surface traction thus increase the tire cornering stiffness ($C_{\alpha f}$ and $C_{\alpha r}$). Assuming regular width tires have a cornering stiffness of around 21000 N/rad

and the wider tires are 25000 N/rad, the effect of adding these to the front vs rear will be plotted below in Figure 26.

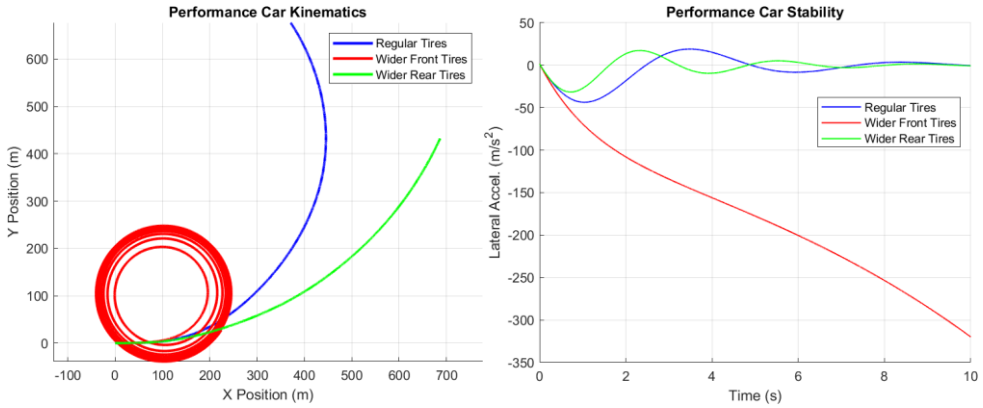


Figure 26: Comparing the placement of wider tires at 300 km/h

Figure 26 demonstrates why it is important to have more grip, or wider, rear tires. At 300 km/h, if the wheel is turned 0.1 radians, it is important to have a stable, controlled turn. The red line demonstrates the instability of oversteer, when the wider tires are attached to the front axle. As seen from the kinematic and stability graphs, the car's lateral acceleration will diverge causing it to become unstable. With regular tires on both front and rear, the blue line shows a relatively stable turn, with a large lateral force and slow convergence. Finally, the green line shows the most understeering with wider rear tires. This is the ideal case as the driver maintains control of the vehicle at this high speed while the lateral acceleration converges faster than the regular tire scenario.

2.8.3 Weight Reduction

The overall mass of the car will be reduced to see the effect on its handling at high speeds. This will assume that the center of mass remains a point mass with its value reduced. At the same speed and steering angle, reducing the mass of the vehicle decreases the turn radius as seen in Figure 27. Depending on the desired handling specifications, for more responsive, tighter turns, less mass would be optimal. Thus, the more mass increases understeering.

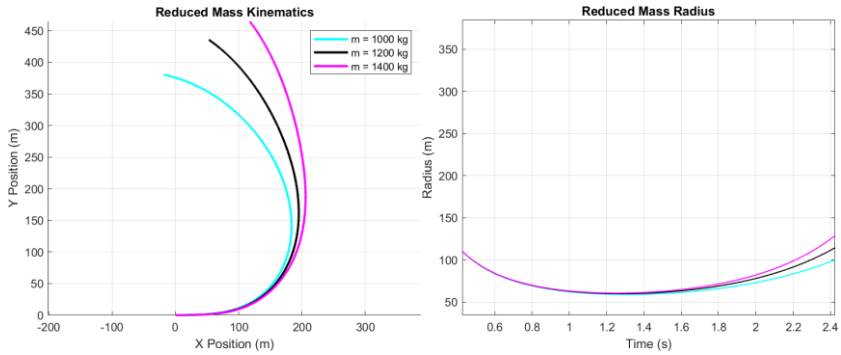


Figure 27: Weight reduction's effect on handling

3.0 Developing a Toyota Testing Center

This section of the report is focused on building a testing field for the handling performance of Toyota vehicles. The goal is to propose a budget and location for the test field of the Toyota RAV4, a compact, 5-seater SUV [6].

3.1 Overview of Proposal

The following will outline the costs and features needed to build a testing field dedicated specifically to handling performance of the RAV4. Using the simulation work of vehicle dynamics above, it can be utilized to choose different scenarios that will push the vehicle to its performance limits. The testing field will be in the Waterloo-Kitchener area. This area was chosen, as the region already has the Toyota Motor Manufacturing Facility. The testing facility will be developed on the outskirts of the city for lower land costs.

3.2 Testing Track and Equipment

As seen in the simulations above it is crucial to test the car under different track layouts. The first part of the track will need a long straight or gentle curving path that will allow the vehicle to reach its max speed. The track should also include corners with varying radii to test the car under situations to tune the understeering. The simulations above show that vehicles tend to lose control when subject to high-speed, sharp-turning situations. These areas of the track will be made of high-grade concrete to improve its durability in the case of dynamic crashes and decrease weathering effects. This will allow winter testing, which is important as shown above, a vehicle driving on lower friction surfaces will often be subject to worse handling performance. The track will also include sections where the ground is not pavement but instead loose dirt or rocks to test the vehicle's handling performance in off road settings, which the Toyota RAV4 is marketed to be capable of.

To accommodate all these features on a single site, a substantial amount of land is necessary. It was found that approximately 50 acres can be bought in Cambridge for $22 million dollars, providing ample space for the planned facility [7]. A simple backyard racetrack costed a US citizen $8 million, and with this being a much larger project, it will be assumed to cost $20 million for construction [8].

The next feature of the test field will be the instrumentation and data acquisition tools needed to track and measure the handling performance of the vehicle. Within the vehicle itself an onboard sensor array using the vehicle CAN bus and other sensors will be needed to track critical metrics such as yaw rate, lateral and linear acceleration, speed and slip angles in real time. Outside of the vehicle and around the track, different cameras and telemetry poles will be placed

in strategic positions to accurately monitor the car's behaviour at all locations to improve data collection accuracy. This setup will be approximated to cost around $4 million.

3.3 Additional Supporting Facilities

In addition to the above, a service and repair facility for rapid maintenance and quick iterative design modification will be needed. The goal of this facility is to put the vehicle through rigorous testing that will push its limits and break certain parts to identify failure points. So, it is crucial to have an onsite repair shop and the ability to ship parts from the local plant at Cambridge. A research and development lab will also be needed to process the data recorded from the tests. This will require lots of computational power to process the data for performance analyzation, like the simulations conducted above. A safety and response building and an onsite mobile unit will need to be included for accidents that may happen during testing.

Assuming there is a research and development team of 10 engineers, paid $100000 a year, a maintenance crew of 8 technicians, paid $80000 a year, a logistics team of 5 for part delivery paid $70000 a year, and a repair team of 5 paid $75000 a year, this comes to approximately $2.5 million a year. The total fixed cost for all the equipment and construction of these buildings will be assumed to be $10 million.

3.4 Final Budget Estimation

Therefore, the total costs for the testing facility will come to an up-front cost of $52 million, and a yearly cost of $2.5 million. The assumed construction timeline will be 2 years.

4.0 Summary and Future Applications

This report provides a detailed analysis of vehicle handling performance under different driving conditions. The primary objective was to apply numerical approximations to solve ordinary differential equations that simulated vehicle dynamics. The accuracy of both Euler's Method and the RK4 method were compared with grid independence checks, showing that RK4 had a greater accuracy and stability than Euler's method. This led to RK4 being utilized throughout the vehicle dynamics studies.

The dynamic behavior of the vehicle was analyzed using a simplified bicycle model, examining how various parameters influence lateral acceleration and yaw rate. Real-world considerations were incorporated to ensure practical applicability. For example, while the theoretical maximum stable speed of the control vehicle was calculated at 228 km/h, this would subject the driver to lateral forces exceeding 5g, which is beyond what an average person can sustain comfortably and safety. To address this, a constraint of 1g was implemented resulting in a safer maximum speed of 97 km/h for everyday driving. This highlights the importance of balancing engineering decisions to optimize both product performance and user experience.

The impact of adding extra weight to the vehicle, both at the rear and front of the vehicle, was simulated to observe changes in handling. Rear-weighted configurations resulted in increased oversteering, reducing the vehicle's turning radius, while front-weighted configurations led to understeering, resulting in wider turns. This highlights the importance of weight distribution in different application for vehicle design.

Furthermore, simulations highlighted the critical role of winter tires. With a reduced friction coefficient representing winter conditions, the maximum speed at which the vehicle can make a controlled turn significantly decreased. For instance, at a steering angle of 0.1 radians, the maximum controlled turn speed was reduced from 96 km/h in summer conditions to 53 km/h in winter conditions. Using winter tires have a higher grip limit increasing the maximum speed the car can drive without losing stability. Also, winter tires give a larger range of stable steering angles so with medium steering angles, the car will stay in control.

To enhance vehicle handling under non-winter conditions, modifications to tire width and vehicle mass were analyzed. As using wider tires increases cornering stiffness, simulations demonstrated that wider rear tires provide greater control by reducing oversteering and promoting stability. In contrast, placing wider tires on the front axle led to instability and unpredictable handling. Further, reducing the vehicle's mass was shown to decrease the turn radius, resulting in more responsive and agile handling. Conversely, heavier vehicles exhibited greater understeering, further emphasizing the influence of mass and tire placement on achieving the desired handling characteristics.

Future applications of this model include further refinement by incorporating additional factors including those assumed negligible in the current model such as aerodynamics and additional passengers or load transfer. The model can also be further developed to assess the effects of different tire materials, road conditions, and advanced suspension and damping systems. Additionally, integrating the model into autonomous vehicle algorithms can allow for active handling control, increasing safety and comfort.

Insights gained from this report can guide vehicle manufacturers in optimizing design parameters such as tire type, tire thickness, mass distribution and other variables for improved stability and comfort. The practical applications extend to motorsport engineering, where precise tuning is essential to maximize performance while maintaining safety and control at high speeds.

Overall, this report built a strong foundation in numerical modeling and simulation analysis, resulting in a deeper understanding of vehicle dynamics and equipping engineers with the tools to innovate and enhance automotive safety and performance.

References

[1] "Vehicle Dynamics & Control - 09 Dynamic bicycle model with linear tires," professorschildbach, 3 May 2020. [Online]. Available: https://youtu.be/35lZlO6NrO0?si=hsMlT80jVGgqWUnY. [Accessed 11 March 2025].

[2] R. Rajamani, Vehicle Dynamics and Control, 2nd ed., New York: Springer, 2012.

[3] Y. Elshebiny, "G-Force in F1: What is it and how many G's do drivers experience during a race," 17 January 2024. [Online]. Available: https://www.gpfans.com/en/f1-news/1010709/f1-g-force/. [Accessed 24 March 2025].

[4] T. Gillespie, Fundamentals of vehicle dynamics, 1st Edition, Warrendale, PA: Society of Automotive Engineers, 1992.

[5] W. F. Milliken and D. L. Milliken, Race Car Vehicle Dynamics, Warrendale, PA: Society of Automotive Engineers, 1995.

[6] Toyota, "2025 RAV4," Toyota, [Online]. Available: https://www.toyota.ca/en/vehicles/rav4/overview/#reviews. [Accessed 11 March 2025].

[7] Realtor.Ca, "475 WITMER STREET," [Online]. Available: https://www.realtor.ca/real-estate/28067756/475-witmer-street-cambridge. [Accessed 24 March 2025].

[8] J. Chow, "This Guy Built an $8 Million Racetrack In His Backyard," 7 June 2022. [Online]. Available: https://johnchow.com/this-guy-built-an-8-million-formula-1-racetrack-in-his-backyard/. [Accessed 24 March 2025].

[9] "Chat GPT," 2025. [Online]. Available: https://chatgpt.com/. [Accessed 24 March 2025].

Statement of Task Division

The tasks in this project were divided among the five group members, ensuring an even distribution of work. Each member contributed to different aspects of the project, as outlined below:

Vikesh Mistry: Developed the two models for Part 2 and wrote much of the MATLAB code and the respective explanations used throughout the project.

Ali Muizz: Led the development and coding for Part 1, proofread the report, and contributed to report writing.

Yasir Ahmed: Conducted background research for model development and wrote the Toyota Testing Center section of the report.

Krypton Purnama: Researched and wrote about conceptual background information, wrote about future applications and wrote and summarized key findings.

Saleem Mohammed Ali: Edited the report and provided support for MATLAB coding in Part 2.

Statement of Generative AI Usage

Generative AI was utilized as a supplementary tool to support our work. It was primarily used for troubleshooting MATLAB code, identifying relevant built-in functions, and clarifying programming concepts. However, final code, analysis, and report writing were completed by the team members.

Appendix A – MATLAB Code

Part 1 – Plotting Sum Solutions

```
clear
clc
close all

% A range of 250 values, from 0 to 5, for x-axis
x = linspace(0, 5, 250);

% The different values used in the power series
N_values = [1, 3, 5, 10, 100];

% Initialize figure
figure;
hold on;

% Plot the exact function (k for black line color)
y_exact = exp(x);
plot(x, y_exact, 'k', 'LineWidth', 2, 'DisplayName', 'Exact');

% A different color for each N value
% Reminder: Lines is a function to produce a matrix of colors
estimateColor = lines(length(N_values));

% Run through the calculations

for i = 1:length(N_values)
   % Resets y each time
   y_approx = zeros(size(x));
   j = N_values(i);

   % Compute the sum
   for n = 0:j
      y_approx = y_approx + (x.^n) ./ factorial(n);
   end

   % Plot approximation
   plot(x, y_approx, 'Color', estimateColor(i, :), 'LineWidth', 1, 'DisplayName', sprintf('N = %d',
j));
```

```
end

% Plot
xlabel('x');
ylabel('y(x)');
legend('Location','northwest');
grid on;
```

Part 1 – Forward Euler Method

```
clear
clc
close all

% Array of time step increments
timeSteps = [0.1, 0.05 0.001];
% Colors
estimateColor = lines(length(timeSteps));
% Initial Condition
y_initial = 1;

% A range of 250 values, from 0 to 5, for x-axis exact solution
x = linspace(0, 5, 250);

% Initialize figure
figure;
hold on;

% Plot the exact function (k for black line color)
y_exact = exp(x);
plot(x, y_exact, 'k', 'LineWidth', 2, 'DisplayName', 'Exact');

% Plots for each time step

for i = 1:length(timeSteps)

    % x values discretized through time step
    x_values = 0:timeSteps(i):5;
    y_approx = zeros(1, length(x_values));
    y_approx(1) = y_initial;
```

```
    % start at 2nd term, 1st term is known from IC
    for j=2:length(x_values)

        y_approx(j) = y_approx(j-1) + y_approx (j-1) * timeSteps(i);

    end

     plot(x_values, y_approx, 'Color', estimateColor(i, :), 'LineWidth', 1, 'DisplayName',
sprintf('Grid Spacing: %3g', timeSteps(i)));

end

% Label Plot
xlabel('x');
ylabel('y(x)');
legend('Location','northwest');
grid on;
```

Part A - Developing Models - Euler Grid Check

```
clc; clear;

m = 1400; %kg
a = 1.14; %m
b = 1.33; %m
Cf = 25000; %N/rad
Cr = 21000; %N/rad
Iz = 2420; %kgm^2
u = 75/3.6; %km/hr

% Define constants for dx2/d2t = Adx/dt + Bdel
A = [-(Cf+Cr)/(m*u), -(a*Cf-b*Cr)/(m*u)-u;
     -(a*Cf-b*Cr)/(Iz*u), -((a^2)*Cf+(b^2)*Cr)/(Iz*u)];

del = 0.1;
B = [Cf/m; (a*Cf)/Iz];
B = del.*B;

% Compute y(t) and psi(t)

F = zeros(4,1);
```

```
y_compare = {1,5};
psi_compare = {1,5};

dt = [1,0.5,0.1,0.01,0.001];
colors = ['r','b','g','c','k'];
figure;
hold on;

for i =1:length(dt)

   t = 0:dt(i):5;

   x = zeros(4,length(t));
   % dv/dt = d^2y/dt^2 = A(1,1)v + A(1,2)w + del*B
   % dw/dt = d^2ψ/dt^2 = A(2,1)v + A(2,2)w + del*B

   % IC at t = 0 (given eq7)
   x(1,1) = 0; %y
   x(2,1) = 0; %psi
   x(3,1) = 0; %v
   x(4,1) = 0; %w

   for n = 1:length(t)-1

      F= [x(3,n);
         x(4,n);
         A(1,1)*x(3,n) + A(1,2)*x(4,n) + B(1);
         A(2,1)*x(3,n) + A(2,2)*x(4,n) + B(2)];

      x(:,n+1) = x(:,n) + dt(i) * F(:,1);

   end

   subplot(2,1,1);
   plot(t, x(3,:), 'color',colors(i), 'LineWidth', 1);
   xlabel('Time (s)');
   ylabel('Lateral Velocity');
   title('Eulers Method Grid Independence Check');
   grid on;
   hold on;
```

```
    subplot(2,1,2);
    plot(t, x(4,:), 'color',colors(i), 'LineWidth', 1);
    xlabel('Time (s)');
    ylabel('Yaw Rate');
    %title('Yaw Rate Grid Independence Check');
    grid on;
    hold on;

    y_compare{i} = x(3,:);
    psi_compare{i} = x(4,:);

end

    legend('dt = 1', 'dt = 0.5', 'dt = 0.1', 'dt = 0.01','dt = 0.001');
```

Part A - Developing Models - Euler Grid Check R1

```
clc; clear;

m = 1400; %kg
a = 1.14; %m
b = 1.33; %m
Cf = 25000; %N/rad
Cr = 21000; %N/rad
Iz = 2420; %kgm^2
u = 75/3.6; %km/hr

% Define constants for dx2/d2t = Adx/dt + Bdel
A = [-(Cf+Cr)/(m*u), -(a*Cf-b*Cr)/(m*u)-u;
     -(a*Cf-b*Cr)/(Iz*u), -((a^2)*Cf+(b^2)*Cr)/(Iz*u)];

del = 0.1;
B = [Cf/m; (a*Cf)/Iz];
B = del.*B;

% IC at t = 0
y0 = 0;
psi0 = 0;
v0 = 0;
w0 = 0;
```

```
F = zeros(4,1);

dt = [1,0.1,0.05,0.025,0.01,0.001];

error_check = zeros(length(dt),51);

e = zeros(length(dt)-1, 51);
e_rms = zeros(length(dt)-1, 1);

for i =1:length(dt)

   t = 0:dt(i):5;

   %vector for i, i+1
   x = zeros(4,length(t));
   x(:,1) = [y0;psi0;v0;w0];

   for n =1:length(t)-1

      F= [x(3,n);
         x(4,n);
         A(1,1)*x(3,n) + A(1,2)*x(4,n) + B(1);
         A(2,1)*x(3,n) + A(2,2)*x(4,n) + B(2)];

      x(:,n+1) = x(:,n) + dt(i) * F(:,1);

      %store lateral vel
      error_check(i,n) = x(3,n+1);

   end

end

for i = 1:length(dt)-1

   dt_coarse = dt(i);
   dt_fine = dt(i+1);

   t_coarse = 0:dt_coarse:5;
   t_fine = 0:dt_fine:5;
```

```
    for j = 1:length(t_coarse)-1

        [~, idx] = min(abs(t_fine - t_coarse(j)));
        e(i, j) = error_check(i, j) - error_check(i+1, idx);

    end

    e_rms(i) = rms(e(i, :));

end

e_rms = log(e_rms);

figure;
plot(log(dt(1:length(e_rms))),e_rms,'b', 'LineWidth', 1.5);
xlabel('Log Time Step (dt)');
ylabel('Log RMS Error Value');
title('Eulers Method Grid Independence Check');
grid on;
```

Part A - Developing Models - RK4 Gid Check

```
clc; clear;

m = 1400; %kg
a = 1.14; %m
b = 1.33; %m
Cf = 25000; %N/rad
Cr = 21000; %N/rad
Iz = 2420; %kgm^2
u = 75/3.6; %km/hr

% Define constants for dx2/d2t = Adx/dt + Bdel
A = [-(Cf+Cr)/(m*u), -(a*Cf-b*Cr)/(m*u)-u;
     -(a*Cf-b*Cr)/(Iz*u), -((a^2)*Cf+(b^2)*Cr)/(Iz*u)];

del = 0.1;
B = [Cf/m; (a*Cf)/Iz];
B = del.*B;

% Compute y(t) and psi(t)
```

```
% IC at t = 0 (given eq7)
y0 = 0;
psi0 = 0;
v0 = 0;
w0 = 0;

F = zeros(4,1);

dt = [1,0.5,0.1,0.01,0.001];
colors = ['r','b','g','c','k'];
figure;
hold on;

for i =1:length(dt)

   t = 0:dt(i):5;

   %vector for i, i+1
   x = zeros(4,length(t));
   x(:,1) = [y0;psi0;v0;w0];

   %vector for i+0.5 (intermediate steps)
   xtemp = zeros(4,1);

   %reset vector array for slopes
   f = {};
   %slope at i
   f{1} = [x(3,1); x(4,1); A(1,1)*x(3,1) + A(1,2)*x(4,1) + B(1);
         A(2,1)*x(3,1) + A(2,2)*x(4,1) + B(2)];

   for n = 1:length(t)-1

      %find xi+.5 and slope at i+.5
      xtemp = x(:,n) + 0.5*dt(i)*f{1};
      f{2} = [xtemp(3); xtemp(4);
         A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
         A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

      %new i+0.5 and slope
      xtemp = x(:,n) + 0.5*dt(i)*f{2};
```

```
    f{3} = [xtemp(3); xtemp(4);
       A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
       A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

    %find xi+1 and slope
    xtemp = x(:,n) + dt(i)*f{3};
    f{4} = [xtemp(3); xtemp(4);
       A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
       A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

    f{5} = (1/6) .* f{1} + (1/3) .* f{2} + (1/3) .* f{3} + (1/6) .* f{4};
    xtemp = x(:,n) + dt(i)*f{5};

    x(:,n+1) = xtemp;
    f{1} = f{5};

  end

  subplot(2,1,1);
  plot(t, x(3,:), 'color',colors(i), 'LineWidth', 1.5);
  xlabel('Time (s)');
  ylabel('Lateral Velocity');
  title('RK4 Grid Independence Check');
  grid on;
  hold on;

  subplot(2,1,2);
  plot(t, x(4,:), 'color',colors(i), 'LineWidth', 1.5);
  xlabel('Time (s)');
  ylabel('Yaw Rate');
  %title('Yaw Rate Grid Independence Check');
  grid on;
  hold on;

end

legend('dt = 1', 'dt = 0.5', 'dt = 0.1', 'dt = 0.01','dt = 0.001');
```

Part A - Developing Models - RK4 Gid Check R1

```
clc; clear;
```

```
m = 1400; %kg
a = 1.14; %m
b = 1.33; %m
Cf = 25000; %N/rad
Cr = 21000; %N/rad
Iz = 2420; %kgm^2
u = 75/3.6; %km/hr

% Define constants for dx2/d2t = Adx/dt + Bdel
A = [-(Cf+Cr)/(m*u), -(a*Cf-b*Cr)/(m*u)-u;
      -(a*Cf-b*Cr)/(Iz*u), -((a^2)*Cf+(b^2)*Cr)/(Iz*u)];

del = 0.1;
B = [Cf/m; (a*Cf)/Iz];
B = del.*B;

% IC at t = 0
y0 = 0;
psi0 = 0;
v0 = 0;
w0 = 0;

F = zeros(4,1);
dt = [1,0.1,0.05,0.01,0.001];

error_check = zeros(length(dt),51);

e = zeros(length(dt)-1, 51);
e_rms = zeros(length(dt)-1, 1);

for i =1:length(dt)

   t = 0:dt(i):5;

   %vector for i, i+1
   x = zeros(4,length(t));
   %x(:,1) = [y0;psi0;v0;w0];

   %vector for i+0.5 (intermediate steps)
   xtemp = zeros(4,1);
```

```
    %reset vector array for slopes
    f = {};
    %slope at i
    f{1} = [x(3); x(4); A(1,1)*x(3) + A(1,2)*x(4) + B(1);
            A(2,1)*x(3) + A(2,2)*x(4) + B(2)];

    for n = 1:length(t)-1

        %find xi+.5 and slope at i+.5
        xtemp = x(:,n) + 0.5*dt(i)*f{1};
        f{2} = [xtemp(3); xtemp(4);
            A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
            A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

        %new i+0.5 and slope
        xtemp = x(:,n) + 0.5*dt(i)*f{2};
        f{3} = [xtemp(3); xtemp(4);
            A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
            A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

        %find xi+1 and slope
        xtemp = x(:,n) + dt(i)*f{3};
        f{4} = [xtemp(3); xtemp(4);
            A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
            A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

        f{5} = (1/6) .* f{1} + (1/3) .* f{2} + (1/3) .* f{3} + (1/6) .* f{4};
        xtemp = x(:,n) + dt(i)*f{5};

        x(:,n+1) = xtemp;
        f{1} = f{5};

        %store lateral vel
        error_check(i,n) = x(3,n+1);

    end
end

for i = 1:length(dt)-1

    dt_coarse = dt(i);
    dt_fine = dt(i+1);
```

```
    t_coarse = 0:dt_coarse:5;
    t_fine = 0:dt_fine:5;

    for j = 1:length(t_coarse)-1

        [~, idx] = min(abs(t_fine - t_coarse(j)));
        e(i, j) = error_check(i, j) - error_check(i+1, idx);

    end

    e_rms(i) = rms(e(i, :));

end

e_rms = log(e_rms);

figure;
plot(log(dt(1:length(e_rms))),e_rms,'b', 'LineWidth', 1.5);
xlabel('Log Time Step (dt)');
ylabel('Log RMS Error Value');
title('RK4 Grid Independence Check');
grid on;
hold off;
```

Part B - Kinematics - Kinematics Steering Radius

```
clc; clear;

m = 1400; %kg
a = 1.14; %m
b = 1.33; %m
Cf = 25000; %N/rad
Cr = 21000; %N/rad
Iz = 2420; %kgm^2

del = 0.1;
B = [Cf/m; (a*Cf)/Iz];
B = del.*B;

% Compute y(t) and psi(t)
dt = 0.001;
```

```
t = 0:dt:100;

%vector for i+0.5 (intermediate steps)
xtemp = zeros(4,1);
%vector array for slopes
f = {};
F = zeros(4,length(t));

u_var = [100];
colors = ['r','b','g','c','k','m',"#EDB120"];

for i = 1:length(u_var)

   %vector for i, i+1
   x = zeros(4,length(t));

   u = u_var(i)/3.6; %m/s

   % Define constants for dx2/d2t = Adx/dt + Bdel
   A = [-(Cf+Cr)/(m*u), -(a*Cf-b*Cr)/(m*u)-u;
     -(a*Cf-b*Cr)/(Iz*u), -((a^2)*Cf+(b^2)*Cr)/(Iz*u)];

   %slope at i
   f{1} = [x(3); x(4); A(1,1)*x(3) + A(1,2)*x(4) + B(1);
         A(2,1)*x(3) + A(2,2)*x(4) + B(2)];

   F_temp = zeros(2,1);
   xy_plot = zeros(2,length(t));

   for n = 1:length(t)-1

      F_temp = [u*cos(x(2,n)) - (x(3,n)+a*x(4,n))*sin(x(2,n));
           (x(3,n)+a*x(4,n))*cos(x(2,n)) + u*sin(x(2,n))];

     %find xi+.5 and slope at i+.5
      xtemp = x(:,n) + 0.5*dt*f{1};
      f{2} = [xtemp(3); xtemp(4);
         A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
         A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

      %new i+0.5 and slope
```

```
    xtemp = x(:,n) + 0.5*dt*f{2};
    f{3} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

    %find xi+1 and slope
    xtemp(:) = x(:,n) + dt*f{3};
    f{4} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

    f{5} = (1/6) .* f{1} + (1/3) .* f{2} + (1/3) .* f{3} + (1/6) .* f{4};
    xtemp(:) = x(:,n) + dt*f{5};

    x(:,n+1) = xtemp;
    f{1} = f{5};
    F(:,n) = f{1};

  end

  radius = u./x(4,:);

  plot(t,radius,'b','LineWidth',1);
  grid on;
  xlim([1,50]);
  xlabel('Time (s)');
  ylabel('Radius (m)');
  hold on;

end

legend('100 km/h', '200 km/h');
title('Radius of Turn - Control Car');
```

Part B - Kinematics - RK4 Kinematics

```
clc; clear;

m = 1400; %kg
a = 1.14; %m
b = 1.33; %m
```

```
Cf = 25000; %N/rad
Cr = 21000; %N/rad
Iz = 2420; %kgm^2
u = 100/3.6; %km/hr

%define constants for dx2/d2t = Adx/dt + Bdel
A = [-(Cf+Cr)/(m*u), -(a*Cf-b*Cr)/(m*u)-u;
     -(a*Cf-b*Cr)/(Iz*u), -((a^2)*Cf+(b^2)*Cr)/(Iz*u)];

del = 0.1;
B = [Cf/m; (a*Cf)/Iz];
B = del.*B;

%compute y(t) and psi(t)

dt = 0.001;
t = 0:dt:10;

x = zeros(4,length(t));
f = {};

%slope at i
f{1} = [x(3); x(4); A(1,1)*x(3) + A(1,2)*x(4) + B(1);
      A(2,1)*x(3) + A(2,2)*x(4) + B(2)];

% IC at t = 0
x0 = [0;0;0;0];

F_temp = zeros(2,1);
xy_plot = zeros(2,length(t));

for n = 1:length(t)-1

      F_temp = [u*cos(x(2,n)) - (x(3,n)+a*x(4,n))*sin(x(2,n));
           (x(3,n)+a*x(4,n))*cos(x(2,n)) + u*sin(x(2,n))];

     %find xi+.5 and slope at i+.5
      xtemp = x(:,n) + 0.5*dt*f{1};
      f{2} = [xtemp(3); xtemp(4);
         A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
         A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];
```

```
    %new i+0.5 and slope
    xtemp = x(:,n) + 0.5*dt*f{2};
    f{3} = [xtemp(3); xtemp(4);
       A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
       A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

    %find xi+1 and slope
    xtemp(:) = x(:,n) + dt*f{3};
    f{4} = [xtemp(3); xtemp(4);
       A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
       A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

    f{5} = (1/6) .* f{1} + (1/3) .* f{2} + (1/3) .* f{3} + (1/6) .* f{4};
    xtemp(:) = x(:,n) + dt*f{5};

    x(:,n+1) = xtemp;
    f{1} = f{5};

    %lat_a(n+1) = f{1}(3);

    xy_plot(:,n+1) = xy_plot(:,n) + dt*F_temp(:);

end

figure;
hold on;
plot(xy_plot(1,:), xy_plot(2,:), 'b', 'LineWidth', 2);
grid on;
axis equal;
xlabel('X Position (m)');
ylabel('Y Position (m)');
legend('100 km/h');
hold off;
```

Part B - Max Speed - Max Speed Eigen

```
clc; clear;

m = 1400; %kg
a = 1.14; %m
```

```
b = 1.33; %m
Cf = 25000; %N/rad
Cr = 21000; %N/rad
Iz = 2420; %kgm^2

del = 0.1;
B = [Cf/m; (a*Cf)/Iz];
B = del.*B;

u_var = 220:0.01:240;
eval = zeros(2,length(u_var));

for i = 1:length(u_var)

   u = u_var(i)/3.6;

   A = [-(Cf+Cr)/(m*u), -(a*Cf-b*Cr)/(m*u)-u;
     -(a*Cf-b*Cr)/(Iz*u), -((a^2)*Cf+(b^2)*Cr)/(Iz*u)];

   eval(:,i) = eig(A);

end

figure;
hold on;
plot(u_var,eval(2,:),'b' ,'LineWidth',2);
grid on;
xlabel('Variable Lateral Speeds');
ylabel('Eigenv1alues');
title('Analytical Maximum Speed')
%legend('Eigenvalues');
```

Part B - Max Speed - RK4 Max Speed

```
clc; clear;

m = 1400; %kg
a = 1.14; %m
b = 1.33; %m
Cf = 25000; %N/rad
Cr = 21000; %N/rad
```

```
Iz = 2420; %kgm^2

del = 0.1;
B = [Cf/m; (a*Cf)/Iz];
B = del.*B;

% Compute y(t) and psi(t)
dt = 0.001;
t = 0:dt:500;

%vector for i, i+1
x = zeros(4,length(t));
%vector for i+0.5 (intermediate steps)
xtemp = zeros(4,1);
%vector array for slopes
f = {};
F = zeros(4,length(t));

u_var = [200,227,228,229];
colors = ['r','b','g','c','k','m',"#EDB120"];

figure;
hold on;

for i = 1:length(u_var)

   x = zeros(4,length(t));
   u = u_var(i)/3.6; %m/s

   % Define constants for dx2/d2t = Adx/dt + Bdel
   A = [-(Cf+Cr)/(m*u), -(a*Cf-b*Cr)/(m*u)-u;
     -(a*Cf-b*Cr)/(Iz*u), -((a^2)*Cf+(b^2)*Cr)/(Iz*u)];

   %slope at i
   f{1} = [x(3); x(4); A(1,1)*x(3) + A(1,2)*x(4) + B(1);
         A(2,1)*x(3) + A(2,2)*x(4) + B(2)];

   for n = 1:length(t)-1

      %find xi+.5 and slope at i+.5
      xtemp = x(:,n) + 0.5*dt*f{1};
```

```
    f{2} = [xtemp(3); xtemp(4);
       A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
       A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

    %new i+0.5 and slope
    xtemp = x(:,n) + 0.5*dt*f{2};
    f{3} = [xtemp(3); xtemp(4);
       A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
       A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

    %find xi+1 and slope
    xtemp(:) = x(:,n) + dt*f{3};
    f{4} = [xtemp(3); xtemp(4);
       A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
       A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

    f{5} = (1/6) .* f{1} + (1/3) .* f{2} + (1/3) .* f{3} + (1/6) .* f{4};
    xtemp(:) = x(:,n) + dt*f{5};

    x(:,n+1) = xtemp;
    f{1} = f{5};
    F(:,n) = f{1};

end

subplot(2,1,1);
plot(t, F(3,:),'color', colors(i), 'LineWidth', 1);
%xlabel('Time (s)');
ylabel('Lateral Accel. (m/s^2)');
ylim([-10,10])
title('RK4 Max Tangential Velocity');
hold on;
grid on;

subplot(2,1,2);
plot(t, x(4,:),'color', colors(i),'LineWidth', 1);
xlabel('Time (s)');
ylabel('Yaw Rate (rad/s)');
hold on;
grid on;
```

```
end

legend('u = 200 km/h', 'u = 227 km/h', 'u = 228 km/h', ...
      'u = 229 km/h');
```

Part B - Max Speed - RK4 Max Speed Safe

```
clc; clear;

m = 1400; %kg
a = 1.14; %m
b = 1.33; %m
Cf = 25000; %N/rad
Cr = 21000; %N/rad
Iz = 2420; %kgm^2

del = 0.1;
B = [Cf/m; (a*Cf)/Iz];
B = del.*B;

% Compute y(t) and psi(t)
dt = 0.001;
t = 0:dt:500;

%vector for i, i+1
x = zeros(4,length(t));
%vector for i+0.5 (intermediate steps)
xtemp = zeros(4,1);
%vector array for slopes
f = {};
F = zeros(4,length(t));

u_var = [96,97];
colors = ['r','g','b','c','k','m',"#EDB120"];

figure;
hold on;

for i = 1:length(u_var)

   x = zeros(4,length(t));
```

```
u = u_var(i)/3.6; %m/s

% Define constants for dx2/d2t = Adx/dt + Bdel
A = [-(Cf+Cr)/(m*u), -(a*Cf-b*Cr)/(m*u)-u;
  -(a*Cf-b*Cr)/(Iz*u), -((a^2)*Cf+(b^2)*Cr)/(Iz*u)];

%slope at i
f{1} = [x(3); x(4); A(1,1)*x(3) + A(1,2)*x(4) + B(1);
      A(2,1)*x(3) + A(2,2)*x(4) + B(2)];

for n = 1:length(t)-1

  %find xi+.5 and slope at i+.5
   xtemp = x(:,n) + 0.5*dt*f{1};
   f{2} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

   %new i+0.5 and slope
   xtemp = x(:,n) + 0.5*dt*f{2};
   f{3} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

   %find xi+1 and slope
   xtemp(:) = x(:,n) + dt*f{3};
   f{4} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

   f{5} = (1/6) .* f{1} + (1/3) .* f{2} + (1/3) .* f{3} + (1/6) .* f{4};
   xtemp(:) = x(:,n) + dt*f{5};

   x(:,n+1) = xtemp;
   f{1} = f{5};
   F(:,n) = f{1};

end

plot(t, F(3,:),'color', colors(i), 'LineWidth', 1);
%xlabel('Time (s)');
```

```
    ylabel('Lateral Accel. (m/s^2)');
    xlabel('Time (s)')
    ylim([-10,10]);
    xlim([-0.5,20]);
    title('Highest Safe Tangential Velocity');
    hold on;
    grid on;

end

legend('u = 96 km/h', 'u = 97 km/h', 'u = 228 km/h', ...
       'u = 229 km/h');
```

Part B - Max Speed - RK4 Performance

```
clc; clear;

m = 1400; %kg
a = 1.14; %m
b = 1.33; %m
Cf = 25000; %N/rad
Cr = 21000; %N/rad
Iz = 2420; %kgm^2
%u = 75/3.6; %km/hr

del = 0.1;
B = [Cf/m; (a*Cf)/Iz];
B = del.*B;

% Compute y(t) and psi(t)

dt = 0.001;
t = 0:dt:5;

% IC at t = 0 (given eq7)
ic = [0;0;0;0];

%vector for i, i+1
x = zeros(4,length(t));
x(:,1) = ic;
```

```
%vector for i+0.5 (intermediate steps)
xtemp = zeros(4,1);

%vector array for slopes
f = {};
F = zeros(4,length(t));
y_a = zeros(size(t));

u_var = [20,50,75,100,200];
colors = ['r','b','g','c','k','m',"#EDB120"];

figure;
hold on;

for i = 1:length(u_var)

   u = u_var(i)/3.6; %m/s

   % Define constants for dx2/d2t = Adx/dt + Bdel
   A = [-(Cf+Cr)/(m*u), -(a*Cf-b*Cr)/(m*u)-u;
     -(a*Cf-b*Cr)/(Iz*u), -((a^2)*Cf+(b^2)*Cr)/(Iz*u)];

   %slope at i
   f{1} = [x(3); x(4); A(1,1)*x(3) + A(1,2)*x(4) + B(1);
         A(2,1)*x(3) + A(2,2)*x(4) + B(2)];

   for n = 1:length(t)-1

      %find xi+.5 and slope at i+.5
      xtemp = x(:,n) + 0.5*dt*f{1};
      f{2} = [xtemp(3); xtemp(4);
         A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
         A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

      %new i+0.5 and slope
      xtemp = x(:,n) + 0.5*dt*f{2};
      f{3} = [xtemp(3); xtemp(4);
         A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
         A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

      %find xi+1 and slope
```

```
        xtemp = x(:,n) + dt*f{3};
        f{4} = [xtemp(3); xtemp(4);
            A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
            A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

        f{5} = (1/6)*f{1} + (1/3)*f{2} + (1/3)*f{3} + (1/6)*f{4};
        xtemp = x(:,n) + dt*f{5};

        F(:,n) = f{5};
        x(:,n+1) = xtemp;
        f{1} = f{5};

        y_a(n) = A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);

    end

    subplot(2,1,1);
    plot(t, y_a(:),'color', colors(i), 'LineWidth', 1);
    xlabel('Time (s)');
    ylabel('Lateral Accel. (m/s^2)');
    title('RK4 with Variable Tangential Velocity');
    hold on;
    grid on;

    subplot(2,1,2);
    plot(t, x(4,:),'color', colors(i),'LineWidth', 1);
    xlabel('Time (s)');
    ylabel('Yaw Rate (rad/s)');
    hold on;
    grid on;

    x = zeros(4,length(t));
    x(:,1) = ic;

end

legend('u = 20 km/h', 'u = 50 km/h', 'u = 75 km/h', ...
      'u = 100 km/h','u = 200 km/h');
```

Part C - Winter Condition - Snow Ice Max Speed

```
clc; clear;

m = 1400; %kg
a = 1.14; %m
b = 1.33; %m
mu = 0.3;
Cf = mu*25000; %N/rad
Cr = mu*21000; %N/rad
Iz = 2420; %kgm^2

% Compute y(t) and psi(t)
dt = 0.001;
t = 0:dt:50;

%vector array for slopes
f = {};
F = zeros(4,length(t));

del = [0.05,0.1,0.5];
u_var = [120,91,44];

colors = ['r','b','g'];

figure;
hold on;

for i = 1:length(u_var)

   %vector for i, i+1
   x = zeros(4,length(t));
   %vector for i+0.5 (intermediate steps)
   xtemp = zeros(4,1);

   B = [Cf/m; (a*Cf)/Iz];

   u = u_var(i)/3.6; %m/s
   B = del(i)*B;

   % Define constants for dx2/d2t = Adx/dt + Bdel
   A = [-(Cf+Cr)/(m*u), -(a*Cf-b*Cr)/(m*u)-u;
     -(a*Cf-b*Cr)/(Iz*u), -((a^2)*Cf+(b^2)*Cr)/(Iz*u)];
```

```
%slope at i
f{1} = [x(3); x(4); A(1,1)*x(3) + A(1,2)*x(4) + B(1);
      A(2,1)*x(3) + A(2,2)*x(4) + B(2)];

for n = 1:length(t)

  %find xi+.5 and slope at i+.5
   xtemp = x(:,n) + 0.5*dt*f{1};
   f{2} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

   %new i+0.5 and slope
   xtemp = x(:,n) + 0.5*dt*f{2};
   f{3} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

   %find xi+1 and slope
   xtemp(:) = x(:,n) + dt*f{3};
   f{4} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

   f{5} = (1/6) .* f{1} + (1/3) .* f{2} + (1/3) .* f{3} + (1/6) .* f{4};
   xtemp(:) = x(:,n) + dt*f{5};

   x(:,n+1) = xtemp;
   f{1} = f{5};
   F(:,n) = f{1};

end

plot(t, F(3,:),'color', colors(i), 'LineWidth', 1);
xlabel('Time (s)');
ylabel('Lateral Accel. (m/s^2)');
title('Max Speed at Different Turn Angles (Winter)');
%xlim([-0.5,10]);
hold on;
grid on;
```

```
end

legend('del = 0.05 (120 km/h)', 'del = 0.1 (91 km/h)',...
      'del = 0.5 (44 km/h)');
hold off;
```

Part C - Winter Condition - Snow Ice Test

```
clc; clear;

m = 1400; %kg
a = 1.14; %m
b = 1.33; %m
Cf = 25000; %N/rad
Cr = 21000; %N/rad
Iz = 2420; %kgm^2
mu = 0.3;

% Compute y(t) and psi(t)
dt = 0.001;
t = 0:dt:50;

%vector array for slopes
f = {};
F = zeros(4,length(t));

del = [0.05,0.1,0.5];
u_var = [120,91,44];

colors = ['r','b','g'];

figure;
hold on;

for i = 1:length(u_var)

   %vector for i, i+1
   x = zeros(4,length(t));
   %vector for i+0.5 (intermediate steps)
```

```
xtemp = zeros(4,1);4,length(t));

Cf = mu*25000;
Cr = mu*21000;

B = [Cf/m; (a*Cf)/Iz];

u = u_var(i)/3.6; %m/s
B = del(i)*B;

% Define constants for dx2/d2t = Adx/dt + Bdel
A = [-(Cf+Cr)/(m*u), -(a*Cf-b*Cr)/(m*u)-u;
  -(a*Cf-b*Cr)/(Iz*u), -((a^2)*Cf+(b^2)*Cr)/(Iz*u)];

%slope at i
f{1} = [x(3); x(4); A(1,1)*x(3) + A(1,2)*x(4) + B(1);
      A(2,1)*x(3) + A(2,2)*x(4) + B(2)];

for n = 1:length(t)-1

  %find xi+.5 and slope at i+.5
   xtemp = x(:,n) + 0.5*dt*f{1};
   f{2} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

   %new i+0.5 and slope
   xtemp = x(:,n) + 0.5*dt*f{2};
   f{3} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

   %find xi+1 and slope
   xtemp(:) = x(:,n) + dt*f{3};
   f{4} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

   f{5} = (1/6) .* f{1} + (1/3) .* f{2} + (1/3) .* f{3} + (1/6) .* f{4};
   xtemp(:) = x(:,n) + dt*f{5};
```

```
    x(:,n+1) = xtemp;
    f{1} = f{5};
    F(:,n) = f{1};

  end

  plot(t, F(3,:),'color', colors(i), 'LineWidth', 1);
  xlabel('Time (s)');
  ylabel('Lateral Accel. (m/s^2)');
  title('Driving on Winter Roads (mu = 0.3)');
  xlim([-0.5,30]);
  hold on;
  grid on;

end

legend('del = 0.05 (120 km/h)', 'del = 0.1 (91 km/h)',...
    'del = 0.5 (44 km/h)');
hold off;
```

Part C - Winter Condition - Snow Ice Turn Radii

```
clc; clear;

m = 1400; %kg
a = 1.14; %m
b = 1.33; %m
%Cf = 25000; %N/rad
%Cr = 21000; %N/rad
Iz = 2420; %kgm^2
mu = 0.3;
Cf = mu*25000;
Cr = mu*21000;

% Compute y(t) and psi(t)
dt = 0.001;
t = 0:dt:5;

%vector array for slopes
f = {};
F = zeros(4,length(t));
```

```
del = [0.05,0.1,0.5];
u_var = [132,96,48];

colors = ['r','b','g'];
radius = zeros(length(del),length(t));

figure;
hold on;

for i = 1:length(u_var)

   %vector for i, i+1
   x = zeros(4,length(t));
   %vector for i+0.5 (intermediate steps)
   xtemp = zeros(4,1);

   B = [Cf/m; (a*Cf)/Iz];

   u = u_var(i)/3.6; %m/s
   B = del(i)*B;

   % Define constants for dx2/d2t = Adx/dt + Bdel
   A = [-(Cf+Cr)/(m*u), -(a*Cf-b*Cr)/(m*u)-u;
     -(a*Cf-b*Cr)/(Iz*u), -((a^2)*Cf+(b^2)*Cr)/(Iz*u)];

   %slope at i
   f{1} = [x(3); x(4); A(1,1)*x(3) + A(1,2)*x(4) + B(1);
         A(2,1)*x(3) + A(2,2)*x(4) + B(2)];

   F_temp = zeros(2,1);
   xy_plot = zeros(2,length(t));

   for n = 1:length(t)-1

      F_temp = [u*cos(x(2,n)) - (x(3,n)+a*x(4,n))*sin(x(2,n));
           (x(3,n)+a*x(4,n))*cos(x(2,n)) + u*sin(x(2,n))];

     %find xi+.5 and slope at i+.5
      xtemp = x(:,n) + 0.5*dt*f{1};
      f{2} = [xtemp(3); xtemp(4);
```

```
        A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
        A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

    %new i+0.5 and slope
    xtemp = x(:,n) + 0.5*dt*f{2};
    f{3} = [xtemp(3); xtemp(4);
        A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
        A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

    %find xi+1 and slope
    xtemp(:) = x(:,n) + dt*f{3};
    f{4} = [xtemp(3); xtemp(4);
        A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
        A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

    f{5} = (1/6) .* f{1} + (1/3) .* f{2} + (1/3) .* f{3} + (1/6) .* f{4};
    xtemp(:) = x(:,n) + dt*f{5};

    x(:,n+1) = xtemp;
    f{1} = f{5};
    F(:,n) = f{1};

    xy_plot(:,n+1) = xy_plot(:,n) + dt*F_temp(:);

  end

  figure(i);
  plot(xy_plot(1,:), xy_plot(2,:), 'color', colors(i), 'LineWidth', 2);
  grid on;
  axis equal;
  xlabel('X Position (m)');
  ylabel('Y Position (m)');
  hold on;

  radius(i,:) = u./x(4,:);

end

%title('Fastest Stable Speed at Different Turn Angles');
%legend('del = 0.05 (71 km/h)', 'del = 0.1 (52 km/h)',...
%       'del = 0.5 (26 km/h)');
```

```
% hold off;
%
% figure;
% for j = 1:length(del)
%     plot(t,radius(j,:),'Color',colors(j),'LineWidth',1);
%     hold on;
% end
% grid on;
% xlim([0.5,10]);
% xlabel('Time (s)');
% ylabel('Radius (m)');
% hold off;
```

Part C - Winter Condition - All In One Kinematics

```
close all; clc; clear;

m = 1400; %kg
a = 1.14; %m
b = 1.33; %m
Cf = 25000; %N/rad
Cr = 21000; %N/rad
Iz = 2420; %kgm^2

% Compute y(t) and psi(t)
dt = 0.001;
t = 0:dt:5;

%vector array for slopes
f = {};
F = zeros(4,length(t));

del = [0.05,0.1,0.5];
%u_var = [120,91,44];
u_var = [132,96,48];

colors = ['k'];
radius = zeros(length(del),length(t));

figure;
```

```
hold on;

for i = 1:length(u_var)

   %vector for i, i+1
   x = zeros(4,length(t));
   %vector for i+0.5 (intermediate steps)
   xtemp = zeros(4,1);

   B = [Cf/m; (a*Cf)/Iz];

   u = u_var(i)/3.6; %m/s
   B = del(i)*B;

   % Define constants for dx2/d2t = Adx/dt + Bdel
   A = [-(Cf+Cr)/(m*u), -(a*Cf-b*Cr)/(m*u)-u;
     -(a*Cf-b*Cr)/(Iz*u), -((a^2)*Cf+(b^2)*Cr)/(Iz*u)];

   %slope at i
   f{1} = [x(3); x(4); A(1,1)*x(3) + A(1,2)*x(4) + B(1);
         A(2,1)*x(3) + A(2,2)*x(4) + B(2)];

   F_temp = zeros(2,1);
   xy_plot = zeros(2,length(t));

   for n = 1:length(t)-1

      F_temp = [u*cos(x(2,n)) - (x(3,n)+a*x(4,n))*sin(x(2,n));
           (x(3,n)+a*x(4,n))*cos(x(2,n)) + u*sin(x(2,n))];

     %find xi+.5 and slope at i+.5
      xtemp = x(:,n) + 0.5*dt*f{1};
      f{2} = [xtemp(3); xtemp(4);
         A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
         A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

      %new i+0.5 and slope
      xtemp = x(:,n) + 0.5*dt*f{2};
      f{3} = [xtemp(3); xtemp(4);
         A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
         A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];
```

```
        %find xi+1 and slope
        xtemp(:) = x(:,n) + dt*f{3};
        f{4} = [xtemp(3); xtemp(4);
           A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
           A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

        f{5} = (1/6) .* f{1} + (1/3) .* f{2} + (1/3) .* f{3} + (1/6) .* f{4};
        xtemp(:) = x(:,n) + dt*f{5};

        x(:,n+1) = xtemp;
        f{1} = f{5};
        F(:,n) = f{1};

        xy_plot(:,n+1) = xy_plot(:,n) + dt*F_temp(:);

    end

    figure(i);
    plot(xy_plot(1,:), xy_plot(2,:), 'color', colors(1), 'LineWidth', 2);
    grid on;
    axis equal;
    xlabel('X Position (m)');
    ylabel('Y Position (m)');
    hold on;

end

%--------------------------------------------------------------------------

%ICEY CONDITION
mu = 0.3;
Cf = mu*25000;
Cr = mu*21000;
Iz = 2420; %kgm^2

%vector array for slopes
f = {};
F = zeros(4,length(t));

del = [0.05,0.1,0.5];
```

```
u_var = [132,96,48];
%u_var = [72,53,26];

colors = ['r','b','g'];

for i = 1:length(u_var)

   %vector for i, i+1
   x = zeros(4,length(t));

   %vector for i+0.5 (intermediate steps)
   xtemp = zeros(4,1);

   B = [Cf/m; (a*Cf)/Iz];

   u = u_var(i)/3.6; %m/s
   B = del(i)*B;

   % Define constants for dx2/d2t = Adx/dt + Bdel
   A = [-(Cf+Cr)/(m*u), -(a*Cf-b*Cr)/(m*u)-u;
     -(a*Cf-b*Cr)/(Iz*u), -((a^2)*Cf+(b^2)*Cr)/(Iz*u)];

   %slope at i
   f{1} = [x(3); x(4); A(1,1)*x(3) + A(1,2)*x(4) + B(1);
         A(2,1)*x(3) + A(2,2)*x(4) + B(2)];

   F_temp = zeros(2,1);
   xy_plot = zeros(2,length(t));

   for n = 1:length(t)-1

      F_temp = [u*cos(x(2,n)) - (x(3,n)+a*x(4,n))*sin(x(2,n));
           (x(3,n)+a*x(4,n))*cos(x(2,n)) + u*sin(x(2,n))];

     %find xi+.5 and slope at i+.5
      xtemp = x(:,n) + 0.5*dt*f{1};
      f{2} = [xtemp(3); xtemp(4);
         A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
         A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];
```

```
    %new i+0.5 and slope
    xtemp = x(:,n) + 0.5*dt*f{2};
    f{3} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

    %find xi+1 and slope
    xtemp(:) = x(:,n) + dt*f{3};
    f{4} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

    f{5} = (1/6) .* f{1} + (1/3) .* f{2} + (1/3) .* f{3} + (1/6) .* f{4};
    xtemp(:) = x(:,n) + dt*f{5};

    x(:,n+1) = xtemp;
    f{1} = f{5};
    F(:,n) = f{1};

    xy_plot(:,n+1) = xy_plot(:,n) + dt*F_temp(:);

  end

  figure(i);
  plot(xy_plot(1,:), xy_plot(2,:), 'color', colors(i), 'LineWidth', 2);
  grid on;
  axis equal;
  legend('Summer','Winter');
  title("Winter vs Summer Turn del = " + num2str(del(i)));
  hold off;

end
```

Part C - Winter Condition - Regular Max Speed

```
clc; clear;

m = 1400; %kg
a = 1.14; %m
b = 1.33; %m
Cf = 25000; %N/rad
```

```
Cr = 21000; %N/rad
Iz = 2420; %kgm^2

% Compute y(t) and psi(t)
dt = 0.001;
t = 0:dt:50;

%vector array for slopes
f = {};
F = zeros(4,length(t));

del = [0.05,0.1,0.5];
u_var = [132,96,48];

colors = ['r','b','g'];

figure;
hold on;

for i = 1:length(u_var)

   %vector for i, i+1
   x = zeros(4,length(t));
   %vector for i+0.5 (intermediate steps)
   xtemp = zeros(4,1);

   B = [Cf/m; (a*Cf)/Iz];

   u = u_var(i)/3.6; %m/s
   B = del(i)*B;

   % Define constants for dx2/d2t = Adx/dt + Bdel
   A = [-(Cf+Cr)/(m*u), -(a*Cf-b*Cr)/(m*u)-u;
     -(a*Cf-b*Cr)/(Iz*u), -((a^2)*Cf+(b^2)*Cr)/(Iz*u)];

   %slope at i
   f{1} = [x(3); x(4); A(1,1)*x(3) + A(1,2)*x(4) + B(1);
         A(2,1)*x(3) + A(2,2)*x(4) + B(2)];

   for n = 1:length(t)-1
```

```
      %find xi+.5 and slope at i+.5
      xtemp = x(:,n) + 0.5*dt*f{1};
      f{2} = [xtemp(3); xtemp(4);
         A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
         A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

      %new i+0.5 and slope
      xtemp = x(:,n) + 0.5*dt*f{2};
      f{3} = [xtemp(3); xtemp(4);
         A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
         A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

      %find xi+1 and slope
      xtemp(:) = x(:,n) + dt*f{3};
      f{4} = [xtemp(3); xtemp(4);
         A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
         A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

      f{5} = (1/6) .* f{1} + (1/3) .* f{2} + (1/3) .* f{3} + (1/6) .* f{4};
      xtemp(:) = x(:,n) + dt*f{5};

      x(:,n+1) = xtemp;
      f{1} = f{5};
      F(:,n) = f{1};

   end

   plot(t, F(3,:),'color', colors(i), 'LineWidth', 1);
   xlabel('Time (s)');
   ylabel('Lateral Accel. (m/s^2)');
   title('Max Speed at Different Turn Angles (Summer)');
   xlim([-0.5,10]);
   hold on;
   grid on;

end

legend('del = 0.05 (132 km/h)', 'del = 0.1 (96 km/h)',...
      'del = 0.5 (48 km/h)');
hold off;
```

Part C - Winter Condition - Regular Turn Radii

```
clc; clear;

m = 1400; %kg
a = 1.14; %m
b = 1.33; %m
Cf = 25000; %N/rad
Cr = 21000; %N/rad
Iz = 2420; %kgm^2

% Compute y(t) and psi(t)
dt = 0.001;
t = 0:dt:5;

%vector array for slopes
f = {};
F = zeros(4,length(t));

del = [0.05,0.1,0.5];
u_var = [132,96,48];

colors = ['k'];
radius = zeros(length(del),length(t));

for i = 1:length(u_var)

   %vector for i, i+1
   x = zeros(4,length(t));
   %vector for i+0.5 (intermediate steps)
   xtemp = zeros(4,1);

   B = [Cf/m; (a*Cf)/Iz];

   u = u_var(i)/3.6; %m/s
   B = del(i)*B;

   % Define constants for dx2/d2t = Adx/dt + Bdel
   A = [-(Cf+Cr)/(m*u), -(a*Cf-b*Cr)/(m*u)-u;
     -(a*Cf-b*Cr)/(Iz*u), -((a^2)*Cf+(b^2)*Cr)/(Iz*u)];
```

```
%slope at i
f{1} = [x(3); x(4); A(1,1)*x(3) + A(1,2)*x(4) + B(1);
      A(2,1)*x(3) + A(2,2)*x(4) + B(2)];

F_temp = zeros(2,1);
xy_plot = zeros(2,length(t));

for n = 1:length(t)-1

   F_temp = [u*cos(x(2,n)) - (x(3,n)+a*x(4,n))*sin(x(2,n));
        (x(3,n)+a*x(4,n))*cos(x(2,n)) + u*sin(x(2,n))];

  %find xi+.5 and slope at i+.5
   xtemp = x(:,n) + 0.5*dt*f{1};
   f{2} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

   %new i+0.5 and slope
   xtemp = x(:,n) + 0.5*dt*f{2};
   f{3} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

   %find xi+1 and slope
   xtemp(:) = x(:,n) + dt*f{3};
   f{4} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

   f{5} = (1/6) .* f{1} + (1/3) .* f{2} + (1/3) .* f{3} + (1/6) .* f{4};
   xtemp(:) = x(:,n) + dt*f{5};

   x(:,n+1) = xtemp;
   f{1} = f{5};
   F(:,n) = f{1};

   xy_plot(:,n+1) = xy_plot(:,n) + dt*F_temp(:);

end
```

```
    figure(i);
    plot(xy_plot(1,:), xy_plot(2,:), 'k', 'LineWidth', 2);
    grid on;
    axis equal;
    legend(sprintf('Summer %g km/h', u_var(i)));
    title("Summer Turn del = " + num2str(del(i)));
    xlabel('X Position (m)');
    ylabel('Y Position (m)');
    hold off;

    radius(i,:) = u./x(4,:);

end

%title('Fastest Stable Speed at Different Turn Angles');
%legend('del = 0.05 (71 km/h)', 'del = 0.1 (52 km/h)',...
%       'del = 0.5 (26 km/h)');
%hold off;

% figure;
% for j = 1:length(del)
%     plot(t,radius(j,:),'Color',colors(j),'LineWidth',1);
%     hold on;
% end
% grid on;
% xlim([0.5,10]);
% xlabel('Time (s)');
% ylabel('Radius (m)');
% hold off;
```

Part C - Winter Condition - Snow Ice

```
clc; clear;

m = 1400; %kg
a = 1.14; %m
b = 1.33; %m
Cf = 25000; %N/rad
Cr = 21000; %N/rad
Iz = 2420; %kgm^2
mu = 0.3;
```

```
% Compute y(t) and psi(t)
dt = 0.001;
t = 0:dt:50;

%vector for i, i+1
x = zeros(4,length(t));
%vector for i+0.5 (intermediate steps)
xtemp = zeros(4,1);
%vector array for slopes
f = {};
F = zeros(4,length(t));

del = [0.05,0.1,0.5];
u_var = [71,52,26];

colors = ['r','b','g'];

figure;
hold on;

for i = 1:length(u_var)

   Cf = mu*25000;
   Cr = mu*21000;

   B = [Cf/m; (a*Cf)/Iz];

   u = u_var(i)/3.6; %m/s
   B = del(i)*B;

   % Define constants for dx2/d2t = Adx/dt + Bdel
   A = [-(Cf+Cr)/(m*u), -(a*Cf-b*Cr)/(m*u)-u;
     -(a*Cf-b*Cr)/(Iz*u), -((a^2)*Cf+(b^2)*Cr)/(Iz*u)];

   %slope at i
   f{1} = [x(3); x(4); A(1,1)*x(3) + A(1,2)*x(4) + B(1);
         A(2,1)*x(3) + A(2,2)*x(4) + B(2)];

   for n = 1:length(t)-1
```

```
      %find xi+.5 and slope at i+.5
      xtemp = x(:,n) + 0.5*dt*f{1};
      f{2} = [xtemp(3); xtemp(4);
         A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
         A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

      %new i+0.5 and slope
      xtemp = x(:,n) + 0.5*dt*f{2};
      f{3} = [xtemp(3); xtemp(4);
         A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
         A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

      %find xi+1 and slope
      xtemp(:) = x(:,n) + dt*f{3};
      f{4} = [xtemp(3); xtemp(4);
         A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
         A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

      f{5} = (1/6) .* f{1} + (1/3) .* f{2} + (1/3) .* f{3} + (1/6) .* f{4};
      xtemp(:) = x(:,n) + dt*f{5};

      x(:,n+1) = xtemp;
      f{1} = f{5};
      F(:,n) = f{1};

   end

   %subplot(2,1,1);
   plot(t, F(3,:),'color', colors(i), 'LineWidth', 1);
   xlabel('Time (s)');
   ylabel('Lateral Accel. (m/s^2)');
   title('Driving on Winter Roads (mu = 0.3)');
   ylim([-3.5,3.5]);
   xlim([-0.5,30]);
   hold on;
   grid on;

   x = zeros(4,length(t));

end
```

```
legend('del = 0.05 (71 km/h)', 'del = 0.1 (52 km/h)',...
      'del = 0.5 (26 km/h)');
```

Part C - Winter Condition - Snow Ice Kinematics

```
clc; clear;

m = 1400; %kg
a = 1.14; %m
b = 1.33; %m
Cf = 25000; %N/rad
Cr = 21000; %N/rad
Iz = 2420; %kgm^2
mu = 0.3;

% Compute y(t) and psi(t)
dt = 0.001;
t = 0:dt:20;

%vector for i, i+1
x = zeros(4,length(t));
%vector for i+0.5 (intermediate steps)
xtemp = zeros(4,1);
%vector array for slopes
f = {};
F = zeros(4,length(t));

del = [0.05,0.1,0.5];
u_var = [71,52,26];

colors = ['r','b','g'];

figure;
hold on;

for i = 1:length(u_var)

   x = zeros(4,length(t));

   Cf = mu*25000;
   Cr = mu*21000;
```

```
B = [Cf/m; (a*Cf)/Iz];

u = u_var(i)/3.6; %m/s
B = del(i)*B;

% Define constants for dx2/d2t = Adx/dt + Bdel
A = [-(Cf+Cr)/(m*u), -(a*Cf-b*Cr)/(m*u)-u;
  -(a*Cf-b*Cr)/(Iz*u), -((a^2)*Cf+(b^2)*Cr)/(Iz*u)];

%slope at i
f{1} = [x(3); x(4); A(1,1)*x(3) + A(1,2)*x(4) + B(1);
      A(2,1)*x(3) + A(2,2)*x(4) + B(2)];

F_temp = zeros(2,1);
xy_plot = zeros(2,length(t));

for n = 1:length(t)-1

   F_temp = [u*cos(x(2,n)) - (x(3,n)+a*x(4,n))*sin(x(2,n));
        (x(3,n)+a*x(4,n))*cos(x(2,n)) + u*sin(x(2,n))];

  %find xi+.5 and slope at i+.5
   xtemp = x(:,n) + 0.5*dt*f{1};
   f{2} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

   %new i+0.5 and slope
   xtemp = x(:,n) + 0.5*dt*f{2};
   f{3} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

   %find xi+1 and slope
   xtemp(:) = x(:,n) + dt*f{3};
   f{4} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

   f{5} = (1/6) .* f{1} + (1/3) .* f{2} + (1/3) .* f{3} + (1/6) .* f{4};
```

```
        xtemp(:) = x(:,n) + dt*f{5};

        x(:,n+1) = xtemp;
        f{1} = f{5};
        F(:,n) = f{1};

        xy_plot(:,n+1) = xy_plot(:,n) + dt*F_temp(:);

    end

    plot(xy_plot(1,:), xy_plot(2,:), 'color', colors(i), 'LineWidth', 2);
    grid on;
    axis equal;
    xlabel('X Position (m)');
    ylabel('Y Position (m)');
    x = zeros(4,length(t));
    xtemp = zeros(4,1);

end

title('Fastest Stable Speed at Different Turn Angles');
legend('del = 0.05 (71 km/h)', 'del = 0.1 (52 km/h)',...
        'del = 0.5 (26 km/h)');
```

Part C - Winter Tires - Winter Tires

```
clc; clear;

m = 1400; %kg
a = 1.14; %m
b = 1.33; %m
Iz = 2420; %kgm^2
u = 70/3.6;

Cf_var = [0,100,20000]; %N/rad
del = [0.3,0.1,0.05];

% Compute y(t) and psi(t)
dt = 0.001;
t = 0:dt:30;
```

```
%vector for i, i+1
x = zeros(4,length(t));
%vector for i+0.5 (intermediate steps)
xtemp = zeros(4,1);
%vector array for slopes
f = {};
F = zeros(4,length(t));

colors = ['b','r','g','c','k','m',"#EDB120"];

figure;
hold on;

radius = zeros(size(x));

for i = 1:length(Cf_var)

   x = zeros(4,length(t));
   xtemp = zeros(4,1);

   Cf = Cf_var(i);
   Cr = Cf;

   % Define constants for dx2/d2t = Adx/dt + Bdel
   A = [-(Cf+Cr)/(m*u), -(a*Cf-b*Cr)/(m*u)-u;
     -(a*Cf-b*Cr)/(Iz*u), -((a^2)*Cf+(b^2)*Cr)/(Iz*u)];

   B = [Cf/m; (a*Cf)/Iz];
   B = del(i).*B;

   %slope at i
   f{1} = [x(3); x(4); A(1,1)*x(3) + A(1,2)*x(4) + B(1);
         A(2,1)*x(3) + A(2,2)*x(4) + B(2)];

   F_temp = zeros(2,1);
   xy_plot = zeros(2,length(t));

   for n = 1:length(t)-1

      F_temp = [u*cos(x(2,n)) - (x(3,n)+a*x(4,n))*sin(x(2,n));
           (x(3,n)+a*x(4,n))*cos(x(2,n)) + u*sin(x(2,n))];
```

```
      %find xi+.5 and slope at i+.5
      xtemp = x(:,n) + 0.5*dt*f{1};
      f{2} = [xtemp(3); xtemp(4);
         A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
         A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

      %new i+0.5 and slope
      xtemp = x(:,n) + 0.5*dt*f{2};
      f{3} = [xtemp(3); xtemp(4);
         A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
         A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

      %find xi+1 and slope
      xtemp(:) = x(:,n) + dt*f{3};
      f{4} = [xtemp(3); xtemp(4);
         A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
         A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

      f{5} = (1/6) .* f{1} + (1/3) .* f{2} + (1/3) .* f{3} + (1/6) .* f{4};
      xtemp(:) = x(:,n) + dt*f{5};

      x(:,n+1) = xtemp;
      f{1} = f{5};
      F(:,n) = f{1};

      xy_plot(:,n+1) = xy_plot(:,n) + dt*F_temp(:);

   end

   plot(xy_plot(1,:), xy_plot(2,:), 'color', colors(i), 'LineWidth', 2);
   grid on;
   axis equal;
   xlabel('X Position (m)');
   ylabel('Y Position (m)');
   hold on;

   radius(i,:) = u./x(4,:);

end
```

```
title('All Season Tires Kinematics');
legend('del = 0.3','del = 0.1','del = 0.05');
hold off;

figure;
hold on;
plot(t,radius(2,:),'r','LineWidth',1);
plot(t,radius(3,:),'g','LineWidth',1);
grid on;
ylim([0,1000]);
xlim([0.1,10]);
xlabel('Time (s)');
ylabel('Radius (m)');
legend('del = 0.1','del = 0.05');
title('All Season Tires Radius of Turn');
hold off;
```

Part C - Winter Tires - Winter Tires 2

```
clc; clear;

m = 1400; %kg
a = 1.14; %m
b = 1.33; %m
Cf_var = [0,5000,5000,20000]; %N/rad
Iz = 2420; %kgm^2
u = 70/3.6;

del = [0.3,0.1,0.05];

% Compute y(t) and psi(t)
dt = 0.001;
t = 0:dt:30;

%vector for i, i+1
x = zeros(4,length(t));
%vector for i+0.5 (intermediate steps)
xtemp = zeros(4,1);
%vector array for slopes
f = {};
F = zeros(4,length(t));
```

```
colors = ['b','r','g','c','k','m',"#EDB120"];

figure;
hold on;

radius = zeros(size(x));

for i = 1:length(Cf_var)

   x = zeros(4,length(t));
   xtemp = zeros(4,1);

   Cf = Cf_var(i);
   Cr = Cf;

   % Define constants for dx2/d2t = Adx/dt + Bdel
   A = [-(Cf+Cr)/(m*u), -(a*Cf-b*Cr)/(m*u)-u;
     -(a*Cf-b*Cr)/(Iz*u), -((a^2)*Cf+(b^2)*Cr)/(Iz*u)];

   B = [Cf/m; (a*Cf)/Iz];
   B = del(i).*B;

   %slope at i
   f{1} = [x(3); x(4); A(1,1)*x(3) + A(1,2)*x(4) + B(1);
         A(2,1)*x(3) + A(2,2)*x(4) + B(2)];

   F_temp = zeros(2,1);
   xy_plot = zeros(2,length(t));

   for n = 1:length(t)-1

      F_temp = [u*cos(x(2,n)) - (x(3,n)+a*x(4,n))*sin(x(2,n));
           (x(3,n)+a*x(4,n))*cos(x(2,n)) + u*sin(x(2,n))];

     %find xi+.5 and slope at i+.5
      xtemp = x(:,n) + 0.5*dt*f{1};
      f{2} = [xtemp(3); xtemp(4);
         A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
         A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];
```

```
        %new i+0.5 and slope
        xtemp = x(:,n) + 0.5*dt*f{2};
        f{3} = [xtemp(3); xtemp(4);
           A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
           A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

        %find xi+1 and slope
        xtemp(:) = x(:,n) + dt*f{3};
        f{4} = [xtemp(3); xtemp(4);
           A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
           A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

        f{5} = (1/6) .* f{1} + (1/3) .* f{2} + (1/3) .* f{3} + (1/6) .* f{4};
        xtemp(:) = x(:,n) + dt*f{5};

        x(:,n+1) = xtemp;
        f{1} = f{5};
        F(:,n) = f{1};

        xy_plot(:,n+1) = xy_plot(:,n) + dt*F_temp(:);

    end

    plot(xy_plot(1,:), xy_plot(2,:), 'color', colors(i), 'LineWidth', 2);
    grid on;
    axis equal;
    xlabel('X Position (m)');
    ylabel('Y Position (m)');
    hold on;

    radius(i,:) = u./x(4,:);

end

title('Winter Tires Kinematics');
legend('del = 0.3','del = 0.1','del = 0.05');
hold off;

figure;
hold on;
plot(t,radius(2,:),'r','LineWidth',1);
```

```
plot(t,radius(3,:),'g','LineWidth',1);
grid on;
ylim([0,1000]);
xlim([0.1,10]);
xlabel('Time (s)');
ylabel('Radius (m)');
legend('del = 0.1','del = 0.05');
title('Winter Tires Radius of Turn');
hold off;
```

Part C - Additional Weight - RK4 Weight 2 Kinematics

```
clc; clear;

m = 1400; %kg
a = 1.14; %m
%b = 1.33; %m
Cf = 25000; %N/rad
Cr = 21000; %N/rad
Iz = 2420; %kgm^2

%extra weight
load = 50; %kg

%find a and b for m+load
a = (m*a)/(m+load);
b = 2.47-a;
m = m + load;

del = 0.1;
B = [Cf/m; (a*Cf)/Iz];
B = del.*B;

% Compute y(t) and psi(t)
dt = 0.001;
t = 0:dt:10;

%vector for i+0.5 (intermediate steps)
xtemp = zeros(4,1);
%vector array for slopes
f = {};
```

```
F = zeros(4,length(t));

u_var = [100,200,300];
colors = ['r','b','g','c','k','m',"#EDB120"];

figure
hold on;

for i = 1:length(u_var)

   %vector for i, i+1
   x = zeros(4,length(t));

   u = u_var(i)/3.6; %m/s

   % Define constants for dx2/d2t = Adx/dt + Bdel
   A = [-(Cf+Cr)/(m*u), -(a*Cf-b*Cr)/(m*u)-u;
     -(a*Cf-b*Cr)/(Iz*u), -((a^2)*Cf+(b^2)*Cr)/(Iz*u)];

   %slope at i
   f{1} = [x(3); x(4); A(1,1)*x(3) + A(1,2)*x(4) + B(1);
         A(2,1)*x(3) + A(2,2)*x(4) + B(2)];

   F_temp = zeros(2,1);
   xy_plot = zeros(2,length(t));

   for n = 1:length(t)-1

      F_temp = [u*cos(x(2,n)) - (x(3,n)+a*x(4,n))*sin(x(2,n));
           (x(3,n)+a*x(4,n))*cos(x(2,n)) + u*sin(x(2,n))];

     %find xi+.5 and slope at i+.5
      xtemp = x(:,n) + 0.5*dt*f{1};
      f{2} = [xtemp(3); xtemp(4);
        A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
        A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

      %new i+0.5 and slope
      xtemp = x(:,n) + 0.5*dt*f{2};
      f{3} = [xtemp(3); xtemp(4);
        A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
```

```
        A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

    %find xi+1 and slope
    xtemp(:) = x(:,n) + dt*f{3};
    f{4} = [xtemp(3); xtemp(4);
        A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
        A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

    f{5} = (1/6) .* f{1} + (1/3) .* f{2} + (1/3) .* f{3} + (1/6) .* f{4};
    xtemp(:) = x(:,n) + dt*f{5};

    x(:,n+1) = xtemp;
    f{1} = f{5};
    F(:,n) = f{1};

    xy_plot(:,n+1) = xy_plot(:,n) + dt*F_temp(:);

  end

  %figure
  plot(xy_plot(1,:), xy_plot(2,:), 'color', colors(i), 'LineWidth', 2);
  hold on;
  grid on;
  axis equal;
  xlabel('X Position (m)');
  ylabel('Y Position (m)');

end

legend('100 km/h', '200 km/h', '300 km/h');
hold off;
```

Part C - Additional Weight - RK4 Weight Check

```
clc; clear;

m = 1400; %kg
%a = 1.14; %m
b = 1.33; %m
Cf = 25000; %N/rad
Cr = 21000; %N/rad
```

```
Iz = 2420; %kgm^2

%extra weight
load = 50; %kg

%find a and b for m+load
b = (m*b)/(m+load);
a = 2.47-b;
m = m + load;

del = 0.1;
B = [Cf/m; (a*Cf)/Iz];
B = del.*B;

% Compute y(t) and psi(t)
dt = 0.001;
t = 0:dt:200;

%vector for i, i+1
x = zeros(4,length(t));
%vector for i+0.5 (intermediate steps)
xtemp = zeros(4,1);
%vector array for slopes
f = {};
F = zeros(4,length(t));

u_var = [100,102,103,104];
colors = ['r','b','g','c','k','m',"#EDB120"];

figure;
hold on;

for i = 1:length(u_var)

   u = u_var(i)/3.6; %m/s

   % Define constants for dx2/d2t = Adx/dt + Bdel
   A = [-(Cf+Cr)/(m*u), -(a*Cf-b*Cr)/(m*u)-u;
     -(a*Cf-b*Cr)/(Iz*u), -((a^2)*Cf+(b^2)*Cr)/(Iz*u)];

   %slope at i
```

```
f{1} = [x(3); x(4); A(1,1)*x(3) + A(1,2)*x(4) + B(1);
      A(2,1)*x(3) + A(2,2)*x(4) + B(2)];

for n = 1:length(t)-1

  %find xi+.5 and slope at i+.5
   xtemp = x(:,n) + 0.5*dt*f{1};
   f{2} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

   %new i+0.5 and slope
   xtemp = x(:,n) + 0.5*dt*f{2};
   f{3} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

   %find xi+1 and slope
   xtemp(:) = x(:,n) + dt*f{3};
   f{4} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

   f{5} = (1/6) .* f{1} + (1/3) .* f{2} + (1/3) .* f{3} + (1/6) .* f{4};
   xtemp(:) = x(:,n) + dt*f{5};

   x(:,n+1) = xtemp;
   f{1} = f{5};
   F(:,n) = f{1};

end

subplot(2,1,1);
plot(t, F(3,:),'color', colors(i), 'LineWidth', 1);
%xlabel('Time (s)');
ylabel('Lateral Accel. (m/s^2)');
title('RK4 with Variable Tangential Velocity');
hold on;
grid on;

subplot(2,1,2);
```

```
    plot(t, x(4,:),'color', colors(i),'LineWidth', 1);
    xlabel('Time (s)');
    ylabel('Yaw Rate (rad/s)');
    hold on;
    grid on;

    x = zeros(4,length(t));

end

legend('u = 100 km/h', 'u = 102 km/h', 'u = 103 km/h', ...
       'u = 104 km/h');
```

Part C - Additional Weight - RK4 Weight Check 2

```
clc; clear;

m = 1400; %kg
a = 1.14; %m
%b = 1.33; %m
Cf = 25000; %N/rad
Cr = 21000; %N/rad
Iz = 2420; %kgm^2

%extra weight
load = 50; %kg

%find a and b for m+load
a = (m*a)/(m+load);
b = 2.47-a;
m = m + load;

del = 0.1;
B = [Cf/m; (a*Cf)/Iz];
B = del.*B;

%compute y(t) and psi(t)
dt = 0.001;
t = 0:dt:20;
```

```
%vector for i+0.5 (intermediate steps)
xtemp = zeros(4,1);
%vector array for slopes
f = {};
F = zeros(4,length(t));

u_var = [50,75,100,200,300];
colors = ['r','b','g','c','k','m',"#EDB120"];

figure;
hold on;

%test = zeros(2,length(u_var));

for i = 1:length(u_var)

   %vector for i, i+1
   x = zeros(4,length(t));
   u = u_var(i)/3.6; %m/s

   %define constants for dx2/d2t = Adx/dt + Bdel
   A = [-(Cf+Cr)/(m*u), -(a*Cf-b*Cr)/(m*u)-u;
     -(a*Cf-b*Cr)/(Iz*u), -((a^2)*Cf+(b^2)*Cr)/(Iz*u)];

   %test(:,i) = eig(A);

   %slope at i
   f{1} = [x(3); x(4); A(1,1)*x(3) + A(1,2)*x(4) + B(1);
         A(2,1)*x(3) + A(2,2)*x(4) + B(2)];

   for n = 1:length(t)-1

     %find xi+.5 and slope at i+.5
      xtemp = x(:,n) + 0.5*dt*f{1};
      f{2} = [xtemp(3); xtemp(4);
         A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
         A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

      %new i+0.5 and slope
      xtemp = x(:,n) + 0.5*dt*f{2};
      f{3} = [xtemp(3); xtemp(4);
```

```
        A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
        A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

    %find xi+1 and slope
    xtemp(:) = x(:,n) + dt*f{3};
    f{4} = [xtemp(3); xtemp(4);
        A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
        A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

    f{5} = (1/6) .* f{1} + (1/3) .* f{2} + (1/3) .* f{3} + (1/6) .* f{4};
    xtemp(:) = x(:,n) + dt*f{5};

    x(:,n+1) = xtemp;
    f{1} = f{5};
    F(:,n) = f{1};

  end

  %subplot(2,2,1);
  plot(t, F(3,:),'color', colors(i), 'LineWidth', 1);
  xlabel('Time (s)');
  ylabel('Lateral Accel. (m/s^2)');
  %title('RK4 Max Tangential Velocity with 50kg in Front Trunk');
  hold on;
  grid on;

end


legend('u = 50 km/h', 'u = 75 km/h', 'u = 100 km/h', ...
    'u = 200 km/h', 'u = 300 km/h');
```

Part C - Additional Weight - RK4 Weight Check 2 Steering

```
clc; clear;

m = 1400; %kg
a = 1.14; %m
%b = 1.33; %m
Cf = 25000; %N/rad
```

```
Cr = 21000; %N/rad
Iz = 2420; %kgm^2

%extra weight
load = 50; %kg

%find a and b for m+load
a = (m*a)/(m+load);
b = 2.47-a;
m = m + load;

del = 0.1;
B = [Cf/m; (a*Cf)/Iz];
B = del.*B;

%compute y(t) and psi(t)
dt = 0.001;
t = 0:dt:20;

%vector for i+0.5 (intermediate steps)
xtemp = zeros(4,1);
%vector array for slopes
f = {};
F = zeros(4,length(t));

u_var = [50,75,100,200,300];
colors = ['r','b','g','c','k','m',"#EDB120"];

figure;
hold on;

%test = zeros(2,length(u_var));

for i = 1:length(u_var)

   %vector for i, i+1
   x = zeros(4,length(t));
   u = u_var(i)/3.6; %m/s

   %define constants for dx2/d2t = Adx/dt + Bdel
   A = [-(Cf+Cr)/(m*u), -(a*Cf-b*Cr)/(m*u)-u;
```

```
  -(a*Cf-b*Cr)/(Iz*u), -((a^2)*Cf+(b^2)*Cr)/(Iz*u)];

%test(:,i) = eig(A);

%slope at i
f{1} = [x(3); x(4); A(1,1)*x(3) + A(1,2)*x(4) + B(1);
      A(2,1)*x(3) + A(2,2)*x(4) + B(2)];

for n = 1:length(t)-1

  %find xi+.5 and slope at i+.5
   xtemp = x(:,n) + 0.5*dt*f{1};
   f{2} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

   %new i+0.5 and slope
   xtemp = x(:,n) + 0.5*dt*f{2};
   f{3} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

   %find xi+1 and slope
   xtemp(:) = x(:,n) + dt*f{3};
   f{4} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

   f{5} = (1/6) .* f{1} + (1/3) .* f{2} + (1/3) .* f{3} + (1/6) .* f{4};
   xtemp(:) = x(:,n) + dt*f{5};

   x(:,n+1) = xtemp;
   f{1} = f{5};
   F(:,n) = f{1};

end

radius = u./x(4,:);

% plot(t,radius,'Color',colors(i),'LineWidth',1);
% grid on;
```

```
    % xlim([1,10]);
    % xlabel('Time (s)');
    % ylabel('Radius (m)');
    % hold on;

    handling = ((m/Cf - m/Cr)*(u*u))./radius;
    plot(1./radius,handling,'Color',colors(i),'LineWidth',1);
    hold on;

end

legend('u = 50 km/h', 'u = 75 km/h', 'u = 100 km/h', ...
       'u = 200 km/h', 'u = 300 km/h');
```

Part C - Additional Weight - RK4 Weight Kinematics

```
clc; clear;

m = 1400; %kg
%a = 1.14; %m
b = 1.33; %m
Cf = 25000; %N/rad
Cr = 21000; %N/rad
Iz = 2420; %kgm^2

%extra weight
load = 50; %kg

%find a and b for m+load
b = (m*b)/(m+load);
a = 2.47-b;
m = m + load;

del = 0.1;
B = [Cf/m; (a*Cf)/Iz];
B = del.*B;

% Compute y(t) and psi(t)
dt = 0.001;
t = 0:dt:20;
```

```
%vector for i, i+1
x = zeros(4,length(t));
%vector for i+0.5 (intermediate steps)
xtemp = zeros(4,1);
%vector array for slopes
f = {};
F = zeros(4,length(t));

u_var = [100,200];
colors = ['b','r','g','c','k','m',"#EDB120"];

figure;
hold on;

for i = 1:length(u_var)

   u = u_var(i)/3.6; %m/s

   % Define constants for dx2/d2t = Adx/dt + Bdel
   A = [-(Cf+Cr)/(m*u), -(a*Cf-b*Cr)/(m*u)-u;
     -(a*Cf-b*Cr)/(Iz*u), -((a^2)*Cf+(b^2)*Cr)/(Iz*u)];

   %slope at i
   f{1} = [x(3); x(4); A(1,1)*x(3) + A(1,2)*x(4) + B(1);
         A(2,1)*x(3) + A(2,2)*x(4) + B(2)];

   F_temp = zeros(2,1);
   xy_plot = zeros(2,length(t));

   for n = 1:length(t)-1

      F_temp = [u*cos(x(2,n)) - (x(3,n)+a*x(4,n))*sin(x(2,n));
           (x(3,n)+a*x(4,n))*cos(x(2,n)) + u*sin(x(2,n))];

     %find xi+.5 and slope at i+.5
      xtemp = x(:,n) + 0.5*dt*f{1};
      f{2} = [xtemp(3); xtemp(4);
         A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
         A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];
```

```
    %new i+0.5 and slope
    xtemp = x(:,n) + 0.5*dt*f{2};
    f{3} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

    %find xi+1 and slope
    xtemp(:) = x(:,n) + dt*f{3};
    f{4} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

    f{5} = (1/6) .* f{1} + (1/3) .* f{2} + (1/3) .* f{3} + (1/6) .* f{4};
    xtemp(:) = x(:,n) + dt*f{5};

    x(:,n+1) = xtemp;
    f{1} = f{5};
    F(:,n) = f{1};

    xy_plot(:,n+1) = xy_plot(:,n) + dt*F_temp(:);

  end

  %figure
  plot(xy_plot(1,:), xy_plot(2,:), 'color', colors(i), 'LineWidth', 2);
  grid on;
  axis equal;
  xlabel('X Position (m)');
  ylabel('Y Position (m)');
  legend('100 km/h')
  x = zeros(4,length(t));
  xtemp = zeros(4,1);

end

legend('100 km/h', '200 km/h');
title('Radius of Turn with 50kg in Front Trunk');
hold off;
```

Part C - Additional Weight - Weight Check 1 Steering Radius

```
clc; clear;

m = 1400; %kg
%a = 1.14; %m
b = 1.33; %m
Cf = 25000; %N/rad
Cr = 21000; %N/rad
Iz = 2420; %kgm^2

%extra weight
load = 50; %kg

%find a and b for m+load
b = (m*b)/(m+load);
a = 2.47-b;
m = m + load;

del = 0.1;
B = [Cf/m; (a*Cf)/Iz];
B = del.*B;

% Compute y(t) and psi(t)
dt = 0.001;
t = 0:dt:100;

%vector for i+0.5 (intermediate steps)
xtemp = zeros(4,1);
%vector array for slopes
f = {};
F = zeros(4,length(t));

u_var = [100];
colors = ['r','b','g','c','k','m',"#EDB120"];

for i = 1:length(u_var)

   %vector for i, i+1
   x = zeros(4,length(t));

   u = u_var(i)/3.6; %m/s
```

```
% Define constants for dx2/d2t = Adx/dt + Bdel
A = [-(Cf+Cr)/(m*u), -(a*Cf-b*Cr)/(m*u)-u;
  -(a*Cf-b*Cr)/(Iz*u), -((a^2)*Cf+(b^2)*Cr)/(Iz*u)];

%slope at i
f{1} = [x(3); x(4); A(1,1)*x(3) + A(1,2)*x(4) + B(1);
      A(2,1)*x(3) + A(2,2)*x(4) + B(2)];

F_temp = zeros(2,1);
xy_plot = zeros(2,length(t));

for n = 1:length(t)-1

   F_temp = [u*cos(x(2,n)) - (x(3,n)+a*x(4,n))*sin(x(2,n));
        (x(3,n)+a*x(4,n))*cos(x(2,n)) + u*sin(x(2,n))];

  %find xi+.5 and slope at i+.5
   xtemp = x(:,n) + 0.5*dt*f{1};
   f{2} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

   %new i+0.5 and slope
   xtemp = x(:,n) + 0.5*dt*f{2};
   f{3} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

   %find xi+1 and slope
   xtemp(:) = x(:,n) + dt*f{3};
   f{4} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

   f{5} = (1/6) .* f{1} + (1/3) .* f{2} + (1/3) .* f{3} + (1/6) .* f{4};
   xtemp(:) = x(:,n) + dt*f{5};

   x(:,n+1) = xtemp;
   f{1} = f{5};
   F(:,n) = f{1};
```

```
    end

    radius = u./x(4,:);

    plot(t,radius,'b','LineWidth',1);
    grid on;
    xlim([1,50]);
    xlabel('Time (s)');
    ylabel('Radius (m)');
    hold on;

end

legend('100 km/h', '200 km/h');
title('Radius of Turn with 50kg in Rear Trunk');
```

Part C - Additional Weight - Weight Check 2 Steering Radius

```
clc; clear;

m = 1400; %kg
a = 1.14; %m
%b = 1.33; %m
Cf = 25000; %N/rad
Cr = 21000; %N/rad
Iz = 2420; %kgm^2

%extra weight
load = 50; %kg

%find a and b for m+load
a = (m*a)/(m+load);
b = 2.47-a;
m = m + load;

del = 0.1;
B = [Cf/m; (a*Cf)/Iz];
B = del.*B;

% Compute y(t) and psi(t)
dt = 0.001;
```

```
t = 0:dt:100;

%vector for i+0.5 (intermediate steps)
xtemp = zeros(4,1);
%vector array for slopes
f = {};
F = zeros(4,length(t));

u_var = [100];
colors = ['r','b','g','c','k','m',"#EDB120"];

for i = 1:length(u_var)

   %vector for i, i+1
   x = zeros(4,length(t));

   u = u_var(i)/3.6; %m/s

   % Define constants for dx2/d2t = Adx/dt + Bdel
   A = [-(Cf+Cr)/(m*u), -(a*Cf-b*Cr)/(m*u)-u;
     -(a*Cf-b*Cr)/(Iz*u), -((a^2)*Cf+(b^2)*Cr)/(Iz*u)];

   %slope at i
   f{1} = [x(3); x(4); A(1,1)*x(3) + A(1,2)*x(4) + B(1);
         A(2,1)*x(3) + A(2,2)*x(4) + B(2)];

   F_temp = zeros(2,1);
   xy_plot = zeros(2,length(t));

   for n = 1:length(t)-1

      F_temp = [u*cos(x(2,n)) - (x(3,n)+a*x(4,n))*sin(x(2,n));
           (x(3,n)+a*x(4,n))*cos(x(2,n)) + u*sin(x(2,n))];

     %find xi+.5 and slope at i+.5
      xtemp = x(:,n) + 0.5*dt*f{1};
      f{2} = [xtemp(3); xtemp(4);
         A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
         A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

      %new i+0.5 and slope
```

```
    xtemp = x(:,n) + 0.5*dt*f{2};
    f{3} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

    %find xi+1 and slope
    xtemp(:) = x(:,n) + dt*f{3};
    f{4} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

    f{5} = (1/6) .* f{1} + (1/3) .* f{2} + (1/3) .* f{3} + (1/6) .* f{4};
    xtemp(:) = x(:,n) + dt*f{5};

    x(:,n+1) = xtemp;
    f{1} = f{5};
    F(:,n) = f{1};

  end

  radius = u./x(4,:);

  plot(t,radius,'b','LineWidth',1);
  grid on;
  xlim([1,50]);
  xlabel('Time (s)');
  ylabel('Radius (m)');
  hold on;

end

legend('100 km/h', '200 km/h');
title('Radius of Turn with 50kg in Front Trunk');
hold off;
```

Part D - Tuning Handling - Handling 1

```
clc; clear;

m = 1400; %kg
a = 1.14; %m
```

```
b = 1.33; %m
Iz = 2420; %kgm^2
u = 300/3.6;

Cf_var = [21000,25000]; %N/rad
Cr_var = [21000];
del = 0.1;

% Compute y(t) and psi(t)
dt = 0.001;
t = 0:dt:10;

%vector for i, i+1
x = zeros(4,length(t));
%vector for i+0.5 (intermediate steps)
xtemp = zeros(4,1);
%vector array for slopes
f = {};
F = zeros(4,length(t));

colors = ['b','r','g','c','k','m',"#EDB120"];

figure;
hold on;

radius = zeros(size(x));

for i = 1:length(Cf_var)

   x = zeros(4,length(t));
   xtemp = zeros(4,1);

   Cf = Cf_var(i);
   Cr = Cr_var(1);

   % Define constants for dx2/d2t = Adx/dt + Bdel
   A = [-(Cf+Cr)/(m*u), -(a*Cf-b*Cr)/(m*u)-u;
     -(a*Cf-b*Cr)/(Iz*u), -((a^2)*Cf+(b^2)*Cr)/(Iz*u)];

   B = [Cf/m; (a*Cf)/Iz];
   B = del.*B;
```

```
%slope at i
f{1} = [x(3); x(4); A(1,1)*x(3) + A(1,2)*x(4) + B(1);
      A(2,1)*x(3) + A(2,2)*x(4) + B(2)];

F_temp = zeros(2,1);
xy_plot = zeros(2,length(t));

for n = 1:length(t)-1

   F_temp = [u*cos(x(2,n)) - (x(3,n)+a*x(4,n))*sin(x(2,n));
        (x(3,n)+a*x(4,n))*cos(x(2,n)) + u*sin(x(2,n))];

  %find xi+.5 and slope at i+.5
   xtemp = x(:,n) + 0.5*dt*f{1};
   f{2} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

   %new i+0.5 and slope
   xtemp = x(:,n) + 0.5*dt*f{2};
   f{3} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

   %find xi+1 and slope
   xtemp(:) = x(:,n) + dt*f{3};
   f{4} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

   f{5} = (1/6) .* f{1} + (1/3) .* f{2} + (1/3) .* f{3} + (1/6) .* f{4};
   xtemp(:) = x(:,n) + dt*f{5};

   x(:,n+1) = xtemp;
   f{1} = f{5};
   F(:,n) = f{1};

   xy_plot(:,n+1) = xy_plot(:,n) + dt*F_temp(:);

end
```

```
    plot(xy_plot(1,:), xy_plot(2,:), 'color', colors(i), 'LineWidth', 2);
    grid on;
    axis equal;
    xlabel('X Position (m)');
    ylabel('Y Position (m)');
    hold on;

    radius(i,:) = u./x(4,:);

end

%title('All Season Tires Kinematics');
%legend('Cf = 21000','Cf = 25000','Cf = 30000');
%hold off;

% figure;
% hold on;
% plot(t,radius(1,:),'b','LineWidth',1);
% plot(t,radius(2,:),'r','LineWidth',1);
% %plot(t,radius(3,:),'g','LineWidth',1);
% grid on;
% xlim([1,5]);
% xlabel('Time (s)');
% ylabel('Radius (m)');
% legend('Cf = 21000','Cf = 25000','Cf = 30000');
% title('All Season Tires Radius of Turn');
% %hold off;
```

Part D - Tuning Handling - Handling Max Speed

```
clc; clear;

m = 1400; %kg
a = 1.14; %m
b = 1.33; %m
Iz = 2420; %kgm^2
u = 300/3.6; %km/hr

Cf_var = [21000,25000,21000]; %N/rad
Cr_var = [21000,21000,25000]; %N/rad
```

```
del = 0.1;

dt = 0.001;
t = 0:dt:10;

% IC at t = 0 (given eq7)
ic = [0;0;0;0];

%vector for i, i+1
%x = zeros(4,length(t));
%x(:,1) = ic;

%vector for i+0.5 (intermediate steps)
xtemp = zeros(4,1);

%vector array for slopes
f = {};
F = zeros(4,length(t));
y_a = zeros(size(t));

colors = ['b','r','g','c','k','m',"#EDB120"];

figure;
hold on;

for i = 1:length(Cf_var)

   x = zeros(4,length(t));
   x(:,1) = ic;

   Cf = Cf_var(i);
   Cr = Cr_var(i);

   % Define constants for dx2/d2t = Adx/dt + Bdel
   A = [-(Cf+Cr)/(m*u), -(a*Cf-b*Cr)/(m*u)-u;
     -(a*Cf-b*Cr)/(Iz*u), -((a^2)*Cf+(b^2)*Cr)/(Iz*u)];

   B = [Cf/m; (a*Cf)/Iz];
   B = del.*B;

   %slope at i
```

```
f{1} = [x(3); x(4); A(1,1)*x(3) + A(1,2)*x(4) + B(1);
      A(2,1)*x(3) + A(2,2)*x(4) + B(2)];

for n = 1:length(t)

   %find xi+.5 and slope at i+.5
   xtemp = x(:,n) + 0.5*dt*f{1};
   f{2} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

   %new i+0.5 and slope
   xtemp = x(:,n) + 0.5*dt*f{2};
   f{3} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

   %find xi+1 and slope
   xtemp = x(:,n) + dt*f{3};
   f{4} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

   f{5} = (1/6)*f{1} + (1/3)*f{2} + (1/3)*f{3} + (1/6)*f{4};
   xtemp = x(:,n) + dt*f{5};

   F(:,n) = f{5};
   x(:,n+1) = xtemp;
   f{1} = f{5};

   y_a(n) = A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);

end

% subplot(2,1,1);
plot(t, y_a(:),'color', colors(i), 'LineWidth', 1);
xlabel('Time (s)');
ylabel('Lateral Accel. (m/s^2)');
hold on;
grid on;
```

```
    % subplot(2,1,2);
    % plot(t, x(4,:),'color', colors(i),'LineWidth', 1);
    % xlabel('Time (s)');
    % ylabel('Yaw Rate (rad/s)');
    % hold on;

end

title('Performance Car Stability');
legend('Regular Tires', 'Wider Front Tires', 'Wider Rear Tires');
hold off;
```

Part D - Tuning Handling - Handling Gear

```
clc; clear;

m = 1400; %kg
a = 1.14; %m
b = 1.33; %m
Iz = 2420; %kgm^2
u = 300/3.6;

Cf_var = [21000]; %N/rad
Cr_var = [25000]; %N/rad
del = 0.1;

% Compute y(t) and psi(t)
dt = 0.001;
t = 0:dt:10;

%vector for i, i+1
x = zeros(4,length(t));
%vector for i+0.5 (intermediate steps)
xtemp = zeros(4,1);
%vector array for slopes
f = {};
F = zeros(4,length(t));

colors = ['b','r','g','c','k','m',"#EDB120"];

%figure;
```

```
hold on;

radius = zeros(size(x));

for i = 1:length(Cr_var)

   x = zeros(4,length(t));
   xtemp = zeros(4,1);

   Cf = Cf_var(1);
   Cr = Cr_var(i);

   %define constants for dx2/d2t = Adx/dt + Bdel
   A = [-(Cf+Cr)/(m*u), -(a*Cf-b*Cr)/(m*u)-u;
     -(a*Cf-b*Cr)/(Iz*u), -((a^2)*Cf+(b^2)*Cr)/(Iz*u)];

   B = [Cf/m; (a*Cf)/Iz];
   B = del.*B;

   %slope at i
   f{1} = [x(3); x(4); A(1,1)*x(3) + A(1,2)*x(4) + B(1);
         A(2,1)*x(3) + A(2,2)*x(4) + B(2)];

   F_temp = zeros(2,1);
   xy_plot = zeros(2,length(t));

   for n = 1:length(t)-1

      F_temp = [u*cos(x(2,n)) - (x(3,n)+a*x(4,n))*sin(x(2,n));
            (x(3,n)+a*x(4,n))*cos(x(2,n)) + u*sin(x(2,n))];

     %find xi+.5 and slope at i+.5
      xtemp = x(:,n) + 0.5*dt*f{1};
      f{2} = [xtemp(3); xtemp(4);
         A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
         A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

      %new i+0.5 and slope
      xtemp = x(:,n) + 0.5*dt*f{2};
      f{3} = [xtemp(3); xtemp(4);
         A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
```

```
        A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

    %find xi+1 and slope
    xtemp(:) = x(:,n) + dt*f{3};
    f{4} = [xtemp(3); xtemp(4);
        A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
        A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

    f{5} = (1/6) .* f{1} + (1/3) .* f{2} + (1/3) .* f{3} + (1/6) .* f{4};
    xtemp(:) = x(:,n) + dt*f{5};

    x(:,n+1) = xtemp;
    f{1} = f{5};
    F(:,n) = f{1};

    xy_plot(:,n+1) = xy_plot(:,n) + dt*F_temp(:);

  end

  plot(xy_plot(1,:), xy_plot(2,:), 'color', colors(i+2), 'LineWidth', 2);
  grid on;
  axis equal;
  xlabel('X Position (m)');
  ylabel('Y Position (m)');
  hold on;

  radius(i,:) = u./x(4,:);

end

title('Performance Car Kinematics')
legend('Regular Tires','Wider Front Tires','Wider Rear Tires');
hold off;

%figure;
%hold on;
%plot(t,radius(1,:),'b','LineWidth',1);
%plot(t,radius(2,:),'r','LineWidth',1);
%plot(t,radius(3,:),'g','LineWidth',1);
% grid on;
% xlim([0.5,5]);
```

```
% xlabel('Time (s)');
% ylabel('Radius (m)');
% legend('Cr = 21000','Cr = 25000','Cr = 30000');
% title('All Season Tires Radius of Turn');
% hold off;
```

Part D - Tuning Handling - Weight Reduction

```
clc; clear;

a = 1.14; %m
b = 1.33; %m
Iz = 2420; %kgm^2
u = 300/3.6;
Cf = 21000; %N/rad
Cr = 21000; %N/rad

m_var = [1000,1200,1400];

del = 0.2;

% Compute y(t) and psi(t)
dt = 0.001;
t = 0:dt:5;

%vector for i, i+1
x = zeros(4,length(t));
%vector for i+0.5 (intermediate steps)
xtemp = zeros(4,1);
%vector array for slopes
f = {};
F = zeros(4,length(t));

colors = ['c','k','m',"#EDB120"];

figure;
hold on;

radius = zeros(size(x));

for i = 1:length(m_var)
```

```
x = zeros(4,length(t));
xtemp = zeros(4,1);

m = m_var(i);

% Define constants for dx2/d2t = Adx/dt + Bdel
A = [-(Cf+Cr)/(m*u), -(a*Cf-b*Cr)/(m*u)-u;
  -(a*Cf-b*Cr)/(Iz*u), -((a^2)*Cf+(b^2)*Cr)/(Iz*u)];

B = [Cf/m; (a*Cf)/Iz];
B = del.*B;

%slope at i
f{1} = [x(3); x(4); A(1,1)*x(3) + A(1,2)*x(4) + B(1);
      A(2,1)*x(3) + A(2,2)*x(4) + B(2)];

F_temp = zeros(2,1);
xy_plot = zeros(2,length(t));

for n = 1:length(t)-1

   F_temp = [u*cos(x(2,n)) - (x(3,n)+a*x(4,n))*sin(x(2,n));
        (x(3,n)+a*x(4,n))*cos(x(2,n)) + u*sin(x(2,n))];

  %find xi+.5 and slope at i+.5
   xtemp = x(:,n) + 0.5*dt*f{1};
   f{2} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

   %new i+0.5 and slope
   xtemp = x(:,n) + 0.5*dt*f{2};
   f{3} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
      A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

   %find xi+1 and slope
   xtemp(:) = x(:,n) + dt*f{3};
   f{4} = [xtemp(3); xtemp(4);
      A(1,1)*xtemp(3) + A(1,2)*xtemp(4) + B(1);
```

```
        A(2,1)*xtemp(3) + A(2,2)*xtemp(4) + B(2)];

    f{5} = (1/6) .* f{1} + (1/3) .* f{2} + (1/3) .* f{3} + (1/6) .* f{4};
    xtemp(:) = x(:,n) + dt*f{5};

    x(:,n+1) = xtemp;
    f{1} = f{5};
    F(:,n) = f{1};

    xy_plot(:,n+1) = xy_plot(:,n) + dt*F_temp(:);

  end

  plot(xy_plot(1,:), xy_plot(2,:), 'color', colors(i), 'LineWidth', 2);
  grid on;
  axis equal;
  xlabel('X Position (m)');
  ylabel('Y Position (m)');
  hold on;

  radius(i,:) = u./x(4,:);

end

%title('All Season Tires Kinematics');
title('Reduced Mass Kinematics');
legend('m = 1000 kg','m = 1200 kg','m = 1400 kg');
hold off;

figure;
for j=1:length(m_var)
  plot(t,radius(j,:),'color', colors(j),'LineWidth',1);
  hold on;
end
grid on;
xlim([1,3]);
xlabel('Time (s)');
ylabel('Radius (m)');
%legend('Cf = 21000','Cf = 25000','Cf = 30000');
title('Reduced Mass Radius');
hold off;
```

Appendix B - Solving the Eigenvalue problem

For solving the second order coupled ODE analytically, the solution can be assumed to take the form:

$$X(t) = C_1 U_1 e^{\lambda_1 t} + C_2 U_2 e^{\lambda_2 t}$$

Where C_1 and C_2 are constants solved for using initial conditions. While U and λ are eigenvectors and eigenvalues respectively. The full derivation for these values is shown below. The important principle is that if one of the eigenvalues is positive, the function X(t) will diverge thus making an unstable system.

$$if\ \lambda > 0, \lim_{t\to\infty} e^{\lambda t} = \infty$$
$$if\ \lambda < 0, \lim_{t\to\infty} e^{\lambda t} = 0$$